Леонид Орлов

Web-сайт без секретов

издание второе, дополненное и исправленное

УДК 004.738.5
ББК 32.973.26-018.2
О66

Рецензенты:
кандидат экономических наук *Б. К. Самсонов*
доктор физико-математических наук *А. В. Наумов*

Орлов Л. В.
О66 Web-сайт без секретов. / Л. В. Орлов. — 2-е изд. — М.: Бук-пресс, 2006. — 512 с.

ISBN 5-9643-0032-4

Создание собственного web-сайта — это прежде всего творчество. В этой сфере можно полностью показать себя, не ограничиваясь ни в содержании, ни в объеме. Почему бы Вам не создать свой собственный web-сайт и не «раскрутить» его во Всемирной компьютерной сети? Тогда воспользуйтесь современными технологиями Интернет и этим достаточно полным пособием, которое написано опытным разработчиком и дизайнером web-приложений.

Одна из «изюминок» книги — описание конкретных приемов работы по созданию web-страниц, публикации их в Интернет и проверке работоспособности всего сайта.

Издание адресовано в первую очередь тем пользователям ПК, которые желают создавать профессиональные сайты в Интернет, а также оно может быть использовано в качестве практического пособия разработчиками web-сайтов, дизайнерами и всеми, кто имеет отношение к web-проектам.

УДК 004.738.5
ББК 32.973.26-018.2

ISBN 5-9643-0032-4

Содержание

Часть 1. Дизайн для Всемирной паутины

Часть 2. Основы создания гипертекстового документа

Часть 3. Описание языка гипертекстовых документов

Часть 4. Редакторы web-страниц

Часть 5. Создание Интернет-портала

Часть 6. CGI, PHP, Perl, MySQL и CMS системы

Часть 7. Life Site CMS — система создания и развития сайтов

Часть 8. «Раскрутка» сайта

Часть 9. Заработок при помощи своего сайта

Часть 10. Уроки мастерства

Часть 11. Тонкости и секреты

Часть 1. Дизайн для Всемирной паутины

Глава 1. Пройдемся по «понятиям»

Web-дизайн — это прежде всего творчество, в котором тебя никто не ограничивает. Только в этой сфере вы можете полностью показать себя, не ограничиваясь, ни в содержании, ни в объеме. Если например брать корреспондентов и журналистов. Они написали материал, долго трудились, и подали его редактору, а редактор взял и выбросил статью, ссылаясь на то, что они написали несколько слов про знаменитого «жулика», который занимает высокий пост в местной Думе. А вам предоставляются все возможности (естественно это не значит, что вы можете материть каждого попавшегося депутата), для того чтобы писать, рисовать и т.д. Потом вывесить это все на сайте и пусть все любуются.

Но в основном web-дизайн означает умение красиво оформить web-страничку на сайте. Если вы написали несколько слов, на обычном default'ном фоне, это не дизайн. Не ограничивайтесь этим. Старайтесь, пытайтесь и у вас все получится.

Причины, по которым люди решают подготовить и опубликовать во Всемирной сети собственный web-ресурс, могут быть совершенно различными. Если речь идет о домашней страничке, то основным движущим фактором для web-мастера является стремление разместить в сети информацию, которую впоследствии можно использовать при поиске работы или интерактивном общении с другими людьми посредством Интернета. Иногда домашняя страничка служит средством для публикации, например литературных произведений, рисунков, музыки или научных исследований автора. Таким способом владелец странички может найти издателя для своих рассказов, организаторов выставки для своих картин, продюсера для музыкального проекта или спонсора, готового финансировать его разработки. В ряде случаев посредством домашней странички можно просто зарабатывать деньги.

Некоммерческий сайт, представляющий посетителям информацию по какой-либо конкретной тематике, вырастает, как правило, хорошо продуманной, грамотно выполненной и регулярно обновляемой домашней странички. Каким бы ни был подобный ресурс — развлекательным порталом, музыкальным сайтом или проектом, посвященным съемке и монтажу любительского видео, он создается обычно с теми же целями, что и домашняя страница. Иногда помимо чисто практических интересов, которыми руководствуется владелец некоммерческого сайта, им движет честолюбие в лучшем смысле этого, то есть стремление обрести популярность и признание у сетевой аудитории. По статистике большинство талантливых и популярных проектов российского Интернета было создано именно честолюбивыми авторами. Не стоит забывать о том, что web-дизайн — это весьма молодая и быстро развивающаяся сфера Интернет-технологий, в которой всегда найдутся высокооплачиваемые вакантные места для толковых специалистов. Налицо тенденция активного освоения Интернета малым и средним бизнесом, в русскоязычную часть Всемирной Сети стали вливаться большие деньги, поскольку данный вид капиталовложений является сегодня одним из наиболее перспективных. По мере насыщения рынка сформируется повышенный спрос на web-дизайнеров и программистов Интернет приложений, а спрос, как известно, рождает предложение.

Что же представляет собой тот «полигон», на котором начинающий web-мастер может испытать собственные силы и способности? Разумеется, это собственная домашняя страничка, ваш первый самостоятельный проект или первый любительский некоммерческий сайт, который вы создаете своими руками.

Коммерческие интернет-проекты организуются с расчетом на получение финансовой прибыли, причем эта прибыль может быть как прямой, так и косвенной. Под косвенной прибылью подразумеваются те неоспоримые преимущества, которые дает грамотно подготовленный web-сайт его владельцам: по сравнению со стоимостью традиционной рекламы, реклама в Интернете практически бесплатна, что с учетом огромной аудитории Всемирной Сети является огромным плюсом. Более того, компания-владелец web-ресурса получает в свое распоряжение именно целевую аудиторию, поскольку ее сайт будут посещать лишь те пользователи, которые нашли его по набору ключевых слов, введенных в форму запроса поисковых машин или в соответствующем тематическом разделе интерактивных каталогов. Web-страница является лучшим и наиболее выгодным решением при поиске как отечественных, так и партнеров и потенциальных клиентов. Никакой иной способ рекламы не даст вам такой приток заинтересованных лиц и заказчиков, как хорошо продуманный и правильно разработанный web-сайт.

Получение прямой прибыли подразумевает создание проекта, изначально рассчитанного на привлечение финансовых средств непосредственно из Интернета. Существуют компании, живущие только за счет, созданных силами их сотрудников, web-ресурсов. Это могут быть поисковые машины или информационные порталы, продающие на своих страницах рекламное пространство; Интернет магазины, посетители которых могут заказать какой-либо товар прямо в режиме on-line, получив его впоследствии по почте; система интерактивных аукционов или виртуальное рекламное агентство. Возможностей зарабатывать деньги в Интернете такое множество, что их трудно было бы даже кратко описать. Как бы то ни было, разработка коммерческого проекта, безусловно требует большого времени и продолжительного, кропотливого труда.

Глава 2. «Вид» дизайна

Итак, уважаемые сайтостроители, скажите мне, сколько видов дизайна вы знаете? А? Так, слышу справа кричат — жесткий... Верно! А еще? Резиновый... Так! А какой лучше? Ой! Не надо так кричать... Давайте разбираться.

Давайте дадим определения, чтобы не было разногласий.

Жесткий дизайн — табличный дизайн, в котором величина каждой ячейки либо задана в пикселях, либо задается картинкой или другим элементом, помещенной в эту ячейку.

Резиновый дизайн- табличный дизайн, в котором ширина одной или нескольких ячеек задается в процентах от ширины окна.

Ну мы с вами, как образованные люди, сразу понимаем, что жесткий дизайн гораздо проще разрабатывать! А что? Все элементы фиксированной ширины, расположены статически друг от друга... Так что тем, кто хочет иметь больший контроль над разметкой страницы, следует разрабатывать страницы фиксированной ширины, которая будет оставаться постоянной для всех пользователей, независимо от размера монитора или изменений размеров окна. Этот подход основан на принципах создания страниц в издательском деле, таких как поддержание постоянной сетки, отношений элементов, расположенных на странице, и удобные длины строк. Красота... Как в журнале!

В этом месте давайте остановимся и пораскинем мозгами, что нам это сулит.

Достоинства:

- Страница будет выглядеть одинаково независимо от разрешения монитора. Это представляет огромный простор для создания сложных и стильных дизайнов.
- Страницы и столбцы с фиксированной шириной обеспечивают лучшее управление длинами строк. Слишком длинные строки неудобны для чтения.

Недостатки:

- У жесткого дизайна есть только одно оптимальное разрешение экрана. Если при разработке упор был сделан на совместимость (оптимальный размер 640x480), то уже на 1024x768 поля занимают почти половину ширины страницы — и чем дальше, тем хуже.
- Если при разработке акцент был сделан на «среднее» разрешение (800x600), то у пользователей маленьких мониторов появляется горизонтальная полоса прокрутки, и часть содержания им становится не видна. Некоторые разработчики ориентировались по своему монитору (1024x768), и у них даже для 800x600 (а это почти 50% всех пользователей!) страница выглядит плохо — а для 640x480 обычно нечитабельна.
- Стремление полностью контролировать отображение страницы означает своего рода выступление против среды. Web — это не печатное издание; у него нет «правильных стандартов», а HTML — язык универсальный с логической разметкой!

М-да... куда ни кинь — всюду клин...

Тут на помощь приходит гибкий дизайн. Web-страницы по умолчанию гибкие. Текст и элементы HTML-файла попадают в окно браузера, заполняя все доступное пространство, вне зависимости от размеров монитора. Если размер окна браузера изменяется, элементы повторно выводятся, чтобы настроиться на новые размеры. В этом и проявляется сущность Web.

Разработчиков поначалу шокирует непредсказуемость места появления элементов страницы, но потом они обычно обучаются обходиться без полного управления страницей.

Итак, что мы имеем?

Достоинства:

- Страницы будут отображаться на мониторах с разным разрешением; гибкую страницу можно настроить для выводов на любом дисплее.
- Заполнено все пространство монитора, отсутствует нежелательное свободное место.
- Дизайн наиболее близок по духу к HTML.

Недостатки:

- На больших мониторах длина строки может оказаться чрезмерной, что ухудшает условия для чтения.
- Результаты гибкого дизайна часто непредсказуемы.
- Браузерам (всем, но особенно Нетскейпу версии 4) очень тяжело переваривать ячейки переменной ширины. Каждому, кто писал совместимый HTML, знакомы ячейки шириной 100%, которые не занимают всю свободную площадь; ячейки фиксированной ширины, которые занимают больше, чем им положено; ячейки с заданной одинаковой шириной, но неизменно выходящие разной ширины и т.д. — перечислять фокусы каждого из браузеров можно очень долго.
- Отливка «резиновой» страницы занимает вдвое-втрое больше времени от аналогичной «жесткой».

Так что теперь видно, что однозначно ответить, какой дизайн подходит именно вам должны вы сами. Если ваша страница имеет фиксированный размер, то в этом нет ничего страшного! Если она рассчитана на разрешение 800x600, и отцентрирована по центру экрана, то даже на больших мониторах она будет смотреться очень прилично, так как не будет наблюдаться эффекта пустого экрана справа. (Но учтите, что при использовании этого метода невозможно точно расположить таблицу относительно фонового изображения.) Единственным недостатком (притом очень крупным) являются шрифты фиксированного размера. В связи с этим дизайн сайта имеет полностью гибкую структуру. Вы должны всегда помнить, что выбор размера шрифта — право пользователя! Ведь страницу вы делаете для него? Тогда пользователь должен выбирать как ему удобно смотреть вашу страницу. Можно использовать также комбинированный подход, когда страница состоит из столбцов (или фреймов), заданных комбинациями абсолютных и относительных размеров. В этом случае при изменении размера окна один столбец сохраняет прежнюю

величину, в то время как остальные изменяют размер и заполняют свободное место. В большинстве случаев наиболее предпочтительно использовать именно этот метод. Подводя итог, еще раз напомним, что выбор в любом случае за вами. В web-дизайне нет строгих законов и правил. Но все же рекомендуется, если есть возможность, использовать блочный, а лучше резиновый дизайн. Да, это дольше и сложнее, но это наиболее удобно пользователю, и близко по духу идее Web.

Глава 3. Критические ошибки

Ниже приводятся 10 ошибок дизайнеров (даже, скорее, верстальщиков), которые способны отпугнуть посетителя с вашей страницы.

1. Долгая загрузка страниц. Пользователь не будет по полчаса ждать загрузки ваших страниц, даже если находит их интересными. Согласитесь, обидно потерять целевого посетителя из-за пары неоптимизированных картинок... Уделите особое внимание оптимизированию размера документов! Первым делом решите, нужны ли картинки эти или нет? Затем все что осталось хорошо оптимизируйте. Вероятно, вам следует использовать специальные программы для оптимизации кода; можно использовать возможности сервера по архивации данных перед отправкой клиенту (но это должен поддерживать ваш хостинг).

В конечном счете, конечно скорость загрузки зависит от множества факторов — от канала хостера и скорости сервера, до пропускной способности модема конечного пользователя; но вы, в любом случае, должны сделать все от вас зависящее, чтобы страница грузилась не больше 20 секунд...

2. Неудачная цветовая схема, шрифт, кривая навигация и верстка сайта. С этим все понятно и добавить нечего. Все, к сожалению, с этим встречались...

3. Использование новейших технологий. Недостаточное тестирование. Пользователь, столкнувшись с ошибкой JavaScript или SSI, вряд ли будет разбираться в чем дело, он просто уйдет.

Прежде чем что-то использовать, кроме HTML, обязательно протестируйте, лучше неоднократно! Еще лучше на локальном сервере. Это касается директив SSI, скриптов, CSS, да и «битые ссылки» очень раздражают...

4. Перегрузка апплетами, скриптами, анимацией, флешем. Многие, особенно начинающие, думают, что чем больше всяких наворотов, тем

«круче». Поверьте, это совсем не так! Можно иногда видеть страницы, где каждый апплет по минуте грузится... Это никуда не годно, пользователь даже не увидеть вашего наворота, он просто уйдет.

5. Всплывающие окна. Если пользователю понадобится открыть новое окно, он это сделает сам, не стоит делать всплывающие окна (pop-up) с рекламой, это только раздражает.

6. Много банеров. Реклама не только раздражает пользователей, но еще и портит дизайн и «утяжеляет» страницу. Помните золотое правило — не больше двух банеров на страницу: один вверху, один внизу. Если посетитель попадет в «засаду диких банеров», то он, скорее всего, даже не станет разбираться, интересная статья или нет, он просто уйдет.

7. Редкое обновление. Людям нужна достоверная информация. Если дата последнего обновления — полтора года назад, то посетитель может посчитать сайт мертвым, а информацию неактуальной и даже не станут ничего читать... Обновляйте сайт как минимум раз в 2 недели, лучше чаще. Если не будет новых материалов, пользователи не будут возвращаться.

8. Грамматические ошибки. Если пишете на русском языке, то будьте любезны писать грамотно! Люди не любят грамматических ошибок, если ошибок много, пользователи не будут к вам возвращаться. Понятно, что не у всех лады с правописанием, но пользуйтесь, хотя бы, проверкой орфографии в Ворде... Не забывайте и про пунктуацию — расставляйте хотя бы очевидные знаки препинания. Соблюдение этого нехитрого правила повысит ваш авторитет в глазах читателя и облегчит чтение и восприятие информации.

9. Отсутствие старой информации. Старая информация не значит ненужная! Не прячьте старые статьи от посетителя — сделайте архив, это может повысить посещаемость ресурса.

10. Оформление текста. Здесь все огрехи верстальщика — неудачный шрифт, слишком длинные абзацы, отсутствие вразумительных заголовков, отсутствие выделения ключевых моментов, и прочее и прочее...

Глава 4. Структура оптимизированных страниц

Ключевые слова на странице играют решающую роль в позиционировании сайта в результатах поисковых систем. Основополагающими факторами ранжирования в поисковых системах являются следующие моменты:

1. Плотность ключевых слов на странице.

Оптимальный вариант частоты ключевых слов — 3-10%, а также присутствие ключевых слов в тэгах «keywords» и «description».

2. Наличие ссылок на ваш сайт извне с использованием ключевых слов в описании.

При оптимизации страницы под определенное ключевое слово вам следует учесть, что поисковая система рассматривает вашу страницу не как отдельный элемент, а как часть организма вашего сайта в целом, т.е. учитывается наличие данного ключевого слова не только на определенной странице, но и на других страницах сайта. При этом, учитываются повторяющиеся элементы на сайте, т.е. панель навигации на сайте с ключевым словом и ссылкой на одну и ту же (оптимизированную) страницу №1 не играет весомой роли, зато наличие на страницах №2, №3 разных текстов с этим ключевым словом учитывается и поднимает вес оптимизированной страницы №1.

В целом, формируйте страницы №2 №3 №..., исходя из следующих правил:

- Эксклюзивный материал каждой страницы (не повторяйте какой-либо текст, уже опубликованный в сети)
- Присутствие ключевого слова в мета-тэгах и title. Надеюсь, здесь все понятно.
- Наличие слов-синонимов на страницах.
- Простой, соответствующий языковым нормам текст. Исключите всевозможный спам (бессмысленное нагромождение глаголов или существительных), используйте только нормальный язык.

В чем же разница между вашей оптимизированной страницей №1 и страницами №2,№3...?

- Самое главное — это присутствие на странице №1 заголовка <H1> вверху, с ключевым словом (не забудьте о css для нужного отображения текста).
- Более высокая плотность ключевого слова на странице по сравнению со страницами №2,3. 3. Ссылки с других сайтов на страницу №1 с цитированием самого ключевого слова. Чем больше таких ссылок на страницу, тем лучше.

Страницы №2,3 играют вспомогательную роль в поднятии рейтинга страницы №1. Регулярно пополняйте такие страницы новыми ма-

териалами — №4,5. Это поможет закрепить рейтинг страницы №1. Также, регулярно обменивайтесь ссылками с другими сайтами. Проследите, чтобы ссылка вела именно на вашу страницу №1. При этом, постоянно меняйте ее описание. Пусть с разных сайтов на вашу страницу ведет разное описание.

Все это нужно для пауков поисковых систем. Различные рекомендации вашей страницы №1 будут учтены поисковыми системами.

Еще о структуре страницы №1. Не обязательно ее часто обновлять. Однако, рекомендуется постоянно пополнять ее ссылками на страницы №2,3.. Как было отмечено ранее, число этих страниц должно постоянно расти. Частые обновления в структуре сайта поднимут вашу страницу №1 еще выше.

И, наконец, о выборе ключевых слов. Прежде, чем оптимизировать страницу под определенное ключевое слово (слова), просмотрите уровень конкуренции в запросах поисковых систем.

Исходя из опыта, необходимо отметить, что есть ключевые слова, по которым можно занять прочные позиции в первой пятерке в Rambler, Yandex и Aport всего за две недели, т.е. при первой индексации созданных страниц.

Глава 5. «Непрошеные» окна

Наверняка, вам часто приходится закрывать «непрошеные» окна браузера, открывающиеся во время навигации по сети. Большинство пользователей ненавидят всплывающие окна, которые раздражают своей навязчивостью и непредсказуемостью. Но если вы все же большой приверженец использования pop-up страниц, можно предложить одно из решений их использования.

Одна из главных целей сайта — удержать посетителя, заставить его вернуться на сайт снова. Для этого мы используем всевозможные «трюки»: предлагаем добавить наш сайт в Избранное, сделать его стартовой страницей пользователя, подписаться на рассылку.

Именно на последнем — рассылке, мы и заострим внимание.

Создайте страницу pop-up.html, на которой будет только форма подписки на рассылку. Измените фон в соответствии с дизайном сайта. Вверху напишите краткий и эффектный заголовок, типа «Эксклюзивные материалы по (вашей теме) в еженедельной рассылке».

То же напишите в title. Все. Добавьте ключевые слова и описания в meta description и keywords — пусть поисковики видят эту страницу.

В каждой странице сайта (кроме pop-up.html) между **<HEAD> </HEAD>** пропишите следующий код:

```
<!------ BEGIN CONSOLE CODE ------->
<SCRIPT language=Javascript>
<!-- var exit=true;
function exitcnsl()
{if (exit)
open("http://www.имясайта.com/имястраницы.html",
"new_window","toolbar=0,location=0,status=0,
menubar=0,scrollbars=0,resizable=0,
width=800,height=600,top=0,left=0");}//-->
</SCRIPT>
<!------ END CONSOLE CODE ------->
```

В тэге **<BODY>** всех страниц пропишите следующий код:

```
onUnload="exitcnsl()"
```

Можете изменять параметры pop-up окна: **scrollbars**, **location**, **menubar**, **resizable**.

Последующее потребует от вас немного терпения: пропишите в каждой **<A HREF>** ссылке на внутреннюю страницу сайта следующий код:

```
onClick="exit=false"
```

Например:

```
<A HREF="index.html" onClick="exit=false">
```

Внешние ссылки можно не снабжать этим кодом, тогда при выходе по ним пользователю также будет открываться pop-up окно. Если же вы снабдите этим кодом абсолютно все (и внутренние, и внешние) ссылки, тогда pop-up окно будет появляться только при закрытии браузера, но не при переходе с сайта.

Не добавляйте другие pop-up страницы, не вставляйте графику в ваше единственное pop-up окно, т.к. пользователи нетерпеливы — никто не будет ждать, пока откроется все окно. Созданная pop-up страница должна буквально вылетать в готовом виде.

Как правило, реклама товаров и услуг, отображаемая в pop-up окнах отталкивает, но предложение подписаться на рассылку может показаться ненавязчивым, и даже необходимым действием, т.к. рассылка — это один из самых удобных способов, по мнению пользователей, получать необходимую информацию.

Встраивайте pop-up окно в страницы сайта периодически — например, неделя с pop-up окном — две недели без него. Ваших повторных посетителей (тех, кто уже подписан на рассылку) будет раздражать постоянное всплывающее окно.

Практика раскрутки сайтов показывает, что использование pop-up окон, таким образом, значительно увеличивает число подписчиков.

Глава 6. Студия web-дизайна

Студия дизайна. Много ли надо чтобы организовать ее? По большому счету можно даже не регистрироваться как юридическое лицо. Ведь что такое организация? Минимум два человека, стремящихся к достижению общей цели. Целью здесь выступает удовлетворение человеческих потребностей в области web-дизайна, а также сопутствующих ей и извлечение прибыли. У вас также могут отсутствовать такие существенные затраты как приобретение (аренда) офиса, необходимого оборудования, программного обеспечения...

Чтобы удовлетворять потребности нужно обладать соответствующими навыками либо самому, либо иметь под рукой людей, которые этими навыками владеют. Рассмотрим основные навыки, необходимые студии для соответствия профессиональному уровню, который и является одним их главных критериев достижения успеха в этом деле.

Прежде всего — это дизайн как таковой.

Студия должна иметь профессионального дизайнера. Здесь важно отметить, что дизайнер и художник — две разных ипостаси. Дизайнер, в отличие от художника использует в своей работе, а нередко и кладет в ее основу не им созданные художественные ценности. Норма в профессиональном дизайне сегодня — использование покупных фотографий и заказной графики, не говоря уже о шрифтах. Дизайнер должен владеть передовыми профессиональными графическими пакетами для обработки и создания растровой, векторной графики, а нередко и специализированным программным обеспечением для рендеринга трехмерных объектов. Т.е. для дизайнера ярко выражен его синтез с компьютерными технологиями.

Художник. Дело обстоит совершенно иначе, когда мы говорим о художнике. Умение рисовать — талант совершенно отдельный от таланта дизайнера, и к тому же в гораздо большей степени врожденный и слабо поддающийся развитию. Работа художника это всегда элемент индивидуальности, особый почерк, недоступное фотографии парадоксальное сочетание начал обобщения и детализации. Привлекательность художественной графики заключается в неизбежной неточности мазков, аморфности пятен краски, брызгах, подтеках... Время от времени в услугах художника появляется необходимость. Он также необходим для разработки логотипов. Качественный, стильный дизайн это очень важно, ибо по одежке встречают. Но это далеко не все.

Программинг. Наличие группы талантливых профессиональных программистов — это крайне важный фактор успеха. Нельзя определить, что важнее: дизайн или программинг, т.к. эти два направления призваны дополнять друг друга. Чистый HTML-код web-страниц, корректное отображения оных в различных популярных версиях браузеров, владение DHTML, Java Script, Java, CSS, XML, Perl, PHP, ASP, VBasic, MySQL, MS SQL, Oracle, Flash — все это мгновенно поднимает студию на качественно иной уровень — уровень профессиональных web-студий, услуги которых оцениваются в тысячи долларов.

Многое зависит от руководителя — лидера студии. Он должен, как минимум, иметь четкое представление об аспектах работы каждого профессионала студии и самому быть профессионалом в одной из областей. Он отвечает за планирование, организацию работ над очередным проектом, мотивацию людей и контроль над ходом выполнения работ. Креативное мышление, четкая координация усилий, вписывание в поставленные временные рамки, справедливое вознаграждение работников соответственно их вкладу в проект — признаки грамотного руководства и успеха студии в целом.

Руководитель — лицо студии. На нем лежит бремя общения с клиентом. Общения корректного, вежливого, внимательного, направленного на демонстрацию профессионализма студии, выяснение потребностей и предпочтений клиента, убедительную презентацию готового проекта.

Численность персонала студии web-дизайна прямо зависит от объема работ. Логично, что чем более раскручено имя студии, тем больше клиентов у студии. Соответственно необходимость в своевременном выполнении заказов диктует дополнительный набор персонала.

Необходимо отметить также, что совокупные знания и средства студии должны обеспечить реализацию таких направлений как интернет-маркетинг — раскрутка сайтов в сети; работу над контентом — информационным наполнением сайтов; консалтинговые услуги — квалифицированные консультации; производство качественных банеров с потенциально высоким CTR; постоянное отслеживание и ориентацию на передовые течения и технологии, имеющие отношение к web-дизай-

ну. В этой связи, на сегодняшний день можно порекомендовать к освоению технологию, за которой будущее — Macromedia Flash.

Секрет успеха для вас может крыться также в осознании одной важной вещи. Интернет имеет способность стирать для своих обитателей такие понятия как географическая зона, уровень жизни. И будь вы хоть чукотской студией web-дизайна (но профессиональной) — вы имеете уникальную возможность предложить свои услуги всему миру и получить за свою работу вознаграждение, эквивалентное стандартам развитых стран с развитой экономикой.

Успех студии web-дизайна делают люди этой студии. Четкая специализация под призмой профессионализма; грамотная координация всех звеньев системы; впечатляющий набор знаний и навыков, доступных студии; качество — как ассоциация имени студии; известность и слава — как совокупность вышеперечисленного — вот те секреты, ведущие к успеху.

Глава 7. «Это» — плохо!

«Кнопки» — начинающие web-строители любят вешать на главную страницу до 10-20 кнопок — счетчики, топы, эмблемы каталогов, ссылки на дружественные сайты. Подумайте, может что-то из этого бесполезное увеличение объема графики.

«Длинные тексты» — иногда видишь «сайт» целиком умещающийся в одном большом файле. Большая ошибка! Пользователь не должен прокручивать экран больше чем на 2,5-3 экрана.

«Фон» — классическая ошибка начинающих — это использование цветистых, а зачастую и разных типов фона в пределах сайта. Старайтесь создать единую атмосферу на сайте, и фон не должен выпячиваться вперед, показывая свою важность. Попробуйте использовать классику — черный текст на белом фоне.

«Выпадающие окна» — использование pop-up окон является нарушением Нетикета, самой обычной реакцией на появление таких окон при каждом клике на ссылке будет уход с такого сайта.

«Горизонталь» — часто при жестком дизайне (все расстояния указываются в пикселях) появляется горизонтальная полоса прокрутки, если web-строитель имеет хороший монитор и видео-карту. Обязательно проверьте свой сайт при разрешении 800x600 при 256 цветах.

«Разные шрифты» — при создании сайтов применяйте правило «Не более пяти шрифтов на странице» — при этом жирные шрифты считаются за отдельный шрифт, тоже касается и их размера.

«Under Construction» — самая бесполезная надпись из всех. Если вы переделываете раздел, лучше убрать ссылки на него. Как правило, на сайтах объема больше среднего (>50 файлов) целесообразно перейти на использование SSI или PHP. Это позволит вам быстро менять общие элементы, в особенности меню.

Глава 8. Дизайн без «дизайна»

Многие люди, только попав в среду дизайна, полагают, что смогут достичь успеха, если научатся делать качественный дизайн. При этом они забывают о главной стороне дизайнерской деятельности — необходимости дизайн продавать. Дизайнер — это профессия, а дизайн — это товар, который надо продать. И продают этот товар дизайнеры. Представляю нестройный хор голосов, протестующих против такой позиции; мол дизайнеры — творцы, а торгашеством должны заниматься агенты и sale-менеджеры. В корне неверная позиция и это поймет каждый, кому приходилось сдавать работу руководителю проекта или показывать ее sale-менеджеру.

Большинство sale-менеджеров заинтересовано только в одном — в продажах. Они сидят на процентах — это их хлеб и их масло. Поэтому они будут хотеть всего, чего захочет их клиент, лишь бы тот купил продукт. Им не до творчества и не до качества, у них одна цель — сделка. Только очень грамотные и дальновидные sale-менеджеры понимают, что продавать некачественные сайты в угоду клиентам не выгодно в долгосрочной перспективе. Только самые талантливые sale-менеджеры понимают, что каждый должен быть профессионалом в своем деле и доверяют дизайнерам. И лишь единицы из них придумывают систему аргументации под работу дизайнера, а не требуют дизайн, соответствующий их системе аргументации. Такие продавцы — подарок для дизайнера, а подарков не бывает много. Чаще дизайнеру приходится доказывать клиенту, своему директору и sale-менеджеру, что его дизайн хорош, и не надо на заглавной странице размещать всякую-батву-которую-клиент-захотел-увидеть. Так что действительно хороший дизайнер — это тот, кто умеет не только сделать качественный дизайн, но и аргументировать свой выбор. Иначе, невнятно пробормотав жалкие оправдания своей действительно качественной работы, дизайнер будет вынужден изуродовать ее в угоду директорско-клиентурной братии или же он просто ли-

шится заказа, что еще хуже. Поэтому повторяю еще раз, дизайнер должен уметь продавать свой дизайн.

Приведем несколько приемов, как это можно сделать грамотно.

Во-первых, терминология.

Люди привыкли к тому, что математика, биология и физика — это сложно, потому что они угробили десять лет в школе, мучаясь над учебниками и домашними заданиями. Дизайну их в школе не учили, не ставили двойки за контрольные. Поэтому люди воспринимают дизайн как бытовую область знаний — там все просто. Вам надо прежде всего сломать этот устоявшийся стереотип. Не стесняйтесь пользоваться профессиональной терминологией в разговоре с клиентом (под клиентом понимается любой человек, которому вы пытаетесь доказать преимущество вашего дизайна). Используйте слова: визуальный ряд, эргономичность, юзабилити, цветовой диссонанс, композиция, структурирование, концепция, цветоделение, гамма, контент, логические блоки, фреймы, текстуры, контрастные образы, метафора, пропускная способность, законы восприятия и т.д. и т.п.

Вся эта терминологическая броня позволит вам сломить сложившееся мнение о простоте дизайна. Клиент, как и любой нормальный человек, в жизни не слышал о законах восприятия; композиция для него — это то, что он так и не смог освоить в пятом классе на уроках рисования; а юзабилити для него вообще темный лес. Благодаря этому клиент впервые для себя откроет очевидные для вас вещи, что дизайн имеет концепцию, цветовую формулу, логическую схему, свой ритм и насыщенность. Тогда и только тогда клиент поймет главное, что он ни черта не понимает в дизайне, что он не профессионал, который платит деньги профессионалам за грамотную работу, которую сам выполнить не способен.

Однако нельзя и переусердствовать с умными словами, чтобы у клиента не сложилось мнение, что его тычут мордой в дерьмо и держат за дебила. В этом случае он может просто обидеться и уйти. Вы должны выглядеть умным и корректным. Это первый залог того, что клиент постесняется лезть к вам со своими неграмотными советами. Просто вежливо и ненавязчиво дайте ему самому понять, что его советы неграмотны. Он это поймет и будет даже счастлив, что нашел такого профессионала и не зря тратит деньги.

Второе правило общения с клиентом заключается в том, чтобы он увидел как вы работаете.

Покажите ему невзначай заготовку изображения в Photoshop или макет сайта в Dreamweaver. При этом постарайтесь, чтобы интерфейс

выглядел максимально навороченным. Откройте побольше менюшек, даже если они и не нужны в принципе, — клиент все равно не поймет. Главное, чтобы он понял как кропотлива и сложна ваша работа. Нужно чтобы клиент пришел к выводу, что он не только нашел нужного человека, но и не переплачивает ему. Поймите, что для клиента знания компьютера ограничивается MS Word и MS Exel. У большинства клиентов есть сознательное мнение, что компьютер — это очень сложно и, одновременно, бессознательная убежденность, что там нет ничего сложнее MS Word. Дайте понять клиенту, что ваши программы по сложности отличаются от тех, в которых он работал, как истребитель отличается от воздушного змея. При этом не давайте ему засиживаться у монитора, а то еще чего доброго догадается, что имидж в некоторой части напускной. Пусть для него ваш труд остается тайной за занавесом.

Третье правило общения с клиентом касается подачи материала. Мы не будем приводить цифр, говорящих какое влияние оказывает упаковка на объемы продаж. Но поверьте, это влияние огромно. Вспомните старую поговорку — «Встречают по одежке...». У покупателя складывается первое впечатление о товаре именно по внешнему виду. Это касается и дизайна. Как ни парадоксально это звучит, но дизайну, как и любому товару, нужна хорошая упаковка. Сохраните материал на дискете или CD. Подпишите цифровой носитель. При этом пусть там будет не только название, но и краткие тех. характеристики. Не поленитесь сделать качественные распечатки вашего дизайна.

Поместите их в красивую папку. Не бойтесь потратить лишнюю пару сотен на оформление своего товара. Но главное снабдите распечатки технической документацией: графики цветовых формул использованных при создании, схемы расположения элементов, логические взаимосвязи, описания композиции. Напечатайте текстовое сопровождение с объяснением своей работы. Сделайте все это и клиенту будет приятно. Он почувствует, что не зря потратил деньги.

Глава 9. Гипертекстовый документ

Под гипертекстовым документом понимают документ, содержащий так называемые ссылки на другой документ. Реализовано все это через протокол передачи гипертекста HTTP (HyperText Transfer Protocol).

Информация в документах Web может быть найдена по ключевым словам. Это означает, что каждый обозреватель Web содержит определенные ссылки, через которые образуются так называемые гиперсвязи,

позволяющие миллионам пользователей Интернет вести поиск информации по всему миру.

Гипертекстовые документы создаются на базе языка HTML (HyperText Markup Language). Этот язык весьма прост, управляющие коды его, которые, собственно, и компилируются обозревателем для отображения на экране, состоят из текста ASCII. Ссылки, списки, заголовки, картинки и формы называются элементами языка HTML.

Глава 10. Универсальный локатор ресурса

Web может открывать доступ к другим ресурсам Интернет, например к электронной почте, FTP, Gopher, WAIS или конференциям Usenet. Одно из таких средств серфинга по Web обеспечивает встроенный в Windows обозреватель Microsoft Internet Explorer. Документ в Интернет ищется по так называемому адресу URL (Uniform Resource Locator), синтаксис которого следующий:

```
protocol://hostport/path
```

hostport – адрес сервера с соответствующим номером порта. Этот параметр отображает так называемую машинную адресацию. Машинная адресация может быть числовой или буквенной.

path – путь.

Вместо аргумента **protocol** может стоять:

- **http** — любая гипертекстовая информация.
- **ftp** — протокол передачи файлов.
- **telnet** — терминальный доступ.
- **gopher** – «предшественник» WWW.
- **afs** — файловая система Интернет.
- **news** — конференции Usenet.
- **wais** — система баз данных Интернет.

Домены Интернет

Все узлы Web классифицированы Международным центром сетевой информации (NIC) на шесть доменов:

- **com** – коммерческие предприятия, например, провайдеры Интернет
- **edu** — образовательные учреждения, колледжи и университеты
- **net** – действующие сети, например, Network Information Center
- **org** – непрофессиональные организации
- **mil** – военные сети
- **gov** – правительственные учреждения, например, white-house.gov

Кроме этого, все доменные имена имеют указатели на страну, в которой расположен данный узел. Например, доменные имена .uk, .jp и .us представляют соответственно Великобританию, Японию и США.

Часть 2. Основы создания гипертекстового документа

Глава 1. Введение

Вы собираетесь писать (или генерировать) гипертекст. Вы можете быть обескуражены тем, что гипертекст потенциально свободен (не связан жесткими рамками). Не пугайтесь. Это не доставит вам сложностей. Во многих случаях, чем проще, тем лучше.

Вы будете писать определенное количество отдельных файлов. Эти файлы будут соединены друг с другом и с внешними документами для завершения вашей работы.

Вы можете расценивать результат своей работы как «документ», ведь если бы он существовал на бумаге, вы назвали бы его документом. В случае online документов, мы можем относиться к каждому отдельному файлу как к документу. Документ может относиться, по аналогии с книгой, к разделу или подразделу или даже сноске.

Документ это минимальный модуль для представления информации. В любой момент времени ваш документ должен быть полностью загружен редактором. Также естественно если вы работаете одновременно над несколькими документами. Это возможно если у вас имеется хороший редактор, позволяющий одновременно открывать несколько документов.

Глава 2. Этикет в «паутине»

Имеется несколько соглашений, делающих web более практичным и удобным. Как администратор сервера, иными словами — webмастер, вы должны быть уверенными в применимости этих правил к вашим данным. Особенно обратите внимание на:

- Подпись вашей работы — особенно welcome page;
- Определение статуса документа.

Ваш администратор сервера нуждается в этих установках для каждого сервера:

Welcome page для гостей

Не существует никакой определенной структуры для данных, которые вы публикуете: вы можете создать такую, которая вам больше понравиться. Однако, наиболее лаконично иметь документ на каждый host (компьютер) который остальные могли бы использовать для получения быстрых сведений (с указателями) о том, какая информация здесь доступна. Вы должны внести строку «pass» в конфигурационный файл сервера для преобразования документа с именем «/» в конкретный документ. Так же как и обзоры доступной информации на сервере, указатели на связанные компьютеры являются хорошей идеей.

Welcome page — Home page?

Welcome page сервера часто называют home page, потому что это хороший выбор для клиента — использовать ее при старте как home (default [по умолчанию]) page. Термин home page означает стартовую точку по умолчанию для вашего браузера. Не смущайтесь этим. Это две отдельные концепции.

Welcome page будет приветствовать новых пользователей, которые хотят познакомиться с содержанием вашего сервера. Это послужит сходным целям для вашей home page, но разница существует для тех, кому она адресована. Часто люди внутри организации используют welcome page как свою home page. В конечном счете они получают представление о том, как их организация выглядит для других людей. Welcome page может содержать разъяснения того, что собой представляет ваш сервер, что может стать пустой тратой пространства на вашей home page для местных пользователей. Так что вы можете захотеть создать отдельную home page для местных пользователей.

Псевдоним для вашего сервера

Если у вас серьезный сервер, то он может быть больше, чем машина, на которой он работает. Попросите администратора вашей сети создать псевдоним для него, чтобы вы могли ссылаться на него, например, как «www.dom.edu», вместо «mysun12.dom.edu». Это значит, что, когда вы меняете машину, вы меняете псевдоним, а связь пользователей с вашими данными по прежнему работает.

В будущем клиенты будут искать локальную «www» машину для использования welcome page, как home page, если в конфигурации явно не указано другое. Это означает, что кто угодно запускающий клиента внутри вашего домена получит одинаковую начальную страницу.

Псевдоним для самого себя

Вам следует создать mail псевдоним «webmaster» на серверной машине, для того, чтобы люди имеющие проблемы с вашим сервером могли легко сообщит вам об этом, используя электронную почту. Это тождественно с псевдонимом «postmaster» для людей имеющих проблемы с вашей машиной.

Делегирование полномочий

Администратор сервера (единственный имеющий пароль root) в принципе имеет возможность включения/выключения сервера и контроля того, что происходило. Однако, мудро иметь четко делегированную ответственность для отдельных сфер вашей документации. Возможно, администратор сервера вообще не несет ответственности за актуальность данных, в этом случае их задача сводится к поддержанию машины в рабочем состоянии.

Домашний стиль

Web напоминает собой травяные корни, без централизованного подчинения, и при этом отлично работает. Это является обязательной частью творчества лиц предоставляющих информацию и той свободой, которой они располагают для выражения своей информации напрямую и настолько быстро, насколько они могут. Читатели очень признательны этой гибкости. Однако, в больших web системах логичность является приятной стороной.

Если вы отвечаете за управление процесса предоставления информации вашей организации, вы должны соблюдать баланс между преимуществами «домашнего стиля» и предоставлением каждой группе или автору свободы творчества. Если вы приняли решения в этой сфере, хорошо бы их записать (не упоминая записать их в web).

Глава 3. Структура

Если у вас есть представление о структуре информации, которую вы хотите донести до читателя, вы возможно имеете ее ментальную орга-

низацию. Обычно, это что-то типа иерархического дерева, глав книги, если вы когда-нибудь писали книгу.

Сохраните эту структуру. Это поможет читателям использовать структуру «дерево» как основу для книги: это даст им представление о том, где они находятся. Вы также можете использовать эту структуру для организации файлов в каталогах.

Вы также должны иметь в виду:

- Читатели заранее должны представлять себе структуру
- Идея перекрывающихся деревьев
- Насколько большим должен быть документ
- Ссылка или копия

Структура для читателя

Всегда считайтесь с аудиторией, для которой вы пишете. Если они новички в предмете, нормальной помощью им будет формирование четкой структуры вашей работы, чтобы они могли самостоятельно в ней разобраться. Например, если вы видите, что описываемый предмет распадается на три отдельные области, то это должно быть видно с первого взгляда.

Если, однако, ваши читатели уже имеют определенные знания предмета, то у них уже самостоятельно сформировалась эта структура. В этом случае они сознательно или подсознательно будут знать где можно найти необходимую информацию. Если ваша структура отличается от их, усиление (развитие) ее может вызвать непонимание и отвратить от вашей работы.

В этом случае вы будете вынуждены сопротивляться сильной тенденции некорректного использования вашей структуры и будете действовать в ущерб всем остальным. В этом случае есть два решения:

- Если вы ориентируетесь на фиксированную аудиторию, которая разделяет ваше видение мира, то старайтесь писать точно в струе этого видения, может быть даже в ущерб собственным взглядам.
- Если вы одновременно пишете для нескольких групп, тогда вы должны поддерживать оба подхода.

Когда вы делаете ссылку, обозначайте ее таким ключевым словом, чтобы некоторые люди могли пропустить ее.

Например, «***Если Вы действительно хотите узнать, как это работает, смотрите Internals guide***», или «***Введение step-by-step можно найти в обучающей части***».

Обеспечьте связи для обоих точек зрения читателей. Ваша работа будет внутренне более связанной, чем простое «дерево», но при тщательной проработке внутренних ссылок, никто не «потеряется» в вашей информации.

Обеспечьте два различных «корня» для вашего дерева. Например, вы можете написать step-by-step обучающую часть и функциональное ссылочное дерево для тех же данных. Оба дерева опираются на нижнем уровне на одни и те же данные, но в то время, как первое разбирается сначала с простейшими понятиями, второе может быть сгруппировано по функциональным признакам. Это все равно, что создать несколько указателей для одной и той же книги. Обучающая часть может также содержать информацию, которой нет в функциональном дереве.

Перекрывающиеся деревья

Новичок начинает обзор дерева сверху-слева и двигается вниз. Когда ему потребуются специфические детали, он будет брать примеры снизу (нижний уровень дерева) и из них ссылку на лежащее в основе определение для каждой детали. Как только он все усвоил, он переходит к чтению материала, имеющего структуру в виде дерева. Фактически, он читает такую же информацию, как и эксперт, который заинтересовался определенной функцией и затем обращается за примером ее использования.

Размер документа

Наиболее важным здесь является, чтобы каждый документ был завершенным с логической точки зрения. Так, не стоит резать одну идею на два произвольных кусочка только для того, чтобы сделать каждый из них меньше. Наоборот, не является хорошим стилем соединять не пересекающиеся между собой идеи, только для того, чтобы увеличить размер документа.

Документ может быть совсем малым по размеру. Например, примечание.

Существует два верхних предела для размера документа. Один из них, что длинный документ требует больше времени для передачи, поэтому читатель не может быстро входить в него и возвращаться назад, как ему бы хотелось. В наибольшей степени это зависит, конечно, от скорости передачи. Другим пределом является сложность для читателя пролистать длинный документ. Читатели, как правило, не в состоянии читать больше нескольких страниц. Они часто смотрят только информацию первой страницы и, если он их не заинтересовал, им очень докучает дальнейшее его пролистывание. Читатели также могут выйти из документа, находясь еще только в самом начале документа.

Читатели обычно пролистывают длинные документы используя scroll bar. Когда scroll bar немного перемещается, документ должен перемещаться на соответственное малое значение, так, чтобы часть оригинального окна осталась видимой. Это позволяет читателю сканировать документ. Если документ оказывается больше, тогда он становится нечитаемым и, при каждом передвижении scroll bar, читатель может потерять место на котором он остановился.

Аргументом в защиту длинных документов может служить легкость просмотра документа непрерывным потоком для читателей, имеющих scroll bars, так что они могут воспринимать его так, как он был написан.

Также они не приносят авторам неприятностей создания (или генерирования) большого количества связей и постоянного их обновления в случае изменения данных. Если создание ссылок является проблемой используйте в документе только одну ссылку — на страницу содержания. Некоторые просмотрщики имеют кнопки «next» и «previous», позволяющие просматривать документы по списку.

Приблизительным руководством по размеру документа может служить:

- Для online help, menu, дающих доступ к другим материалам лучше не превышать 24 линии. Проверьте это утверждение сидя за текстовым терминалом.
- Для текстовых документов — примерно 2.5 листа формата A4.

Ссылка или копия?

Когда вы создаете информационную системы, которая будет ссылаться на какие то другие материалы, будьте осторожны, прежде чем начать копирование.

Имеется несколько причин оставить информацию на прежнем месте:

- Когда она обновится, вы должны будете найти способ выяснения этого и создания свежей копии или вы останетесь с просроченной копией материалов.

- Если вы чувствуете, что ваша копия будет легче для доступа, помните, что она относительна. У вас появятся пользователи из других мест которые смогут найти закрытый оригинал. Если при обращении к оригиналу возникают проблемы, вы можете найти другой сервер (например свой) в качестве места хранения.

Причины для копирования:

- Если информация носит временный характер, например статья, есть смысл ее скопировать
- У вас может появиться желание ссылать на определенную версию чего-либо, что может быть изменено и не сохранено в архиве.

Вам следует быть очень осторожными прежде чем ссылаться на ваши личные коллекции, которые дублируют официальные коллекции:

- Internet FAQ (Frequently Asked Questions)
- RFCs (Request For Comments)
- Information by subject
- The «best» URLs on the web. Создайте личный список указателей ваших интересов.

Глава 4. Внутри документа

Вы должны попытаться:

- Подписывать свою работу
- Дать ей статус
- Делать ссылки на страницы контекста.
- Используйте независимый от контекста заголовок
- Формат, независимый от устройства вывода
- Пишите так, чтобы вашу работу можно было бы печатать
- Пишите читаемый текст несмотря на ссылки
- Старайтесь не говорить о технических деталях

Подписывайтесь!

Важным аспектом информации, который помогает сохранять ее актуальность, является то, что она позволяет установить своего автора. Это легко сделать с помощью гипертекста — все, что от вас требуется, это установить ссылку на страницу о авторе (или просто на телефонную книгу авторов).

Создайте страницу для себя, с вашим почтовым адресом и номером телефона. В конце файла, за который вы ответственны, сделайте маленькую заметочку — укажите только свои инициалы — и создайте ссылку на свою авторскую страницу. Адресный стиль (обычно текст, выровненный по правой границе) вполне подойдет для этого.

Ваша авторская страница является традиционным местом помещения отказа от прав, заметок об авторском праве и т.д., требуемые законом или традициями. Это спасает сами ваши заметки и послания от длинных подписей.

Статус вашей информации

Некоторая информация носит строго определенный характер, другая может быть поспешно объединенной и незавершенной. И та и другая может быть полезна читателю, так что не стесняйтесь публиковать незавершенную или устаревшую информацию — она может оказаться незаменимой. Однако, не забывайте устанавливать ее статус. Когда она последний раз обновлялась? Полная ли она? Что она охватывает? Для телефонной книги, например, что за люди присутствуют в ней?

Не обязательно добавлять это описание в каждый документ. Этого можно не делать, если, например, обзорная страница вашей работы уже содержит всю эту информацию.

Вы конечно можете дать оттенок статуса текста через его язык: синтаксические ошибки, пропущенные заглавные буквы, «расслабленная» грамматика, все это указывает на неофициальность стиля. Осторожно используйте глаголы «shall» и «should» (для английского текста), а инструкция Long Capitalised Noun Phrases (LCNPs) в конце концов создаст впечатление изучения ISO стандарта.

Датируйте

В некоторых случаях может быть полезно ставить дату создания и последнего изменения вашей работы. (Заметим, что это относится к такого рода вещам, которые могут быть выполнены сервером автоматически при минимальном программировании).

Выясняйте, когда установка даты может избавить читателя от использования просроченной информации.

Ссылка на контекст

Главным различием между написанием части последовательного текста и online документа является то, что читатель может попасть в него откуда угодно. Даже если вы только создали ссылки на документ только из одного места, любой другой человек может пожелать ссылаться на какую то определенную точку и таким образом сделать ссылку на некоторую часть вашей работы из своей собственной. Так что вы не можете положиться на читателя, что он последует вашим путем изучая вашу работу.

Конечно, если ваша информация носит обучающий характер, будет важно сохранять последовательность документов так, как это установлено вами для первичного ознакомления. Вы можете не пожелать угождать специально тем, кто произвольно прыгает по вашей работе, но следует при этом оставить им достаточное количество ключевых слов чтобы они окончательно не потерялись. Вот несколько путей, чтобы этого добиться:

- Следите чтобы ваш текст и словарь описывали сами себя. Начиная документ со слов «Следующая мысль рассматривает...» или «Единственным решением этой проблемы является...» конечно собьет с толку.
- Иногда введение нового слова или понятия относится к контексту и может быть ссылкой на общую информацию. Например: в документации проекта WWW, первое появление понятия WWW ссылается на центральный документ проекта.
- Советы по навигации вверху или внизу документа могут давать подробные указатели. Примером может служить текущий документ.

В момент написания вами документа было бы неплохо помнить о том, что вы когда-нибудь можете от него отказаться.

Навигационные рисунки (icons)

Icons выполняют роль навигационных советчиков. Очень эффективно иметь одинаково составленные картинки на протяжении всей работы, всегда (за исключением начальной страницы) ссылающихся назад на первую страницу.

Вы можете убить этим двух зайцев: это дает согласованность вашей работе, так что читатель всегда знает, когда он находится внутри нее а когда нет, также это дает ему быстрый путь возврата к началу работы.

Вы можете сделать то же самое и с секциями, так что на верху (или внизу) каждой страницы вы можете поместить небольшой ряд картинок, первая — переход на самый первый документ, вторая — к началу главы, третья — к началу раздела в главе.

Заголовок

Заголовок документа определяется элементом — TITLE. Элемент TITLE должен появляться в HEAD документа.

У любого документа может быть только один заголовок. Он должен идентифицировать содержание документа в довольно широком контексте.

Заголовок не является частью текста документа, но он относится ко всему документу. Он не может содержать ссылок, меток параграфов или подчеркиваний. Заголовок может использоваться для определения строк в history list, для заголовков окон, показывающих отдельные документы, и т.д. Он обычно не отображается в тексте собственно документа. Этим различаются titles и headings. Идеальная длина заголовка — менее 64 знаков. Таким образом, многие приложения будут показывать заголовки документов в окне для заголовков, в меню, и т.д. — там, где место ограничено. Не существует ограничений на длину заголовков (так как они могут быть автоматически сгенерированными из других данных), но информационные провайдеры должны помнить, что слишком длинные заголовки могут обрезаться.

Независимость от устройства

Гипертекст, который вы пишете запоминается в терминах языка HTML, которые не содержат информации о шрифтах, форматах параграфа и заполнителях (метод заполнения пустого пространства), которые должны быть использованы для изображения документа на экране.

Это дает большие преимущества для успешного переноса вашего документа на различные платформы, включая простые текстовые терминалы.

Вы должны учитывать, что разные клиенты используют различные заполнители и шрифты. Вы должны быть осторожны в использовании структурных элементов, таких как заголовки и списки в том виде, для которого они предназначены. Если вам не нравится то, как вашу информацию представляет определенный клиент, не пытайтесь исправить

положение использованием нестандартных элементов и не пытайтесь наполнить пространство пустыми элементами. Это может привести к различной интерпретации различными клиентами и выглядеть очень странно. Во многих случаях вы можете сконфигурировать как клиент выводит каждый элемент

Например:

- Всегда используйте уровни заголовков в следующем порядке, один заголовок 1-го уровня в начале документа, если необходимо, то несколько заголовков 2-го уровня, и далее, если необходимо, несколько заголовков 3-го уровня внутри каждого заголовка 2-го уровня. Если вам не нравится то, как форматируется заголовок 2-го уровня, настройте это в своем клиенте, но не перепрыгивайте на следующий уровень заголовков.
- Не делайте дополнительных пробелов или пустых линий в вашем тексте только для «раздувания» документа, исключая преформатированную часть (тэг PRE).
- Не ссылайтесь в вашем тексте на особенности определенного просмотрщика. Просить кого-то «нажмите здесь» не имеет смысл без мышки, также как и просьба кого-то «выберите ссылку по номеру» опровергается фактом того, что вы используете не line-mode просмотрщик. Оставьте только ссылку. Инструкции излишни, если пользователь сам знает как выбирать ссылки.

Следуя этим указаниям вы можете обнаружить, что конечный результат появляется на вашем экране не в том виде, в котором вы бы хотели, но возможно вашим читателям повезет больше.

Печать гипертекста

В идеальном мире бумага перестанет быть необходимой. Из этого следует, что будет достаточно времени чтобы написать гипертекстовую версию документа и также сделать отдельную версию на бумаге. Однако, в условиях реального мира вы возможно захотите генерировать любые печатные и online документы из одного файла.

Предполагая, что HTML файлы будут эталонными, вы будете генерировать из них версию для печати, создавая один длинный документ и возможно печатая его оттранслировав в формат некоторого текстового процессора, например, TeX. Возможно не сейчас, но когда-нибудь, такое желание у вас возникнет.

Старайтесь избегать ссылок в тексте на специфические моменты online гипертекста. «***Смотрите раздел независимость от устройства***» лучше, чем «***За подробностями по идеям о независимости от устройств, нажмите здесь***.». Фактически, мы говорим о форме подачи *независимости от устройств*.

К сожалению, рекомендованная практика подписи каждого документа и построения навигационных ссылок имеет тенденцию портить печатные копии, таким образом вы можете развивать способы удаления подобных вещей если ваши документы следуют общему формату.

Читаемый гипертекст

Первое — это синдром _HERE_, т.е.:

```
Информацию о Blah Blah Blah можно получить нажав here
```

где слово **here** ссылка. Этот стиль неуклюж, когда вы нажимаете **here**, вы должны «посмотреть по сторонам», чтобы убедиться, что нажимаете правильно.

Совет: когда вы создаете свою HTML страницу, убедитесь что то, на что нужно нажать фактически является чем то типа заголовка для того материала, на который вы перейдете, нажав. Т.е., говорите

```
Информация о Blah Blah Blah теперь доступна.
```

И используйте:

```
Информация о способах поиска доступна.
```

Вместо

```
Если хотите получить информацию о способах поиска, выберите
эту ссылку.
```

Не так уж и плохо, но все же неудобно если кто-то будет использовать словарное слово, как ссылку, одновременно говоря о «ссылке».

```
Здесь ссылки на страницы кредитов и технических деталей
```

вместо этого попытайтесь написать что-то типа:

```
Огромное спасибо различным людям за их содействие,
технические детали этой системы теперь доступны всем.
```

Т.е., делайте ваши HTML страницу такой, чтобы вы могли читать ее даже если вы не проходите по всем ссылкам.

Избегайте технических деталей

Сообщения об Интернет службах обычно следуют на страницах для информации о том, как использовать FTP, почтовых службы и т.д. для получения информации. WWW было создано для того, чтобы избежать необходимости использования этого. Соблазном является вырезать все эти инструкции и оставить ссылку типа:

```
Это WWW доступ к нашему большому FTP архиву, который ранее был
доступен только через FTP, NFS и mail. Эта коллекция включает
множество программных средств и текстов, авторские права на
которые уже закончились.
```

Web обычно читают люди, которые либо не нуждаются, либо, что бывает чаще, не хотят знать о FTP, NFS и даже WWW! Лучше написать следующее:

```
Наш архив включает множество программных средств и текстов, ав-
торские права на которые уже закончились.
```

Описывайте предмет рассуждений чаще, чем механизмы и протоколы, при этом не забывайте о краткости текста, которая способна привлечь читателей.

Даже если вы работаете с метафорами web, используйте ссылки, но не говорите о них. Например:

```
Вы можете найти больше информации в части, посвященной обучению,
которая имеет ссылку на home-page.
```

Очевидно будет лучше написать:

```
Обучающая часть содержит больше информации об этом.
```

Еще одна общая идея

```
Обучающая часть содержит разделы Урок 1, Урок 2 и Урок 3.
```

Дайте читателю передохнуть и позвольте ему обратиться непосредственно к тому разделу, который ему необходим!

Глава 5. Тестируйте свой документ

В каком-то смысле ваш гипертекст похож на книгу, которую вы должны откорректировать. Также как и на программу, которую необходимо проверить.

По крайней мере дайте его прочитать кому-нибудь из той группы людей, для которых был написан ваш документ, в целях обратной связи. Другой вариант:

- Прочитайте документ несколькими различными просмотрщиками, чтобы убедиться, что вы его создали независимым от устройства вывода.
- Курируйте круг ваших читателей. Вы можете это делать путем анализа log-file вашего сервера. Вы можете обнаружить, что некоторые части не читаются, возможно потому, что люди ищут их не в том месте. Вы можете увидеть, что люди часто заходят в какой-то материал и сразу же из него выходят. Если вам понятно, что они искали, вы можете сделать ключевые слова ссылки более информативными. (Не забывайте сохранять информацию из log-file конфиденциальной до тех пор, пока вы не удалили из него информацию о пользователях).
- Объясняйте, намерены ли вы принимать критику или предложения читателей и каким способом они могут вам ее послать.
- Просите людей решать проблемы используя документ, и, в случае успеха, информировать об этом. Если они потерпели неудачу, выясните, что они искали и относится ли это вообще к документу.

Затраты времени на тест

Проверка занимает время. Решение, сколько времени посвящать тестированию, основано на том качестве документа, которое вы хотите достигнуть. Вы балансируете между временем читателей и своими усилиями. Если ваш документ «продает» идею или, если вы продаете документ, или обеспечиваете сервис, вы захотите сделать это для читателя как можно проще. Если много людей прочитают вашу работу, небольшое время, затраченное ваши, сэкономит читателям много времени.

Если, однако, вы описываете в документе какую-то непонятную часть системы, которой кроме вас больше никто не интересуется, или, в том случае, если вы чувствуете, что читатели рады иметь хоть какую-то информацию о данном предмете, нет смысла на проверку этой части. Если кто-то действительно нуждается в этой информации, он может смириться с некоторыми недочетами и просмотреть все ваши ссылки чтобы понять, что вы хотели написать. Это может быть наиболее эффективным.

Очень много информации, которая предназначается для быстрого ознакомления или создания файлов на ее основе, окажется важной для будущих читателей. Лучше, чтобы эта информация была доступна хотя бы в незаконченной форме, чем если она совсем не будет доступна. До

появления электронных технологий, усилия на публикацию сырого материала приводили к тому, что эту информацию мало кто видел, или к браку; и могли быть расценены как оскорбление читателя, так как публиковалось то, что было низкого качества.

В настоящее время публикация идет на всех уровнях и, как документы высокого качества, так и недоработанные, имеют свою ценность. Поэтому, делая ссылки, указывайте на текущий уровень документа дабы избежать разочарования читателей.

Просмотр log-file сервера даст информацию о том, какие документы вызывают наибольший интерес. Вы можете эффективнее использовать свое время, посвящая его улучшению качества именно этих документов. Конечно, анализ log-file сервера тоже займет некоторое время!

Проверка вашего HTML

Если вы используете программное обеспечение для редактирования гипертекста, ваши файлы должны соответствовать действующему стандарту HTML. На текущий момент, многие люди редактируют HTML файлы как и обычные текстовые и должны самостоятельно контролировать соответствие написанного правилам HTML. Если вы относитесь к этой категории, то вам будет не лишним пропускать ваши файлы через HTML-checker. Также неплохой идеей является использование HTML средств нового поколения для единичной проверки выхода HTML-checker.

Глава 6. Таблица или фреймы

Если вы хотите сделать сайт, то первой проблемой, которая должна появиться, будет выбор использования нескольких фреймов, или взять таблицу, как основу сайта. Самое простое не ошибиться в выборе — рассмотреть все плюсы и минусы, и решить для себя — фреймы или таблица. Итак, первое — фреймы:

- Меню, логотип — отдельные файлы, что очень удобно (при любом изменении, не надо менять в каждом файле одни и те же строчки)
- Одни и те же элементы (меню, логотип) загрузятся только один раз и не будут каждый раз перезагружаться
- Меню или логотип (то, что в отдельных фреймах) всегда будут видны пользователю

- «Нераспознавание» фреймов поисковыми системами
- «Незнание» фреймов, как инструмента HTML старыми браузерами
- Неправильная работа новых браузеров с фреймами (вплоть до «зависания» системы)

Второе — использование таблиц:

- Правильная работа всех браузеров с таблицами
- Корректная обработка поисковыми системами страниц с таблицами
- Загрузка сразу всей таблицы, что ограничивает размер файла до 20-30 Kb
- Использование одинаковых элементов (например, меню) в каждом файле

Выбор структуры

На самом деле, неважно, что вы взяли за основу — таблицу или фреймы. Простые таблицы имитируют все нормальные (общепринятые) варианты структуры сайта: каждый цвет, каждая ячейка — структурная единица (меню, заголовок, рекламное место под банеры и т.д.). Если у вас за основу взята таблица — то каждая ячейка будет соответствовать ячейке на нашей таблице. Если вы делаете через фреймы — то каждая ячейка будет являться отдельным фреймом.

Глава 7. Перекодировщики кириллицы

лПДЙТПЧЛБ — ЬФП ЧБЦОП. В смысле, кодировка — это важно. Именно таким образом будет выглядеть данная фраза, будучи записанной в кодировке KOI8-R. Очевидно, что если разработчик какой-либо web-страницы ненароком спутает кодировку или просто установит ее значение неправильно, посетители будут долго любоваться на подобный бессмысленный набор символов, но прочитать его, увы, не смогут.

Как вы уже знаете, некоторые http-серверы умеют автоматически изменять кодировку web-страниц в зависимости от настроек пользовательского браузера, однако на самом сервере документы хранятся в какой-либо одной кодировке. Если конкретной информации о том, в какой кодировке следует загружать страницы на сервер, администрация

узла не предоставляет, в девяноста процентах случаев из ста по умолчанию используется кодовая страница windows-1251, а в остальных это, скорее всего, KOI8-R. Известно, что программное обеспечение, работающее под управлением операционной системы Microsoft Windows, в том числе текстовый редактор Notepad, сохраняет файлы на диск в кодировке windows-1251. Как быть, если согласно требованиям сайта, предоставляющего вам web-хостинг, необходимо размещать данные в другой кодировке? Вот для этого и существуют многочисленные программы- перекодировщики кириллицы.

ConvHTML

Программа ConvHTML, автоматически перекодирует текстовые файлы и документы HTML из Windows 1251 в KOI8-Ru обратно. Интерфейс ее предельно прост.

Рабочее окно ConvHTML разделено на две вертикальные части: в левой указывается список документов, подлежащих перекодированию, а в правой отображается результат работы программы. Панель инструментов в интерфейсе ConvHTML отсутствует, вместо нее имеется селектор, с помощью которого можно указать, в какую кодировку вы намерены транслировать файл. Конвертация web-страниц с использованием программы ConvHTML осуществляется достаточно быстро: в меню Файл следует выбрать пункт Открыть и указать нужный документ, после чего программа спросит вас, желаете ли вы перекодировать только этот документ или все файлы, содержащиеся в данной директории. Для того чтобы выполнить конвертацию документа, необходимо выбрать пункт Перекодировать и сохранить в том же меню. Программа сохранит отконвертированный файл под старым именем в исходной папке.

К достоинствам утилиты ConvHTML можно отнести простой и доступный интерфейс, возможность работы с файловыми списками, возможность преобразовывать из кодировки в кодировку как html-документы с расширениями .htm и .html, так и текстовые файлы.

Главным недостатком является то, что данная программа умеет обращаться только с двумя кодовыми страницами, которых на практике используется значительно больше, а также то, что она сохраняет файл с использованием его первоначального имени, замещая при этом старый, и не дает возможности сохранить перекодированный документ в другой директории.

Совет: Преобразуя составляющие ваш сайт html-документы в другую кодировку, сделайте копию папки, в которой они хранятся. Если процесс конвертации прошел с ошибкой или впоследствии у вас возникла необходимость внести изменения в уже перекодированный документ,

вы можете использовать исходную копию, заместив ею старый файл и перекодировав донную web-страницу заново.

Программа ConvHTML доступна для свободного копирования на многочисленных сайтах Интернета, предлагающих посетителям бесплатное программное обеспечение.

SNK DEcode

Программа-перекодировщик SNK DEcode была создана специалистами российской компании «Тихая Гавань М», она доступна для бесплатной загрузки на сайте http://www.book.ru/snk. В отличие от утилиты ConvHTML SNKDEcode является полнофункциональным приложением, предназначенным для трансляции кириллицы из одной кодировки в другую. Эта программа поддерживает следующие направления конвертации файлов: KOI8-R ➪ Windows-1251 и обратно, Alt DOS ➪ Windows-1251 и обратно, KOI8-R ➪ Windows-1251, ISO ➪ Windows-1251, MAC ➪ Windows-1251 и, наконец, Unicode ➪ Windows-1251. Возможна также конвертация файлов по заданной пользователем схеме, с преобразованием какого-либо файла из одной кодировки в другую в произвольном порядке.

SNK DEcode поддерживает конвертацию текстовых файлов и документов HTML, файлов RTF, а также текстовых файлов с произвольным расширением. Главная панель содержит стандартные кнопки очистки рабочего пространства программы, открытия файла, копирования и вставки фрагментов кода из буфера обмена, сохранения текущего документа в произвольную директорию, вывода обрабатываемого кода на принтер, а также несколько других весьма полезных кнопок. Функция Перезагрузить в начальном виде позволяет восстановить состояние окон после внесения в документы каких-либо изменений, функция Запомнить — сделать «моментальный снимок» рабочего пространства программы, которое восстанавливается при нажатии кнопки **Перезагрузить** в начальном виде, и, наконец, функция **Поменять местами** меняет местами рабочие окна. Перекодировка осуществляется нажатием соответствующей кнопки на панели инструментов. При выборе пункта **По схеме** программа выводит на экран диалоговое окно, позволяющее пользователю задать произвольный алгоритм трансляции файла.

Пользовательская схема перекодировки формируется следующим образом: в меню, появляющемся при нажатии на расположенную в левом верхнем углу окна кнопку, следует выбрать исходную кодировку, а в меню, появляющемся при нажатии на кнопку, расположенную правее, — кодировку, в которую осуществляется трансляция. Включение пары кодировок в список осуществляется нажатием кнопки **Добавить**, отправ-

ка созданного списка на исполнение — нажатием кнопки **Выполнить**. Сформированный таким образом список кодовых пар отображается в левом поле диалогового окна настройки пользовательской схемы трансляции, правое окно содержит перечень сохраненных на диске схем. Управлять этим перечнем можно с использованием кнопок **Сохранить** и **Удалить**. При нажатии на первую из них текущая схема записывается на диск, а при помощи второй можно уничтожить неиспользуемую схему. Кнопка **Очистить** удаляет все компоненты списка кодовых пар и, наконец, кнопка **Закрыть** закрывает данное диалоговое окно.

Рабочее пространство программы SNK DEcode разделено на два вертикальных поля аналогично программе ConvHTML: в одном из них отображается исходный текст преобразуемого файла, в другом — его отконвертированная версия. Помимо основных программных свойств SNK DEcode имеет еще целый ряд полезных дополнительных функций, вызываемых с применением меню **Сервис**. Для их использования необходимо выделить при помощи мыши в окне исходного текста какую-либо символьную последовательность и выбрать соответствующий пункт указанного меню. Функции **Верхний регистр** и **Нижний регистр** преобразуют символы выделенной фразы соответственно в заглавные или строчные, функции **Lat — Cyr** и **Cyr — Lat** изменяют латинские символы на символы кириллицы и обратно в соответствии с их расположением на клавиатуре компьютера. Команда **Убрать пробелы** удаляет все лишние пробелы из выделенного участка преобразуемого документа, сокращая итоговый размер файла, и, наконец, функция **Убрать тэги** удаляет из исходного файла все элементы разметки HTML, оставляя в нем лишь содержательный текст.

Не лишена эта программа и существенных недостатков. Главное упущение разработчиков SNK DEcode заключается в том, что данное приложение умеет открывать файлы с расширением .htm, но, увы, не распознает документы с расширением .html, что вызывает ряд ощутимых неудобств у пользователей.

Часть 3. Описание языка гипертекстовых документов

Глава 1. Гипертекстовый язык

В настоящее время существует масса редакторов Web-страниц, которые не требуют от вас знаний основ HTML. Но для того чтобы уметь профессионально подготавливать гипертекстовые документы, вы должны знать их внутреннее строение, то есть код документа HTML.

HTML позволяет вам формировать различную гипертекстовую информацию на основе структурированных документов.

Обозреватель определяет сформированные ссылки и, через протокол передачи гипертекста HTTP, открывает доступ к вашему документу другим пользователям Интернет. Разумеется, для успешной реализации всего этого необходим софт, полностью совместимый с WWW и поддерживающий HTML.

HTML-документ — это обычный текстовой файл. Используя, например, обозреватель Netscape Navigator вы можете просмотреть результат вашей работы, просто загрузив в него созданный на основе синтаксиса HTML текстовой файл.

Гипертекстовый язык предоставляет только информацию для чтения. Это означает, что редактировать Web-страницы может лишь тот, кто их создал, а не простой пользователь Сети сетей. Впрочем, если забежать немного вперед, можно сказать, что используя общий шлюзовой интерфейс (CGI) можно добавлять некоторые операторы HTML в сгенерированную страницу.

Глава 2.
Термины гипертекста

Самый смак гипертекстового языка – это ссылки. В мировой паутине вы просто нажимаете на ссылку и мгновенно оказываетесь в другой точке земного шара на выбранной вами страничке.

По традиции всех классических руководств по HTML мы приводим самый простой гипертекстовый документ.

```
<TITLE>Пример простого HTML документа</TITLE>
<H1>Здесь размещен заголовок первого уровня</H1>
Добро пожаловать в Internet!
Первый и последний параграф.<P>
```

В этом примере мы использовали следующие термины гипертекста (так называемые тэги):

<TITLE> — тэг, использующийся для определения заголовка.

<H1> — тэг заголовка.

<P> — тэг метки параграфа.

В языке описания гипертекстовых документов все тэги парные. В конечном тэге присутствует слэш, который сообщает обозревателю о завершении. Но! Существует одно исключение из этого правила пар:

В природе не существует тэга **</P>**.

Не все тэги совместимы с обозревателями. Если обозреватель не понимает тэг, то он его просто пропускает.

Итак, документ HTML это заголовок:

```
<html>
<head> Заголовок </head>
<body>
...
```

и текст

```
...
</body>
```

с названием:

```
<head>
<title> Название </title>
</head>
```


Название документа

Это не правило, и даже не закон, это факт:

Любой документ HTML имеет название.

По названию вашего документа HTML другие обозреватели могут найти информацию. Место для названия всегда определено – оно находится вверху экрана, и отдельно от содержимого документа. Максимальная длина названия – 40 символов.

Форматирование

Форматирование может быть непосредственным или авторским. Если вы используете тэг **<pre>**, то форматирование считается авторским:

```
<body>
<pre>
```

Следующие тэги присущи непосредственному форматированию:

<p> — параграф.

<hr> — горизонтальная линия.

**
** — обрыв строки.

Заголовки и подзаголовки

Язык HTML позволяет вам работать с шестью уровнями заголовков. Первый заголовок – самый главный. На него обращается особое внимание. Остальные заголовки могут быть оформлены, например, жирным шрифтом или прописными буквами.

В HTML первый заголовок обозначается как **<H1>**:

```
<Hn>Текст</Hn>
```

Под **n** понимается уровень заголовка, то есть числа 1, 2, 3, 4, 5 или 6.

В HTML первый заголовок может совпадать с названием документа.

Списки

Списки подразделяются на:

Ненумерованные

```
<ul>
<li> Элемент списка
</ul>
```

Нумерованные

```
<ol>
<li> Элемент списка
</ol>
```

С описаниями

```
<DL>
<DT> Собака (элемент)
<DD> Друг человека (описание элемента)
</DL>
```

Вложенные

```
<UL>
<LI> Примус
<OL>
<LI> Другой примус
...
</OL>
<LI>
...
</UL>
```

Выделение текста

Текст в документе HTML может быть выделен одним из следующих способов:

<cite> — цитата **</cite>**

<code> — программный код **</code>**

<dfn> — определение **</dfn>**

**** — логический акцент ****

<kbd> — ввод с клавиатуры **</kbd>**

<samp> — сообщения компьютера **</samp>**

**** — сильный акцент ****

<var> — переменные **</var>**

Один большой параграф

В HTML разбиение на строки *не принципиально*. Это означает, что вы можете разбить строки вашего документа в любом его месте. Связано это с тем, что в гипертекстовом документе идущие подряд отбивки превращаются в одну. Но! Если отбивка сделана после тэга **<P>**, то она учитывается. Если какой-нибудь тэг **<H>** игнорируется, то отбивка также

учитывается. В остальных случаях обозреватель будет пропускать отбивки.

Ссылки

HTML позволяет вам связать текст или картинку с другими гипертекстовыми документами. Текст, как правило, выделяется цветом или оформляется подчеркиванием.

Для этого используется тэг **<A>**. Помните, что после буквы A должен стоять пробел.

Чтобы сформировать ссылку:

- наберите **<A**
- введите **HREF="filename">**
- наберите после > текст гипертекстовой ссылки
- наберите тэг **</A>**

Один из вариантов гипертекстовой ссылки может выглядеть так:

```
<A HREF="BobAnapa.html">Bob</A>
```

Здесь слово **Bob** ссылается на документ **BobAnapa.html**, образуя гипертекстовую ссылку.

Если документ, формирующий ссылку, находится в другой директории, то подобная ссылка называется относительной:

```
<A HREF="BobAnapa/BobMoscow.html">Bob</A>
```

Если вы хотите указать полное имя файла, то вам необходимо использовать синтаксис UNIX.

Ссылки можно формировать на основе так называемого универсального локатора ресурса, то есть используя следующий синтаксис:

```
protocol: //hostport/path
```

Предварительное форматирование текста

Тэг **<PRE>** позволяет сформировать текст, оформленный моноширинным шрифтом.

Используйте этот тэг для оформления листингов программ.

Расширенные цитаты

Тэг **<BLOCKQUOTE>** позволяет вам включить цитату в уединенный объект.

Адрес

Тэг **<ADDRESS>** позволяет сформировать информацию об авторе документа HTML.

Принудительный перевод строки

Тэг **
** переводит только одну строку, то есть без дополнительного пробела.

Горизонтальные разделители

Тэг **<HR>** формирует горизонтальную линию по всей ширине окна.

Встроенные изображения

Вы можете встраивать в ваш документ картинки. Синтаксис встроенной картинки следующий:

```
<IMG SRC=image_URL>
```

Здесь **image_URL** есть указатель на файл картинки, синтаксис которого совпадает с синтаксисом ссылки HTML.

Глава 3. Использование звуков

Для того, чтобы вставить в вашу страничку звуковой файл, например, midi-файл, используйте следующую конструкцию:

```
<EMBED SRC="bob1.mid" WIDTH="140" HEIGHT="50" ALIGN="MIDDLE" BORDER="0" AUTOSTART=TRUE>
```

Это одна строка.

В этом тэге были использованы следующие параметры:

WIDTH

Параметр, определяющий ширину midi-плейера.

HEIGHT

Параметр, определяющий высоту midi-плейера.

BORDER

Ширина рамки midi-плейера.

AUTOSTART

Запустить midi-плейер сразу после того, как загрузится документ HTML.

Глава 4. Создание графического меню

Используя технологию распределения ссылок по картинке, вы можете, например, создать графическое меню из одной большой картинки таким образом, чтобы каждый элемент системы меню содержал определенный URL.

Распределение ссылок по картинке описывается в тэге IMG следующим параметром:

```
<IMG SRC="url" USEMAP="url#map_name">
```

Здесь аргумент **USEMAP** задает расположение схемы распределения **map_name в URL**.

Если URL не указан, то поиск схемы **map_name** ведется в текущем документе.

Код схемы может выглядеть так:

```
<MAP NAME="map_name">
<AREA [SHAPE=" shape "] COORDS="x,y,..." [HREF=" reference "]
[NOHREF]>
</MAP>
```

Здесь были использованы следующие тэги:

<AREA>

Определить для данного URL область на картинке посредством параметров SHAPE и COORDS.

SHAPE

Форма области. Вы можете выделить область на картинке так:

- **default** — стандартная форма
- **rect** — прямоугольник
- **circle** — круг
- **poly** — многоугольник произвольной формы

COORDS

Координаты области. Задаются в пикселах. Отсчет начинается с нуля. Круг имеет три координаты, прямоугольник — четыре, а для многоугольника вы должны описать каждый его угол в двух координатах. Например, область, имеющая размеры 50 на 50 пикселов, описывается так:

```
<AREA COORDS="0,0,54,54" ...>
HREF="url"
```

Определить ссылку на схеме, то есть вписать URL.

NOHREF

Указать, что в данной области картинки отсутствует ссылка. Этот параметр работает всегда, когда не определен параметр HREF.

</MAP>

Закончить описание схемы распределения ссылок по картинке.

Глава 5. Текстовые стили

В HTML слова и строки кодируются *логическими* и *физическими* стилями.

Физические стили форматируют текст.

Логические стили форматируют через определение в гипертекстовом документе некоторого значения. Это в частности означает, что тэг заголовка первого уровня не содержит информации о размере шрифта и гарнитуре. Поэтому, чтобы изменить символьное форматирование заголовка вы должны модифицировать заголовок первого уровня. Через логические (в том числе и символьные) тэги вы можете сформировать согласованный гипертекстовый документ, то есть определить заголовок первого уровня в качестве только **<H1>** (без информации о гарнитуре шрифта и его кегле).

Логические стили

Ниже мы представляем примеры логических стилей документа HTML.

<DFN>

Определить слово. Как правило, курсив.

<EM>

Усилить акцент. Как правило, курсив.

<CITE>

Заголовок чего-то большого и хорошего. Курсив.

<CODE>

Компьютерный код. Моноширинный шрифт.

<KBD>

Текст, введенный с клавиатуры. Моноширинный жирный шрифт.

<SAMP>

Сообщение программы. Моноширинный шрифт.

<STRONG>

Ну очень важные участки. Жирный шрифт.

<VAR>

Замена переменной на число. Курсив.

Физические стили

Гипертекстовый документ может быть оформлен с использованием следующих стилей:

<B>

Полужирный

<I>

Курсив

<TT>

Моноширинный

Специальные символы

Символы, которые не могут быть введены в текст документа непосредственно через клавиатуру называются специальными символами. Для таких символов существуют особые тэги.

Четыре символа — знак меньше <, знак больше >, амперсанд **&** и двойные кавычки **"** имеют специальное значение внутри HTML и следовательно не могут быть использованы в тексте в своем обычном значении.

Скобки используются для обозначения начала и конца HTML тэгов, а амперсанд используется для обозначения так называемой escape-последовательности. Для использования одного из этих символов введите одну из следующих escape-последовательностей:

&lt

Знак меньше.

&gt

Знак больше.

&amp

Амперсанд.

&quot

Кавычки.

Глава 6. Общий интерфейс и формы языка

Общий шлюзовой интерфейс (Common Gateway Interface) позволяет работать с данными сервера Web в интерактивном режиме. Сервер Web через CGI запускает поисковую программу и пересылает обработанные данные назад. Сама программа CGI хранится в каталоге CGI-BIN. Это означает, что файл из каталога CGI-BIN всегда исполняемый файл. Если CGI- программа, например, взаимодействует с системой управления базой данных, то пользователь может получать некоторую интересующую его информацию в интерактивном режиме.

Это тривиально, но факт: CGI-программы создаются посредством CGI. Код программы пишется, как правило, на языке описания сценариев Perl. Perl является интерпретируемым языком, интерпретатор которого соответствует операционной системе.

Передача данных от сервера к программе CGI осуществляется сервером через командную строку и переменные окружения.

Таким образом, сервер через общий шлюзовой интерфейс запускает программу CGI и пересылает ей вводимые пользователем данные. Сами данные вводятся через так называемые формы HTML.

Форма представляет собой гипертекстовую страницу с одним или несколькими полями данных и специальной кнопкой для передачи введенной информации.

Как и код любого гипертекстового документа, код формы начинается с тэга **FORM ACTION = /SGI-BIN/EXAMPLE.PL** и заканчивается тэгом **/FORM**.

ACTION

Аргумент ACTION — это URL программы CGI, то есть /SGI-BIN/EXAMPLE.PL.

METHOD

Метод, используемый для запроса данных.

Этот параметр задает режим передачи данных из формы в программу CGI.

Основные режимы передачи — **GET**, **HEAD** и **POST**. Программа CGI должна поддерживать один из этих режимов, иначе обработки данных не произойдет.

В режиме **GET** данные входят через **URL** в строку запроса. Например, если программа обработки данных **BOB.PL** лежит в каталоге **CGI-BIN**, то запрос HTML пойдет на сервер через **ACTION** следующим образом:

```
FORM ACTION=/CGI-BIN/BOB.PL METHOD=GET
```

Теперь сервер знает, где находится программа **BOB.PL**, поэтому он ее запустит в режиме **GET**.

Программы CGI получают данные от переменных окружения и посылают выходные данные через общий шлюзовой интерфейс обратно пользователю. Например, после ввода тэга **A HREF=BOB.HTML** на сервер пойдет запрос **GET /BOB.HTML**. Заголовок **GET** определяет получение документа **BOB.HTML** в корневом каталоге сервера.

Относительно аргумента POST

CGI реализован в программах, поддерживающих Unix и некоторые приложения Windows. CGI для Windows реализуется лишь в том случае, если сервер Web способен декодировать данные тех форм HTML, которые пересылаются в режиме POST. Это можно сделать двумя способами:

URL-Encoded. Данные формы пересылаются на сервер в виде HTML.

Multipart Form Data. Данные формы пересылаются на сервер в виде MIME-сообщения.

INPUT TYPE=TEXT

Поместить в форму текстовое поле данных.

NAME=NAME

Определить имя текстового поля данных NAME.

MXLENGTH=NUMBER

Размер текстового поля данных. Вместо NUMBER вы можете ввести целое число.

RADIO

Определить кнопку переключения.

NAME=PRODTYPE

Определить логическое поле PRODTYPE.

INPUT TYPE=CHECKBOX

Определить флажок для протокола передачи.

ACCEPT

Метод, используемый для интерпретации пересылаемых файлов. Файлы могут пересылаться в виде ASCII или HTML.

Количество заголовков **ACCEPT** соответствует типам данных MIME (Multipurpose Internet Mail Extensions). Заголовок **ACCEPT: TYPE/SUB-TYPE {parameters}** пересылается как значение параметра **ACCEPT**. Каждый тип данных имеет собственный параметр **ACCEPT**.

Глава 7. Фреймы

Технология фреймирования в HTML позволяет просматривать в одном окне обозревателя несколько гипертекстовых документов. Один фрейм отображает только один гипертекстовый документ.

Создание фрейма

Создание фрейма осуществляется через тэг **<FRAMESET>**. Тэг **<BODY>** в этом случае не используется.

Тэг **</FRAMESET>** заканчивает выполнение кода после тэга **<FRAMESET>**.

Далее идет стандартное применение HTML.

Описание фрейма

Тэг **<FRAME SRC="Name1">** позволяет описать первый фрейм, т.е. присвоить имя гипертекстовому документу. Второй фрейм описывается тэгом **<FRAME SRC="Name2" NAME="Main">**.

Если обозреватель не поддерживает фреймы

Если обозреватель того или иного пользователя не поддерживает фреймы, то между тэгом **<NOFRAMES>** и тэгом **</NOFRAMES>** заносится текст, который распознает обозреватель.

Свойства фреймов

Тэг **<FRAMESET COL="N1, N2,..>** позволяет определить количество фреймов и задать размер фреймов в процентах от размера окна обозревателя или зафиксировать эти размеры в пикселах.

Тэг с большими возможностями

Тэг **<FRAME>**, имеющий самое большое количество атрибутов, позволяет настроить свойства фрейма. Ниже описываются эти атрибуты.

NAME=

Имя фрейма.

MARGINWIDTH=

Горизонтальный отступ (от 1 до 6) между фреймом и его границей.

MARGINHEIGHT=

Вертикальный отступ (от 1 до 6) между фреймом и его границей.

SCROLLING=

Прокрутка фрейма. Податрибут **SCROLLING=YES** позволяет создать полосы прокрутки, **SCROLLING=NO** — указывает обозревателя, что полосы прокрутки отсутствуют в данном фрейме, а податрибут **SCROLLING=AUTO** позволяет отображать полосы прокрутки в зависимости от свойств обозревателя.

NORESIZE

Фиксированный размер фрейма.

SRC=

Задать гипертекстовый документ для этого фрейма.

TARGET=Name

Открыть ссылку во фрейме с именем Name.

Используя фреймы, позволяющие разбивать Web-страницы на скроллируемые подокна, вы можете значительно улучшить внешний вид и функциональность информационных систем и Web-приложений. Каждое подокно, или фрейм, может иметь ряд свойств.

Каждый фрейм имеет свой URL, что позволяет загружать его независимо от других фреймов. Каждый фрейм имеет собственное имя (параметр **NAME**), позволяющее переходить к нему из другого фрейма. Размер фрейма может быть изменен пользователем прямо на экране при помощи мыши (если это не запрещено указанием специального параметра). Данные свойства фреймов позволяют создавать продвинутые интерфейсные решения, такие как:

- Размещение статической информации, которую автор считает необходимым постоянно показывать пользователю, в одном статическом фрейме. Это может быть графический логотип фирмы, copyright, набор управляющих кнопок.
- Помещение в статическом фрейме оглавления всех или части WEB-документов, содержащихся на WEB-сервере, что позволяет пользователю быстро находить интересующую его информацию.
- Создавать окна результатов запросов, когда в одном фрейме находится собственно запрос, а в другом результаты запроса.
- Создавать формы типа «мастер-деталь» для WEB-приложений, обслуживающих базы данных.

Синтаксис фреймов

Формат документа, использующего фреймы, внешне очень напоминает формат обычного документа, только вместо тэга **BODY** используется контейнер **FRAMESET**, содержащий описание внутренних HTML-документов, содержащий собственно информацию, размещаемую во фреймах.

```
<HTML>
<HEAD>...</HEAD>
<FRAMESET>...</FRAMESET>
</HTML>
```


Однако, фрейм-документ является специфичным видом HTML-документа, поскольку не содержит элемента **BODY** и какой-либо информационной нагрузки соответственно. Он описывает только фреймы, которые будут содержать информацию (кроме случая двойного документа).

Представим общий синтаксис фреймов:

```
<FRAMESET COLS="value" | ROWS="value">
<FRAME SRC="url1">
<FRAME ...>
. . .
</FRAMESET>
```

Общий контейнер **FRAMESET** описывает все фреймы, на которые делится экран. Вы можете разделить экран на несколько вертикальных или несколько горизонтальных фреймов. Тэг **FRAME** прописывает каждый фрейм в отдельности. Рассмотрим более детально каждый компонент.

FRAMESET

```
<FRAMESET [COLS="value" | ROWS="value"]>
```

Тэг **<FRAMESET>** имеет завершающий тэг **</FRAMESET>**. Все, что может находиться между этими двумя тэгами, это тэг **<FRAME>**, вложенные тэги **<FRAMESET>** и **</FRAMESET>**, а также контейнер из тэгов **<NOFRAMES>**, который позволяет строить двойные документы для браузеров, поддерживающих фреймы и не поддерживающих фреймы.

Данный тэг имеет два взаимоисключающих параметра: **ROWS** и **COLS**.

- **ROWS**

```
ROWS="список-определений-горизонтальных-подокон"
```

Данный тэг содержит описания некоторого количества подокон, разделенные запятыми. Каждое описание представляет собой числовое значение размера подокна в пикселах, процентах от всего размера окна или связанное масштабное значение. Количество подокон определяется количеством значений в списке. Общая сумма высот подокон должна составлять высоту всего окна (в любых измеряемых величинах). Отсутствие атрибута **ROWS** определяет один фрейм, величиной во все окно браузера.

Синтаксис используемых видов описания величин подокон:

- **value**

Простое числовое значение определяет фиксированную высоту подокна в пикселах. Это далеко не самый лучший способ описания вы-

соты подокна, поскольку различные браузеры имеют различный размер рабочего поля, не говоря уже о различных экранных разрешениях у пользователя. Если вы, все же, используете данный способ описания размера, то настоятельно рекомендуется сочетать его с каким-либо другим, чтобы в результате вы точно получили 100%-ное заполнение окна браузера вашего пользователя.

- **value%**

Значение величины подокна в процентах от 1 до 100. Если общая сумма процентов описываемых подокон превышает 100, то размеры всех фреймов пропорционально уменьшаются до суммы 100%. Если, соответственно, сумма меньше 100, то размеры пропорционально увеличиваются.

- **value***

Вообще говоря, значение value в данном описании является необязательным. Символ «*» указывает на то, что все оставшееся место будет принадлежать данному фрейму. Если указывается два или более фрейма с описанием «*» (например «*,*»), то оставшееся пространство делится поровну между этими фреймами.

Если перед звездочкой стоит цифра, то она указывает пропорцию для данного фрейма (во сколько раз од будет больше аналогично описанного чистой звездочкой). Например, описание «3*,*,*», говорит, что будет создано три фрейма с размерами 3/5 свободного пространства для первого фрейма и по 1/5 для двух других.

- **COLS**

```
COLS="список-определений-горизонтальных-подокон"
```

То же самое, что и **ROWS**, но делит окно по вертикали, а не по горизонтали.

Внимание! Совместное использование данных параметров может привести к непредсказуемым результатам. Например, строка:

```
<FRAMESET ROWS="50%,50%" COLS "50%,50%">
```

может привести к ошибочной ситуации.

Примеры:

```
<FRAMESET COLS="50,*,50">
```

описывает три фрейма, два по 50 точек справа и слева, и один внутри этих полосок.


```
<FRAMESET ROWS="20%,3*,*">
```

описывает три фрейма, первый из которых занимает 20% площади сверху экрана, второй 3/4 оставшегося от первого фрейма места (т.е. 60% всей площади окна), а последний 1/4 (т.е. 20% всей площади окна).

```
<FRAMESET ROWS="*,60%,*">
```

аналогично предыдущему примеру.

Тэги **<FRAMESET>** могут быть вложенными, т.е. например:

```
<FRAMESET ROWS="50%,50%">
<FRAMESET COLS="*,*"
</FRAMESET>
</FRAMESET>
```

FRAME

```
<FRAME SRC="url" [NAME="frame_name"] [MARGINWIDTH="nw"] [MARGIN-
HEIGHT="nh"] [SCROLLING=yes|no|auto] [NORESIZE]>
```

Данный тэг определяет фрейм внутри контейнера **FRAMESET**.

```
SRC="url"
```

Описывает URL документа, который будет отображен внутри данного фрейма. Если он отсутствует, то будет отображен пустой фрейм.

```
NAME="frame_name"
```

Данный параметр описывает имя фрейма. Имя фрейма может быть использовано для определения действия с данным фреймом из другого HTML-документа или фрейма (как правило, из соседнего фрейма этого же документа). Имя обязательно должно начинаться с символа. Содержимое поименованных фреймов может быть задействовано из других документов при помощи специального атрибута **TARGET**.

```
MARGINWIDTH="value"
```

Это атрибут может быть использован, если автор документа хочет указать величину разделительных полос между фреймами сбоку. Значение value указывается в пикселах и не может быть меньше единицы. По умолчанию данное значение зависит от реализации поддержки фреймов используемым клиентом браузером.

```
MARGINHEIGHT="value"
```

То же самое, что и **MARGINWIDTH**, но для верхних и нижних величин разделительных полос.

```
SCROLLING="yes | no | auto"
```

Этот атрибут позволяет задавать наличие полос прокрутки у фрейма. Параметр *yes* указывает, что полосы прокрутки будут в любом случае присутствовать у фрейма, параметр *no* наоборот, что полос про-

крутки не будет. *Auto* определяет полосы прокрутки только при их необходимости (значение по умолчанию).

```
NORESIZE
```

Данный атрибут позволяет создавать фреймы без возможности изменения размеров. По умолчанию, размер фрейма можно изменить при помощи мыши так же просто, как и размер окна Windows. **NORESIZE** отменяет данную возможность. Если у одного фрейма установлен атрибут **NORESIZE**, то у соседних фреймов тоже не может быть изменен размер со стороны данного.

NOFRAMES

Данный тэг используется в случае, если вы создаете документ, который может просматриваться как браузерами, поддерживающими фреймы, так и браузерами, их не поддерживающими. Данный тэг помещается внутри контейнера **FRAMESET**, а все, что находится внутри тэгов **<NOFRAMES>** и **</NOFRAMES>** игнорируется браузерами, поддерживающими фреймы.

Рассмотрим реализацию фреймов для подобного разбиения окна:

```
<FRAMESET ROWS="*,*">
<NOFRAMES>
<H1>Ваша версия WEB-браузера не поддерживает фреймы!</H1>
</NOFRAMES>
<FRAMESET COLS="65%,35%">
<FRAME SRC="link1.html">
<FRAME SRC="link2.html">
</FRAMESET>
<FRAMESET COLS="*,40%,*">
<FRAME SRC="link3.html">
<FRAME SRC="link4.html">
<FRAME SRC="link5.html">
</FRAMESET>
</FRAMESET>
```

Глава 8. Планирование и взаимодействие фреймов

С появлением фреймов сразу возникает вопрос: «А как сделать так, чтобы нажимая на ссылку в одном фрейме инициировать появление информации в другом?»

Ответом на данный вопрос является планирование взаимодействия фреймов (далее — планирование). Каждый фрейм может иметь собственное имя, определяемое параметром **NAME** при описании данного фрейма. Существует, также, специальный атрибут — **TARGET**, позволяющий определять, к какому фрейму относится та или иная операция. Формат данного атрибута следующий:

```
TARGET="windows_name"
```

Данный атрибут может встречаться внутри различных тэгов:

TARGET в тэге A

Это самое прямое использование **TARGET**. Обычно, при активизации пользователем ссылки соответствующий документ появляется в том же окне (или фрейме), что и исходный, в котором была ссылка. Добавление атрибута **TARGET** позволяет произвести вывод документа в другой фрейм. Например:

```
<A HREF="mydoc.html" TARGET="Frame1">
Переход в фрейм 1 </A>
```

TARGET в тэге BASE

Размещение **TARGET** в тэге **BASE** позволит вам не указывать при описании каждой ссылки фрейм-приемник документов, вызываемых по ссылкам. Это очень удобно, если в одном фрейме у вас находится меню, а в другой — выводится информация. Например:

Документ 1

```
<FRAMESET ROWS="20,*">
<FRAME SRC="doc2.htm" NAME="Frame1">
<FRAME SRC="doc3.htm" NAME="Frame2">
</FRAMESET>
```

Документ 2

```
<HTML>
<HEAD>
<BASE TARGET="Frame2">
</HEAD>
<BODY>
<A HREF="url1"> Первая часть</A>
<A HREF="url2"> Вторая часть</A>
</BODY>
</HTML>
```

TARGET в тэге AREA

Также можно включать тэг **TARGET** в описание ссылки при создании карты изображения.

Например:

```
<AREA SHAPE="circle" COORDS="100,100,50" HREF="http://www.softex-
press.com" TARGET="Frame1">
```

TARGET в тэге FORM

То же относится и к определению формы. В данном случае, после обработки переданных параметров формы результирующий документ появится в указанном фрейме.

```
<FORM ACTION="url" TARGET="window_name">
```

Внимание! Имя окна (фрейма) в параметре **TARGET** должно начинаться с латинской буквы или цифры. Также необходимо помнить, что существуют зарезервированные имена для разрешения специальных ситуаций.

Глава 9. Зарезервированные имена фреймов

Зарезервированные имена фреймов служат для разрешения специальных ситуаций. Все они начинаются со знака подчеркивания. Любые другие имена фреймов, начинающиеся с подчеркивания будут игнорироваться браузером.

```
TARGET="_blank"
```

Данное значение определяет, что документ, полученный по ссылке будет отображаться в новом окне браузера.

```
TARGET="_self"
```

Данное значение определяет, что документ, полученный по ссылке будет отображаться в том же фрейме, в котором находится ссылка. Это имя удобно для переопределения окна назначения, указанного ранее в тэге **BASE**.

```
TARGET="_parent"
```

Данное значение определяет, что документ, полученный по ссылке будет отображаться в родительском окне, вне зависимости от параметров **FRAMESET**. Если родительского окна нет, то данное имя аналогично «_self».

```
TARGET="_top"
```

Данное значение определяет, что документ, полученный по ссылке будет отображаться на всей поверхности окна, вне зависимости от наличия фреймов. Использование данного параметра удобно в случае вложенных фреймов.

Глава 10. Создание документа HTML

В HTML документы записываются в ASCII формате и поэтому могут быть созданы и отредактированы в любом текстовом редакторе (например, Emacs или vi на UNIX машинах, или любом редакторе на IBM PC).

Пример документа в HTML

Любой гипертекст похож на книгу и может быть разбит на отдельные структурные элементы: Собственно документ, главы, параграфы, пункты, подпункты, абзацы.

Для каждого из этих элементов в HTML существуют определенные стили, описывающие в каком виде пользователь увидит текст на экране. Пусть мы создали файл minihtml.html:

```
<BODY>
<TITLE>Пример HTML-текста</TITLE>
<H1>Глава 1</H1>
<H2>Параграф 1.</H2>
Добро пожаловать в HTML! Здесь мы расскажем, как надо и как не
надо писать гипертексты.
<H2>Параграф 2.</H2>
</BODY>
```

Итак, вы уже поняли, что заголовок документа начинается с **<TITLE>** и заканчивается **</TITLE>**. Заголовок первого уровня (главы) выделяется символами **<H1>** и **</H1>**. Заголовки последующих уровней (параграфы, пункты, подпункты и т.п.) — символами **<Hx>** и **</Hx>**, где **x** — числа 2, 3, ... Абзац — символами **<P>** (в версиях HTML, действующих сейчас, символа **</P>** не существует! Но! Он может появится в последующих версиях!)

Внимание! HTML не различает, какими буквами набраны символы форматирования: **<title>** равносильно **<TITLE>** или **<TiTlE>**. Не все стили поддержаны всеми WWW-браузерами. Если браузер не поддерживает стиль, то он его игнорирует. (Поэтому не страшно, если вы уже сейчас начнете пользоваться при форматировании абзацев символом.

Основные элементы

Основной текст отделяется от сопроводительного символами: **<BODY> </BODY>.**

Заголовки документов

Каждый HTML-документ должен иметь заголовок, он показывается отдельно и используется, прежде всего, для идентификации документа (например, при поиске). Заголовок должен описывать цель документа и содержать не больше 5-6 слов. Практически во всех браузерах заголовок документа виден в верхней части экрана (окна).

Для выделения заголовка служат символы:

```
<HEAD> <TITLE>
Заголовок
</TITLE> </HEAD>
```

Заголовки разделов документов

HTML имеет шесть уровней заголовков, имеющих номера с 1 по 6 (заголовок первого уровня является заголовком высшего уровня). По сравнению с нормальным текстом, заголовки выделяются шрифтом — размером и толщиной букв. Первый заголовок в каждом документе должен быть выделен **<H1>**. Синтаксис заголовков:

```
<Hx>
Текст заголовка
</Hx>
```

где **x** — число от 1 до 6, определяющее уровень заголовка.

Абзацы

В отличие от документов в большинстве текстовых процессоров, прерывания строк и слов в HTML-файлах не существенны. Обрыв слова или строки может происходить в любом пункте в вашем исходном файле, при просмотре это прерывание будет проигнорировано. Напомним, что в нашем примере, первый параграф был представлен как **<H2>Параграф 1.</H2> Добро пожаловать в HTML! Здесь мы расскажем, как надо и как не надо писать гипертексты.**

В исходном файле два предложения находятся на двух разных строках. Web-браузер игнорирует это прерывание строки и начинает новый абзац только, после знака **< P>**. Однако, чтобы сохранить удобочитаемость в исходных HTML-файлах, мы рекомендуем заголовки размещать на отдельных строках, а абзацы отделять пустыми строками (в дополнение к **<P>**). Также заметим, что хотя в действующих версиях HTML нет признака форматирования **</P>**, мы вам рекомендуем его употреблять, поскольку он может быть введен в последующих версиях. К ошибке это не приведет, поскольку все незнакомые символы браузер просто игнорирует. В противном случае, вам в последствии, может быть,

придется переделывать ваши HTML-документы. В версии HTML+ предлагается, по аналогии с описанием заголовков, использовать как открывающий, так и закрывающий знаки абзаца.

Соединение с другими документами

Главное преимущество HTML состоит в его способности связываться с другими документами. Браузер выделяет (обычно цветом и/или подчеркиванием) ключевые слова, являющиеся гипертекстовыми ссылками (гиперссылками). Описывается ссылка на другой документ следующим образом:

```
<A HREF="имя файла"> Текст, который будет служить как обращение к
другому документу</A>.
```

Приведем пример такой гипертекстовой ссылки:

```
Пример HTML-текста
```

Здесь ключевые слова «Пример HTML-текста» являются гиперссылкой на файл с расширением .html, который лежит в той же директории, что и текущий документ. Вы можете ссылаться на документ, лежащий в любой директории, описав к нему полный путь. Так, например, ссылку на файл NJStats.html, лежащий в поддиректории AtlanticStates можно описать как:

```
<A HREF="AtlanticStates/NJStats.html">New Jersey</A>
```

Это так называемые относительные ссылки. Вы также можете использовать абсолютное имя файла (полный путь). В общем случае, использование ссылки по абсолютному имени файла более предпочтительно.

URL

Итак — URL это аббревиатура от Uniform Resource Locator. В него входит, кроме названия файла и директории: сетевой адрес машины и метод доступа к файлу. С помощью URL можно запускать удаленные программы, и передавать им значения. На этом принципе построены шлюзы в другие интернетовские сервисы: finger, archie и т.д. Здесь представлены несколько наиболее часто используемых типов URL. Допустим файл с именем «online15.zip» лежит на ftp сервере ftp.simtel.ru в директории /pub/doc/services/, тогда URL этого файла будет выглядеть так: file://ftp. simtel.ru/pub/doc/services/online15.zip. URL директории, в которой лежит файл: file://ftp.simtel.ru/pub/doc/services/, а URL корневой директории ftp сервера ftp.simtel.ru выглядит вот так: file://ftp.simtel.ru/

Gopher URL's не так разнообразны, как файловые. Это связано с ограниченностью этого сервиса. Для того чтобы описать, например,

gopher сервер узла gopher.kiae.su необходим URL: gopher://gopher.kiae.su/. Некоторые gopher-сервера могут находиться на нестандартном номере порта (по умолчанию обычно используется 70 порт) тогда он должен указываться: gopher://gopher.banzai.edu:1234/, где 1234 — номер порта.

Если вы внимательно посмотрите на исходники какого-нибудь гипертекстового документа, и обратите внимание на то, как указаны ссылки на другие URL то заметите, что встречаются два вида:

```
1. <A Href="http://www.simtel.ru/news/snews.http">News</A>
2. <A Href="aaa.html">AAA</A>
```

Первый — это полный URL, а второй — частичный. Частичный URL указывает на документ который находится на том же сервере и в той же директории, что и документ в котором встречается эта ссылка.

Обращение к определенным разделам других документов

Гиперссылки могут также использоваться для соединения с определенными разделами документов. Предположим, мы хотим соединить документ A с первой главой документа B, для чего нам необходимо создать именованную гиперссылку в документе B. Здесь вы можете увидеть

```
<A NAME = "Глава 1">Главу 1</a> Текст первой главы.
```

Теперь, описывая ссылку в документе A, надо включить не только имя файла documentB.html но также и имя гиперссылки, отделяемое символом (#): Здесь вы можете увидеть текст

```
<A HREF = "documentB.html#Глава1"> Главы 1 </A> документа B.
```

Теперь, «кликнув» в слово «Главы 1» в документе A, вы переходите непосредственно в Главу 1 в документе B.

Соединения с разделами текущего документа

Техника соединения аналогична описанной выше, только опускается имя файла. Вот, например, связь с Главой 1 внутри того же самого файла (Документ B) — это **<A HREF = "#Глава1">Глава 1</A>** текущего документа.

Дополнительные возможности форматирования

Выше было описано, как создавать простые HTML-документы. Для более сложных документов, HTML имеет некоторые дополнительные возможности форматирования.

Списки

HTML поддерживает ненумерованные, нумерованные списки и списки определений.

- **Ненумерованные списки**

Ненумерованный список:

```
<UL><LI>список пунктов </UL>
```

Например:

```
<UL> <LI> яблоки <LI> бананы </UL>
```

- **Нумерованные списки**

Нумерованный список идентичен ненумерованному списку, только вместо **<UL>** используется **<OL>**.

```
<OL> <LI> апельсины <LI> персики <LI> виноград </OL>
```

Браузер автоматически нумерует элементы такого списка.

Списки определений

Список определений обычно состоит из чередования термина **<DT>** и определения **<DD>**. Обычно Web-браузеры определения располагает на новой строке. Приведем пример списка определений:

```
<DT> NCSA
<DD> NCSA (National Center for Supercomputing Applications).
<DT> CTC
<DD> CTC (Cornell Theory Center).
</DL>
```

Вложенные списки

Списки могут быть произвольно вложены, хотя разумнее было бы практически ограничиться тремя уровнями вложенных списков.

Приведем пример вложенных списков:

```
<DD><UL> </DD>
<DD><LI> Крупные города России: </DD>
<DD><UL> </DD>
<DD><LI> Москва </DD>
<DD><LI> Санкт-Петербург </DD>
<DD></UL> </DD>
<DD><LI> Крупные города Украины: </DD>
<DD><UL> </DD>
<DD><LI> Киев </DD>
<DD></UL> </DD>
<DD></UL>
```

Авторский стиль редактирования

Как мы уже говорили выше, в общем случае, текст документа формируется браузером, игнорируя пробелы и переносы строк. Используя **<PRE>** можно описать в тексте заданный авторский стиль. (То есть пробелы и пустые строки показаны как пробелы и пустые строки, и строки будут прерываться там же что и в исходном HTML-файле.) Это полезно, например, для изображения программ:

```
#!/bin/csh
cd $SCR
cfs get mysrc.f:mycfsdir/mysrc.f
cfs get myinfile:mycfsdir/myinfile
fc -O2 -o mya.out mysrc.f
mya.out
cfs save myoutfile:mycfsdir/myoutfile rm *
```

Адреса

<ADDRESS> используется, чтобы определить автора документа и способы контакта с ним (например, e-mail адрес). Обычно это последний пункт в файле.

Например, последняя строка этого документа выглядит:

```
<ADDRESS> Введение в HTML/ НИИЯФ МГУ/ lenka@srdlan.npi.msu.su
</ADDRESS>
```

Внимание! **<ADDRESS>** не используется для почтового адреса.

Стили

Можно описывать специальными стилями отдельные слова или предложения. Имеются два типа стилей: логический и физический. Логические стили определяют текст согласно заданному значению, в то время как физические стили определяют некоторые участки текста.

Зачем существуют два стиля, если они могут дать одинаковый результат на экране? В качестве ответа на этот вопрос сошлемся на аксиому: «Доверьтесь вашему браузеру».

В идеале, в SGML, содержание отделяется от заглавия. Таким образом, SGML определяет строку как заголовок, но не определяет, что заголовок должен быть написан, например, жирным шрифтом с размером букв 24 пункта, и должен быть расположен в верхней части страницы. Преимущество этого подхода (это подобно в концепции стиля в большинстве текстовых процессоров) в том, что если вы решаете заменить стиль заголовка — все, что вы должны сделать — это изменить определение заголовка в Web браузере.

Другое преимущество стилей в том, что, например, удобнее определить что-нибудь как **<H1>**, чем помнить, каким шрифтом надо описывать заголовок. Это же истинно и для отдельных символов. Например, рассмотрим **<STRONG>**. Большинство браузеров рассматривают это как жирный шрифт в тексте. Однако, возможно, что читатель предпочел бы, чтобы в этом разделе это выделялось, например, другим цветом. Таким образом, стили дают пользователю большую свободу в выборе шрифтов.

Логические стили

- **<DFN>** — служит для описания определений.
- **<EM>** — служит для выделения слов.
- **<CITE>** — служит для выделения заголовков книг, фильмов, цитат и т.п.
- **<CODE>** — служит для выделения программных кодов, текстов программ и т.п. Изображается шрифтом фиксированной ширины.
- **<KBD>** — используется для ввода с клавиатуры пользователя. Может быть изображено жирным шрифтом.
- **<SAMP>** — используется для машинных сообщений. Изображается шрифтом фиксированной ширины.
- **<STRONG>** — служит для *особого* выделения слов. Обычно выделяется жирным шрифтом.
- **<VAR>** — используется для символьных переменных.

Физические стили

Существуют физические способы выделения — автор задает стиль написания текста, описывая шрифт в исходном HTML-документе.

Вы можете задать:

- **<B>**, **</B>** — жирный шрифт.
- **<I>**, **</I>** — наклонный шрифт.
- **<TT>**, **</TT>** — фиксированный шрифт (шрифт заданной ширины).

Специальные символы

Символы <, >, & и " имеют в HTML особое значение, как символы форматирования. Но иногда нам необходимо использовать их в тексте по своему прямому назначению.

Для их введения в текст, вы должны использовать:

- **& lt;** — левая скобка <
- **& gt;** — правая скобка >
- **& amp;** — &
- **& quot;** — "

Замечание: Специальные символы чувствительны к регистру: **нельзя** использовать **& LT;** вместо **& lt;**.

Прерывание строки

Используя **
** вы можете перейти на новую строку, не начиная нового абзаца (в большинстве браузер абзацы выделяются дополнительными пустыми строками).

Например:

```
Институт Ядерной Физики <BR> Московского Государственного
Университета <BR>
```

даст на экране:

```
Институт Ядерной Физики
Московского Государственного Университета
```

Горизонтальная линия

Используя **<HR>** вы можете разделить текст горизонтальной чертой.

Внутренние рисунки

Большинство Web браузеров могут показывать рисунки X Bitmap (XBM) или GIF формата вместе с текстом.

Поскольку каждый рисунок занимает много времени на отображение на экране (что замедляет показ документа), то мы не рекомендуем вам включать слишком большое количество или чрезмерно большие по размеру рисунки в ваш HTML-документ.

Чтобы включить рисунок, надо описать гиперссылку на него:

```
<IMG SRC=image_URL>
```

где **image_URL** — URL .gif или .xbm файла, содержащего рисунок. Таким образом, синтаксис ссылки на рисунок аналогичен синтаксису гиперссылки HREF.

Автоматически, рисунок выравнивается по нижнему краю сопровождающего его текста, но вы можете задавать взаимное расположение рисунка и текста:

- выравнивание по нижнему краю (делается браузером по умолчанию).
- выравнивание по верхнему краю.

Форматирование положения рисунка задается включением в гиперссылку пункта **ALIGN =**:

```
<IMG ALIGN=top SRC=image_URL>
```

Также возможны типы выравнивания:

```
ALIGN = MIDDL
ALIGN = CENTER
```

Если браузер не поддерживает рисунки

Некоторые WWW-браузеры, (например, используемые на VT100-терминалах) не могут показывать рисунки. Пользователи смогут увидеть только текст, заданный в пункте гиперссылки **ALT =**. Сопроводительный текст должен быть включен в кавычки. Например:

```
<IMG SRC="logo.gif" ALT = "logo.gif">
```

В этом случае пользователь увидит только текст "logo.gif".

Внешние рисунки, звуки и мультипликация

Если вы не хотите, чтобы рисунок замедлял загрузку основного WWW-документа, вы можете поместить рисунок в отдельный документ, написав на него гиперссылку. В этом случае пользователь сам должен решить — смотреть или не смотреть ему этот рисунок:

Рассмотрим более длинный пример HTML-документа:

```
<HEAD>
TITLE>Более длинный пример</TITLE>
</HEAD>
<BODY>
<H1>Более длинный пример</H1>
Это простой HTML-документ. Это первый абзац.
<P>Это второй абзац, он демонстрирует некоторые возможности HTML
по выделению слов. Это слово написано <I>наклонным</I> шрифтом.
Это слово написано <B>жирным</B> шрифтом.
Здесь Вы можете увидеть картинку: <IMG SRC="1_cool.gif">
<P>Это третий абзац, он демонстрирует использование гиперссылок.
Это гиперссылка на файл minihtml.html, содержащий простой пример
```

```
HTML-документа: <A HREF="minihtml.html">Пример HTML-текста</A>.<P>
<H2>Заголовок второго уровня</H2>
Дальнейший текст будет написан шрифтом фиксированной ширины: <P>
<PRE> On the stiff twig up there Hunches a wet black rook
Arranging and rearranging its feathers in the rain ...
</PRE>
Это ненумерованный список, состоящий из двух элементов:
<P>
<UL>
<LI> смородина
<LI> черника
</UL>
Конец документа. <P>
<ADDRESS>Irina Pankova (iii@vyt.hi.msu.au)</ADDRESS> </DD>
</BODY>
```

Глава 11. Формы в HTML документах

Некоторые WWW браузер позволяют пользователю, заполнив специальную форму, возвращающую полученное значение, выполнять некоторые действия на вашем WWW-сервере. Когда форма интерпретируется WEB-браузером, создается специальные экранные элементы GUI, такие, как поля ввода, checkboxes, radiobuttons, выпадающие меню, скроллируемые списки, кнопки и т.д. Когда пользователь заполняет форму и нажимает кнопку «Подтверждение» (SUBMIT — специальный тип кнопки, который задается при описании документа), информация, введенная пользователем в форму, посылается HTTP-серверу для обработки и передаче другим программам, работающим под сервером, в соответствии с CGI (Common Gateway Interface) интерфейсом.

Когда вы описываете форму, каждый элемент ввода данных имеет тэг **<INPUT>**. Когда пользователь помещает данные в элемент формы, информация размещается в разделе **VALUE** данного элемента.

Синтаксис

Все формы начинаются тэгом **<FORM>** и завершаются тэгом **</FORM>**.

```
<FORM METHOD="get|post" ACTION="URL"> Элементы_формы_и_другие_эле-
менты_HTML </FORM>
```


METHOD

Метод посылки сообщения с данными из формы. В зависимости от используемого метода вы можете посылать результаты ввода данных в форму двумя путями:

GET: Информация из формы добавляется в конец URL, который был указан в описании заголовка формы. Ваша CGI-программа (CGI-скрипт) получает данные из формы в виде параметра переменной среды QUERY_STRING. Использование метода **GET** не рекомендуется.

POST: Данный метод передает всю информацию о форме немедленно после обращения к указанному URL. Ваша CGI-программа получает данные из формы в стандартный поток ввода. Сервер не будет пересылать вам сообщение об окончании пересылки данных в стандартный поток ввода; вместо этого используется переменная среды CONTENT_LENGTH для определения, какое количество данных вам необходимо считать из стандартного потока ввода. Данный метод рекомендуется к использованию.

ACTION

ACTION описывает URL, который будет вызываться для обработки формы. Данный URL почти всегда указывает на CGI-программу, обрабатывающую данную форму.

Тэги формы

TEXTAREA

Тэг **<TEXTAREA>** используется для того, чтобы позволить пользователю вводить более одной строки информации (свободный текст). Вот пример использования тэга **<TEXTAREA>**:

```
<TEXTAREA NAME="address" ROWS=10 COLS=50>Москва, Тверская улица,
д.7, офис 1 </TEXTAREA>
```

Атрибуты, используемые внутри тэга **<TEXTAREA>** описывают внешний вид и имя вводимого значения. Тэг **</TEXTAREA>** необходим даже тогда, когда поле ввода изначально пустое. Описание атрибутов:

NAME — имя поля ввода

ROWS — высота поля ввода в символах

COLS — ширина поля ввода в символах

Если вы хотите, чтобы в поле ввода по умолчанию выдавался какой-либо текст, то необходимо вставить его внутри тэгов **<TEXTAREA>** и **</TEXTAREA>**.

INPUT

Тэг **<INPUT>** используется для ввода одной строки текста или одного слова. Атрибуты тэга:

CHECKED — означает, что **CHECKBOX** или **RADIOBUTTON** будет выбран.

MAXLENGTH — определяет количество символов, которое пользователи могут ввести в поле ввода. При превышении количества допустимых символов браузер реагирует на попытку ввода нового символа звуковым сигналом и не дает его ввести. Не путать с атрибутом **SIZE**. Если **MAXLENGTH** больше чем **SIZE**, то в поле осуществляется скроллинг. По умолчанию значение **MAXLENGTH** равно бесконечности.

NAME — имя поля ввода. Данное имя используется как уникальный идентификатор поля, по которому, впоследствии, вы сможете получить данные, помещенные пользователем в это поле.

SIZE — определяет визуальный размер поля ввода на экране в символах.

SRC — URL, указывающий на картинку (используется совместно с атрибутом **IMAGE**).

VALUE — присваивает полю значение по умолчанию или значение, которое будет выбрано при использовании типа **RADIO** (для типа **RADIO** данный атрибут обязателен)

TYPE — определяет тип поля ввода. По умолчанию это простое поле ввода для одной строки текста. Остальные типы должны быть явно, их полный список приведен ниже:

CHECKBOX

Используется для простых логических (**BOOLEAN**) значений. Значение, ассоциированное с именем данного поля, которое будет передаваться в вызываемую CGI-программу, может принимать значение **ON** или **OFF**.

HIDDEN

Поля данного типа не отображаются браузером и не дают пользователю изменять присвоенные данному полю по умолчанию значение. Это поле используется для передачи в CGI-программу статической информации, как то ID пользователя, пароля или другой информации.

IMAGE

Данный тип поля ввода позволяет вам связывать графический рисунок с именем поля. При нажатии мышью на какую-либо часть рисунка будет немедленно вызвана ассоциированная форме CGI-программа. Значения, присвоенные переменной **NAME** будут выглядеть так — создается две новых переменных: первая имеет имя, обозначенное в поле **NAME** с добавлением **.x** в конце имени. В эту переменную будет помещена X-координата точки в пикселах (считая началом координат левый верхний угол рисунка), на которую указывал курсор мыши в момент нажатия, а переменная с именем, содержащимся в **NAME** и добавленным **.y**, будет содержать Y-координату. Все значения атрибута **VALUE** игнорируются. Само описание картинки осуществляется через атрибут **SRC** и по синтаксису совпадает с тэгом **<IMG>**.

PASSWORD

То же самое, что и атрибут **TEXT**, но вводимое пользователем значение не отображается браузером на экране.

RADIO

Данный атрибут позволяет вводить одно значение из нескольких альтернатив. Для создания набора альтернатив вам необходимо создать несколько полей ввода с атрибутом **TYPE="RADIO"** с разными значениями атрибута **VALUE**, но с одинаковыми значениями атрибута **NAME**. В CGI-программу будет передано значение типа **NAME=VALUE**, причем **VALUE** примет значение атрибута **VALUE** того поля ввода, которое в данный момент будет выбрано (будет активным). При выборе одного из полей ввода типа **RADIO** все остальные поля данного типа с тем же именем (атрибут **NAME**) автоматически станут невыбранными на экране.

RESET

Данный тип обозначает кнопку, при нажатии которой все поля формы примут значения, описанные для них по умолчанию.

SUBMIT

Данный тип обозначает кнопку, при нажатии которой будет вызвана CGI-программа (или URL), описанная в заголовке формы. Атрибут **VALUE** может содержать строку, которая будет высвечена на кнопке.

TEXT

Данный тип поля ввода описывает однострочное поле ввода. Используйте атрибуты **MAXLENGTH** и **SIZE** для определения максимальной длинны вводимого значения в символах и размера отображаемого поля ввода на экране (по умолчанию принимается 20 символов).

Меню выбора в формах

Под меню выбора в формах понимают такой элемент интерфейса, как **LISTBOX**. Существует три типа тэгов меню выбора для форм:

Select — пользователь выбирает одно значение из фиксированного списка значений, представленных тэгами **OPTION**. Данный вид представляется как выпадающий **LISTBOX**.

Select single — то же самое, что и **Select**, но на экране пользователь видит одновременно три элемента выбора. Если их больше, то предоставляется автоматический вертикальный скроллинг.

Select multiple — позволяет выбрать несколько элементов из **LISTBOX**.

SELECT

Тэг **SELECT** позволяет пользователю выбрать значение из фиксированного списка значений. Обычно это представлено выпадающим меню.

Тэг **SELECT** имеет один или более параметр между стартовым тэгом **<SELECT>** и завершающим **</SELECT>**.По умолчанию, первый элемент отображается в строке выбора. Вот пример тэга **<SELECT>**:

```
<FORM>
<SELECT NAME=group>
<OPTION> AT 386
<OPTION> AT 486
<OPTION> AT 586
</SELECT>
</FORM>
```

SELECT SINGLE

Тэг **SELECT SINGLE** — это то же самое, что и **Select**, но на экране пользователь видит одновременно несколько элементов выбора (три по умолчанию). Если их больше, то предоставляется автоматический вертикальный скроллинг. Количество одновременно отображаемых элементов определяется атрибутом **SIZE**. Пример:

```
<FORM>
<SELECT SINGLE NAME=group SIZE=4>
<OPTION> AT 386
<OPTION> AT 486
<OPTION> AT 586
<OPTIONS> Pentium PRO
</SELECT>
</FORM>
```


SELECT MULTIPLE

Тэг **SELECT MULTIPLE** похож на тэг **SELECT SINGLE**, но пользователь может одновременно выбрать более чем один элемент списка. Атрибут **SIZE** определяет количество одновременно видимых на экране элементов, атрибут **MULTIPLE** — максимальное количество одновременно выбранных элементов. Пример:

```
<FORM>
<SELECT SINGLE NAME=group SIZE=4 MULTIPLE=2>
<OPTION> AT 386
<OPTION> AT 486
<OPTION> AT 586
<OPTIONS> Pentium PRO
</SELECT>
</FORM>
```

Если выбрано одновременно несколько значений, то серверу передаются соответствующее выбранному количество параметров **NAME=VALUE** с одинаковыми значениями **NAME**, но разными **VALUE**.

Отправление файлов при помощи форм

Формы можно использовать для отправки не только небольших информационных сообщений в виде параметров, а также и для отправки файлов.

Внимание! Поскольку данная возможность требует поддержки получения файлов WEB-сервером, то, соответственно, необходимо, чтобы сервер поддерживал получение файлов!

Например:

```
<FORM ENCTYPE="multipart/form-data" ACTION="url" METHOD=POST>
```

Отправить данный файл:

```
<INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Отправить файл">
</FORM>
<FORM action="http://pandemonium.cs.nstu.ru/~gun/docs/sites.htm"
encType="multipart/form-data" method="post">
```

Отправить данный файл:

```
<INPUT name="userfile" type="file">
<INPUT type="submit" value="Отправить файл">
</FORM>
```

Глава 12. HTML 4.0

Предмет гордости этой версии — новые средства для разработки Web-страниц и расширяемость, а также незначительное число нестандартных кодов.

Если в течение последних лет вы внимательно следили за развитием HTML, то, вероятно, у вас сложилось мнение, что этот жизненно важный язык кодирования — сердце самой Web — не что иное, как причудливая смесь хороших, не очень хороших и превосходных, но рассчитанных только на один браузер средств. Вы недалеки от истины. В каждой следующей версии браузеров появляются все новые и новые тэги и контейнеры, и начинает казаться, будто консорциум World Wide Web Consortium (W3C) будет вечно (по крайней мере, пока жива Web) заниматься лишь внедрением этих готовых элементов в официальную спецификацию HTML.

Именно так и должно быть, если учесть, что HTML — это «официально утвержденная спецификация», а не «просто новейшие средства». Однако для дизайнеров Web-страниц такие задержки оборачиваются множеством проблем. Главная из них — это необходимость подготавливать одни и те же страницы в двух вариантах — для Microsoft Internet Explorer и для Netscape Navigator. Многие HTML-тэги предназначены только для одного из этих браузеров и не работают в другом. Существование подобных тэгов вносит хаос в общую концепцию единых стандартов для Web.

В 1996 г. W3C выпустил стандарт HTML 3.2. Основная цель принятия данной версии — объединить и официально утвердить значительное число фирменных тэгов Microsoft и Netscape, поскольку они уже получили широкое распространение. Но совершенно ясно, что подобная мера была временной, рассчитанной на то, чтобы пользователи могли пережить период ожидания кардинально новой версии HTML. Эта версия, в течение нескольких месяцев носившая кодовое название Cougar (Пума), отчасти предназначалась для обуздания процесса появления расширений HTML от отдельных фирм. Конечно, абсолютного успеха этой версии достичь не удастся, поскольку Microsoft и Netscape слишком ревностно стремятся быть первыми и единственными игроками на этом поприще с правом на собственные нововведения в HTML. Однако удалось добиться по крайней мере возможности контролировать дальнейшее расширение этого основного авторского инструмента Web.

Контролируемое расширение — это лишь одна из задач, стоявших перед разработчиками HTML 4.0. Другая задача — удовлетворить возрастающую потребность в более совершенных средствах для макетирования страниц и программируемых функциях. Эту задачу также удалось решить. В итоге появилась новая спецификация HTML, которая даже при первом знакомстве производит впечатление более зрелого продукта, чем предыдущие. И как раз вовремя! Как только появились сведения о том, что на подходе спецификация XML (Extensible Markup Language), положение HTML оказалось под угрозой. Возможно, с выпуском новой версии HTML взлет XML будет менее стремительным, что могло бы привести к мирному сосуществованию обеих спецификаций.

Расширяемый HTML

До настоящего времени основная проблема HTML заключалась в том, что от версии к версии перечень его тэгов и контейнеров (container — набор парных тэгов, обрамляющих содержательную часть конструкции, например:

```
<CENTER>...</CENTER>
```

для выравнивания по центру) не менялся. Диапазон возможностей разработчика по оформлению и программированию Web-страниц постоянно расширялся, однако опытные авторы всегда ощущали их недостаток, причем потребность в дополнительных средствах была насущной.

Чтобы решить эту проблему, фирма Netscape (тогда имевшая название Mosaic Communications Corp.) стала время от времени предлагать свои собственные HTML-элементы; впервые они вошли в версию 2.0 ее браузера (сначала он назывался Netscape). Благодаря его популярности, эти управляющие коды стали все чаще применяться в Web, и помешать этому процессу уже не могли сторонники чистоты стандарта HTML. В свою очередь, Microsoft добавила в готовящийся к выпуску браузер Internet Explorer собственные тэги, и вскоре стали появляться Web-узлы, где были предусмотрены «ветвления» для пользователей Internet Explorer и Navigator. В худшем случае для обращения к узлам годился только один из этих двух браузеров.

Короче говоря, расширение возможностей HTML всегда сопряжено с трудностями. Для этого нужно сначала разработать сами элементы (что не так уж трудно), а затем подготовить соответствующие модули для браузера или внести в него изменения, с тем чтобы он надлежащим образом воспроизводил эти тэги (что значительно сложнее). Эффективного решения этой задачи удалось добиться только для браузеров Microsoft и Netscape, поскольку за последние несколько лет именно им

было отдано практически повсеместное предпочтение пользователей. Но ни Microsoft, ни Netscape не отвечают за утверждение стандартов, поэтому де-факто им приходится сначала вводить свои расширения, а затем вносить предложение в W3C об их включении в официальную спецификацию HTML. Официально принятая версия HTML 3.2 по сути дала лучшие из уже существующих тэгов.

В HTML 4.0 предпринят другой подход. Вместо включения в спецификацию максимально возможного числа уже существующих, но официально не утвержденных элементов, для разработчиков предусматривается определенная свобода самостоятельно вносить дополнения в HTML. Хотя по степени расширяемости язык HTML 4.0 не идет ни в какое сравнение с XML, он по крайней мере поддается этому. Решается подобная задача достаточно просто — с помощью нового элемента **object**. Контейнер **OBJECT** несет браузеру информацию о том, что имеется элемент одного из нескольких типов, обычно относящийся к данным мультимедиа. Кроме того, он содержит сведения о том, должен ли браузер пытаться воспроизвести этот объект, или эти полномочия необходимо передать какой-то внешней программе.

В состав контейнера **OBJECT** может входить три основных вида информации: адрес прикладной программы, предназначенный для воспроизведения объекта; сами обрабатываемые данные и любые используемые для этого параметры. Сначала браузер пробует полностью следовать всем заданным командам; если же его попытки заканчиваются неудачно, воспроизведение этого элемента он берет на себя.

Одним из примеров элементов типа **OBJECT** служит графический файл. До сих пор для обращения с ними использовался тэг **IMG**, и по-прежнему нужда в нем не отпала. Однако в целях согласованности подхода изображения рассматриваются как объекты определенных типов, и не исключено, что в конце концов тэг **IMG** выйдет из обращения вообще. Оцените разницу:

```
<IMG SRC=http://www.mycomputer.com/images/
1997/meeting01.jpg>
```

и

```
<OBJECT data="http://www.mycomputer.com/
images/1997/meeting01.jpg" type="image/jpg">
```

Следующий пример показывает новый способ встраивания апплет, имеющийся в HTML 4.0. Обратите внимание — сейчас допускаются оба варианта, но в дальнейшем элемент **APPLET**, по всей видимости, будет снят с вооружения:


```
<APPLET code="songviewer.class" width="550" height="600">Click
here to view the sheet music in a Java applet</APPLET>
<OBJECT codetype="application/octet-stream"
classid="java:songviewer.class" width="550" height="600">Click
here to view the sheet music in a Java applet</OBJECT >
```

Как вы видите на двух этих примерах, прежний метод программирования проще и компактнее. Тогда почему приоритет отдается контейнеру **OBJECT**? Во-первых, немалое значение имеет то, что такой способ обработки объектов обеспечивает HTML более высокую степень согласованности: все типы не-HTML-файлов могут обрабатываться как объекты; в результате нет необходимости запоминать множество имен HTML-элементов. Процедуры разработки средствами HTML становятся более структурированными.

Во-вторых, что тоже немаловажно, теперь пользователям придется, вероятно, реже обращаться к диалоговому окну **Download** (Загрузка файла), когда они щелчком мыши выбирают связь с файлом, тип которого не зарегистрирован в ОС. Поскольку контейнер **OBJECT** содержит также информацию о MIME-типе, операционная система вполне обходится без сведений о том, как обрабатывать файлы этого типа. Конфигурацию браузера можно настроить таким образом, чтобы указанные в элементе **OBJECT** типы обрабатывались через область вспомогательных программ, и в идеале Microsoft и Netscape будут совершенствовать такую конфигурацию в последующих версиях браузеров.

Наконец, с появлением контейнера **OBJECT** язык HTML становится формально расширяемым. Отныне нет нужды обновлять браузеры для обслуживания новых тэгов; они должны лишь надлежащим образом обрабатывать тэг **OBJECT**; а это решается путем настройки конфигурации, а не подготовки исходного текста программ. Однако учтите, что такая возможность вовсе не означает, что расширяется набор средств браузеров. Как и прежде, они отображают файлы только определенных типов, поэтому необходимость во внешних прикладных программах все же может возникнуть. Однако в результате выигрывает разработчик: предложив новый тип файла, он может составить программу просмотра, которая будет отвечать за его отображение, предоставить эту программу для загрузки из сети и предусмотреть связи с ней с помощью контейнера **OBJECT**.

Тем не менее важно соблюдать меру. По расширяемости HTML очень далек от XML, который по сути дела описывает языки разметки, или Document Type Definitions (DTDs) (описания типов документов). Фактически весь HTML — это всего лишь одна спецификация DTD. Присущая XML возможность задавать новые DTD означает, что этот

язык обеспечивает гораздо более широкое разнообразие параметров, чем когда бы то ни было будет в тэге **OBJECT** языка HTML. Однако тэг OBJECT — это только начало, и он, есть надежда, ослабит поток фирменных, рассчитанных на конкретный браузер тэгов.

Полнофункциональные экранные формы

По сравнению с экранными формами офисных приложений формы, созданные средствами HTML, казались просто примитивными. Они имели упрощенный вид и скудные функциональные возможности. Разработчики HTML-форм обходили своим вниманием необходимость обеспечить такие средства, какие мы привыкли видеть в развитых приложениях. Среди них, например, средства для подготовки клавиатурных команд быстрого вызова, для обновления интерфейса по мере заполнения полей, надписи на кнопках и т.д.

В HTML 4.0 вы сможете значительно лучше оснастить средствами свои формы. Например, атрибут **accesskey** позволяет назначить каждому полю некоторое сочетание клавиш; это избавляет нас от необходимости непрерывно щелкать мышью для выбора полей, как в предыдущих версиях HTML. Помимо прежних **Submit** (Отправить) и **Reset** (Обновить) возможны и другие типы кнопок (задаваемые с помощью элемента **BUTTON**); теперь с помощью атрибута **disabled** (недоступна) можно сделать их «блеклыми», — т.е. процесс заполнения форм становится более интуитивно понятным. Можно задавать любые надписи на кнопках, используя элемент **LABEL**, или с применением элемента **FIELDSET** группировать связанные поля. Можно присваивать отдельным полям атрибут readonly (только для чтения). Исключительно полезный новый атрибут **onchange-INPUT** обеспечивает проверку вводимой в форму строки.

Конечно, такие функции, как верификация ввода, имелись и в предыдущих версиях HTML, однако раньше для этого выполнялся обмен данными с сервером. Приходилось щелкать на кнопке **Submit** и ждать отклика с сервера с подтверждением корректности введенной строки. В HTML 4.0 проверка ввода в экранные формы проводится локально, тем самым ускоряется работа с формами и возникает меньше проблем при обработке.

Новые табличные функции

Таблицы в предыдущих версиях HTML, как и формы, значительно уступали по своим возможностям таблицам из систем управления базами данных или электронным таблицам. HTML 4.0 отличается от своих предшественников гораздо более гибкими и полезными средствами для компоновки таблиц. Например, атрибут **align** предусматривает не только

стандартные варианты выравнивания — по правой и левой границе, по центру и по ширине, но также при использовании признака **char** обеспечивает выравнивание текста по отношению к заданному символу. Скажем, с применением **align char** можно выровнять все числа по знаку десятичных точек. Атрибуты **frame** и **rules** предназначены для управления внешним видом таблиц. С помощью первого можно задать рамку вокруг указанных сторон таблицы (скажем, ее правой и левой сторонами); тогда как, употребив атрибут **rules**, вы проведете линии между ячейками, строками или отдельными группами ячеек.

Появление элемента **COLGROUP** по сути дела означает возможность объединять в группу несколько столбцов, чтобы присвоить им одинаковые ширину, направление размещения текста, стиль, выравнивание и команды событий. Для отдельных групп строк могут задаваться заголовки и итоги, которые при прокрутке таблицы остаются неподвижными. Более того, поместив контейнеры **THEAD** и **TFOOT** перед контейнером **TBODY**, можно добиться того, чтобы заголовок и итоги появлялись до начала заполнения ячеек самой таблицы, что позволяет пользователям узнать ее в тот момент, когда она воспроизводится в браузере.

Сценарии для автоматизации

Новый контейнер **<SCRIPT>** языка HTML 4.0 обеспечивает возможность составлять сценарии. В нем указывается URL-адрес сценария, который будет выполняться, и сообщается, какой язык сценариев должен использовать браузер. Предусматривается два основных языка: JavaScript и VBScript. С помощью элемента **META** можно задать язык по умолчанию для всех сценариев в данном документе или для конкретного сценария. Сценарии исполняются либо при загрузке документа, либо, если применяются атрибуты intrinsic event, всякий раз, когда происходит определенное событие (например, при щелчке мышью).

С помощью элемента **SCRIPT** можно решать несколько разных задач. Встроив сценарий в экранную форму, вы добьетесь автоматического заполнения ее полей в тот момент, когда пользователь щелкает мышью или вводит определенное значение в какое-то из полей. Допустим, конечная дата должна отличаться от начальной на один день, тогда с помощью конкретного сценария будет автоматически заполняться поле конечной даты после считывания значения начальной даты. Можно подготовить сценарии, изменяющие содержимое документа по мере его загрузки, за счет последовательной загрузки конкретных элементов. Кроме того, можно назначить кнопкам сценарии и, таким образом, сформировать интерактивную страницу с интерфейсом на основе кнопок. Поскольку сценарии исполняются локально, нет необходимости в подключении к серверу.

Встроенные кадры

Кадры (фреймы) — это важный элемент при разработке HTML-страниц. Однако до сих пор кадр окаймлял лишь границы документа. В HTML 4.0 можно встраивать кадр внутрь текстового блока — эффективный способ для размещения одного HTML-документа в другом. Синтаксис очень прост. Покажем пример:

```
<IFRAME src="info.html" width "300"
height "500" scrolling="auto" frameborder="2"></IFRAME>
```

Помещенный в HTML-документ, этот текст содержит информацию для браузера о том, что нужно обратиться к исходному документу (**src=filename.html**) и вывести его в текущем документе в виде кадра заданного размера. Допускается применение любых атрибутов элемента **frame**, за исключением **noresize**, поскольку изменение размеров встроенных кадров запрещено.

Другие возможности

Кроме перечисленных средств, HTML 4.0 содержит множество других. Например, акцент был сделан и на интернационализации. Глобальная природа Web означает, что здесь фигурируют документы на самых разных естественных языках. Однако до сих пор представление национальных символов было проблематичным. Спецификация HTML 4.0 подчиняется международным соглашениям RFC 2070 «Internationalization of the HyperText Markup Language» (Правила адаптации языка разметки гипертекста к национальным системам), а для выбора таблицы кодировки символов документа — стандарту ISO/IEC:10646. Это обеспечивает расширенный набор нелатинских символов.

Кроме того, в HTML 4.0 нашел воплощение проект CSS (Cascading Style Sheets — Каскадные таблицы стилей). Консорциум W3C стремится повысить привлекательность CSS, чтобы разработчики наконец отказались от применения таблиц для компоновки страниц. Даже ранние версии CSS обладали более, чем достаточными возможностями для решения данной задачи. Благодаря CSS в целом ряде документов можно использовать одни и те же шрифты, цвета, выравнивания и другие элементы форматирования. В результате можно добиться единого стиля оформления для Web-узлов.

Совокупность всех названных свойств позволяет говорить о принципиально новой ступени в развитии HTML. Версия 4.0 избавлена от многих прежних ограничений, касавшихся расширяемости, разметки страниц, интерактивности и, конечно, взаимоотношений клиента и сервера. Все это означает, что во время работы в Web вы получите массу новых, ярких впечатлений.

В HTML 4.0 имеется улучшенная поддержка разных направлений письма и направления справа налево, таблицы с большим количеством возможностей и новые свойства форм, обеспечивая лучшие возможности доступа для людей с физическими недостатками.

Эта версия HTML разработана с помощью экспертов в области интернационализации, так что документы можно писать на любом языке и легко передавать их по всему миру.

Важным шагом стало принятие стандарта ISO/IEC:10646 в качестве набора символов для документов HTML. Это наиболее содержательный стандарт в мире, в котором решены вопросы представления национальных символов, направления письма, пунктуации и других языковых вопросов.

HTML теперь предоставляет лучшую поддержку различных языков в одном документе. Это обеспечивает более эффективное индексирование документов для поисковых машин, типографию высшего качества, преобразование текста в речь, более удобные переносы и т.д.

Доступность

Поскольку сообщество Web растет, и возможности и умения его членов различаются, очень важно, чтобы основные технологии соответствовали потребностям. Язык HTML разработан так, чтобы сделать Web-страницы более доступными для пользователей с физическими недостатками. В HTML 4.0 имеются следующие дополнения, продиктованные соображениями доступности:

- усилено разделение структуры и представления документа, что побуждает использовать таблицы стилей вместо элементов и атрибутов представления языка HTML.
- улучшены формы, включена возможность назначения клавиш доступа, возможность семантической группировки управляющих элементов формы, семантической группировки вариантов в тэге SELECT и активные метки.
- добавлена возможность разметки текстового описания включенного объекта (с помощью элемента **OBJECT**).
- введен новый механизм действия изображений-карт на стороне клиента (элемент **MAP**), который позволяет авторам интегрировать изображения и текстовые ссылки.
- альтернативный текст для изображений, включаемых с помощью элемента **IMG**, обязателен.

- добавлена поддержка атрибутов **title** и **lang** во всех элементах.
- добавлена поддержка элементов **ABBR** и **ACRONYM**.
- более широкий диапазон целевых устройств (телетайп, шрифт Бройля и т.д.) для использования в таблицах стилей.
- улучшены таблицы, включена поддержка заголовков, групп столбцов и механизмов для упрощения невизуального представления документа.
- добавлены длинные описания таблиц, изображений, кадров и т.д.

Авторы, разрабатывающие страницы с учетом доступности, получат не только эту возможность, но также и некоторые другие: хорошо разработанные документы HTML с разделенными структурой и представлением будут легче адаптироваться к новым технологиям.

Таблицы

Теперь авторы имеют большую власть над структурой и компоновкой таблицы (например, группы столбцов). Возможность дизайнеров рекомендовать ширину столбцов позволяет браузерам отображать данные таблицы постепенно (по мере получения) и не ждать всю таблицу до создания изображения.

Составные документы

В HTML теперь имеется стандартный механизм для внедрения объектов и приложений в документы HTML. Элемент **OBJECT** (а также более специфичные элементы, его преемники, **IMG** и **APPLET**) обеспечивает механизм включения в документ изображений, видеофайлов, звуковых файлов, математических выражений, специализированных приложений и других объектов. Он также позволяет авторам указывать иерархию или альтернативный способ создания изображения для браузеров, не поддерживающих указанный способ создания изображения.

Таблицы стилей

Таблицы стилей упрощают разметку HTML и существенно снижают участие языка HTML в представлении документа. Они предоставляют как авторам, так и пользователям возможность управлять представлением документов — шрифтами, выравниванием, цветами и т.д.

Информацию о стиле можно указать для отдельных элементов или групп элементов, в документе HTML или во внешних таблицах стилей.

Механизмы связи таблиц стилей с документами не зависят от языка таблиц стилей.

До появления таблиц стилей возможности управления созданием изображения у авторов были ограничены. В HTML 3.2 был включен ряд атрибутов и элементов для управления выравниванием, размером шрифта и цветом текста. Авторы также использовали для компоновки страниц таблицы и изображения. Поскольку на обновление браузеров у пользователей уйдет довольно долгое время, эти средства еще будут использоваться в течение какого-то времени. Однако поскольку таблицы стилей обеспечивают более мощные механизмы представления, World Wide Web Consortium существенно сократит число элементов и атрибутов представления в HTML. В этой спецификации элементы и атрибуты, которые могут быть впоследствии исключены, помечены как «нежелательные». Они сопровождаются примерами по достижению того же эффекта с помощью других элементов или таблиц стилей.

Скрипты

С помощью скриптов авторы могут создавать динамичные Web-страницы (например, «интеллектуальные формы», изменяющиеся по мере заполнения их пользователем) и использовать HTML как средство построения сетевых приложений.

Механизмы, обеспечивающие включение скриптов в документы HTML, не зависят от языка скриптов.

Печать

Иногда авторы хотят упростить для пользователей печать текущего документа. Если документ является частью другого документа, отношения между ними можно описать с помощью элемента HTML LINK или языка описания ресурсов (Resource Description Language — RDF) W3C.

Функции

Если попытаться выделить многочисленные новые функции и особенности последней версии HTML, эту информацию будет трудно переварить. Однако в действительности многие «новые» средства не такие уж и новые, хотя HTML, эволюционируя от версии 3.2 к версии 4.0, претерпел значительные изменения. Но такие браузеры, как Internet Explorer и Netscape Navigator, уже поддерживают ключевые средства но-

вой версии HTML, а ряд других новых функций представляет собой упрощенный или усовершенствованный вариант существующих структур HTML, облегчающих работу с современными web-технологиями.

Основные усовершенствования коснулись фреймов, мультимедиа, таблиц и форм, а также работы со сценариями и таблицами стилей.

Фреймы

Одним из самых значительных усовершенствований в последней версии HTML стало распознавание стандартных и встроенных фреймов. Конечно, фреймы не представляют собой ничего нового. На самом деле, стандартные фреймы появились еще в Netscape Navigator 2.0. С тех пор в разных версиях браузеров предлагались различные усовершенствования. К счастью, для совместимости с HTML 4.0 браузеры должны будут поддерживать фреймы, определенные в данной спецификации.

С помощью стандартных фреймов можно создавать web-страницы с «мини-окнами», причем каждое из них может иметь свое информационное наполнение. Обычно такое мини-окно отделяется от остального экрана рамкой и выводится с полосами прокрутки. Кроме того, фрейм можно вставить непосредственно в блок текста на web-странице (как встроенное изображение). Такой фрейм называется встроенным (in-line) и создается с помощью тэга **iframe**. Многие атрибуты встроенных и стандартных фреймов совпадают, другие добавлены для на стройки размера и выравнивания окна фрейма. Для использования встроенного фрейма нужно выбрать место и добавить элемент **iframe**. С помощью атрибута **src** можно задать исходный документ, а затем указать позицию фрейма. В следующем примере в документ вставляется простой встроенный фрейм в 500 пикселов высотой и 325 пикселов шириной:

```
<iframe src="samples.html" width="325" height="500"
align="right"></iframe>
```

Как и в случае со стандартными фреймами, с помощью атрибута **frameborder** нетрудно задать фрейм без рамки. Установив этот атрибут равным 1, можно вывести рамку, а присвоив ему значение 0, — скрыть ее. Для достижения требуемого эффекта исходный документ должен помещаться внутри фрейма. В противном случае браузер выводит на экран фрейм с рамкой и полосами прокрутки. Следует отметить, что тэг **iframe** позволяет определить выводимую в браузере информацию, даже если он несовместим с HTML 4.0. Для этого достаточно включить ее в сам элемент **iframe**. Например:

```
<iframe src="samples.html" width="325" height="500"
frameborder="0">
```



```
<P>Normally, a product sample is displayed in this space.
However, your browser doesn't support HTML 4.0.</P>
</iframe>
```

Совместимые с HTML 4.0 браузеры будут игнорировать подобный текст, а несовместимые выводят его, игнорируя тэг **iframe**. И еще одно замечание по встроенным фреймам: присваивая фрейму имя, можно дать ссылки, которые выводят в этом фрейме другие файлы. Такие гиперссылки для фрейма называются целевыми. Цель указывается атрибутом фрейма **name**. Имя же присваивается как значение атрибута **target** с соответствующей гиперссылкой.

Указание цели во встроенных фреймах с помощью значения атрибута name

```
<HTML>
<HEAD>
<TITLE>Inline Frames Example</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<DIV>
<H1 align="center">Calculators R Us</H1>
<IFRAME name="samp" src="samples.html" width="325" height="500"
align="right" frameborder="0">
</IFRAME>
<P>Наш каталог содержит свыше 500 продуктов, образцы которых можно получить по Сети. Попробуйте демоверсии продуктов и подумайте. Сегодня предлагается 25%-я скидка. Выберите нужные вам продукты:
</P>
<A href="feature1.html" target="samp"> S5000 Scientific Plus</A>
<BR>
<A href="feature2.html" target="samp"> E2800 Engineering Max</A>
<BR>
<A href="feature3.html" target="samp"> M3500 Statistical
Plus</A></P>
</DIV>
</BODY>
</HTML>
```

Мультимедиа

Еще один важный шаг вперед — элемент **object**, предусматривающий тэг общего назначения для работы с различными типами информации, такими, как Java-апплеты, встроенное видео, потоковые звук и изображение. В конечном счете он должен заменить тэги для разных типов информации мультимедиа. Элемент **object** можно будет использовать

вместо тэгов **applet**, **img** и **embed**. Его удобно применять даже для вывода на экран в окне браузера обычных текстовых файлов. В этом случае он во многом напоминает встроенный фрейм.

Первоначально элемент **object** был предложен в качестве компромиссного варианта вместо тэгов, предназначенных для конкретных целей, таких, как **img** и **applet**. К сожалению, хотя разработчики браузеров видели необходимость стандартного способа включения мультимедиа в web-страницы, они не всегда были единодушны в том, каким способом это нужно делать. Кроме того, в течение какого-то времени тэг **object** не являлся частью официальной спецификации HTML. Наконец, в HTML 4.0 элемент **object** все же принят как решение для будущей реализации в web-страницах объектов мультимедиа.

Браузеры, совместимые с HTML 4.0, без проблем интерпретируют тэги **<object>**, однако старые браузеры могут этого не делать, и вы не будете знать, как именно будут выводиться на экран и работать внедренные объекты мультимедиа. Для обратной совместимости может потребоваться включить в элемент **object** тэги для конкретных типов мультимедиа. Такие тэги должны следовать сразу за тэгом **< object>** В следующем примере, если браузер не понимает тэг **<object>**, он попытается использовать тэг:

```
<object data="cougar.bmp" type="image/bmp">
<embed src="cougar.bmp">
</object>
```

Таблицы

Большая часть нововведений в данной области, появившихся в HTML 3.0, сохранилась в HTML 4.0, но появились и дополнительные средства. Стало возможным группировать столбцы и определять (в начале таблицы) их свойства. Кроме того, допускается определение заголовка, нижнего колонтитула и содержимого разделов таблицы.

Элементы **th** и **td** позволяют определить подзаголовок или ячейки данных, а также указать браузерам, как должны выглядеть столбцы в таблице. Для этого используются два элемента: тэг **colgroup** создает структурную группу, устанавливающую характеристику столбцов в этой группе, а тэг **col** позволяет задать общие для группы атрибуты. В совокупности эти элементы позволяют браузерам немедленно начинать вывод на экран столбцов, постепенно (по мере загрузки) формируя таблицу.

Использовать такие элементы в таблице проще, чем кажется. Они поддерживают идентичный набор атрибутов, предоставляя достаточную свободу в определении структур группы. Атрибутам **width**, **cellhalign** и **cel-**

lvalign присваивается заданная по умолчанию ширина, выравнивание ячейки по горизонтали и вертикали соответственно, а **span** задает число столбцов, совместно использующих атрибуты элемента. Чтобы сделать таблицы более динамичными, можно применять атрибуты таблицы стилей и встроенные события.

Существует несколько разных способов задания структуры для идентичных групп столбцов. Имейте в виду, что если вместо абсолютной ширины (в пикселах) применяется относительная ширина столбцов, то нужно задать и ширину всей таблицы. Для этого используется атрибут **width** элемента **table**. Можно также сгруппировать строки таблицы, задав для них общий заголовок, нижний колонтитул и содержимое разделов с помощью элементов **thead**, **tfoot** и **tbody**. Заголовок и нижний колонтитул содержат информацию о столбцах таблицы, а в теле таблицы находятся строки данных.

В идеальном случае браузеры будут использовать такое разделение для интеллектуального вывода таблиц. Например, если таблица выходит за пределы текущего окна, то браузер может позволить читателю прокручивать тело таблицы, а заголовок и нижний колонтитул раздела будут оставаться на экране. При печати большой таблицы он будет выводить заголовки и нижние колонтитулы разделов на каждом листе.

Использовать эти элементы не трудно. Разделы **header** (заголовок) и **footer** (нижний колонтитул) включаются в начало таблицы, после чего определяется тело раздела. Поскольку разделы заголовка и нижнего колонтитула следуют первыми, браузеры могут выводить таблицу на экран, даже если она еще загружается. Для заголовков, нижних колонтитулов и содержимого разделов таблицы применяются тэги **end**, хотя в данной спецификации они не обязательны.

Использование элементов COLGROUP и COL

```
//Установка ширины каждой колонки отдельно
<COLGROUP>
<COL width="100">
<COL width="100">
<COL width="100">
<COL width="75">
<COL width="75">
<COL width="75">
</COLGROUP>
//Использование элементов span для сокращения кода
<COLGROUP>
<COL span="3" width="100">
<COL span="3" width="75">
</COLGROUP>
```

```
//Назначение ширины колонки и промежутка
//непосредственно в группах колонок
<COLGROUP span="3" width="100">
</COLGROUP>
<COLGROUP span="3" width="75">
</COLGROUP></xmp>
Применение заголовков и нижних колонтитулов помогает браузеру
структурировать таблицу</P>
<xmp>
<TABLE BORDER=2 WIDTH=50%>
<COLGROUP>
<COL width="100">
<COL width="75">
<COL width="75">
<COL width="75">
</COLGROUP>
<THEAD>
<TR> <TH> </TH> <TH>1996</TH> <TH>1997</TH> <TH>1998</TH> </TR>
</THEAD>
<TFOOT>
<TR><TD>Данные предоставлены компанией только для статистических
целей.</TD></TR>
</TFOOT>
<TBODY>
<TR> <TH>Неделя 1</TH> <TD>252</TD> <TD>267</TD> <TD>289</TD>
</TR>
<TR> <TH>Неделя 2</TH> <TD>194</TD> <TD>197</TD> <TD>205</TD>
</TR>
<TR> <TH>Неделя 3</TH> <TD>212</TD> <TD>225</TD> <TD>234</TD>
</TR>
<TR> <TH>Неделя 4</TH> <TD>145</TD> <TD>176</TD> <TD>179</TD>
</TR>
<TR> <TH>Неделя 5</TH> <TD>167</TD> <TD>182</TD> <TD>193</TD>
</TR>
<TR> <TH>Неделя 6</TH> <TD>185</TD> <TD>201</TD> <TD>205</TD>
</TR>
<TR> <TH>Неделя 7</TH> <TD>197</TD> <TD>207</TD> <TD>213</TD>
</TR>
. . .
<TR> <TH>Неделя 52</TH> <TD>203</TD> <TD>221</TD> <TD>279</TD>
</TR>
</TBODY>
</TABLE>
```

Формы

Если вы работали с предыдущими версиями HTML, то знаете, что формы с момента их появления в этом языке изменились незначительно,

их нельзя было назвать «дружественными пользователю», и они уже давно нуждались в усовершенствовании. На этот раз спецификация HTML устраняет, наконец, их наиболее очевидные недостатки.

Теперь формы имеют немало новых элементов, но давайте сосредоточимся на некоторых наиболее полезных из них: табличных индексах и клавишах доступа. Табличные индексы позволяют легко перемещаться (в порядке табуляции) по полям формы с помощью клавиатуры. Это делается путем спецификации порядка табуляции для каждого элемента формы, что позволяет использовать клавишу **Tab** для прямого и обратного перемещения по полям формы. В следующем примере с помощью атрибута **tabindex** задается последовательный порядок табуляции.

```
<form action="cgi-bin/data.pl" method="post">
<P>Name: <input tabindex="1" type="text" name="userName"></P>
<P>Email: <input tabindex="2" type="text" name="userEmail"></P>
<P>Phone: <input tabindex="3" type="text" name="userPhone"></P>
<P><input tabindex="4" type="submit">
<input tabindex="5" type="reset"></P>
</form>
```

Когда пользователь нажимает на клавишу **Tab**, он попадает сначала в поле **userName**, затем в поле **userEmail** и т.д. Порядок табуляции может включать в себя любое число от 0 до 32767. Браузеры используют числовое значение для определения следующего или предыдущего поля. Некоторым полям можно также присваивать и клавиши доступа. Клавиши доступа позволяют быстро перемещаться на определенные поля в форме. Например, если полю **userName** в предыдущем примере присвоена клавиша **N**, то, нажав ее, можно сразу оказаться в этом поле. Работа клавиш доступа зависит от операционной системы. В системах Microsoft Windows, кроме клавиши доступа, обычно нужно нажимать на клавишу **Alt**. Таким образом, чтобы попасть в поле **userName** в системе Windows, нужно нажать **Alt-N**. Браузеры должны выводить клавишу доступа на экран в поле ввода, причем каким-то уникальным способом, например, подчеркнутым или жирным шрифтом.

Присвоить клавишу доступа можно с помощью атрибута **accesskey**, но лучше помочь браузеру определить метод визуализации клавиши доступа, пометив поля ввода и присвоив этой метке клавишу доступа. Метки — новое средство HTML 4.0. Помечая поле формы, вы присоединяете к нему информацию. Для идентификации соответствующего поля формы метки используют атрибут **for**. Значение данного атрибута должно соответствовать значению атрибута **id** в этом поле.

В следующем примере показаны метки с текстовыми полями ввода:

```
<form action="cgi-bin/data.pl" method="post">
<label for="name" accesskey="N">Name: </label>
<input id="name" type="text">
<label for="email" accesskey="E">Email: </label>
<input id="email" type="text">
<label for="phone" accesskey="P">Phone: </label>
<input id="phone" type="text">
<P><input accesskey="S" type="submit"> <input accesskey="R"
type="reset"></P>
</form>
```

Еще одно полезное усовершенствование: элемент **button**. Он создает нажимаемую экранную кнопку, аналогичную кнопкам **Reset** и **Submit** в формах. Между тем, к этим новым кнопкам можно добавить содержимое (чего стандартные экранные кнопки не допускали). Таким образом, новые кнопки теперь могут содержать изображения, текст и другие навороты. Чтобы добавить их, достаточно вставить «наполнение» между открывающим тэгом **<button>** и закрывающим тэгом **</button>**. Воспользовавшись сказанным, в предыдущем примере стандартные кнопки **Submit** и **Reset** можно заменить элементом **button**. Добавление текста и изображений позволит сделать эти элементы более привлекательными:

```
. . .
<button tabindex="4" accesskey="S" name="submit" type="submit">
Enter your name in the database! <img src="go.gif" alt="Submit
it!">
</button>
<button tabindex="5" accesskey="R" name="reset" type="reset">
Start over! <img src="ohno.gif" alt="Try again!">
</button>
```

Не забывайте о том, что старые браузеры не могут использовать элемент **button**. Вместо нажимаемой экранной кнопки вы увидите только ее содержимое. Преодолеть этот недостаток можно включением в элемент **button** полей **submit** и **reset**:

```
<button name="submit" type="submit">
<input type="submit"></button>
<button name="reset" type="reset">
<input type="reset"></button>
```


Сценарии

Сценарии — ключевое средство интерактивного взаимодействия в Web, однако в спецификации HTML механизмы их создания не были определены достаточно хорошо. В результате браузеры обрабатывают сценарии во многом несогласованно. К счастью, в HTML 4.0 реализован более полный подход.

Одно из наиболее важных изменений заключается в спецификации сценариев на web-страницах. Элемент **meta** позволяет определить заданный по умолчанию язык сценариев для всех сценариев страницы. Это делается с помощью спецификации значения в заголовке **Conte** заданное по умолчанию значение, браузер попытается извлечь его из соответствующего поля фактического заголовка HTTP, установленного web-сервером. После определения языка сценария для сценариев указывается тип содержимого MIME. Чаще всего типом содержимого является сценарий VBScript или text/javascript (сценарий JavaScript), однако можно использовать и другие допустимые типы, такие, как text/tcl (сценарий TCL).

Чтобы указать в качестве заданного по умолчанию языка JavaScript, используйте выражение:

```
<meta http-equiv="Content-Script-Type" content="text/javascript">
```

С помощью элемента **script** можно переопределить используемый по умолчанию язык сценариев в любом месте страницы, однако атрибут **language** в данной спецификации HTML применять не рекомендуется. Вместо этого нужно использовать атрибут **type**, позволяющий задать тип содержимого MIME для сценариев, например,

```
<script type="text/vbscript" >
```

Между тем, не следует забывать, что при указании атрибута **type** некоторые старые браузеры не смогут правильно обрабатывать ваши сценарии. Пока все это урегулируется, нужно иметь в виду, что, хотя атрибут **type** вполне законен, он не является обязательным.

Еще одно важное изменение, касающееся сценариев, состоит в том, что большинство элементов HTML теперь поддерживают широкий спектр атрибутов событий. Событие происходит автоматически при определенном условии. Некоторые события вызываются действием пользователя (например, нажатием клавиши или щелчком кнопкой мыши), другие — браузерами (например, когда браузер завершает загрузку изображения). Атрибуты событий позволяют задать условия, при которых должен обнаруживаться элемент. Например, можно создать в таблице ячейки, активизируемые щелчком мыши, абзацы текста, выделяемые при перемещении на них курсора, и многое другое.

Существует несколько способов работы с событиями. Для этого удобно использовать такие атрибуты событий, как **onclick** и **onkeypress**. Они позволяют распознать событие и передать его функции в сценарии. Следующий пример показывает, как использовать атрибут события **onmouseover** с тэгом:

```
<a href="main.html" onmouseover="show('Посетите нашу адресную
страницу')">Main</a>
```

Когда указатель мыши перемещается на фиксированный текст, вызывается функция **show()**, и этот текст передается ей. Как вы обнаружите на практике, лучше всего использовать атрибуты именно так, когда одна и та же функция вызывается многократно.

Если применяются события, специфические для HTML 4.0, то полезно проверить браузер на совместимость. Обычно несовместимые браузеры игнорируют атрибуты событий, которых они не понимают, и в результате ожидаемая вами функция поддерживаться не будет. Проектируя web-страницу с учетом этой особенности, можно постараться избежать подобных проблем.

Таблицы стилей

Таблицы стилей (CSS, cascading style sheets) предусматривают инструменты, необходимые для создания привлекательных web-страниц. Они позволяют управлять цветом текста и фона и позиционировать содержательные материалы, а также предусматривают множество других интересных функций. До появления таблиц стилей для публикации web-страниц с подобными свойствами приходилось полагаться на существующие структуры HTML. Чтобы изменить фоновый цвет страницы, применялся атрибут **bgcolor** элемента **body**. Если необходимо от центрировать текст на странице — элемент **center** и т.д.

К сожалению, когда информация о представлении документа комбинируется с его содержательными материалами, получается излишне сложный и трудно обслуживаемый документ. Данное важное нововведение направлено на постепенное сокращение презентационных элементов и атрибутов в HTML-документах, поскольку таблицы стилей позволяют отделить представление документа от его фактического содержимого. Элементы и атрибуты, которые могут исчезнуть из спецификации HTML, помечаются как выходящие из употребления. Один из таких элементов — **font**, применяемый для указания цвета текста, его гарнитуры и размера шрифта. В числе других **basefont** (для задания информации о назначаемом по умолчанию шрифте), **center** (центрирующий объекты на странице) и несколько других элементов, таких, как **u** (подчеркивание) и **s** (перечеркивание текста).

Кроме включения элементов в список выходящих из употребления, некоторые элементы в данной спецификации объявлены устаревшими, а другие — не рекомендуемыми к использованию. Устаревшие элементы отсутствуют в спецификации HTML, и нет никакой гарантии, что браузер будет их поддерживать. Хотя ясно, что устаревшие элементы использовать не следует, не вполне понятно, что такое не рекомендуемые элементы, отсутствующие в списке устаревших и выходящих из употребления элементов. Спецификация просто не поощряет их применение.

В число устаревших элементов входят **listing** (коды листингов), **plaintext** (листинг простого текста) и **xmp** (examples). Вместо них следует применять элемент **pre**, позволяющий использовать заранее форматированный текст. Не рекомендуются такие элементы, как **big** (крупный текст), **small** (мелкий текст), **tt** (моноширинный шрифт), **i** (курсив), **b** (жирный). Хотя эти элементы отсутствуют в списке выходящих из употребления, спецификация советует вместо них применять таблицы стилей.

Список элементов, выходящих из употребления, довольно обширен. Например, в него попали атрибуты **background**, **bgcolor**, **link**, **text**, **alink** и **vlink** тэга **body**. В числе таких элементов **border** (для изображений и объектов), **clear** (разрыв строки), **noshade** (горизонтальные линейки). Этот список можно продолжить. В общем, если вам хочется применить атрибут для представления информации, то, вероятно, лучше воспользоваться вместо него таблицей стилей.

Определения типов документов

Разработчики HTML 4.0 тщательно продумали вопрос поддержки элементов и атрибутов определения типов документов (Document Type Definition, DTD). Они устанавливают правила и определяют структуры, которые можно использовать в совместимых документах. В HTML 4.0 определены три DTD: строгое, свободное и фреймовое.

Спецификация HTML 4.0 требует, чтобы для web-страницы было задано одно из этих определений. Для этого в первую строку описания типа документа включается DTD. Если вы все же предпочтете не задавать DTD, то имейте в виду, что совместимые с HTML 4.0 браузеры по умолчанию используют строгое определение DTD.

Основная цель строгого определения DTD состоит в том, чтобы отделить визуальное оформление от фактического контента. Это делается с помощью таблиц стилей, управляющих представлением web-страниц. Строгое определение DTD не включает в себя выходящих из употребления элементов/атрибутов или структур, применяемых для со-

здания фреймов. Таким образом, это наиболее ограничивающее определение DTD. Если ваш браузер совместим с HTML 4.0, и нужно протестировать web-страницу со строгим DTD, воспользуйтесь следующим описанием типа:

```
<!doctype HTML public "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
```

Свободное определение DTD не ограничивает применения элементов или атрибутов для представления документа и рассматривается как промежуточный этап (поэтому его называют также переходным определением DTD). В этом определении можно использовать любой выходящий из употребления элемент/атрибут. Для проверки web-страницы с таким определением включите в нее следующее описание:

```
<!doctype HTML public "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
```

Фреймовое определение DTD предназначено для web-страниц с фреймами. Эта версия поддерживает все структуры, допустимые в свободном определении DTD, а также структуры, необходимые для фреймов. Для спецификации такого DTD используйте описание:

```
<!doctype HTML public "-//W3C//DTD HTML 4.0 Frameset//EN"
"http://www.w3.org/TR/REC-html40/frameset.dtd">
```

Глава 13. Тэги

Тэг **<A>** является основным тэгом гипертекста. Выделенный этим тэгом текст является либо началом (направлением), либо местом назначения гипертекстовой ссылки. Обязательными атрибутами данного тэга являются только **NAME** или **HREF**.

Атрибуты:

HREF

Если присутствует атрибут **HREF**, тогда текст заключенный в тэг **<A>** является гипертекстовой ссылкой. Если пользователь выберет данную ссылку, ему будет показан документ, адрес (URL) которого указан в атрибуте **HREF**.

Адрес может быть расширен с помощью конструкции **адрес#имя якоря**. В этом случае данная ссылка будет указывать на другой тэг **<A>**, атрибут **NAME** которого принимает значение имя якоря. В некоторых программах просмотра этот способ работает только внутри одного документа.

Существуют несколько видов адресов (префиксов адресов) поддерживаемых программами просмотра, например, следующие:

<A HREF="http://:"> — ссылка на другой документ WWW.

<A HREF="ftp://:"> — ссылка на FTP сайт. В большинстве случаев подключение идет к общедоступным сайтам.

<A HREF="mailto::"> — данная ссылка откроет установленную у пользователя по умолчанию почтовую программу (почтовый клиент). Некоторые программы просмотра страниц (NCSA Mosaic) поддерживают атрибут **TITLE** для этого тэга, через который можно передать тему (**subject**) в создаваемое сообщение. Большинство браузеров передают **subject** следующим образом:

```
<A HREF = "mailto:nsolov@mail.ru?subject = The HTML Help is a
greatest book">текст ссылки</A>
```

A MS IE 4 поддерживает возможность указать и другие поля письма:

```
<A HREF = "mailto:nsolov@mail.ru?subject = Подписка на ново-
сти&body = Хочу получать новости с раз в неделю">Текст ссылки</A>
```

Однако следует помнить, что этот механизм может работать не со всеми почтовыми программами.

<A HREF="news::"> — ссылка на группу новостей

<A HREF="view-source://..."> — это специфическая возможность NetScape, позволяет открыть окно просмотра источника (HTML кода) документа.

NAME

Если этот атрибут присутствует, он позволяет использовать этот тэг в качестве цели в другой ссылке. Значение этого атрибута является идентификатором якоря. В качестве идентификатора может использоваться произвольная строка, но она должна быть уникальна в пределах одного документа. Для создания ссылки необходимо использовать: **имя файла#значение NAME**. В некоторых браузерах возможно использование предопределенных расширений — например **#top** указывает на начало документа.

TITLE

Используется только для информационных целей (за исключением совместного использования с **mailto**). Значение данного атрибута будет появляться как всплывающая подсказка, когда пользователь будет задерживать указатель мыши над данным элементом. Работает только в IE.

URN

В настоящее время не используется.

TARGET

Окна программ просмотра могут иметь имена, и ссылки из одного окна могут ссылаться на другие окна по именам. Также именуются и отдельные фреймы, на которые делится окно. Если активизируется ссылка с указанием имени окна, то документ будет открыт в этом окне, а если окно с таким именем еще не существует, то оно будет создано и к нему можно будет обращаться по этому имени.

Атрибут **TARGET** может принимать одно из следующих значений:

- **Window_name** — название окна или фрейма, заданное явно при его создании. Если окна или фрейма с таким названием не существует ссылка будет открыта в новом окне и ему будет присвоено это название, по которому к нему можно будет обращаться в дальнейшем.
- **_self** — открывает ссылку в том же окне
- **_parent** — открывает ссылку в родительском фрейме
- **_top** — открывает ссылку в самом «верхнем» окне.
- **_blank** — открывает ссылку в новом окне, при этом этому окну не присваивается имени как в первом случае.

ACCESSKEY

Данный атрибут может указывать горячую клавишу для доступа к ссылке, т.е. ссылка активируется нажатием комбинации **ALT+ ACCESSKEY**, как во многих стандартных Windows приложениях. Однако указанная буква никак не выделятся в тексте ссылки, вам нужно сделать это самостоятельно, например так:

```
<A HREF="new.htm" ACCESSKEY="W">W<SPAN STYLE="
{text-decoration:none}">hat's New</SPAN></A>
```

К сожалению, при использовании кириллицы данный атрибут не работает. Поддерживается только IE.

LANGUAGE

Атрибут **LANGUAGE** принимает значения **Javascript** или **Vbscript**, чтобы указать какой язык сценариев использовать в данном тэге. Этот атрибут полезно использовать, если у вас есть несколько обработчиков событий связанных с данным тэгом.

Таким образом, запись

```
<A HREF="url.htm" onclick="javascript:return false">Link text</A>
```

и запись

```
<A HREF="url.htm" LANGUAGE="Javascript" onclick="return
false">Link text</A>
```

полностью эквивалентны. Кстати эта конструкция подавляет переход по указанной ссылке.

LANG

Используется для указания, какой язык использует данный тэг **<A>**. Может принимать любое значение из стандартных аббревиатур ISO.

CLASS

Используется для указания какой класс предварительно определенной таблицы стилей (CSS) используется.

ID

Может использоваться как уникальный идентификатор ссылки в сценариях либо в таблицах стилей.

STYLE

Применяется в случае использовании встроенных стилей, например:

```
<A STYLE = "color : #CC00CC" HREF = "http://www.microsoft.com">
http://www.microsoft.com/ </A>
```

Другие атрибуты

REL
REV
METHOD
DATAFLD
DATASRC

Глава 14. Тэг <A>, использование в сценариях

Свойства тэга <A>

Каждый объект ссылка (**<A HREF=:>**) поддерживает следующие свойства, поддерживаемые всеми объектами:

- className
- document
- id
- innerHTML
- innerText
- isTextEdit
- lang
- language
- offsetHeight
- offsetLeft
- offsetParent
- offsetTop
- offsetWidth
- outerHTML
- outerText
- parentElement
- parentTextEdit
- sourceIndex
- style
- tagName
- title.

Кроме них поддерживается еще ряд свойств, присущих только объекту **<A HREF=:>**. Доступ ко всем свойствам можно получить как из встроенного перехватчика событий (например **<A HREF=":" onmouseover="Javascript::">**), так и используя ID данного объекта, или через коллекцию объектов **Links**.

Следует помнить, что большинство перечисленных свойств доступны только в модели DHTML браузера IE4. Поддержка различными браузерами указана в скобках в описании каждого свойства.

- **accessKey** — (Internet Explorer 4.0+)
- **datafld** — (Internet Explorer 4.0+)

- **datasrc** — (Internet Explorer 4.0+)
- **hash** — (Internet Explorer 3.0+, Netscape 2.0+)
- **host** — (Internet Explorer 3.0+, Netscape 2.0+)
- **hostname** — (Internet Explorer 3.0+, Netscape 2.0+)
- **href** — (Internet Explorer 3.0+, Netscape 2.0+)
- **Methods** — (Internet Explorer 4.0+)
- **name** — (Internet Explorer 3.0+, Netscape 2.0+)
- **pathname** — (Internet Explorer 3.0+, Netscape 2.0+)
- **port** — (Internet Explorer 3.0+, Netscape 2.0+)
- **protocol** — (Internet Explorer 3.0+, Netscape 2.0+)
- **rel** — (Internet Explorer 4.0+)
- **rev** — (Internet Explorer 4.0+)
- **search** — (Internet Explorer 3.0+, Netscape 2.0+)
- **target** — (Internet Explorer 3.0+, Netscape 2.0+)
- **urn** — (Internet Explorer 4.0+)

Методы тэга <A>

Помимо стандартных методов DHTML, таких как **click**, **contains**, **getAttribute**, **insertAdjacentHTML**, **insertAdjacentText**, **removeAttribute**, **scrollIntoView**, **setAttribute**, объект **<A>** поддерживает также следующие методы:

- **blur** — (Internet Explorer 4.0+)
- **focus** — (Internet Explorer 4.0+)

События тэга <A>

Помимо стандартных событий DHTML, таких как **onclick**, **ondblclick**, **ondragstart**, **onfilterchange**, **onhelp**, **onkeydown**, **onkeypress**, **onkeyup**, **onmousedown**, **onmousemove**, **onmouseout**, **onmouseover**, **onmouseup**, **onselectstart**, объект **<A>** поддерживает следующие события:

- **onblur** — (Internet Explorer 4.0+)
- **onfocus** — (Internet Explorer 4.0+)

Глава 15.
Создание документов в формате HTML 4.0

Авторам и разработчикам для работы с HTML 4.0 рекомендуется ознакомиться со следующими общими принципами.

Разделение структуры и представления

HTML происходит из SGML, который всегда был языком определения структурной разметки. По мере развития HTML все большее количество его элементов и атрибутов для представления заменяется другими механизмами, в частности, таблицами стилей. Опыт показывает, что отделение структуры документа от аспектов его представления снижает стоимость обслуживания широкого диапазона платформ, носителей и т.д. и упрощает изменение документов.

Универсальность доступа к Web

Чтобы сделать свой Web-сервер доступным для всех пользователей, особенно для пользователей с физическими недостатками, авторы должны предполагать, как их документы могут отображаться на различных платформах: речевых браузерах, программах чтения азбуки Бройля и т.д. Мы не рекомендуем авторам ограничивать творческий процесс, но рекомендуем предусматривать альтернативные методы подачи информации. HTML предлагает ряд таких механизмов (например, атрибут **alt**, атрибут **accesskey** и т.д.)

Авторам также следует иметь в виду, что к их документам могут обращаться пользователи с другой конфигурацией компьютеров. Для корректной интерпретации документов авторам следует включать в свои документы информацию о языке и направлении письма в тексте, о кодировке документа и прочую подобную информацию.

Помощь в последовательном создании изображений

При тщательной разработке таблиц и использовании новых возможностей HTML 4.0 авторы могут ускорить отображение документов браузером. Авторы могут прочесть здесь о том, как создавать таблицы для последовательного представления (элемент **TABLE**). Разработчики могут получить информацию об алгоритмах последовательного представления в замечаниях о таблицах в приложении.

Глава 16.
SGML и HTML

SGML — это система определения языков разметки. Авторы размечают свои документы, представляя информацию о структуре, представлении и семантике в одном документе. HTML является одним из примеров языка разметки. Вот пример документа на языке HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
  "http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
 <HEAD>
  <TITLE>My first HTML document</TITLE>
 </HEAD>
 <BODY>
  <P>Hello world!
 </BODY>
</HTML>
```

Документ HTML состоит из *раздела заголовка* (здесь — между тэгами **<HEAD>** и **</HEAD>**) и *тела* (здесь — между заголовками **<BODY>** и **</BODY>**). Заголовок документа отображается в заголовке (вместе с другой информацией о документе), а содержимое документа находится в теле. В этом примере тело документа состоит только из одного абзаца, помеченного **<P>**.

Каждый язык разметки, определенный в SGML, называется приложением SGML. Приложение SGML характеризуется:

Объявлением SGML

Объявление указывает, какие символы и разделители могут отображаться в приложении.

Определением типа документа (DTD)

DTD определяет синтаксис конструкций разметки. DTD может включать дополнительные определения, например, **character entity references**.

Спецификацией, описывающей семантику, используемую в разметке

Эта спецификация также налагает синтаксические ограничения, которые невозможно выразить при помощи DTD. Экземпляры документа содержат данные (содержимое) и разметку. Каждый экземпляр содержит ссылку на DTD, которое должно использоваться для интерпретации.

Спецификация HTML 4.0 включает объявление SGML, три определения типа документов и список character references.

Конструкции SGML, используемые в HTML

Элементы

Определение типа документа SGML объявляет типы элементов, представляющие структуры или желательное поведение. HTML включает типы элементов, представляющие абзацы, гипертекстовые ссылки, списки, таблицы, изображения и т.д.

Каждое объявление типа элемента обычно включает три части: начальный тэг, содержимое и конечный тэг.

Имя элемента отображается в начальном тэге (пишется <имя-элемента>) и в конечном тэге (пишется </имя-элемента>); не забывайте про слэш перед именем элемента в конечном тэге. Например, начальные и конечные тэги элемента **UL** определяют список:

```
<UL>
<LI><P>...элемент списка 1...
<LI><P>...элемент списка 2...
</UL>
```

Некоторые типы элементов HTML позволяют авторам опускать конечные тэги (например, типы элементов **P** and **LI**). Несколько типов элементов также позволяют опускать начальные тэги; например, **HEAD** и **BODY**. HTML DTD указывает для каждого типа элемента, являются ли начальный и конечный тэги обязательными.

Некоторые типы элементов HTML не имеют содержимого. Например, элемент перехода на следующую строку **BR** не имеет содержимого; его роль — прерывание строки текста. Такие пустые элементы никогда не имеют конечных тэгов. Определение типа документа и текст спецификации указывают, является ли тип элемента пустым (не имеет содержимого) или, если он может иметь содержимое, что является допустимым содержимым.

Имена элементов всегда учитывают регистр. Информацию о правилах, управляющих элементами, (например, что они могут быть вложенными соответствующим образом, конечный тэг закрывает все опущенные начальные тэги вплоть до соответствующего ему начального тэга находятся в стандарте SGML.

Например, следующий абзац:

```
<P>Это первый абзац.</P>
...элемент блока...
```


можно перезаписать без конечного тэга:

```
<P>Это первый абзац.
...элемент блока...
```

поскольку начальный тэг **<P>** закрывается следующим элементом блока. Точно так же, если абзац включен в элемент блока, например:

```
<DIV>
<P>Это абзац.
</DIV>
```

конечный тэг включающего элемента блока (здесь — **</DIV>**) служит также конечным тэгом открытого начального тэга **<P>**.

Элементы — это не тэги. Некоторые люди называют элементы тэгами. Помните, что элемент — это одно, а тэг (начала или конца, неважно) — другое. Например, элемент **HEAD** всегда присутствует, даже если начальный и конечный тэги **HEAD** отсутствуют.

Атрибуты

С элементами могут быть связаны свойства, называемые атрибутами, которые могут иметь значения (стандартные или устанавливаемые авторами или сценариями). Пары атрибут/значение помещаются перед закрывающей скобкой «>» начального тэга элемента. В начальном тэге элемента может быть любое число (допустимых) пар атрибут/значение, разделенных пробелами. Они могут указываться в любом порядке.

В данном примере для элемента **H1** установлен атрибут **id**:

```
<H1 id="section1">
Это определенный заголовок, спасибо атрибуту id
</H1>
```

По умолчанию в SGML необходимо, чтобы все значения атрибутов были разделены с помощью двойных (десятичный код ASCII 34) или одинарных кавычек (десятичный код ASCII 39). Одинарные кавычки могут включаться в значение атрибута, если значение отделяется двойными кавычками, и наоборот. Авторы могут также использовать цифровые ссылки на символы для представления двойных (") и одинарных кавычек ('). Для двойных кавычек авторы могут также использовать **character entity reference "**.

В определенных случаях авторы могут указывать значение атрибута без кавычек. Значение атрибута может включать только буквы (a-z и A-Z), цифры (0-9), знаки переноса (десятичный код ASCII 45) и точки (десятичный код ASCII 46). Рекомендуется всегда использовать кавычки.

Значения атрибутов обычно учитывают регистр. Определение каждого атрибута в списке атрибутов указывается, учитывает ли значение регистр.

Ссылки на символы

Ссылки на символы — это числовые или символьные имена символов, которые могут быть включены в документ HTML. Они удобны для обращения к редко используемым символам или к символам, которые трудно или невозможно вводить в средствах разработки документов. Ссылки на символы начинаются со знака «**&**» и заканчиваются точкой с запятой «**;**». Вот некоторые примеры:

```
"&lt;"
```

представляет знак <.

```
"&gt;"
```

представляет знак >.

```
"&quot;"
```

представляет знак ".

```
"&#229;"
```

(десятичное число) представляет букву @.

```
"&#1048;"
```

(десятичное число) представляет кириллическую букву «I».

```
"&#x6C34;"
```

(шестнадцатеричное число) представляет китайский знак воды.

Комментарии

Комментарии в HTML имеют следующий синтаксис:

```
<!-- это комментарий -->
<!-- это тоже комментарий,
 он занимает несколько строк -->
```

Проблемы между открывающим разделителем разметки (<!) и открывающим разделителем комментария (--) недопустимы, но их можно использовать между закрывающим разделителем комментария (--) и закрывающим разделителем разметки (>). Распространенной ошибкой является включение строки символов переноса (---) в комментарий. Следует избегать использования в комментариях двух или более символов переноса.

Информация в комментариях не имеет специального значения (например, ссылки на символы не интерпретируются).

Глава 17. Как читать HTML DTD

Каждое объявление элемента и атрибута в этой спецификации сопровождается фрагментом его определения типа документа. Мы решили включить фрагменты DTD в спецификацию вместо того, чтобы искать более доступные, но более длинные и менее точные средства описания свойств элементов.

Комментарии DTD

В DTD комментарии могут занимать несколько строк. В DTD комментарии отделяются парой меток «--», например,

```
<!ELEMENT PARAM - O EMPTY  -- значение именованного свойства -->
```

Здесь комментарий «значение именованного свойства» объясняет использование типа элемента **PARAM**. Комментарии в DTD носят информативный характер.

Определения Parameter entity

HTML DTD начинается с ряда определений **parameter entity**. Определение **parameter entity** определяет макрос, на который можно ссылаться в любом месте DTD. Эти макросы не отображаются в документах HTML, они отображаются только в DTD. Другие типы макросов, называемые ссылками на символы, могут использоваться в тексте документа в формате HTML или в значениях атрибутов.

Когда на **parameter entity** ссылаются в DTD по имени, он разворачивается в строку.

Определение parameter entity начинается с ключевого слова **<!ENTITY %**, за которым следует имя **entity**, строка в кавычках, в которую разворачивается **entity** и наконец закрывающий **>**. Экземпляры **parameter entities** в DTD начинаются со знака «**%**», затем идет имя **parameter entity** и заканчивается необязательным символом «**;**».

В следующем примере определяется, в какую строку будет разворачиваться «**%fontstyle;**».

```
<!ENTITY % fontstyle "TT | I | B | BIG | SMALL">
```

Строка, в которую разворачивается **parameter entity**, может содержать другие имена **parameter entity**. Эти имена разворачиваются рекурсивно. В следующем примере «**%inline;**» **parameter entity** включает «**%fontstyle;**», «**%phrase;**», «**%special;**» и «**%formctrl;**» **parameter entities**.

```
<!ENTITY % inline "#PCDATA | %fontstyle; | %phrase; | %special; |
%formctrl;">
```

Вы часто будете встречать в HTML DTD два DTD **entities**: «**%block;**» «**%inline;**». Они используются, если модель содержимого включает **block-level** и **inline elements** соответственно.

Объявления элементов

HTML DTD состоит из объявлений типов элементов и их атрибутов. Объявление начинается с ключевого слова **<!ELEMENT** и заканчивается символом >. Между ними указываются:

Имя элемента

Если конечный тэг элемента необязателен Два символа переноса после имени элемента означают, что начальный и конечный тэги являются обязательными. Один символ переноса, за которым следует буква «**O**», указывает, что конечный тэг можно опустить. Две буквы «O» указывают на то, что можно опустить как начальный, так и конечный тэги.

Содержимое элемента

Если таковое имеется. Допустимым содержимым элемента называется его модель содержимого. Типы элементов, не имеющие содержимого, называются пустыми элементами. Модель содержимого для таких типов элементов объявляется при помощи ключевого слова «**EMPTY**».

В этом примере:

```
<!ELEMENT UL - - (LI)+>
```

объявляется тип элемента UL.

Два знака переноса указывают, что начальный тэг **<UL>** и конечный тэг **</UL>** для этого элемента обязательны.

Модель содержимого для типа элемента

По крайней мере один элемент **LI**. В примере показано объявление пустого типа элемента — объявляется тип элемента IMG:

```
<!ELEMENT IMG - O EMPTY>
```

Знак переноса, за которым следует буква «**O**», указывает, что конечный тэг можно опустить, но если модель содержимого — это «**EMPTY**», это правило усиливается, и конечный тэг должен быть опущен.

Ключевое слово «**EMPTY**» означает, что экземпляры этого типа не должны иметь содержимого.

Определения модели содержимого

Модель содержимого описывает, что может содержаться в экземпляре типа элемента. Определения модели содержимого могут включать:

- Имена допустимых или запрещенных типов элементов (например, элемент **UL** содержит экземпляры типа элемента **LI**, а тип элемента **P** не может включать другие элементы **P**).
- DTD **entities** (например, элемент **LABEL** включают экземпляры «**%inline;**» **parameter entity**).
- Текст документа (указываемый SGML-конструкцией «**#PCDATA**»). Текст может включать ссылки на символы. Помните, что они начинаются с **&** и заканчиваются точкой с запятой (например, «**Hergé's adventures of Tintin**» содержит ссылку на **character entity** для символа «е со знаком ударения»).

Модель содержимого элемента указывается с использованием следующего синтаксиса:

```
( ... )
```

Разделяет группы.

```
A | B
```

Происходит A или B, но не оба.

```
A , B
```

Происходят A и B в указанном порядке.

```
A & B
```

Происходят A и B в любом порядке.

```
A?
```

A происходит ноль или один раз.

```
A*
```

A происходит ноль или более раз.

```
A+
```

A происходит один или несколько раз.

Вот некоторые примеры HTML DTD:

```
<!ELEMENT UL - - (LI)+>
```

Элемент **UL** должен содержать один или несколько элементов **LI**.

```
<!ELEMENT DL - - (DT|DD)+>
```

Элемент **DL** должен содержать один или несколько элементов **DT** или **DD** в любом порядке.

```
<!ELEMENT OPTION - O (#PCDATA)>
```

Элемент **OPTION** может содержать только текст и такие **entities** как **&** — это определяется типом данных SGML **#PCDATA**.

Некоторые типы элементов HTML используют дополнительную функцию SGML для исключения элементов из модели содержимого. Исключенным элементам предшествует символ переноса. Явные исключения имеют приоритет по отношению к допустимым элементам.

Ниже в примере **-(A)** означает, что элемент **A** не может находиться в другом элементе **A** (то есть ссылки не могут быть вложенными).

```
<!ELEMENT A - - (%inline;)* -(A)>
```

Помните, что тип элемента **A** является частью DTD **parameter entity «%inline;»**, но явно исключается выражением **-(A)**. Точно так же следующее объявление типа элемента **FORM** запрещает вложенные формы:

```
<!ELEMENT FORM - - (%block;|SCRIPT)+ -(FORM)>
```

Объявления атрибутов

Объявление атрибутов, которые может иметь элемент, начинается с ключевого слова **<!ATTLIST**. За ним следует имя элемента **in question**, список определений атрибутов и закрывающий символ >. Каждое определение атрибута — это тройка, определяющая:

Имя атрибута

Тип значения атрибута или явный набор допустимых значений. Значения, определенные явным образом при помощи DTD, учитывают регистр. Является ли значение атрибута по умолчанию неявным (ключевое слово «**#IMPLIED**»), то есть значение по умолчанию устанавливается браузером (в некоторых случаях с использованием наследования от родительских элементов); всегда обязательным (ключевое слово «**#REQUIRED**»); или фиксированным данным значением (ключевое слово «#FIXED»). Некоторые определения атрибутов явным образом указывают значение атрибута по умолчанию.

В следующем примере атрибут **имя** определен для элемента **MAP**. Атрибут на является обязательным для этого элемента.

```
<!ATTLIST MAP
 name  CDATA  #IMPLIED
 >
```


Тип значений, допустимых для этого атрибута, дается как **CDATA**, тип данных SGML. **CDATA** — это текст, который может содержать ссылки на символы.

В следующих примерах показано несколько определений атрибутов:

```
rowspan  NUMBER  1   -- number of rows spanned by cell --
http-equiv NAME  #IMPLIED -- HTTP response header name --
id   ID   #IMPLIED -- document-wide unique id --
valign  (top|middle|bottom|baseline) #IMPLIED
```

Для атрибута **rowspan** необходимы значения типа **NUMBER**. Значение по умолчанию дается явно — «**1**». Для необязательного атрибута **http-equiv** необходимы значения типа **NAME**. Для необязательного атрибута **id** необходимы значения типа **ID**. Необязательный атрибут **valign** ограничен значениями из набора **{top, middle, bottom, baseline}**.

DTD entities в определениях атрибутов

Определения атрибутов могут также содержать ссылки на **parameter entity**.

В примере мы видим, что список определений атрибутов для элемента **LINK** начинается с «**%attrs;**» **parameter entity**.

```
<!ELEMENT LINK - O EMPTY    -- a media-independent link -->
<!ATTLIST LINK
 %attrs;         -- %coreattrs, %i18n, %events --
 charset  %Charset;  #IMPLIED -- char encoding of linked resource
--
 href  %URI;   #IMPLIED -- URI for linked resource --
 hreflang %LanguageCode; #IMPLIED -- language code --
 type  %ContentType; #IMPLIED -- advisory content type --
 rel   %LinkTypes; #IMPLIED -- forward link types --
 rev   %LinkTypes; #IMPLIED -- reverse link types --
 media  %MediaDesc; #IMPLIED -- for rendering on these media --
 >
Start tag: required, End tag: forbidden
```

«**%attrs;**» **parameter entity** определен следующим образом:

```
<!ENTITY % attrs "%coreattrs; %i18n; %events;">
```

«**%coreattrs;**» **parameter entity** в определении «**%attrs;**» разворачивается следующим образом:

```
<!ENTITY % coreattrs
 "id   ID    #IMPLIED -- document-wide unique id --
 class  CDATA   #IMPLIED -- space separated list of classes --
```

```
 style  %StyleSheet; #IMPLIED -- associated style info --
 title  %Text;   #IMPLIED -- advisory title/amplification --"
 >
```

The «**%attrs;**» **parameter entity** определен для удобства, поскольку эти атрибуты определены для большинства типов элементов HTML.

Таким же образом DTD определяет «**%URI;**» **parameter entity** как расширение строки «**CDATA**».

```
<!ENTITY % URI "CDATA"
 -- a Uniform Resource Identifier,
  see [URI]
 -->
```

Как показано в этом примере, **parameter entity** «**%URI;**» предоставляет читателям DTD больше информации, чем для типа данных, ожидаемого для этого атрибута. Похожие **entities** определены для «**%Color;**», «**%Charset;**», «**%Length;**», «**%Pixels;**» и т.д.

Boolean attributes

Некоторые атрибуты играют роль булевых переменных (например, атрибут **selected** для элемента **OPTION**). Их наличие в начальном тэге элемента подразумевает, что значением атрибута является «истина». Их отсутствие означает «ложь».

Булевы атрибуты могут принимать только одно значение: собственно имя атрибута (например, **selected="selected"**).

В этом примере атрибут **selected** определяется как булев.

```
selected  (selected) #IMPLIED -- reduced inter-item spacing --
```

Для атрибута устанавливается значение «истина», поскольку он находится в начальном тэге элемента:

```
<OPTION selected="selected">
...contents...
<OPTION>
```

В HTML булевы атрибуты могут быть в минимизированной форме — в начальном тэге элемента находится только значение атрибута. Таким образом, **selected** можно установить, написав:

```
<OPTION selected>
```

вместо:

```
<OPTION selected="selected">
```

Авторам следует знать, что многие браузеры распознают только минимизированную форму булевых атрибутов и не распознают полную.

Глава 18. Представление документа в формате HTML

Для обеспечения возможности взаимодействия сетей SGML требует от каждого приложения (включая HTML) указания набора символов документа.

Документ включает:

Репертуар

Набор абстрактных символов, таких как латинская буква «A», кириллическая буква «I», китайский иероглиф «вода» и т.д.

Коды

Набор целочисленных ссылок на символы репертуара.

Каждый документ SGML (включая каждый документ HTML) — это последовательность символов из репертуара. Компьютерные системы идентифицируют каждый символ по его коду; например, в наборе символов ASCII коды 65, 66 и 67 означают символы «A», «B» и «C» соответственно.

Набора символов ASCII недостаточно для такой глобальной информационной системы, как Web, поэтому HTML использует более полный набор символов, называемый Универсальным набором символов (Universal Character Set — UCS). Этот стандарт определяет репертуар тысяч символов, используемых во всем мире.

Набор символов — это посимвольный эквивалент Unicode 2.0. Оба эти стандарта время от времени обновляются, пополняются новыми символами, об изменениях следует узнавать на соответствующих серверах Web. О спецификации HTML Unicode также упоминается при обсуждении других вопросов, таких как алгоритм двунаправленного текста.

Набора символов документа, однако, недостаточно, чтобы браузеры могли корректно интерпретировать документы HTML при типичном обмене — закодированные как последовательность байт в файле или во время передачи по сети. Браузеры должны также знать кодировки символов, которые использовались для преобразования потока символов документа в поток байт.

Кодировки символов

Кодировки символов в ряде спецификаций имеют другие названия (что может вызвать некоторую путаницу). Однако это понятие в Ин-

тернет означает примерно одно и то же. Одно и то же имя — «**charset** — набор символов» — используется в заголовках протоколов, атрибутах и параметрах, ссылающихся на символы и использующих одни и те же значения.

Параметр «**charset**» идентифицирует кодировку символов, которая является способом преобразования последовательности байт в последовательность символов. Это преобразование естественно вписывается в схему деятельности Web: серверы отправляют документы HTML браузерам в виде потока байт; браузеры интерпретируют их как последовательность символов. Способы преобразования могут меняться от простого соответствия один к одному до сложных схем или алгоритмов переключения.

Простой техники кодировки «один байт — один символ» недостаточно для текстовых строк с широким репертуаром символов. Кроме кодировок всего набора символов (например, UCS-4), имеются некоторые другие кодировки частей.

Выбор кодировки

Средства разработки (например, текстовые редакторы) могут кодировать документы HTML в кодировках по своему выбору, и этот выбор существенно зависит от соглашений, используемых системным программным обеспечением. Эти средства могут использовать любую удобную кодировку, включающую большинство символов в документе, при условии, что кодировка корректно помечена. Некоторые символы, не включенные в эту кодировку, можно представить с помощью ссылок на символы. Это всегда относится к набору символов документа, а не к кодировке символов.

Серверы и прокси могут изменять кодировку символов (что называется транскодированием) на лету для выполнения запросов браузерам в заголовке запроса HTTP «**Accept-Charset**». Серверы и прокси не должны обслуживать документ в кодировке, включающей весь набор символов документа.

Широко используемые в Web кодировки:

- ISO-8859-1 (также называется «Latin-1»; используется для большинства западноевропейских языков)
- ISO-8859-5 (с поддержкой кириллицы)
- SHIFT_JIS (японская кодировка)
- EUC-JP (еще одна японская кодировка)

- UTF-8 (вариант кодировки ISO 10646, использующий разное число байт для разных символов).

Названия кодировок символов не учитывают регистр, так что, например, «SHIFT_JIS», «Shift_JIS» и «shift_jis» эквивалентны.

Соответствующие браузеры должны корректно отображать в Unicode все символы в любых кодировках, которые они могут распознавать.

Замечания об определенных кодировках

Когда текст HTML передается в UTF-16 (charset=UTF-16), текстовые данные должны передаваться в сетевом порядке байт («big-endian», байт высшего порядка — первый) в соответствии с ISO10646 и UNICODE.

Более того, чтобы повысить вероятность правильной интерпретации, рекомендуется передавать документы UTF-16, всегда начиная с символа «неразделяющий пробел нулевой ширины» (шестнадцатеричный код FEFF, также называется «Меткой порядка байтов» (Byte Order Mark — BOM)), при обращении байт становится шестнадцатеричным FFFE, никогда не назначаемым символом. Таким образом, браузер, получивший шестнадцатеричный код FFFE в качестве первых байтов текста будет знать, что в остальном тексте байты нужно обратить.

Не следует использовать формат трансформации UTF-1.

Указание кодировки символов

Как сервер определяет, какая кодировка символов применяется в документе? Некоторые серверы проверяют первые несколько байт документа или сверяются с базой данных известных файлов и кодировок. Многие современные серверы Web предоставляют администраторам больше возможностей управления конфигурацией набора символов, чем старые серверы. Администраторы серверов Web должны при возможности использовать следующие механизмы для отправки параметра «**charset**», но должны позаботиться о том, чтобы не установить для документов ошибочное значение параметра «**charset**».

Как браузер узнает, какая использовалась кодировка символов? Эту информацию предоставляет сервер. Лучшим способом проинформировать браузер о кодировке символов документа — использовать параметр «**charset**» в поле заголовка «**Content-Type**» протокола HTTP. Например, следующий заголовок HTTP объявляет, что используется кодировка EUC-JP:

```
Content-Type: text/html; charset=EUC-JP
```

Протокол HTTP считает ISO-8859-1 кодировкой символов по умолчанию, если параметр «**charset**» в поле заголовка «**Content-Type**» отсутствует. На практике эта рекомендация бесполезна, поскольку некоторые серверы не позволяют отправлять параметр «**charset**», а некоторые могут не быть сконфигурированы для отправки этого параметра. Поэтому браузеры не должны предполагать никакого значения параметра «**charset**».

Для указания ограничений сервера или конфигурации документы HTML могут включать явную информацию о кодировке символов документа; для предоставления такой информации браузерам может использоваться элемент **META**.

Например, чтобы указать, что кодировкой символов в текущем документе является EUC-JP, включите следующее объявление **META**:

```
<META http-equiv="Content-Type" content="text/html;
charset=EUC-JP">
```

Объявление **META** должно использоваться, только если кодировка символов упорядочена так, что символы ASCII стоят на своем месте (по крайней мере, при разборе элемента **META**). Объявления **META** должны быть в тексте как можно раньше в элементе **HEAD**.

В случаях, когда ни протокол HTTP, ни элемент **META** не предоставляют информации о кодировке документа, HTML предоставляет атрибут **charset** для некоторых элементов. Объединив все эти механизмы, автор может существенно повысить шансы на то, что, когда пользователь загружает ресурс, браузер распознает кодировку символов.

Подводя итоги, соответствующие браузеры при определении кодировки символов документа (от высшего приоритета к низшему) должны руководствоваться следующими источниками в соответствии с приоритетом:

- Параметр «**charset**» протокола HTTP в поле «**Content-Type**».
- Объявление **META**, в котором для «**http-equiv**» установлено «**Content-Type**» и установлено значение для «**charset**».
- Атрибут **charset** устанавливается на элемент, обозначающий внешний ресурс.

Кроме этого списка приоритетов, браузер может использовать эвристические установки и установки пользователя. Например, многие браузеры используют эвристику для распознавания различных кодировок, используемых для японского языка.

Браузеры обычно имеют определяемую пользователем локальную кодировку по умолчанию, которую они используют, если нет указаний кодировки.

Браузеры могут обеспечивать механизм, позволяющий пользователям изменять некорректную информацию о наборе символов. Однако если браузер предлагает такой механизм, он должен предлагать его только для просмотра, а не для изменения, во избежание создания Web-страниц с некорректным параметром «**charset**».

Если в каком-то приложении нужно использовать символы, не входящие в кодировку, этим символам должна быть назначена персональная зона во избежание конфликтов с настоящей или будущими версиями стандарта. Однако это не рекомендуется из соображений переносимости.

Ссылки на символы

Данная кодировка символов может не содержать все символы из набора символов документа. Для таких кодировок или для таких конфигураций оборудования и программного обеспечения, не позволяющих пользователям вводить определенные символы, авторы могут использовать ссылки на символы SGML. Ссылки на символы — это независимый от кодировки механизм ввода любых символов.

Ссылки на символы в HTML могут принимать две формы:

- Числовые ссылки на символы (десятичные или шестнадцатеричные).
- Ссылки на комбинации символов.

Ссылки на символы в комментариях не имеют значения; они являются только данными комментариев.

HTML обеспечивает другие способы представления символов, в частности, встроенные изображения.

В SGML можно в некоторых случаях не использовать заключительный символ «**;**» после ссылки на символы (например, в символе переноса строки или непосредственно перед тэгом).

В других обстоятельствах их нельзя удалять (например, в середине слова).

Мы предлагаем использовать «**;**» всегда во избежание проблем с браузерами, для которых этот символ обязателен.

Числовые ссылки на символы

Числовые ссылки на символы указывают код символа в наборе символов документа. Числовые ссылки на символы могут также принимать две формы:

- Синтаксис «**&#D;**», где **D** — десятичное число, указывает символ Unicode с десятичным номером D.
- Синтаксис «**&#xH;**» или «**&#XH;**», где H — шестнадцатеричное число, указывает на символ Unicode с шестнадцатеричным номером H. Шестнадцатеричные числовые ссылки учитывают регистр.

Вот некоторые примеры числовых ссылок на символы:

```
&#229;
```

(десятичное) представляет букву «а» с кружком сверху (используемую, например, в норвежском языке).

```
&#xE5;
```

(шестнадцатеричное) представляет тот же символ.

```
&#Xe5;
```

(шестнадцатеричное) представляет тот же символ.

```
&#1048;
```

(десятичное) представляет кириллическую заглавную букву «I».

```
&#x6C34;
```

(шестнадцатеричное) представляет китайский иероглиф «вода».

Комбинации ссылок на символы

Чтобы дать авторам более инициативный способ использования символов, HTML предлагает набор **character entity references**. Комбинации ссылок на символы используют символические имена, так что авторам не придется запоминать коды. Например, комбинация **å** обозначает символ «а» нижнего регистра с кружком сверху; **å** легче запомнить, чем **å**.

HTML 4.0 не определяет **character entity reference** для каждого символа. Например, для кириллической буквы «**I**» нет **character entity reference**.

Комбинации ссылок на символы учитывают регистр. Так, **Å** указывает на другой символ (А с кружком верхнего регистра), а не на **å** (а с кружком нижнего регистра).

Четыре ссылки нужно упомянуть специально, поскольку они часто используются для указания специальных символов:

```
&lt;
```

представляет знак <.

```
&gt;
```

представляет знак >.

```
&amp;
```

представляет символ &.

```
&quot;
```

представляет знак ".

Авторы, которые хотят поместить в текст символ <, должны использовать ссылку **<** (десятичный код ASCII 60) во избежание возможной путаницы с началом тэга (открывающий разделитель начального тэга). Точно так же следует использовать **>** (десятичный код ASCII 62) вместо >, чтобы избежать проблем со старыми версиями браузеров, некорректно принимающих их за окончание тэга (закрывающий разделитель тэга).

Авторам следует использовать **&** (десятичный код ASCII 38) вместо «**&**» во избежание путаницы со ссылками на символы (открывающий разделитель **entity reference**). Авторам также следует использовать **&** в значениях атрибутов, поскольку ссылки на символы внутри значений атрибута **CDATA** разрешены.

Некоторые авторы используют **character entity reference** «**"**» для кодирования экземпляров двойных кавычек ("), поскольку этот символ может использоваться для разделения значений атрибутов.

Неотображаемые символы

Возможно, браузер не сможет отобразить все символы в документе, например, из-за отсутствия соответствующего шрифта или если символ имеет значение, которое не может быть выражено во внутренней кодировке браузера и т.д.

Поскольку в этом случае есть несколько вариантов, этот документ не предписывает определенной тактики. В зависимости от применения непечатные символы могут также обрабатываться дополнительной системой отображения, а не самим приложением. В случае более сложного поведения, например, настроенного для определенного сценария или языка, рекомендуем следующее поведение для браузеров:

- Примите явно видимый, но незаметный механизм для предупреждения пользователя об отсутствующих ресурсах.
- Если отсутствующие символы представляются в другом числовом представлении, используйте шестнадцатеричную (не десятичную) форму, поскольку эта форма используется в стандартах наборов символов.

Основные типы данных HTML

Каждое определение атрибута включает информацию об учете регистра его значениями. Информация о регистре представляется следующими ключами:

CS

Значение учитывает регистр (то есть браузеры по-разному интерпретируют «a» и «A»).

CI

Значение не учитывает регистр (то есть браузеры одинаково интерпретируют «a» и «A»).

CN

Значение не зависит от регистра, например, потому что это число или символ из набора символов документа.

CA

Само определение элемента или атрибута дает информацию о регистре.

Если значением атрибута является список, ключи применяются к каждому значению в списке, если не указано обратное.

Основные типы SGML

В определении типа документа определяется синтаксис содержимого элемента HTML и значений атрибутов с использованием меток SGML (например, **PCDATA**, **CDATA**, **NAME**, **ID** и т.д.).

Вот обобщенная информация о ключах:

CDATA — это последовательность символов из набора символов документа, она может включать **character entities**. Браузеры должны интерпретировать значения атрибутов следующим образом:

- Заменять **character entities** на символы;

- Игнорировать перевод строки;
- Заменять каждый возврат каретки или табуляцию на один пробел.

Браузеры могут игнорировать пробелы в начале и в конце значений атрибута **CDATA** (например, « myval » интерпретируется как «myval»). Авторы не должны объявлять значения атрибутов с пробелами в начала или в конце.

На некоторые атрибутов HTML 4.0 со значениями атрибутов **CDATA** спецификация налагает дополнительные ограничения на множество допустимых значений атрибутов, не выраженные в DTD.

Хотя элементы **STYLE** и **SCRIPT** используют **CDATA** для своей модели данных, для этих элементов браузеры должны обрабатывать **CDATA** по-другому. Разметка и **entities** должны считаться текстом и передаваться в приложение как есть. Первое вхождение последовательности символов «</» (открывающий разделитель конечного тэга) считается концом содержимого элемента. В допустимых документах это будет конечный тэг элемента.

Метки **ID** и **NAME** должны начинаться с буквы (A-Z, a-z), за которой может следовать любое число букв, цифр (0-9), символов переноса (-), символов подчеркивания (_), двоеточий (:) и точек (.).

IDREF и **IDREFS** — это ссылки на метки **ID**, определенные другими атрибутами. **IDREF** — одиночная метка, а **IDREFS** — разделенный пробелами список меток.

Метки **NUMBER** должны содержать по крайней мере одну цифру (0-9).

Текстовые строки

Ряд атрибутов (**%Text;** в DTD) принимают текст, который предназначается для чтения людьми.

URI

URI включают URL. Относительные URI разрешаются до полных URI с использованием основного URI. URI представляются в DTD комбинацией символов **%URI;**.

URI вообще учитывают регистр. Могут быть URI, или части URI, в которых регистр не имеет значения (например, имена машин), но идентификация их может быть непроста. Пользователи должны всегда считать, что URI учитывают регистр (чтобы не ошибиться).

Цвета

Значение атрибута типа «**color**» (**%Color;**) относится к определениям цветов. Значение цвета может быть шестнадцатеричным числом (которому предшествует знак диеза) или одним из следующих шестнадцати названий цветов. Названия цветов учитывают регистр.

Названия цветов и значения RGB

Black = #000000

Green = #008000

Silver = #C0C0C0

Lime = #00FF00

Gray = #808080

Olive = #808000

White = #FFFFFF

Yellow = #FFFF00

Maroon = #800000

Navy = #000080

Red = #FF0000

Blue = #0000FF

Purple = #800080

Teal = #008080

Fuchsia = #FF00FF

Aqua = #00FFFF

То есть, значения **#800080** и «**Purple**» оба означают пурпурный цвет.

Замечания об использовании цветов

Хотя цвета могут существенно добавлять информации в документ и повышать удобство чтения, при использовании цветов имейте в виду следующие основные принципы:

- Использование элементов и атрибутов HTML для указания цвета нежелательно. Вместо этого следует использовать таблицы стилей.

- Не используйте комбинации цветов, вызывающие проблемы у пользователей.
- Если вы используете изображение в качестве фона или устанавливаете цвет фона, не забудьте установить и цвета текста.

Цвета, указанные в элементах **BODY** и **FONT** и в **bgcolor** в таблицах выгладят по-разному на разных платформах (на рабочих станциях, Mac, Windows и на панелях LCD и CRT), поэтому не рассчитывайте на определенный эффект. В будущем поддержка цветовой модели вместе с цветовыми профилями ICC должна устранить эти проблемы.

При возможности принимайте общие соглашения.

Длины

HTML определяет три типа значений длины для атрибутов:

Пикселы

Значение (**%Pixels;** в DTD) — это целое, представляющее число пикселов (на экране, на бумаге). Таким образом, значение «50» означает пятьдесят пикселов.

Длина

Значение (**%Length;** в DTD) может быть **%Pixel;** или доля вертикального или горизонтального расстояния в процентах. Таким образом, значение «**50%**» означает половину доступного пространства.

МультиДлина

Значение (**%MultiLength;** в DTD) может быть **%Length;** или относительной длиной. Относительная длина имеет форму «**i***», где «**i**» — целое число. При распределении пространства между элементами, конкурирующими за это пространство, браузеры сначала отводят место для длин, определенных в пикселах и процентах, а затем делят оставшееся место между относительными длинами. Каждая относительная длина получает часть доступного пространства, пропорциональную целому числу, предшествующему «*». Значение «*» эквивалентно «**1***». Таким образом, если имеется 60 пикселов пространства после того, как браузер распределит пространство для длин, определенных в пикселах и процентах, а конкурирующими относительными длинами являются **1***, **2*** и **3***; **1*** получит 10 пикселов, **2*** — 20 пикселов, а **3*** — 30 пикселов.

Значения длин не учитывают регистр.

Типы содержимого (типы MIME)

«Тип носителя» указывает природу связанного ресурса. Далее будет использоваться термин «тип содержимого» вместо «типа носителя» в соответствии с его использованием. Более того, «тип носителя» может означать носитель, на котором браузер генерирует документ.

Этот тип представлен в DTD с помощью **%ContentType;**.

Типы содержимого учитывают регистр.

Примеры типов содержимого включают «**text/html**», «**image/png**», «**image/gif**», «**video/mpeg**», «**audio/basic**», «**text/tcl**», «**text/javascript**» и «**text/vbscript**».

Тип содержимого «**text/css**», хотя он и не зарегистрирован в IANA, должен использоваться, если связываемым элементом является таблица стилей.

Коды языков

Значения атрибутов, типом которых является код языка (**%LanguageCode** в DTD), относится к коду языка. В кодах языков пробелы недопустимы.

Коды языков учитывают регистр.

Кодировки символов

Атрибуты «**charset**» (**%Charset** в DTD) относятся к кодировкам символов. Значениями должны быть строки (например, «**euc-jp**») из реестра IANA.

Имена кодировок символов учитывают регистр.

Отдельные символы

Определенные атрибуты вызывают отдельный символ из набора символов документа. Эти атрибуты имеют тип **%Character** в DTD.

Отдельные символы можно указать с помощью ссылок на символы (например, «**&**»).

Дата и время

Кодировка ISO позволяет много вариантов представления даты и времени. Один из таких форматов для определения допустимых строк дата/время (**%Datetime** в DTD) следующий:

```
ГГГГ-ММ-ДДТчч:мм:ссУЧП
```


где:

ГГГГ — год из четырех цифр

ММ — месяц из двух цифр (01 — январь и т.д.)

ДД — день из двух цифр (01 — 31)

чч — две цифры часов (00 — 23)

мм — две цифры минут (00 — 59)

сс — две цифры секунд (00 — 59)

УЧП = указатель часового пояса

Указатели часового пояса:

Z

означает **UTC** (Общее скоординированное время). «**Z**» должно быть в верхнем регистре.

+чч:мм

указывает, что местное время отстоит на **чч** часов и **мм** минут от **UTC** вперед.

-чч:мм

указывает, что местное время отстает на **чч** часов и **мм** минут от **UTC**.

Указанные компоненты должны присутствовать в точности, с точно такой же пунктуацией. Помните, что буква «T» отображается в строке литерально (она должна быть в верхнем регистре), для указания начала времени.

Если генерирующее приложение не знает времени с точностью до секунды, для секунд может использоваться значение «00» (при необходимости также для минут и для часов).

Типы ссылок

Авторы могут использовать следующие распознаваемые типы ссылок, перечисленные здесь вместе с условными интерпретациями. В DTD **%LinkTypes** означает список типов ссылок, разделенных пробелами. Символы пробелов в типах ссылок не допускаются.

Эти типы ссылок не учитывают регистр, т.е. «**Alternate**» означает то же, что и «**alternate**».

Браузеры, поисковые машины и т.д. могут интерпретировать эти типы ссылок несколькими способами. Например, браузеры могут предоставлять доступ к связанным документам с помощью навигационной панели.

Alternate

Обозначает альтернативные версии документа, в котором находится ссылка. Вместе с атрибутом **lang** означает переведенную версию документа. Вместе с атрибутом **media** означает версию, созданную для другого носителя.

Stylesheet

Обозначает внешнюю таблицу стилей. Используется вместе с типом ссылки «**Alternate**» для таблиц стилей, выбираемых пользователем.

Start

Обозначает первый документ в наборе. Этот тип ссылки сообщает поисковым машинам о том, какой документ автор считает началом набора.

Next

Обозначает следующий документ в линейной последовательности документов. Браузеры могут предварительно загружать документ «**next**» для сокращения времени загрузки.

Prev

Обозначает предыдущий документ в упорядоченной серии документов. Некоторые браузеры также поддерживают синоним «**Previous**».

Contents

Обозначает документ, служащий содержанием. Некоторые браузеры также поддерживают синоним **ToC** (из «**Table of Contents**»).

Index

Обозначает документ, являющийся указателем текущего документа.

Glossary

Обозначает документ — глоссарий терминов, относящихся к текущему документу.

Copyright

Обозначает замечание об авторском праве для текущего документа.

Chapter

Обозначает документ, являющийся главой в наборе документов.

Section

Обозначает документ, являющийся разделом в наборе документов.

Subsection

Обозначает документ, являющийся подразделом в наборе документов.

Appendix

Обозначает документ, являющийся приложением в наборе документов.

Help

Обозначает документ, содержащий справку (более подробная информация, ссылки на другие информационные ресурсы и т.д.)

Bookmark

Обозначает закладку. Закладка — это ссылка на ключевую точку в расширенном документе. Атрибут **title** может использоваться, например, для пометки закладки. Помните, что в каждом документе можно определить несколько закладок.

Авторы могут определить дополнительные типы ссылок, не описанные в этой спецификации. При этом они должны использовать профиль для указания соглашений, используемых для определения типов ссылок.

Дескрипторы носителей

Ниже приведен список распознаваемых дескрипторов носителей (%**MediaDesc** в DTD).

screen

Предназначен для экранов компьютеров, не разделенных на страницы.

tty

Предназначен для носителя с фиксированной сеткой для символов, таких как телетайпы, терминалы или переносные устройства с ограниченными возможностями отображения.

tv

Предназначен для устройств типа телевизора (низкое разрешение, цвета, ограниченные возможности прокрутки).

projection

Предназначен для проекторов.

handheld

Предназначен для карманных устройств (небольшой экран, монохромный, растровая графика, ограниченный диапазон).

print

Предназначен для страничных, непрозрачных материалов и документов, просматриваемых на экране в режиме предварительного просмотра печати.

braille

Предназначен для тактильных устройств с алфавитом Бройля.

aural

Предназначен для синтезаторов речи.

all

Для всех устройств.

В будущих версиях HTML могут быть введены новые значения и разрешены параметризованные значения. Для упрощения введения этих расширений соответствующие спецификации браузеры должны иметь возможность анализировать значение атрибута **media** следующим образом:

Значение — это разделенный запятыми список элементов. Например,

```
media="screen, 3d-glasses, print and resolution > 90dpi"
```

отображается в:

```
"screen"
"3d-glasses"
"print and resolution > 90dpi"
```

Каждый элемент усекается перед первым символом, не являющимся буквой кодировки US ASCII (a-z, A-Z) (десятичные коды Unicode 65-90, 97-122), цифрой (0-9) (шестнадцатеричные коды Unicode 30-39) или знаком переноса (45).

В данном примере получается:

```
"screen"
"3d-glasses"
"print"
```

Затем с учетом регистра проводится сверка с набором определенных выше типов дескрипторов. Браузеры могут игнорировать несовпадающие элементы. В данном примере останутся только элементы **screen** и **print**.

Таблицы стилей могут включать вариации в зависимости от носителя (например, конструкция CSS **@media**). В таких случаях имеет смысл использовать «**media=all**».

Данные сценария

Данные сценария (**%Script;** в DTD) могут быть содержимым элемента **SCRIPT** и значением атрибутов внутренних событий. Браузеры не должны оценивать данные сценариев в разметке HTML, а должны передавать эти данные ядру сценариев.

Учет регистра в данных сценариев зависит от языка сценариев.

Помните, что данные сценариев, являющиеся содержимым элемента, не могут содержать ссылок на символы, но данные сценария, являющиеся значением атрибута, могут.

Данные таблиц стилей

Данные таблиц стилей (**%StyleSheet;** в DTD) могут быть содержимым элемента **STYLE** и значением атрибута **style**. Браузеры не должны оценивать данные стилей в разметке HTML.

Учет регистра данных стиля зависит от языка таблиц стилей.

Помните, что данные таблиц стилей, являющиеся содержимым элемента, не могут включать ссылки на символы, но данные таблиц стилей, являющиеся значением атрибута, могут включать их.

Целевые имена кадров

За исключением приведенных ниже зарезервированных имен, целевые имена кадров (**%FrameTarget;** в DTD) должны начинаться с алфавитных символов (a-z, A-Z). Браузеры должны игнорировать все остальные имена.

Следующие **target names** зарезервированы и имеют специальные значения.

_blank

Браузеры должны загружать документ в новое окно без имени.

_self

Браузеры должны загружать документ в тот же кадр, в котором находится ссылающийся на него документ.

_parent

Браузеры должны загружать документ в непосредственный родительский кадр этого кадра во **FRAMESET**. Это значение эквивалентно **_self**, если текущий кадр не имеет родительского кадра.

_top

Браузеры должны загружать документ в полное окно (закрывая все остальные кадры). Это значение эквивалентно **_self**, если у текущего кадра нет родительского кадра.

Глава 19. Глобальная структура документа

Документ в формате HTML 4.0 состоит из трех частей:

- строки, содержащей информацию о версии HTML
- раздела заголовков (определяемого элементом **HEAD**)
- тела, которое включает собственно содержимое документа. Тело может вводиться элементом **BODY** или элементом **FRAMESET**.

Перед каждым элементом или после каждого элемента может находиться пустое пространство (пробелы, переход на новую строку, табуляции и комментарии). Разделы 2 и 3 должны отделяться элементом HTML.

Вот пример простого документа HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
 "http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
 <HEAD>
  <TITLE>Мой первый документ HTML</TITLE>
 </HEAD>
 <BODY>
  <P>Всем привет!
```



```
 </BODY>
</HTML>
```

Информация о версии HTML

В документе HTML должна быть объявлена используемая в нем версия языка HTML. Объявление типа документа указывает определение типа документа (DTD), используемое в этом документе.

HTML 4.0 определяет три DTD, так что авторы должны включать в свои документы одно из следующих объявлений типов. Разница между DTD заключается в поддерживаемых ими элементах.

HTML 4.0 **Strict** DTD (строгое определение) включает все элементы и атрибуты, не являющиеся нежелательными и не использующиеся в документах с кадрами. Для документов, использующих это DTD, используйте такое объявление типа документа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-tml40/strict.dtd">
```

HTML 4.0 **Transitional** DTD (переходное определение) включает все, что включено в строгое DTD, а также нежелательные элементы и атрибуты (большинство из которых относится к визуальному представлению). Для документов, использующих это DTD, используйте такое объявление типа документа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
```

HTML 4.0 **Frameset** DTD (определение для кадров) включает все, что включено в переходное DTD, а также кадры. Для документов, использующих это DTD, используйте такое объявление типа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
"http://www.w3.org/TR/REC-html40/frameset.dtd">
```

URI в каждом объявлении типа документа позволяет браузерам загрузить DTD и все необходимые **entity sets**. Следующие URI указывают на DTD и **entity sets** для HTML 4.0, поддерживаемого W3C:

```
"http://www.w3.org/TR/REC-html40/strict.dtd"
```

— строгое DTD по умолчанию

```
"http://www.w3.org/TR/REC-html40/loose.dtd"
```

— переходное DTD

```
"http://www.w3.org/TR/REC-html40/frameset.dtd"
```

— DTD для документов, использующих кадры

```
"http://www.w3.org/TR/REC-html40/HTMLlat1.ent"
```

— Latin-1 entities

```
"http://www.w3.org/TR/REC-html40/HTMLsymbol.ent"
```

— Symbol entities

```
"http://www.w3.org/TR/REC-html40/HTMLspecial.ent"
```

— Special entities

Связь между общими идентификаторами и файлами можно указать с использованием файла каталога, за которым следует формат, рекомендуемый Открытым консорциумом SGML.

Элемент HTML

```
<!ENTITY % html.content "HEAD, BODY">
<!ELEMENT HTML O O (%html.content;) -- корневой элемент документа
-->
<!ATTLIST HTML
 %i18n;         -- lang, dir -->
```

Начальный тэг: не обязательный, Конечный тэг: не обязательный.

Определения атрибутов

```
версия = cdata[CN]
```

Нежелателен. Значение этого атрибута указывает версию HTML DTD, которой подчиняется этот документ. Этот атрибут является нежелательным, поскольку он является избыточным при наличии информации о версии, указываемой в объявлении типа документа.

Атрибуты, определяемые в любом другом месте

```
lang (информация о языке), dir (направление текста)
```

После объявления типа документа остальная часть документа HTML содержится в элементе HTML. Таким образом, типичный документ HTML имеет такую структуру:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
...Здесь идут заголовок, тело и т.д...
</HTML>
```

Заголовок документа

Элемент HEAD

```
<!-- %head.misc;, определенный ранее как
"SCRIPT|STYLE|META|LINK|OBJECT" -->
<!ENTITY % head.content "TITLE & BASE?">
```



```
<!ELEMENT HEAD O O (%head.content;) +(%head.misc;) - заголовок
документа -->
<!ATTLIST HEAD
 %i18n;         -- lang, dir --
 profile  %URI;    #IMPLIED -- каталог метаинформации -->
```

Начальный тэг: не обязателен. Конечный тэг: не обязателен.

Определения атрибутов

```
profile = uri [CT]
```

Этот атрибут указывает местоположение одного или нескольких профилей метаданных, отделяемых пробелами. Для расширения в будущем браузеры должны предполагать, что значение является списком, хотя в как правило значащим считается только первый URI.

Атрибуты, определяемые в любом другом месте

```
lang (информация о языке), dir (направление текста)
```

Элемент **HEAD** содержит информацию о текущем документе, такую как заголовок, ключевые слова, которые могут использоваться поисковыми машинами, и другие данные, которые не считаются содержимым документа. Браузеры обычно не используют при генерации элементы из раздела **HEAD**. Однако они могут предоставлять пользователям информацию из раздела **HEAD** с помощью своих собственных механизмов.

Элемент TITLE

```
<!-- Элемент TITLE не считается частью текста.
 Он должен отображаться, например, в качестве заголовка страницы
или окна. В документе должен быть ровно один заголовок. -->
<!ELEMENT TITLE - - (#PCDATA) -(%head.misc;) -- document title --
>
<!ATTLIST TITLE %i18n>
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Атрибуты, определяемые в любом другом месте

```
lang (информация о языке), dir (направление текста)
```

Каждый документ HTML должен иметь элемент **TITLE** в разделе **HEAD**.

Авторы должны использовать элемент **TITLE** для идентификации содержимого документа. Поскольку пользователи часто обращаются к документам за пределами контекста, авторам следует обеспечивать заголовки в широком контексте. Таким образом, вместо заголовков типа «Введение», ничего не говорящих о контексте, авторам следует использовать заголовки типа «Введение в средневековое пчеловодство».

Из соображений доступности браузеры всегда должны делать содержимое элемента **TITLE** доступным пользователям (включая элементы **TITLE** в кадрах). Механизм этого зависит от браузера (например, в виде заголовка или произносимый). Заголовки могут включать **character entities** (для символов со знаком ударения, специальных символов и т.д.), но не могут содержать другой разметки. Вот образец заголовка документа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD>
<TITLE>Исследование динамики популяции </TITLE>
... другие элементы заголовка...
</HEAD>
<BODY>
... тело документа...
</BODY>
</HTML>
```

Атрибут title

Определения атрибутов

```
title = text [CS]
```

Этот атрибут предлагает информацию об элементе, для которого он устанавливается. В отличие от элемента **TITLE**, который предоставляет информацию обо всем документе и может присутствовать в тексте только один раз, атрибут **title** может сопровождать любое число элементов. Узнать, поддерживает ли элемент этот атрибут, можно в определении элемента.

Значения атрибута **title** могут использоваться браузерами в генерации изображения по-разному. Например, визуальные браузеры часто отображают заголовок как подсказку (краткое сообщение, которое появляется, если вы указываете на объект). Аудио-браузеры могут проговаривать информацию заголовка. Например, установка этого атрибута для ссылки позволяет браузерам (визуальным и невизуальным) сообщить пользователям о природе связанного ресурса:

```
...текст...
Вот фотография
<A href="http://someplace.com/neatstuff.gif" title="Me scuba div-
ing">
как я нырял в прошлом году
</A>
...еще текст...
```


Атрибут **title** играет дополнительную роль при использовании с элементом **LINK** для назначения внешней таблицы стилей.

Для улучшения качества синтеза речи в случае плохой обработки стандартными механизмами будущие версии HTML могут включать атрибут для кодирования фонематической информации.

Метаданные

HTML позволяет авторам указывать метаданные — информацию о документе вместо содержимого документа — множеством способов.

Например, чтобы указать автора документа, можно использовать элемент **META** следующим образом:

```
<META name="Author" content="Иван Иванов">
```

Элемент **META** задает свойство (здесь «Author (Автор)») и назначает ему значение (здесь — «Иван Иванов»).

Значение свойства и набор допустимых значений этого свойства должны определяться в относительном словаре, называемом профилем. Например, профиль, разработанный для помощи в индексировании документов для поисковых машин может определять такие свойства как «**author**», «**copyright**», «**keywords**» и т.д.

Задание метаданных

В общем случае задание метаданных состоит из двух шагов:

- Объявление свойства и его значения. Это можно сделать двумя способами:
  - Из документа с помощью элемента **META**.
  - Не из документа с помощью ссылки на метаданные через элемент **LINK**.
- Сославшись на профиль, в котором определяются свойства и их допустимые значения. Для назначения профиля используйте атрибут профиль элемента **HEAD**.

Помните, что поскольку профиль определяется для элемента **HEAD**, этот профиль применяется ко всем элементам **META** и **LINK** в заголовке документа.

Браузеры не обязательно должны поддерживать механизмы метаданных.

Элемент META

```
<!ELEMENT META – O EMPTY     -- общая метаинформация -->
<!ATTLIST META

 %i18n;          -- lang, dir, для использования с содержимым --
 http-equiv NAME    #IMPLIED -- имя заголовка ответа HTTP --
 name  NAME    #IMPLIED -- имя метаинформации --
 content  CDATA    #REQUIRED – связанная информация --
 scheme  CDATA    #IMPLIED -- выбор формы содержимого --
 >
```

Начальный тэг: обязателен. Конечный тэг: запрещен.

Определения атрибутов

Для следующих атрибутов допустимые значения и их интерпретация зависят от профиля:

```
name = name [CS]
```

Этот атрибут определяет имя свойства.

```
content = cdata [CS]
```

Этот атрибут определяет значение свойства.

```
scheme = cdata [CS]
```

Этот атрибут дает имя схеме, используемой для интерпретации значения свойства.

```
http-equiv = name [CI]
```

Этот атрибут может использоваться вместо атрибута **name**. Серверы HTTP используют этот атрибут для сбора информации для заголовков сообщений ответов HTTP.

Атрибуты, определяемые в любом другом месте

```
lang (информация о языке), dir (направление текста)
```

Элемент **META** может использоваться для идентификации свойств документа (например, автора, срок истечения, список ключевых слов и т.д.) и назначения им значений.

Каждый элемент **META** задает пару свойство/значение. Атрибут **name** определяет свойства, а атрибут **content** — значение.

Например, в следующем объявлении устанавливается значение свойства **Author**:

```
<META name="Author" content="Дэйв Рэггетт">
```


Атрибут **lang** может использоваться с элементом **META** для указания языка значения атрибута **content**. Это позволяет синтезаторам речи использовать правила произношения для разных языков.

В этом примере имя автора объявляется на французском языке:

```
<META name="Author" lang="fr" content="Arnaud Le Hors">
```

Элемент **META** — это общий механизм задания метаданных. Однако некоторые элементы и атрибуты HTML уже обрабатывают некоторые части метаданных и могут использоваться авторами вместо элементов **META** для указания этих частей: элементы **TITLE**, **ADDRESS**, **INS** и **DEL**, атрибут **title** и атрибут **cite**.

Если свойство, заданное с помощью элемента **META**, принимает значение URI, некоторые авторы предпочитают указывать метаданные с помощью элемента **LINK**. Таким образом, следующее объявление:

```
<META name="DC.identifier"
content="ftp://ds.internic.net/rfc/rfc 1866.txt">
```

можно также записать следующим образом:

```
<LINK rel="DC.identifier"
   type="text/plain"
href="ftp://ds.internic.net/rfc/rfc1866.txt">
```

META и заголовки HTTP

Атрибут **http-equiv** может использоваться вместо атрибута **name**; он особенно важен, если документы загружаются по протоколу передачи гипертекста (HTTP). Серверы HTTP могут использовать имя свойства, указанное в атрибуте **http-equiv** для создания заголовка в ответе HTTP.

В следующем примере объявление **META**:

```
<META http-equiv="Expires" content="Tue, 20 Aug 1996 14:25:27
GMT">
```

вернет следующий заголовок HTTP:

```
Expires: Tue, 20 Aug 1996 14:25:27 GMT
```

Это может использоваться кэш-памятью для определения того, когда следует загрузить новую копию связанного документа.

Некоторые браузеры поддерживают использование элемента **META** для обновления текущей страницы по истечении указанного числа секунд с возможностью замены на другой URI.

```
<META http-equiv="refresh"
content="3,http://www.acme.com/intro.html">
```

content — это число, указывающее задержку в секундах, за которым следует URI, который нужно загрузить по прошествии этого времени. Этот механизм широко используется для создания кратковременных заставок. Однако поскольку некоторые браузеры не поддерживают этот механизм, авторам следует включить в заставку возможность перейти на следующую страницу (чтобы они на «зависли» на заставке).

META и поисковые машины

Основной способ использования элемента **META** — задание ключевых слов, которые поисковые машины могут использовать для улучшения результатов поиска. Если информация о документе представлена в нескольких элементах **META** в зависимости от языка, поисковые машины могут фильтровать атрибут **lang** и отображать результаты поиска с использованием выбранного пользователем языка. Например,

```
<-- Для говорящих на американском английском -->
<META name="keywords" lang="en-us"
content="vacation, Greece, sunshine">
<-- Для говорящих на британском английском -->
<META name="keywords" lang="en"
content="holiday, Greece, sunshine">
<-- для русскоязычных пользователей -->
<META name="keywords" lang="fr"
content="отпуск, Греция, солнце">
```

Эффективность поисковых машин можно также повысить с использованием элемента **LINK** для задания ссылок на переводы документа на другие языки, ссылки на версии документа для другого носителя (например, PDF), и, если документ является частью набора, ссылки на соответствующую начальную точку для просмотра набора.

META и PICS

Platform for Internet Content Selection (Платформа для выбора содержимого Интернет) — это инфраструктура для связывания меток (метаданных) с содержимым Интернет. Созданная для помощи родителям и учителям в управлении доступом детей к Интернет, она также упрощает другое использование меток, включая управление подписью кодов, секретностью и правами интеллектуальной собственности.

Этот пример иллюстрирует использование объявления **META** для включения метки PICS 1.1:

```
<HEAD>
 <META http-equiv="PICS-Label" content='
 (PICS-1.1 "http://www.gcf.org/v2.5"
 labels on "1994.11.05T08:15-0500"
 until "1995.12.31T23:59-0000"
 for "http://w3.org/PICS/Overview.html"
ratings (suds 0.5 density 0 color/hue 1))
'>
<TITLE>... название документа ...</TITLE>
</HEAD>
```


META и информация по умолчанию

Элемент **META** может использоваться для указания информации по умолчанию для документа в следующих случаях:

- Язык сценариев по умолчанию.
- Язык таблиц стилей по умолчанию.
- Кодировка символов документа.

В следующем примере для документа указывается кодировка символов ISO-8859-5:

```
<META http-equiv="Content-Type" content="text/html;
charset=ISO-8859-5">
```

Профили метаданных

Атрибут профиль элемента **HEAD** указывает местоположение профиля метаданных. Значением атрибута **profile** является URI. Браузеры могут использовать этот URI двумя способами:

- Как глобально уникальное имя. Браузеры могут распознавать имя (не загружая в действительности профиль) и выполнять некоторые действия на базе известных соглашений для этого профиля. Например, поисковые машины могут обеспечивать интерфейс для поиска в каталогах документов HTML, где все эти документы используют один и тот же профиль для представления записей каталога.
- Как ссылку. Браузеры могут разыменовывать URI и выполнять некоторые действия на базе определений из профиля (например, авторизовать использование профиля в текущем документе HTML).

В этом примере используется гипотетический профиль, определяющий полезные свойства для индексирования документов.

Для свойств, определяемых этим профилем — включая «**author**», «**copyright**», «**keywords**» и «**date**» — значения устанавливаются с помощью последовательных объявлений **META**.

```
<HEAD profile="http://www.acme.com/profiles/core">
<TITLE>How to complete Memorandum cover sheets</TITLE>
 <META name="author" content="John Doe">
 <META name="copyright" content="&copy; 1997 Acme Corp.">
 <META name="keywords" content="corporate,guidelines,cataloging">
 <META name="date" content="1994-11-06T08:49:37+00:00">
 </HEAD>
```

Атрибут scheme позволяет авторам предоставлять браузерам дополнительный контекст для корректной интерпретации метаданных. Иногда такая дополнительная информация может иметь важное значение, например, если метаданные указаны в другом формате. Например, автор может указать дату в формате «10-9-97» (неоднозначно); означает ли это 9 октября 1997 г. или 10 сентября 1997 г.? Значение атрибута **scheme "Month-Date-Year"** устранит неоднозначность.

В других случаях атрибут **scheme** может предоставлять браузерам полезную, но не столь важную информацию.

Например, следующее объявление **scheme** поможет браузерам определить, что значение свойства «**identifier**» — номер кода ISBN:

```
<META scheme="ISBN" name="identifier" content="0-8230-2355-9">
```

Значения атрибута **scheme** зависят от свойства **name** и связанного профиля. Примером профиля является **Dublin Core**. Этот профиль определяет набор рекомендуемых свойств для электронных библиографических описаний и предназначен для обеспечения интероперабельности в несопоставимых моделях описаний.

Глава 20. Тело документа

Элемент BODY

```
<!ELEMENT BODY O O (%block;|SCRIPT)+ +(INS|DEL) - тело документа
-->
<!ATTLIST BODY
 %attrs;         -- %coreattrs, %i18n, %events --
 onload   %Script; #IMPLIED -- документ загружен --
 onunload  %Script; #IMPLIED -- документ удален --
 >
```

Начальный тэг: не обязателен. Конечный тэг: не обязателен.

Определения атрибутов

```
background = uri[CT]
```

Нежелателен. Значение этого атрибута — URI, указывающий на изображение. Это изображение является фоном (для визуальных браузеров).

```
text = color[CI]
```

Нежелателен.Этот атрибут устанавливает цвет текста (для визуальных браузеров).

```
link = color [CI]
```

Нежелателен. Этот атрибут устанавливает цвет текста гипертекстовых ссылок, по которым вы не переходили (для визуальных браузеров).

```
vlink = color [CI]
```

Нежелателен. Этот атрибут устанавливает цвет текста ссылок, по которым вы переходили (для визуальных браузеров).

```
alink = color [CI]
```

Нежелателен. Этот атрибут устанавливает цвет текста ссылок, когда они выбраны пользователем (для визуальных браузеров).

Атрибуты, определяемые в любом другом месте

```
id, class (идентификаторы в пределах документа)
lang (информация о языке), dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле)
bgcolor (цвет фона)
onload, onunload (внутренние события)
onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmouse-
move, onmouseout, onkeypress, onkeydown, onkeyup (внутренние со-
бытия)
```

В теле документа располагается содержание документа. Содержимое может представляться браузером несколькими способами. Например, для визуальных браузеров вы можете считать тело документа полотном, на котором отображается содержимое: текст, изображения, цвета, рисунки и т.д. Для аудио-браузеров оно может произноситься. Поскольку предпочтительным способом задания представления документов теперь являются таблицы стилей, атрибуты представления в тэге **BODY** являются нежелательными.

Пример нежелательного использования

В следующем фрагменте кода HTML показано использование нежелательного атрибута.

Он устанавливает белый цвет фона, черный цвет текста и красный цвет гиперссылок изначально, цвет фуксии при активизации ссылок и коричневый для ссылок, по которым вы переходили.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 "http://www.w3.org/TR/REC-html40/loose.dtd">
<HTML>
<HEAD>
 <TITLE>Динамика популяции</TITLE>
</HEAD>
<BODY bgcolor="white" text="black"
 link="red" alink="fuchsia" vlink="maroon">
 ... тело документа...
</BODY>
</HTML>
```

Используя таблицы стилей, того же эффекта можно достичь следующим образом:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD>
 <TITLE>Динамика популяции</TITLE>
 <STYLE type="text/css">
 BODY { background: white; color: black}
 A:link { color: red }
 A:visited { color: maroon }
 A:active { color: fuchsia }
 </STYLE>
</HEAD>
<BODY>
 ... тело документа...
</BODY>
</HTML>
```

Использование внешних (связываемых) таблиц стилей обеспечивает гибкость при изменении представления без пересмотра источника документа HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD>
 <TITLE>Динамика популяции</TITLE>
 <LINK rel="stylesheet" type="text/css" href="smartstyle.css">
</HEAD>
```



```
<BODY>
 ... тело документа...
</BODY>
</HTML>
```

Кадры и тела документов HTML. В документах, содержащих кадры, элемент **BODY** заменяется элементом **FRAMESET**.

Идентификаторы элементов: атрибуты id and class

Определения атрибутов

```
id = name [CS]
```

Этот атрибут назначает элементу имя. Имя в пределах документа должно быть уникальным.

```
class = cdata-list [CS]
```

Этот атрибут назначает элементу имя класса или набор имен классов. Одно и то же имя или имена классов могут быть назначены любому числу элементов. Несколько имен классов должны быть разделены пробелами.

Атрибут **id** назначает элементу уникальный идентификатор (который может проверяться синтаксическим анализатором SGML). Например, следующие абзацы распознаются по значениям их атрибутов **id**:

```
<P id="myparagraph"> Этот абзац имеет уникальное имя.</P>
<P id="yourparagraph"> Этот абзац тоже имеет уникальное имя.</P>
```

Атрибут **id** имеет в HTML несколько ролей:

- Способ выбора таблицы стиля.
- Назначение цели (якорь) для гипертекстовых ссылок.
- Средство ссылки на определенный элемент сценария.
- Имя объявленного объекта **OBJECT**.
- В целях обработки браузерами (например, для полей идентификации при извлечении данных из страниц HTML в базу данных, преобразовании документов HTML в другие форматы и т.д.).

Атрибут **class**, с другой стороны, назначает одно или несколько имен классов элементу; при этом элемент может называться принадлежащим к этим классам. Имя класса может использоваться несколькими экземплярами элемента. Атрибут **class** имеет в HTML несколько ролей:

- Способ выбора таблицы стиля (когда автор хочет назначить информацию о стиле набору элементов).

- Для общей обработки браузерами.

Далее элемент **exampleSPAN** используется вместе с атрибутами **id** и **class** для пометки сообщений документа. Сообщения отображаются в английской и русской версиях.

```
<!-- английские сообщения -->
<P><SPAN id="msg1" class="info" lang="en">Variable declared
twice</SPAN>
<P><SPAN id="msg2" class="warning" lang="en">Undeclared vari-
able</SPAN>
<P><SPAN id="msg3" class="error" lang="en">Bad syntax for vari-
able name</SPAN>
<!-- русские сообщения -->
<P><SPAN id="msg1" class="info" lang="fr">Переменная объявлена
дважды</SPAN>
<P><SPAN id="msg2" class="warning" lang="fr">Переменная не объяв-
лена</SPAN>
<P><SPAN id="msg3" class="error" lang="fr">Синтаксическая ошибка в
имени переменной</SPAN>
```

Следующие правила стиля CSS сообщат браузерам о необходимости отображения информации зеленым цветом, предупреждений — желтым, а сообщений об ошибках — красным:

```
SPAN.info { color: green }
SPAN.warning { color: yellow }
SPAN.error { color: red }
```

Помните, что русское «**msg1**» и английское «**msg1**» не могут отображаться в одном документе, поскольку они используют одно и то же значение атрибута **id**. Авторы могут извлечь дальнейшую пользу, используя атрибут **id** для усовершенствования представления отдельных сообщений, указания их в качестве целей (якорей) и т.д.

Почти каждому элементу HTML может быть назначен идентификатор и информация о классе.

Предположим, мы пишем документ о языке программирования. Этот документ должен включать ряд отформатированных примеров. Для форматирования примеров мы используем элемент **PRE**. Мы также назначаем цвет фона (зеленый) всем экземплярам элемента **PRE**, принадлежащим классу «**example**».

```
<HEAD>
<TITLE>... название документа...</TITLE>
<STYLE type="text/css">
PRE.example { background : green }
```



```
</STYLE>
</HEAD>
<BODY>
<PRE class="example" id="example-1">
...код примера...
</PRE>
</BODY>
```

Установив атрибут **id** для этого примера, мы можем:

- создать на него гиперссылку;
- использовать информацию о стиле, отличную от определенной в таблицы, для одного экземпляра информации о стиле.

Атрибут **id** использует одно пространство имен с атрибутом **name**, если он используется для имен якорей.

Элементы уровня блока и встроенные элементы

Некоторые элементы HTML, которые могут присутствовать внутри тэга **BODY**, называются элементами «уровня блока», в то время как другие — «встроенными» (также называемыми элементами «уровня текста»).

Модель содержимого

Обычно элементы уровня блока могут содержать встроенные элементы и другие элементы уровня блока. Обычно встроенные элементы могут содержать только данные и другие встроенные элементы. Этому структурному различию свойственна идея о том, что элементы блока создают «большие» структуры, чем встроенные элементы.

Форматирование

По умолчанию элементы уровня блока форматируются иначе, чем встроенные элементы. Обычно элементы уровня блока начинаются с новой строки, а встроенные элементы — нет.

Направление

По техническим причинам, затрагивающим алгоритм двунаправленного текста (UNICODE), элементы уровня блока и встроенные элементы различаются способами наследования информации о направлении. Таблицы стилей обеспечивают средства задания отображения произвольных элементов, включая и то, генерируется ли элемент как блочный или встроенный. В некоторых случаях, например, в случае

встроенного стиля для элементов списка, это может быть полезным, но вообще говоря, авторам следует избегать такого переопределения интерпретации элементов языка HTML.

Изменение традиционных выражений представления для элементов уровня блока и встроенных элементов влияет на алгоритм двунаправленного текста.

Группировка элементов: элементы DIV и SPAN

```
<!ELEMENT DIV - - (%flow;)*    -- общий контейнер языка/стиля -->
<!ATTLIST DIV
 %attrs;         -- %coreattrs, %i18n, %events --
 >
<!ELEMENT SPAN - - (%inline;)*   -- общий контейнер языка/стиля -
->
<!ATTLIST SPAN
 %attrs;         -- %coreattrs, %i18n, %events --
 >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Определения атрибутов

```
href = uri [CT]
```

Этот атрибут указывает ресурс, предоставляющий дополнительную информацию о содержимом элемента **DIV** или **SPAN**.

Атрибуты, определяемые в любом другом месте

```
id, class (идентификаторы в пределах документа)
lang (информация о языке), dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле information)
align (выравнивание)
onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmouse-
move, onmouseout, onkeypress, onkeydown, onkeyup (внутренние
события)
```

Элементы **DIV** и **SPAN** вместе с атрибутами **id** и **class** обеспечивают общий механизм добавления в документы структуры. Эти элементы определяют встраиваемую информацию (**SPAN**) или информацию уровня блока (**DIV**), но не налагают никаких других выражений для представления контекста. Таким образом, авторы могут использовать эти элементы с таблицами стилей, атрибутами **lang** и т.д.

Предположим, вы хотите сгенерировать документ в формате HTML на основе базы данных информации о клиентах. Поскольку HTML не включает элементов для идентификации таких объектов как

«клиент», «номер телефона», «адрес электронной почты» и т.д., мы используем элементы **DIV** и **SPAN** для достижения нужных эффектов структуры и представления. Для структурирования информации мы могли использовать элемент **TABLE** следующим образом:

```
<!-- Пример данных из базы данных о клиентах: -->
<!-- Имя: Ivan Ivanov, Тел.: (095) 185-3332, Email: II@oo.org -->
<DIV id="client-ivanov" class="client">
<P><SPAN class="client-title">Информация о клиенте:</SPAN>
<TABLE class="client-data">
<TR><TH>Фамилия:<TD>Ivanov</TR>
<TR><TH>Имя:<TD>Ivan</TR>
<TR><TH>Тел:<TD>(095) 185-3332</TR>
<TR><TH>Email:<TD>ii@oo.org</TR>
</TABLE>
</DIV>
<DIV id="client-petrov" class="client">
<P><SPAN class="client-title">Информация о клиенте:</SPAN>
<TABLE class="client-data">
<TR><TH>Фамилия:<TD>Petrov</TR>
<TR><TH>Имя:<TD>Petr</TR>
<TR><TH>Тел:<TD>(253) 115-5420</TR>
<TR><TH>Email:<TD>pp@coucou.com</TR>
</TABLE>
</DIV>
```

Затем мы легко сможем добавить объявление таблицы стиля для настройки представления этих записей.

Визуальные браузеры обычно помещают символ перевода строки перед и после элементов **DIV**, например:

```
<P>aaaaaaaaa<DIV>bbbbbbbbb</DIV><DIV>vvvvv<P>vvvvv</DIV>
```

что обычно генерируется как:

```
aaaaaaaaa
bbbbbbbbb
vvvvv

vvvvv
```

Заголовки: Элементы H1, H2, H3, H4, H5, H6

```
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
<!--
 Существует шесть уровней заголовков - с H1 (наиболее важный) до
H6 (наименее важный).
```

```
-->
<!ELEMENT (%heading;) - - (%inline;)* -- заголовки -->
<!ATTLIST (%heading;)
 %attrs;         -- %coreattrs, %i18n, %events --
 >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Атрибуты, определяемые в любом другом месте

```
id, class (идентификаторы в пределах документа)
lang (информация о языке), dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле)
align (выравнивание)
onclick, ondblclick, onmousedown, onmouseup, onmouseover,
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
(внутренние события)
```

Заголовок кратко описывает содержание раздела, которому он предшествует. Информация из заголовка может использоваться браузерами, например, для автоматического построения оглавления документа.

В языке HTML существует шесть уровней заголовков: **H1** — наиболее важный — и **H6** — наименее важный. Визуальные браузеры обычно отображают более важные заголовки более крупным шрифтом.

В следующем примере показано, как использовать элемент **DIV** для того, чтобы связать заголовок с последующим разделом документа. Это позволит вам определить стиль для раздела (цвет фона, шрифт и т.д.) с использованием таблиц стилей.

```
<DIV class="section" id="forest-elephants" >
<H1>Лесные слоны</H1>
<P>В этом разделе мы обсуждаем менее известных лесных слонов.
...продолжение раздела...
<DIV class="subsection" id="forest-habitat" >
<H2>Ариал</H2>
<P>Лесные слоны не живут в деревьях, а среди них.
...продолжение раздела...
</DIV>
</DIV>
```

Эту структуру можно представить с использованием информации о стиле, например:

```
<HEAD>
<TITLE>... название документа ...</TITLE>
```



```
<STYLE type="text/css">
DIV.section { text-align: justify; font-size: 12pt}
DIV.subsection { text-indent: 2em }
H1 { font-style: italic; color: green }
H2 { color: green }
</STYLE>
</HEAD>
```

Пронумерованные разделы и ссылки

Язык HTML не генерирует номера разделов из заголовков. Это может выполняться браузерами. Вскоре языки описания таблиц стилей, такие как CSS, предоставят авторам возможность управления генерацией номеров разделов.

Некоторые люди считают пропуск уровней заголовков дурным тоном. Они принимают порядок заголовков H1 H2 H1, но не принимают порядок H1 H3 H1, поскольку пропущен уровень H2.

Элемент ADDRESS

```
<!ELEMENT ADDRESS - - (%inline;)* -- информация об авторе -->
<!ATTLIST ADDRESS
 %attrs;         -- %coreattrs, %i18n, %events --
 >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Атрибуты, определяемые в любом другом месте

```
id, class (идентификаторы в пределах документ)
lang (информация о языке), dir (направление текста)
onclick, ondblclick, onmousedown, onmouseup, onmouseover,
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
(внутренние события)
```

Элемент **ADDRESS** может использоваться авторами для указания контактной информации или основной части документа, такой как форма. Этот элемент часто находится в начале или в конце документа.

Например, страница на сервере W3C, относящаяся к языку HTML, может включать следующую контактную информацию:

```
<ADDRESS>
<P><A href="../People/Raggett/">Dave Raggett</A>,
 <A href="../People/Arnaud/">Arnaud Le Hors</A>,
contact persons for the <A href="Activity">W3C HTML
Activity</A><BR>
$Date: 1997/12/16 05:38:14 $
</ADDRESS>
```

Информация о языке и направление текста

Определения атрибутов

```
lang = код языка [CI]
```

Этот атрибут указывает основной язык значений атрибутов элементов и секстового содержимого. По умолчанию значение этого атрибута не установлено.

Информация о языке, указанная с помощью атрибута **lang**, может использоваться браузером для управления генерацией изображения различными способами. Некоторые ситуации, в которых указываемая автором информация о языке, может быть полезна:

- Помощь поисковым машинам
- Помощь синтезаторам речи
- Помощь браузерам в выборе вариантов глифов для высококачественной типографии
- Помощь браузеру в выборе набора кавычек
- Помощь браузеру в вопросах переноса, лигатур и интервалов
- Помощь программа проверки грамматики и орфографии

Атрибут **lang** указывает код содержимого элемента и значений атрибутов; относится ли он к данному атрибуту, зависит от синтаксиса и семантики атрибута и от операции.

Атрибут **lang** предназначен для того, чтобы позволить браузерам более осмысленно генерировать изображение на основе принятой культурной практики для данного языка. Это не подразумевает, что браузеры должны генерировать символы, не являющиеся типичными для конкретного языка, менее осмысленным способом; браузеры должны пытаться сгенерировать се символы, независимо от значения, указанного в атрибуте **lang**.

Например, если в русском тексте должен появиться символ греческого алфавита:

```
<P><Q lang="ru">"Эта супермощность была результатом &gamma;-
радиации,</Q> объяснил он.</P>
```

Браузер:

- должен попытаться сгенерировать русский текст соответствующим образом (например, в соответствующих кавычках)

- попытаться сгенерировать символ ?, даже если это не русский символ.

Коды языков

Значением атрибута **lang** является код языка, идентифицирующий естественный разговорный язык, который устно, письменно или иным образом используется для передачи информации между людьми. Компьютерные языки явным образом исключены из кодов языков.

Кратко говоря, коды языков состоят из первичного кода и ряда подкодов, который может быть пустым:

```
код-языка = первичный-код ( «-» подкод )*
```

Вот несколько примеров кодов языков:

en

английский

en-US

американская версия английского.

en-cockney

кокни (диалект английского).

i-navajo

навахо (язык американских индейцев).

x-klingon

Первичный код «x» обозначает экспериментальный код языка

Двухбуквенные первичные коды зарезервированы для сокращений языков по стандарту ISO639. Сюда входят коды **fr** (французский), **de** (немецкий), **it** (итальянский), **nl** (голландский), **el** (греческий), **es** (испанский), **pt** (португальский), **ar** (арабский), **he** (иврит), **ru** (русский), **zh** (китайский), **ja** (японский), **hi** (хинди), **ur** (урду) и **sa** (санскрит).

Любой двухбуквенный подкод считается кодом страны в стандарте ISO3166.

Наследование кодов языков

Элемент наследует информацию о коде языка в следующем порядке старшинства (от высшего к низшему):

Атрибут **lang**, установленный для самого элемента.

Самый близкий родительский элемент, для которого установлено значение атрибута **lang** (то есть атрибут **lang** наследуется).

Заголовок HTTP «**Content-Language**» может конфигурироваться на сервере. Например:

```
Content-Language: en-cockney
```

В примере, показанном ниже, первичным языком документа является французский («fr»). Один абзац объявлен на испанском языке («es»), после чего язык снова становится французским. В следующий абзац включена японская фраза («ja»), после чего язык опять изменяется на французский.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML lang="fr">
<HEAD>
<TITLE>Un document multilingue</TITLE>
</HEAD>
<BODY>
...текст интерпретируется как французский...
<P lang="es">... текст интерпретируется как испанский...
<P>... текст опять интерпретируется как французский...
<P>...французский текст, в котором попадается <EM lang="ja">фраг-
мент на японском</EM>, а здесь опять начинается французский...
</BODY>
</HTML>
```

Ячейки таблицы могут наследовать значения атрибута **lang** не от родителя, а из первой ячейки объединения.

Интерпретация кодов языков

В контексте HTML код языка должен интерпретироваться браузерами как иерархия знаков, а не один знак. Если браузер генерирует изображение в соответствии с информацией о языке (скажем, сравнивая языковые коды в таблицах стилей и значения атрибута **lang**), он всегда должен находить точное соответствие, но должен также принимать во внимание первичные коды. Таким образом, если значение атрибута **lang** «**en-US**» установлено для элемента HTML, браузер должен сначала выбрать информацию о стиле, совпадающую с «**en-US**», а затем сгенерировать более общее значение «**en**».

Иерархия кодов языков не гарантирует понимания всех языков с общими префиксами людьми, бегло говорящими на одном или нескольких из этих языков. Она помогает пользователю запросить эту общность, когда для пользователя она является истинной.

Указание направления текста и таблиц: атрибут dir

Определения атрибутов

```
dir = LTR | RTL [CI]
```

Этот атрибут задает основное направление нейтрального в смысле направления текста (например, текста, который не наследует направленность, как определено в UNICODE) и направление таблиц. Возможные значения:

- **LTR**: Слева направо.
- **RTL**: Справа налево.

Кроме задания языка документа с помощью атрибута **lang**, авторы могут указать основное направление (слева направо или справа налево) частей текста, таблицы и т.д. Это делается с помощью атрибута **dir**.

UNICODE назначает направление символам и определяет (сложный) алгоритм для определения соответствующего направления текста. Если документ не содержит отображаемых справа налево символов, браузер не должен использовать двунаправленный алгоритм UNICODE. Если документ содержит такие символы, и если браузер и отображает, он должен использовать двунаправленный алгоритм.

Хотя в Unicode определены специальные символы, отвечающие за направление текста, HTML предлагает конструкции разметки высшего уровня, выполняющие те же функции: атрибут **dir** (не спутайте с элементом **DIR**) и элемент **BDO**. Таким образом, чтобы привести цитату на иврите, проще написать

```
<Q lang="he" dir="rtl">...цитата на иврите...</Q>
```

чем с эквивалентными ссылками Unicode:

```
&#x202B;&#x05F4;...цитата на иврите...&#x05F4;&#x202C;
```

Браузеры не должны использовать атрибут **lang** для определения направления текста.

Атрибут **dir** наследуется, и его можно переопределить.

Использование разметки направленности HTML с символами Unicode

Авторы и разработчики средств создания HTML-документов должны знать о возможных конфликтах, возникающих при использовании атрибута **dir** со встроенными элементами (включая **BDO**) одновременно с соответствующими символами форматирования UNICODE.

Предпочтительнее использовать только один метод. Метод с использованием разметки гарантирует структурную целостность документа и устраняет некоторые проблемы с редактированием двунаправленного текста HTML в простых текстовых редакторах, но некоторое программное обеспечение может лучше использовать символы UNICODE. Если используются оба метода, следует хорошо позаботиться о правильном вложении разметки и символов, иначе результаты могут быть непредсказуемыми.

Приоритет над двунаправленным алгоритмом: элемент BDO

```
<!ELEMENT BDO - - (%inline;)*    -- I18N BiDi over-ride -->
<!ATTLIST BDO
 %coreattrs;        -- id, class, style, title --
 lang  %LanguageCode; #IMPLIED -- код языка --
 dir   (ltr|rtl)  #REQUIRED - направление --
>
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Определения атрибутов

```
dir = LTR | RTL [CI]
```

Этот обязательный атрибут указывает основное направление текстового содержимого элемента. Это направление имеет приоритет по отношению к наследуемому направлению символов, как определено в UNICODE. Возможные значения:

- **LTR**: Направление слева направо.
- **RTL**: Направление справа налево.

Атрибуты, определяемые в любом другом месте

```
lang (информация о языке)
```

Двунаправленного алгоритма и атрибута **dir** обычно достаточно для управления изменением направления внедренного текста. Однако в некоторых ситуациях двунаправленный алгоритм может привести к некорректному представлению. Элемент **BDO** позволяет авторам отключать двунаправленный алгоритм для выбранных фрагментов текста.

Элемент **BDO** следует использовать в сценариях, где необходим абсолютный контроль над последовательностью (например, многоязыковые номера частей). Атрибут **dir** для этого элемента является обязательным. Авторы могут также использовать специальные символы Unicode ля того, чтобы избежать использования двунаправленного алгоритма — **LEFT-TO-RIGHT OVERRIDE** (202D) или **RIGHT-TO-LEFT**

OVERRIDE (202E). Символ **POP DIRECTIONAL FORMATTING** (шестнадцатеричный код 202C) заканчивает любую последовательность, используемую для обхода двунаправленного алгоритма.

Помните, что при использовании атрибута **dir** во встроенных элементах (включая **BDO**) одновременно с соответствующими символами форматирования, могут возникать конфликты.

Существуют специальные соглашения относительно использования значений параметра «**charset**» для указания обработки двунаправленности в почте MIME, в частности для отличия визуальной, явной и неявной направленности. Значение параметра «ISO-8859-8» (для иврита) обозначает визуальную кодировку, «ISO-8859-8-i» обозначает неявную двунаправленность, а «ISO-8859-8-e» обозначает явную направленность.

Поскольку HTML использует двунаправленный алгоритм Unicode, соответствующие документы, использующие кодировку ISO 8859-8, должны помечаться как «ISO-8859-8-i». Явное управление направлением в HTML также возможно, но его нельзя выразить в ISO 8859-8, поскольку не следует использовать «ISO-8859-8-e».

Значение «ISO-8859-8» подразумевает, что документ отформатирован визуально, и некоторая разметка будет использоваться неправильно (например, **TABLE** с выравниванием по правому краю без разбивки строк), чтобы гарантировать правильное отображение для более старых браузеров, не поддерживающих двунаправленность. При необходимости документы можно изменить (и одновременно они будут корректно отображаться в старых версиях браузеров), добавив, где нужно, разметку **BDO**. Вопреки сказанному, кодировка ISO-8859-6 (арабская) не представляет визуального порядка.

Ссылки на символы для управления направлением и объединением

Поскольку иногда возникает двусмысленность относительно некоторых символов (например, символов пунктуации), UNICODE включает символы для правильного определения назначения. Unicode также включает некоторые символы для управления объединением при необходимости (например, в некоторых ситуациях с арабскими символами). HTML 4.0 включает для этих символов ссылки на символы.

Следующее **DTD** определяет представление некоторых объектов направления:

```
<!ENTITY zwnj CDATA "&#8204;"--=нулевая ширина без объединения-->
<!ENTITY zwj CDATA "&#8205;"--=объединитель нулевой ширины-->
<!ENTITY lrm CDATA "&#8206;"--=метка слева направо-->
```

```
<!ENTITY rlm CDATA "&#8207;"--=метка справа налево-->
```

Объект **zwnj** используется для блокировки объединения в тех контекстах, где объединение произойдет, но оно происходить не должно. Объект **zwj** имеет обратное действие; он производит объединение в случае, когда оно не предполагается, но должно произойти. Например, арабская буква «НЕН» используется для сокращения «Hijri», названия исламской системы летоисчисления. Поскольку отдельный иероглиф «НЕН» в арабской письменности выглядит как цифра пять, для того, чтобы не путать букву «НЕН» с последней цифрой пять в годе, используется исходная форма буквы «НЕН». Однако, нет последующего контекста (например, буквы для объединения), с которым можно объединить «НЕН». Символ **zwj** предоставляет такой контекст.

Точно так же в персидских текстах буква может иногда объединяться с последующей буквой, в то время как в рукописном тексте этого быть не должно. Символ **zwnj** используется для блокировки объединения в таких случаях.

Символы порядка — **lrm** и **rlm**, используются для определения направления нейтральных по отношению к направлению символов. Например, если двойные кавычки ставятся между арабской (справа налево) и латинской (слева направо) буквами, направление кавычек неясно (относятся ли они к арабскому или к латинскому тексту?). Символы **lrm** и **rlm** имеют свойство направления, но не имеют свойств ширины и разделения слов/строк.

Отражение глифов символов

Вообще двунаправленный алгоритм не отражает глифы символов и не влияет на них. Исключением являются такие символы как скобки. Если отражение желательно, например, для египетских иероглифов, греческих знаков или специальных эффектов дизайна, можно сделать это с помощью стилей.

Таблица стилей и двунаправленность

Вообще использование таблиц стилей для изменения визуального представления элемента с уровня блока до встроенного и наоборот используется в прямом направлении. Однако, поскольку двунаправленный алгоритм использует различия встроенных элементов/элементов уровня блока, во время преобразования нужно быть внимательным.

Если встроенный элемент, не имеющий атрибута **dir**, преобразуется в стиль элемента уровня блока с помощью таблицы стилей, для определения основного направления блока он наследует атрибут **dir** от ближайшего родительского элемента блока.

Если элемент блока, не имеющий атрибута **dir**, преобразуется к стилю встроенного элемента с помощью таблицы стилей, результирующее представление должно быть эквивалентным, в терминах двунаправленного форматирования, форматированию, получаемому путем явного добавления атрибута **dir** (которому назначено унаследованное значение) преобразованному элементу.

Глава 21. Списки

Язык HTML предлагает авторам несколько механизмов создания списков информации. В каждом списке должен быть один или несколько элементов списков. Списки могут содержать:

- Неупорядоченную информацию.
- Упорядоченную информацию.
- Определения.

Предыдущий список, например, не упорядочен, он создан с помощью элемента **UL**:

```
<UL>
<LI>Неупорядоченную информацию.
<LI>Упорядоченную информацию.
<LI>Определения.
</UL>
```

Упорядоченный список, создаваемый с помощью элемента **OL**, может содержать информацию, в которой важен порядок, например, рецепт:

```
1. Тщательно смешать сухие ингредиенты.
2. Влить жидкость.
3. Смешивать 10 минут.
4. Выпекать в течение часа при температуре 300 градусов.
```

Списки определений, создаваемые с помощью элемента **DL**, могут содержать ряд пар термин/определение (хотя списки определений могут иметь и иные применения). Например, список определений можно использовать в рекламе изделия:

```
Низкая цена
Новая модель этого изделия существенно дешевле предыдущей!
Проще работа
Мы изменили изделие, так что с ним теперь легко работать!
```

Безопасно для детей
Вы можете оставить своих детей в комнате, и изделие не причинит им вреда (не гарантируется).

На языке HTML он определяется следующим образом:

```
<DL>
<DT><STRONG>Низкая цена</STRONG>
<DD> Новая модель этого изделия существенно дешевле предыдущей!
<DT><STRONG>Проще работа</STRONG>
<DD>Мы изменили изделие, так что с ним теперь легко работать!
<DT><STRONG> Безопасно для детей </STRONG>
<DD> Вы можете оставить своих детей в комнате, и изделие не причинит им вреда (не гарантируется).
</DL>
```

Списки могут быть вложенными, разные типы списков можно использовать вместе, как в следующем примере, где список определений содержит неупорядоченный список (ингредиенты) и упорядоченный список (процедуру):

Ингредиенты:

- 100 г муки
- 10 г сахара
- 1 стакан воды
- 2 яйца
- соль, перец

Процедура:

1. Тщательно смешайте сухие ингредиенты.
2. Влейте жидкие ингредиенты.
3. Смешивайте 10 минут.
4. Выпекайте в течение часа при температуре 300 градусов.

Примечания:
Можно добавить изюм.

Точное представление трех типов списков зависит от браузера. Не стоит использовать списки для создания отступов в тексте. Это делается с помощью таблиц стилей.

Неупорядоченные списки (UL), упорядоченные списки (OL) и элементы списков (LI)

```
<!ELEMENT UL - - (LI)+      -- неупорядоченный список -->
<!ATTLIST UL
 %attrs;         -- %coreattrs, %i18n, %events --
 >
```



```
<!ELEMENT OL - - (LI)+      -- упорядоченный список -->
<!ATTLIST OL
 %attrs;         -- %coreattrs, %i18n, %events --
 >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

```
<!ELEMENT LI - O (%flow;)*     -- элемент списка -->
<!ATTLIST LI
 %attrs;         -- %coreattrs, %i18n, %events --
 >
```

Начальный тэг: обязателен. Конечный тэг: не обязателен.

Определения атрибутов

`type = информация о стиле [CI]`

Нежелателен. Этот атрибут устанавливает стиль элемента списка. Доступные в настоящее время значения предназначены для визуальных браузеров. Возможные значения описаны ниже (включая информацию о регистре).

`start = число [CN]`

Нежелателен. Только для **OL**. Этот атрибут задает начальный номер первого элемента в упорядоченном списке. По умолчанию начальный номер — «1». Помните, что, хотя значением этого атрибута является целое число, соответствующая метка может быть нецифровая. Если в качестве стиля выбраны латинские буквы верхнего регистра (A, B, C, ...), start=3 означает «C». Если в качестве стиля выбраны римские цифры нижнего регистра, start=3 означает «iii» и т.д.

`value = число [CN]`

Нежелательно. Только для **LI**. Этот атрибут устанавливает номер текущего элемента списка. Помните, что, хотя значением атрибута является целое число, соответствующая метка может быть нечисловая.

`compact [CI]`

Нежелателен. Если этот логический атрибут установлен, он сообщает визуальным браузерам о том, что генерировать список нужно более компактно. Интерпретация этого атрибута зависит от браузера.

Атрибуты, определяемые в любом другом месте

`id, class (идентификаторы в пределах документа)`
`lang (информация о языке), dir (направление текста)`
`title (заголовок элемента)`
`style (встроенная информация о стиле)`
`onclick, ondblclick, onmousedown, onmouseup, onmouseover,`

onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
(внутренние события)

Упорядоченные и неупорядоченные списки генерируются одинаково за исключением того, что визуальные браузеры нумеруют упорядоченные списки. Браузеры могут представлять эти номера несколькими способами. Элементы неупорядоченного списка не нумеруются.

Оба эти типа списков состоят из последовательностей элементов списков, определяемых элементом **LI** (конечный тэг которого можно опустить).

В этом примере показана общая структура списка.

```
<UL>
 <LI> ... первый элемент списка...
 <LI> ... второй элемент списка...
 ...
</UL>
```

Списки могут быть вложенными.

Пример нежелательного использования

```
<UL>
<LI> ... Уровень один, номер один...
<OL>
<LI> ... Уровень два, номер один...
<LI> ... Уровень два, номер два...
<OL start="10">
<LI> ... Уровень три, номер один...
</OL>
<LI> ... Уровень два, номер три...
</OL>
<LI> ... Уровень один, номер два...
</UL>
```

Информация о порядке номеров

В упорядоченных списках невозможно продолжать нумерацию автоматически из предыдущего списка или убрать нумерацию для некоторых элементов. Однако авторы могут пропустить несколько элементов списка, установив для них атрибут **value**. Нумерация для последующих элементов списка продолжается с нового значения. Например:

```
<ol>
<li value="30"> элемент списка номер 30.
<li value="40"> элемент списка номер 40.
<li> элемент списка номер 41.
</ol>
```


Списки определений: элементы DL, DT и DD

```
<!-- списки определений - DT - термин, DD - его определение -->

<!ELEMENT DL - - (DT|DD)+    -- список определений -->
<!ATTLIST DL
 %attrs;         -- %coreattrs, %i18n, %events --
 >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

```
<!ELEMENT DT - O (%inline;)*    -- термин -->
<!ELEMENT DD - O (%flow;)*     -- определение -->
<!ATTLIST (DT|DD)
 %attrs;         -- %coreattrs, %i18n, %events --
 >
```

Начальный тэг: обязателен. Конечный тэг: не обязателен.

Атрибуты, определяемые в любом другом месте

id, class (идентификаторы в пределах документа)
lang (информация о языке), dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле)
onclick, ondblclick, onmousedown, onmouseup, onmouseover,
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
(внутренние события)

Списки определений незначительно отличаются от других типов списков — тем, что элементы состоят из двух частей: термина и определения. Термин обозначается с помощью элемента **DT** и может иметь только встроенное содержимое. Описание указывается с помощью элемента **DD**, имеющего содержимое уровня блока.

Пример:

```
<DL>
 <DT>Dweeb
 <DD>young excitable person who may mature
 into a <EM>Nerd</EM> or <EM>Geek</EM>
 <DT>Cracker
 <DD>hacker on the Internet
 <DT>Nerd
 <DD>male so into the Net that he forgets
 his wife's birthday
</DL>
```

Вот пример с несколькими терминами и определениями:

```
<DL>
```

```
 <DT>Center
 <DT>Centre
 <DD> A point equidistant from all points on the surface of a
sphere.
 <DD> In some field sports, the player who holds the middle posi-
tion on the field, court, or forward line.
</DL>
```

Другим применением элемента **DL**, например, может быть разметка диалогов, где каждый элемент **DT** означает говорящего, а в каждом элементе **DD** содержатся его слова.

Визуальное отображение списков

Таблицы стилей предоставляют большие возможности управления форматированием списков (например, в отношении нумерации, соглашений, используемых в разных языках, отступов и т.д.). Визуальные браузеры обычно сдвигают вложенные списки соответственно уровню вложенности. Для элементов **OL** и **UL** атрибут **type** определяет параметры генерации для визуальных браузеров.

Для элемента **UL** возможными значениями атрибута **type** являются **disc**, **square** и **circle**. Значение, используемое по умолчанию, зависит от уровня вложенности текущего списка. Эти значения не учитывают регистр. Представление каждого значения зависит от браузера. Браузеры должны пытаться представлять «disc» в виде небольшого заполненного кружка, «circle» — в виде окружности, а «square» в виде небольшого квадрата.

Графические браузеры могут генерировать их как:

- ● для значения «disc»
- ❍ для значения «circle»
- ❑ для значения «square»

Для элемента **OL** возможные значения атрибута **type** приведены в следующей таблице (они учитывают регистр):

Type	Стиль нумерации
1 арабские цифры	1, 2, 3, ...
a буквы нижнего регистра	a, b, c, ...
A буквы верхнего регистра	A, B, C, ...
i римские цифры в нижнем регистре	i, ii, iii, ...
I римские цифры в верхнем регистре	I, II, III, ...

Помните, что использование атрибута **type** нежелательно, и стили списков должны определяться с помощью таблиц стилей.

Например, с помощью **CSS** можно указать, что стиль нумерации для элементов списка в нумерованном списке — римские цифры нижнего регистра. В приведенном ниже примере каждый элемент **OL**, принадлежащий классу «withroman», обозначается римской цифрой.

```
<STYLE type="text/css">
OL.withroman { list-style-type: lower-roman }
</STYLE>
<BODY>
<OL class="withroman">
<LI> Шаг один ...
<LI> Шаг два ...
</OL>
</BODY>
```

Генерация списка определений также зависит от браузера. Например, список:

```
<DL>
 <DT>Dweeb
 <DD>young excitable person who may mature
 into a <EM>Nerd</EM> or <EM>Geek</EM>
 <DT>Cracker
 <DD>hacker on the Internet
 <DT>Nerd
 <DD>male so into the Net that he forgets
 his wife's birthday
</DL>
```

может генерироваться следующим образом:

```
Dweeb
 young excitable person who may mature into a Nerd or Geek
Cracker
  hacker on the Internet
Nerd
  male so into the Net that he forgets his wife's birthday
```

Элементы DIR и MENU

Использование элементов **DIR** и **MENU** нежелательно.

Атрибуты, определяемые в любом другом месте

```
id, class (идентификаторы в пределах документа)
lang (информация о языке), dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле)
onclick, ondblclick, onmousedown, onmouseup, onmouseover,
```

```
onmousemove, onmouseout, onkeypress, onkeydown, onkeyup
(внутренние события)
```

Элемент **DIR** предназначался для создания многостраничных списков каталогов. Элемент **MENU** предназначался для использования в списках меню, состоящих из одного столбца. Оба эти элемента имеют ту же структуру, что и элемент **UL**, различаясь только при генерации. На практике браузеры генерируют списки **DIR** или **MENU** точно так же, как список **UL**.

Настоятельно рекомендуется использовать вместо этих элементов элемент **UL**.

Глава 22. Таблицы стилей

Таблицы стилей представляют наибольшее достижение для дизайнеров Web-страниц, расширяя возможности улучшения внешнего вида страниц. В научных средах, в которых и зародилась Web, люди более сосредоточены на содержании документов, нежели на их представлении. По мере открытия Web прочими людьми ограничения HTML стали источником разочарований, и авторам пришлось уклоняться от стилистических ограничений HTML. Хотя намерения и были добрыми — улучшение представления Web-страниц, — технологии имели нежелательные побочные эффекты. Эти технологии работали только для некоторых, только иногда, но не для всех и не всегда. Сюда включаются:

- Использование собственных расширений HTML
- Преобразование текста в изображения
- Использование изображений для управления пустым пространством
- Использование таблиц для размещения объектов на станице
- Написание программ вместо использования HTML

Эти технологии существенно усложняют Web-страницы, ограничивают гибкость, создают проблемы взаимодействия и создают сложности для людей с физическими недостатками.

Таблицы стилей решают эти проблемы, одновременно превосходя ограниченные механизмы представления в HTML. Таблицы стилей упрощают определение интервалов между строками текста, отступов, цветов, используемых для текста и фона, размера и стиля шрифтов и другой

информации. Например, следующая таблица стилей CSS (хранящаяся в файле «special.css») зеленый устанавливает цвет текста абзаца и окружает его сплошной красной рамкой:

```
P.special {
color : green;
border: solid red;
}
```

Авторы могут связывать таблицы стилей с исходным документом HTML с помощью элемента **LINK**:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
 "http://www.w3.org/TR/REC-html40">
<HTML>
 <HEAD>
 <LINK href="special.css" rel="stylesheet" type="text/css">
 </HEAD>
 <BODY>
 <P class="special">В этом абзаце текст должен быть зеленым.
 </BODY>
</HTML>
```

HTML 4.0 обеспечивает поддержку следующих функций таблиц стилей:

Гибкое размещение информации о стиле

Помещение таблиц стилей в отдельные файлы упрощает их повторное использование. Иногда полезно включать инструкции по представлению в документ, к которому они применяются, в начало документа или в атрибуты элементов в теле документа. Для упрощения управления стилем сайта применяется использование заголовков HTTP для установки таблиц стилей, применяемых к документу.

Независимость от языков таблиц стилей

HTML не привязан к конкретному языку таблиц стилей. Это позволяет использовать широкий диапазон таких языков, например, простые языки для большинства пользователей и более сложные для более специализированных случаев. Во всех примерах, приведенных ниже, используется язык CSS (Каскадные таблицы стилей), но можно использовать и другие языки.

Каскады

Эта возможность обеспечивается некоторыми языками таблиц стилей, такими как **CSS**, для объединения информации о стиле из не-

скольких источников. Это может быть, например, корпоративные положения о стиле, стили, общие для группы документов, а также стили, специфичные для одного документа. С использованием раздельного хранения эти таблицы стилей могут использоваться повторно, что упрощает работу авторов и повышает эффективность сетевого кэширования. Каскад определяет упорядоченную последовательность таблиц стилей, в которой правила более поздних таблиц имеют приоритет над более ранними. Не все языки таблиц стилей поддерживают каскады.

Зависимость от устройств

HTML позволяет авторам разрабатывать документы независимо от устройств. Это позволяет пользователям обращаться к Web-страницам с использованием различных устройств, например, графических дисплеев для компьютеров под управлением Windows, Macintosh OS и X11, телевизионных устройств, специальным образом адаптированных телефонов и портативных устройств на базе PDA, речевых браузеров и тактильных устройств на базе азбуки Бройля.

Таблицы стилей, напротив, применяются к конкретным устройствам или группам устройств. Таблица стилей, предназначенная для экрана, может применяться при печати, но бесполезна для речевых браузеров. Это позволяет вам определить широкие категории устройств, к которым применяется конкретная таблица стилей и позволяет браузерам избежать загрузки ненужных таблиц стилей. Языки таблиц стилей могут включать функции описания зависимости от устройств в одной таблице.

Альтернативные стили

Авторы могут предлагать читателям несколько способов просмотра документа. Например, таблица стилей для представления компактных документов с мелким шрифтом, или таблица, задающая крупные шрифты для удобства чтения. Авторы могут указать предпочитаемую таблицу стилей, а также альтернативные таблицы для определенных пользователей или устройств. Браузеры должны предоставлять пользователям возможность выбора одной из альтернативных таблиц или отключать все таблицы стилей.

Вопросы производительности

Некоторые пользователи высказывали сомнения относительно скорости работы таблиц стилей. Например, загрузка внешней таблицы стилей может привести к задержке общего представления материала для пользователя. Подобные ситуации возникают и в том случае, если в заголовок документа включен длинный набор правил относительно стиля.

Эти проблемы решаются путем предоставления авторам возможности включать инструкции по представлению в каждый элемента HTML. Благодаря этому информация о представлении всегда доступна ко времени представления элемента браузером.

Во многих случаях авторы воспользуются преимуществами использования общей таблицы стилей для группы документов. В этом случае распределение правил стиля в документе приведет к снижению производительности по сравнению с использованием связанной таблице стилей, поскольку для большинства документов таблицы стилей уже будет находиться в локальном кэше. К этому эффекту приведет общедоступность хороших таблиц стилей.

Как добавить стиль в HTML

Документы в формате HTML могут содержать правила таблиц стилей непосредственно или могут импортировать таблицы стилей.

В HTML можно использовать все языки таблиц стилей. Простого языка таблиц стилей может быть достаточно для большинства пользователей, в то время как другие языки могут подходить для более специализированных задач. В примерах ниже используется язык «Каскадные таблицы стилей», сокращенно CSS.

Синтаксис данных стиля зависит от языка таблицы стилей.

Установка языка таблицы стилей по умолчанию

Авторы должны указывать язык таблицы стилей для информации о стиле, связанной с документом HTML. Для установки языка таблицы стилей для документа по умолчанию следует использовать элемент **META**. Например, чтобы установить по умолчанию язык **CSS**, следует поместить в раздел **HEAD** следующее объявление:

```
<META http-equiv="Content-Style-Type" content="text/css">
```

Язык таблиц стилей по умолчанию можно также установить с помощью заголовков HTTP. Показанное выше объявление с использованием тэга **META** эквивалентно заголовку HTTP:

```
Content-Style-Type: text/css
```

Браузеры должны определять язык таблиц стилей по умолчанию для документа в соответствии со следующими шагами (от высшего приоритета к низшему):

- Если в объявлении **META** задается «**Content-Style-Type**», язык таблиц стилей определяет последнее объявление в потоке символов.

- В противном случае, если «**Content Style-Type**» задается в заголовках HTTP, язык таблиц стилей определяет последний заголовок в потоке символов.
- В противном случае по умолчанию используется язык «**text/css**».

Документы, включающие элементы, в которых устанавливается атрибут **style**, но не определяется язык таблиц стилей по умолчанию, являются некорректными. Средства разработки должна генерировать информацию о языке таблиц стилей по умолчанию (обычно с помощью объявлений **META**), чтобы браузеры не полагались на язык по умолчанию «**text/css**».

Встроенная информация о стиле

Определения атрибутов

```
style = style [CN]
```

Этот атрибут определяет информацию о стиле текущего элемента.

Атрибут **style** определяет информацию о стиле одного элемента. Язык таблиц стилей встроенных правил стиля определяется языком таблиц стилей по умолчанию. Синтаксис данных стиля зависит от языка таблиц стилей.

В данном примере устанавливается информация о цвете и размере шрифта текста определенного абзаца.

```
<P style="font-size: 12pt; color: fuchsia">Что за прелесть эти
таблицы стилей!
```

В **CSS** объявления свойств имеют форму «имя: значение» и разделяются точкой с запятой.

Атрибут **style** может использоваться для применения определенного стиля к отдельному элементу HTML. Если стиль повторно используется для нескольких элементов, авторы должны использовать элемент **STYLE** для перегруппировки этой информации. Для оптимальной гибкости авторам следует определять стили во внешних таблицах стилей.

Информация о стиле в заголовке: элемент STYLE

```
<!ELEMENT STYLE - - %StyleSheet  -- информация о стиле -->
<!ATTLIST STYLE
 %i18n;         -- lang, dir, для использования с заголовком --
 type  %ContentType;
#REQUIRED -- тип содержимого языка стилей --
 media  %MediaDesc; #IMPLIED -- для использования с этими
```



```
устройствами --
 title  %Text;   #IMPLIED -- рекомендуемый заголовок --
 >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Определения атрибутов

```
type = content-type [CI]
```

Этот атрибут определяет язык таблиц стилей для содержимого элемента и имеет приоритет над языком таблиц стилей, используемых. Язык таблиц стилей указывается как тип содержимого (например, «**text/css**»). Авторы должны указать значение для этого атрибута; для него нет значения по умолчанию.

```
media = дескрипторы устройств [CI]
```

Этот атрибут задает целевое устройство для информации о стиле. Это может быть один дескриптор устройства или список дескрипторов, разделенных запятыми. По умолчанию устанавливается значение «**screen**».

Атрибуты, определяемые в другом месте

```
lang (информация о языке), dir (направление текста)
```

Элемент **STYLE** позволяет авторам помещать правила таблиц стилей в раздел **head** документа. В HTML допустимо любое число элементов **STYLE** в разделе **HEAD**. Браузеры, не поддерживающие таблицы стилей или не поддерживающие определенный язык таблиц стилей, используемый в элементе **STYLE**, не должны показывать элемент **STYLE**. Ошибкой будет генерировать его содержимое как часть текста документа. Некоторые языки таблиц стилей поддерживают синтаксис для того, чтобы не показывать содержимое несоответствующим браузерам.

Синтаксис данных стиля зависит от языка таблицы стилей.

Некоторые реализации таблиц стилей могут поддерживать большее разнообразие правил для элемента **STYLE**, чем в атрибуте **style**. Например, в **CSS** правила могут объявляться в элементе **STYLE** для:

- Всех экземпляров определенного элемента языка HTML (например, для всех элементов **P**, всех элементов **H1** и т.д.)
- Всех экземпляров элемента HTML, принадлежащих определенному классу (т.е. для атрибута **class** в котором установлено определенное значение).
- Отдельных экземпляров элемента языка HTML (т.е. для атрибута **id** которого установлено определенное значение).

Правила приоритета и наследования правил таблиц стилей зависят от языка таблиц.

Следующее объявление **CSS STYLE** приводит к появлению границы вокруг всех элементов **H1** в документе и центрированию их на странице.

```
<HEAD>
 <STYLE type="text/css">
 H1 {border-width: 1; border: solid; text-align: center}
 </STYLE>
</HEAD>
```

Чтобы указать, что эта информация о стиле должна применяться только к элементам **H1** определенного класса, можно изменить определение следующим образом:

```
<HEAD>
 <STYLE type="text/css">
 H1.myclass {border-width: 1; border: solid; text-align: center}
 </STYLE>
</HEAD>
<BODY>
 <H1 class="myclass"> Наш стиль влияет на этот заголовок уровня
H1</H1>
 <H1> А на этот заголовок наш стиль не влияет</H1>
</BODY>
```

И, наконец, для ограничения области действия информации о стиле единственным экземпляром элемента **H1**, установите атрибут **id**:

```
<HEAD>
 <STYLE type="text/css">
 #myid {border-width: 1; border: solid; text-align: center}
 </STYLE>
</HEAD>
<BODY>
 <H1 class="myclass"> На этот заголовок H1 стиль не влияет</H1>
 <H1 id="myid"> А на этот влияет </H1>
 <H1> На этот опять не влияет </H1>
</BODY>
```

Хотя информация о стиле может устанавливаться почти для всех элементов HTML, два элемента, **DIV** и **SPAN**, особенно полезны тем, что они не накладывают никакой семантики представления (кроме block-level vs. inline). Вместе с таблицами стилей эти элементы позволяют пользователям неограниченно расширять язык HTML, особенно при использовании атрибутов **class** и **id**.

В следующем примере элемент **SPAN** используется для установки малых прописных букв для стиля шрифта первых нескольких слов абзаца.

```
<HEAD>
 <STYLE type="text/css">
 SPAN.sc-ex { font-variant: small-caps }
 </STYLE>
</HEAD>
<BODY>
<P><SPAN class="sc-ex">Первые несколько</SPAN> слов абзаца
выделены малыми прописными буквами.
</BODY>
```

В следующем примере мы используем элемент **DIV** и атрибут **class** для установки выравнивания текста для ряда абзацев, составляющих введение в научную статью. Информация о стиле может повторно использоваться для других разделов введения путем установки атрибута **class** в любом месте документа.

```
<HEAD>
 <STYLE type="text/css">
 DIV.Abstract { text-align: justify }
 </STYLE>
</HEAD>
<BODY>
 <DIV class="Abstract">
 <P>The Chieftain product range is our market winner for the com-
ing year. This report sets out how to position Chieftain against
competing products.
 <P>Chieftain replaces the Commander range, which will remain on
the price list until further notice.
 </DIV>
</BODY>
```

Типы устройств

HTML позволяет авторам создавать документы, использующие характеристики устройства, на котором будет представляться документ (например, графические дисплеи, телевизионные экраны, переносные устройства, речевые браузеры, тактильные устройства на базе азбуки Бройля и т.д.). С помощью атрибута **media** авторы могут позволить браузерам загружать и применять таблицы стилей выборочно.

Объявления в следующем примере применяются к элементам **H1**. При показе на проекторе во время встречи все экземпляры будут отображаться синим цветом. При печати они будут отцентрированы.

```
<HEAD>
 <STYLE type="text/css" media="projection">
 H1 { color: blue}
 </STYLE>
 <STYLE type="text/css" media="print">
 H1 { text-align: center }
 </STYLE>
```

Этот пример добавляет звуковые эффекты для устройства речевого вывода:

```
<STYLE type="text/css" media="aural">
A { cue-before: uri(bell.aiff); cue-after: uri(dong.wav)}
</STYLE>
</HEAD>
```

Управление устройствами особенно интересно при использовании с внешними таблицами стилей, поскольку браузеры могут сэкономить время, загружая из сети только таблицы стилей, применяющиеся к текущему устройству. Например, речевые браузеры могут не загружать таблицы стилей, разработанные для визуального представления.

Внешние таблицы стилей

Авторы могут отделять таблицы стилей от документов HTML. Это дает следующие преимущества:

- Авторы и менеджеры Web-сайтов могут совместно использовать таблицы стилей в ряде документов (и сайтов).
- Авторы могут изменять таблицы стилей без изменения документа.
- Браузеры могут загружать таблицы стилей выборочно (в зависимости от описаний устройств).

Предпочитаемые и альтернативные таблицы стилей

HTML позволяет авторам связывать с документом любое число внешних таблиц стилей. Язык таблиц стилей определяет взаимодействие нескольких внешних таблиц стилей (например, правила «каскадов» CSS).

Авторы могут указать ряд взаимоисключающих таблиц стилей, называемых альтернативными. Пользователи могут выбирать таблицы, которые им больше нравятся. Например, автор может указать один стиль

для небольших экранов, другой — для слабовидящих пользователей (например, с использованием крупного шрифта). Браузеры должны предоставлять пользователям возможности выбора одной из альтернативных таблиц.

Автор может указать, что одна из альтернатив является предпочтительной. Браузеры должны применять предпочитаемые автором таблицы стилей, если пользователь не выбрал другую альтернативу.

Авторы могут сгруппировать несколько альтернативных таблиц стилей (включая предпочитаемые автором) под одним именем стиля. Если пользователь выбирает именованный стиль, браузер должен применять все таблицы стилей с этим именем. Браузеры не должны применять альтернативные таблицы стилей с другим именем стиля.

Авторы также могут указать постоянные таблицы стилей, которые браузеры должны применять в дополнение к альтернативным таблицам стилей.

При применении таблицы стилей браузеры должны учитывать дескрипторы устройств.

Браузеры также должны позволять пользователям полностью отключать таблицы стилей автора; в этом случае браузер не должен применять ни одну из таблиц стилей.

Указание внешних таблиц стилей

Авторы указывают внешние таблицы стилей с помощью атрибутов элемента **LINK**:

- Установите в атрибуте **href** местоположение файла таблицы стилей. Значением атрибута **href** должен быть URI.
- Установите для атрибута **type** значение, указывающее язык связанного ресурса (таблицы стилей). Это позволяет браузерам не загружать таблицы стилей, использующие неподдерживаемые языки.
- Укажите, является ли таблица стилей постоянной, предпочитаемой или альтернативной:
  - Чтобы таблицы была постоянной, установите для атрибута **rel** значение «**stylesheet**», и не устанавливайте атрибут **title**.
  - Чтобы таблица была предпочитаемой, установите для атрибута **rel** значение «**stylesheet**», и дайте таблице имя с помощью атрибута **title**.

- Чтобы указать альтернативную таблицу, установите для атрибута **rel** значение «**alternate stylesheet**» а дайте таблице имя с помощью атрибута **title**.

Браузеры должны обеспечивать пользователям средства просмотра и выбора таблицы стилей из списка альтернатив. Для атрибута **title** рекомендуется устанавливать значение, которое будет представлять эту таблицу в списке.

В этом примере мы сначала определяем постоянную таблицу стилей, находящуюся в файле mystyle.css:

```
<LINK href="mystyle.css" rel="stylesheet" type="text/css">
```

Установка атрибута **title** назначает ее предпочитаемой автором таблицей:

```
<LINK href="mystyle.css" title="Compact" rel="stylesheet"
type="text/css">
```

Добавление ключевого слова «**alternate**» а атрибут **rel** сделает ее альтернативной таблицей стилей:

```
<LINK href="mystyle.css" title="Medium" rel="alternate stylesheet"
type="text/css">
```

Авторы также могут использовать для установки предпочитаемой таблицы стилей элемент **META**. Например, чтобы установить предпочитаемую таблицу стилей «**compact**» (см. предыдущий пример), авторы могут включить в элемент **HEAD** следующую строку:

```
<META http-equiv="Default-Style" content="compact">
```

Предпочитаемую таблицу стилей можно также указать с помощью заголовков HTTP. Объявление **META** выше эквивалентно заголовку HTTP:

```
Default-Style: "compact"
```

Если предпочитаемая таблица стилей указывается двумя или более элементами **META** или заголовками HTTP, преимущество имеет последнее объявление. Считается, что заголовки HTTP обрабатываются раньше, чем объявления **HEAD**.

Если предпочитаемая таблица стилей задается двумя или более элементами **LINK**, преимущество имеет первая.

Предпочитаемые таблицы стилей, задаваемые с помощью **META** или заголовков HTTP имеют преимущество над таблицами, задаваемыми элементом **LINK**.

Каскады таблиц стилей

Каскадные языки таблиц стилей, такие как **CSS**, позволяют использовать информацию о стиле из нескольких источников. Однако не все языки таблиц стилей поддерживают каскады. Чтобы определить каскад, авторы указывают последовательность элементов **LINK** и/или **STYLE**. Каскад информации таблиц стилей производится в порядке указания элементов в разделе **HEAD**.

В следующем примере мы определяем две альтернативные таблицы стилей с именем «**compact**». Если пользователь выбирает стиль «**compact**», браузер должен применять обе внешние таблицы, а также постоянную таблицу «common.css». Если пользователь выбирает стиль «**big print**», применяться будут только альтернативная таблица «bigprint.css» и постоянная таблица «common.css».

```
<LINK rel="alternate stylesheet" title="compact"
href="small-base.css" type="text/css">
<LINK rel="alternate stylesheet" title="compact"
href="small-extras.css" type="text/css">
<LINK rel="alternate stylesheet" title="big print"
href="bigprint.css" type="text/css">
<LINK rel="stylesheet" href="common.css" type="text/css">
```

Вот пример каскада, в котором задействованы оба элемента — **LINK** и **STYLE**.

```
<LINK rel="stylesheet" href="corporate.css" type="text/css">
<LINK rel="stylesheet" href="techreport.css" type="text/css">
<STYLE type="text/css">
 p.special { color: rgb(230, 100, 180) }
</STYLE>
```

Каскады, зависящие от устройств

Каскад может включать таблицы стилей, применяемые к различным устройствам. Элементы **LINK** и **STYLE** могут использоваться с атрибутом media. Браузер несет ответственность за отфильтровывание таблиц стилей, не применяющихся к текущему устройству.

В следующем примере мы определяем каскад, в котором таблица стилей «**corporate**» представляется в нескольких версиях: одна для печати, другая для экранного представления, третья для речевых браузеров (полезная, например, при чтении электронной почты в машине). Таблица «**techreport**» применяется ко всем устройствам. Цветная **rule**, определяемая элементом **STYLE**, используется для печати и для экрана, но не для звукового представления.

```
<LINK rel="stylesheet" media="aural" href="corporate-aural.css"
type="text/css">
<LINK rel="stylesheet" media="screen" href="corporate-screen.css"
type="text/css">
<LINK rel="stylesheet" media="print" href="corporate-print.css"
type="text/css">
<LINK rel="stylesheet" href="techreport.css" type="text/css">
<STYLE type="text/css">
 p.special { color: rgb(230, 100, 180) }
</STYLE>
```

Наследование и каскады

Если браузер собирается представлять документ, ему необходимо найти значения для свойств стиля, например, семейство шрифтов, начертание, размер шрифта, длину строки, цвет текста и т.д.

Точный механизм зависит от языка таблиц стилей, но в общем применяется следующее: механизм раскодирования используется, если к элементу применяется ряд правил стиля. Этот механизм позволяет браузеру сортировать правила по специфичности и определять, какое правило нужно применить. Если правило невозможно найти, следующий шаг зависит от наследования свойства. Не все свойства могут наследоваться. Для этих свойств язык таблиц стилей обеспечивает значения по умолчанию для использования в случае отсутствия явных правил для конкретного элемента.

Если свойство может наследоваться, браузер проверяет непосредственно элемент, в который вложен текущий элемент, и ищет правило, применяющееся к нему. Этот процесс продолжается до тех пора, пока применимое правило не будет обнаружено. Этот механизм обеспечивает компактное задание таблиц стилей. Например, авторы могут указать семейство шрифтов для всех элементов в разделе **BODY** с помощью одного правила для элемента **BODY**.

Как скрыть информацию о стиле от браузеров

Некоторые языки таблиц стилей поддерживают синтаксис, позволяющий авторам скрывать содержимое элементов **STYLE** от несоответствующих браузеров. В данном примере для **CSS** показано, как можно скрыть содержимое элементов **STYLE**, чтобы гарантировать, что более старые несоответствующие данной версии браузеры не будут представлять их в виде текста.

```
<STYLE type="text/css">
<!--
```



```
 H1 { color: red }
 P { color: blue}
 -->
</STYLE>
```

Привязка таблиц стилей с помощью заголовков HTTP

Менеджеры Web-серверов могут сконфигурировать сервер таким образом, чтобы таблица стилей применялась к группе страниц. Заголовок HTTP **Link** действует так же, как элемент **LINK**, с теми же атрибутами и значениями. Несколько заголовков **Link** соответствуют нескольким элементам **LINK** в том же порядке. Например,

```
Link: <http://www.acme.com/corporate.css>; REL=stylesheet
```

соответствует:

```
<LINK rel="stylesheet" href="http://www.acme.com/corporate.css">
```

Можно задать несколько альтернативных стилей с помощью нескольких заголовков **Link**, а затем использовать атрибут **rel** для определения стиля по умолчанию.

В следующем примере стиль «**compact**» применяется по умолчанию, поскольку в нем отсутствует ключевое слово «**alternate**» для атрибута **rel**.

```
Link: <compact.css>; rel="stylesheet"; title="compact"
Link: <bigprint.css>; rel="alternate stylesheet"; title="big
print"
```

Это работает и при отправке документов HTML по электронной почте. Некоторые браузеры электронной почты могут изменять порядок заголовков. Чтобы защитить стиль от изменения порядка каскадов для таблиц, задаваемых заголовками **Link**, авторы могут использовать объединение заголовков для объединения нескольких экземпляров одного и того же поля заголовка. Кавычки необходимы только в случае, если значения атрибутов включают пробелы. Используйте **SGML entities** для ссылок на символы, недопустимые в заголовках HTTP или электронной почты или символов, которые могут быть изменены при передаче через шлюзы.

Элементы **LINK** и **META** определяются как встреченные раньше явного элемента **LINK** и **META** в разделе **HEAD** документа.

Часть 4. Редакторы web-страниц

Глава 1. Основные требования

Как выжить в «войне браузеров»

Причина, по которой разработчики Web-узлов неохотно используют WYSIWYG-редакторы, состоит в том, что ведущие Web-браузеры Netscape Navigator и Microsoft Internet Explorer (IE) по-разному интерпретируют ключевые стандарты Web, принятые консорциумом World Wide Web Consortium (W3C) или находящиеся на стадии рассмотрения. Соответственно, в этих браузерах применяются разные форматы данных, которые различаются по ряду важнейших параметров.

Например, язык стилей CSS был утвержден в качестве стандарта более двух лет назад. Однако Microsoft IE поддерживает только 80% функций CSS1, а Netscape Navigator — еще меньше. Что же говорить о более новых технологиях, таких как HTML 4, Dynamic HTML, CSS2, JavaScript и других! Здесь степень расхождения еще больше.

Некоторые Web-дизайнеры вообще отказываются от применения HTML-редакторов и готовят код Web-страниц с помощью простого текстового редактора. Главным недостатком WYSIWYG-редакторов является то, что разработчик не имеет полного контроля над собственным кодом.

Непрекращающаяся «война браузеров» вынуждает разработчиков идти на всевозможные ухищрения, чтобы создавать Web-узлы, которые можно просматривать с помощью обоих браузеров, и конечно, их не устраивает такое положение. Они активно борются за свои права и даже создали общественную организацию Web Standards Project, которая требует поддержки Web-стандартов всеми браузерами. Microsoft и Netscape выражают готовность к обеспечению совместимости с основными Web-технологиями, но не торопятся выполнять эти обещания.

А пока ситуация не изменилась, разработчикам приходится отказываться от применения простых и удобных WYSIWYG-редакторов в

пользу специальных средств, предусматривающих возможность ручного кодирования Web-страниц.

Такие редакторы позволяют полностью контролировать процесс подготовки кода, непосредственно работать с HTML-тэгами и устанавливать собственные правила компоновки Web-страниц. Это не значит, что все приходится делать вручную. Профессиональные редакторы снабжены многочисленными программами-мастерами, помогающими разработчикам создавать таблицы, фреймы и другие сложные элементы. Кроме того, обеспечивается предварительный просмотр создаваемой Web-страницы в специальном окне (в режиме WYSIWYG).

Профессиональные HTML-редакторы прогрессируют так быстро, что разработчикам трудно уследить за последними новинками. Жестокая конкуренция в области средств подготовки Web-страниц и быстрое развитие Web-технологий заставляют производителей постоянно совершенствовать HTML-редакторы, дополняя их новыми функциями и расширяя возможности.

Основное требование к графическому редактору, который будет применяться для web-графики, это поддержка форматов JPEG, а особенно GIF, в т.ч. прозрачных, а для создания банеров почти обязательна поддержка анимированных GIF-ов, и как следствие возможность работы с многокадровыми и(или) многослойными изображениями.

Что касается обычной графики для web-страниц, то не советуем вам использовать анимированные гифы, если вы не супер-профи в области дизайна. Не вдаваясь в детали, заметим только, что 90% хороших профессионально сделанных сайтов обходятся без нее. И наоборот — 90% подобных «фишек» на web-страничках выдают непрофессионализм их создателей, и только раздражают посетителей.

Слегка оживить страничку лучше с помощью JavaScript, но и тут нужно чувство меры, и конечно понимание самого скрипта. Конечно, первой части этого требования удовлетворяют практически все ведущие графические редакторы, как растровые так и векторные. Надеемся, разница между ними вам известна, так что заметим лишь, что растровые пакеты, в целом, для web-графики подходят больше. Что касается создания ГИФ-анимации, то тут большинство монстров типа Фотошопа (Adobe Photoshop) или Корела (Corel Draw) пока не поспевают за временем, и этого делать не умеют. Впрочем большинство фирм-разработчиков графического ПО уже выпустили специализированные программы для Web.

Поскольку писать HTML-код можно в любом текстовом редакторе, под определение HTML-редактора подпадает практически каждая программа, способная сохранять текст как ASCII. Для написания до-

машней странички размером в 100 строк вам может хватить и «Блокнота» со справочником по HTML, однако, если ваша задача заключается в создании и/или поддержке ресурса на 20 и более страниц, вам, скорее всего, понадобятся функции автоматизации рутинной работы — цветовая разметка, расширенные поиск/замена, возможность просмотра редактируемых документов, поддержка макрокоманд и множество других.

Вот некоторые из таких редакторов...

Agile HTML Editor

Первое, что бросается в глаза, — легкость редактирования атрибутов тэгов. Для того чтобы получить к ним доступ, достаточно щелкнуть правой кнопкой мыши по тэгу. При этом появляется меню, предоставляющее возможность редактирования атрибутов тэга. Стоит заметить, что содержание этого меню напрямую зависит от контекста, в котором редактируемый тэг находится. Еще одна хорошая функция — **Snippets**, позволяющая вынести в специальное меню наиболее часто употребляемые куски кода. Тэги в Agile HTML Editor условно подразделены на четыре группы: **Quick**, **Structure**, **Formatting** и **Advanced**.

Группа **Quick** включает в себя наиболее используемые тэги, вроде **<img>**, **<a>**, **<p>** и др. Добавить сюда свои тэги, равно как и удалить имеющиеся почему-то нельзя. Группа **Structure** содержит тэги, определяющие структуру документа, **Formatting** — управляющие форматированием документа. Загадкой остается группа **Advanced**, в которую поместили все тэги, не вошедшие в первые три группы. Таким образом, в одной куче оказались элементы таблиц, форм, различные ActiveX-элементы и Java-апплеты, что вызывает некоторую путаницу.

Agile HTML Editor, как и большинство HTML-редакторов, имеет средства раскраски кода, предусмотрена возможность работы с шаблонами. Хотя, по заявлению разработчика, редактор «знает» и HTML 4.0, HTML 3.2 и HTML 2.0, а также особенности представления разных версий HTML в разных версиях двух популярных браузеров, в программу встроена документация только по HTML 3.2 и его реализации в Microsoft Internet Explorer 3.0. В качестве средств поиска/замены использованы стандартные функции «Блокнота», недостаточные для эффективной работы с содержимым даже небольшого сайта. Еще один отмеченный недостаток продукта — невозможность переопределить «горячие» клавиши. В целом же, несмотря на недоработки, Agile HTML Editor оставляет приятное впечатление, он вполне подходит для мелкой редактуры, однако пользоваться им «на производстве» пока рано.

Arachnophilia

Несмотря на свои небольшие размеры, Arachnophilia по основным параметрам успешно конкурирует с таким монстрами, как Homesite и Hotdog Professional, оставляя остальные редакторы далеко позади. Как и большинство программ, Arachnophilia делит все множество тэгов на несколько групп: **Fonts**, **Forms**, **Tables**... Все достаточно логично, хотя, например, сочетания Frames/JavaScript, Graphics/Misc, Links/Sound, Structure/Lists кажутся неудачными. С другой стороны, Arachnophilia позволяет редактировать как пункты меню, так и целые панели, удалять и создавать их одним щелчком мыши. Программа имеет встроенный, работающий в реальном времени механизм просмотра **Instant View**, использующий процессор Internet Explorer 3.0 и выше, а также собственный ftp-клиент, который возможно настроить так, что он будет отслеживать все изменения, сделанные на локальном компьютере, и в автоматическом режиме заменять соответствующие документы на web-сервере их новыми версиями.

Стоит отметить, что с помощью Arachnophilia возможна работа не только с HTML-документами, но и с Perl- и CGI-скриптами, Java-кодом, а также ASCII- и RTF-текстом. Конечно, не предполагается, что с помощью HTML-редактора вы станете создавать программы с тысячами строк кода или многостраничные rtf-документы — для этого существует специализированное ПО, однако для мелкой редактуры этих документов Arachnophilia сгодится. Механизм поиска/замены выполнен «на твердую четверку»: есть возможность поиска/замены во всех открытых документах, счетчик найденных соответствий. Имеется поддержка шаблонов, пошаговое руководство по созданию HTML-документов и JavaScript-сценариев, гибкие настройки клавиатурных сочетаний. Существенный недостаток — слабые средства раскраски кода: выделяются всего три группы: «тэг», «не тэг» и «параметр тэга». Тем не менее, Arachnophilia — один из лучших существующих на сегодняшний день HTML-редакторов, а в соотношении «цена/производительность/размер» он безусловный лидер.

Homesite

На сегодняшний день Homesite является одним из трех самых популярных среди профессионалов HTML-редакторов — число полученных им наград исчисляется десятками. Перечислять все функциональные возможности Homesite вследствие их огромного количества не представляется возможным, поэтому остановимся лишь на самых выдающихся, об остальных, например раскраске кода, работе с шаблонами или встроенном механизме просмотра, можно сказать, что они выполнены на самом высоком уровне. Homesite — это своего рода конструктор:

настраивается здесь практически все, от вида и содержания панелей инструментов до создания собственных спецификаций HTML. В Homesite есть все, что нужно практически любому HTML-кодеру или HTML-редактору. С помощью этого редактора можно не только верстать код, но и работать со структурой сайта. Для этого предусмотрен специальный инструментарий «**Проекты**», имеющий множество полезных компонентов, из которых особенно хочется отметить собственный ftp-клиент, ничем не уступающий даже специализированным программам, а по некоторым параметрам даже превосходящий их.

Хочется остановиться на механизме поиска/замены, аналогов которому нет, пожалуй, ни в одном существующем на сегодняшний день ПО, за исключением некоторых сред программирования. У программы есть не только стандартные «поиск/замена» в выделенном фрагменте кода, во всем тексте, сверху и снизу от курсора, полнотекстовый с учетом и без учета регистра, но и «расширенные», с возможностью работы со всеми открытыми документами, рекурсивно со всеми документами в указанном каталоге. Но самое главное: в механизме поиска/замены от Allaire есть практически полная поддержка регулярных выражений.

В Homesite встроен модуль проверки орфографии английского языка, а также есть возможность работы со словарями MS Word (в том числе и русским). Список поддерживаемых спецификаций исключителен: HTML 3.2, 4.0, Internet Explorer, Netscape Navigator, Gold Fusion, а также Synchronized Multimedia Integration Language (SMIL). Кроме инспекторов кода и правописания в Homesite предусмотрена проверка корректности ссылок. Если вы не сильны в HTML, можете воспользоваться многочисленными помощниками или средствами визуальной разработки, однако последнее использовать не рекомендуется (не советуют это делать и сами разработчики, позиционируя WYSIWYG-редактор Homesite лишь как эксперимент). Если же вы все-таки воспользуетесь этими средствами, к вашим услугам функция **Clear Sweep**, называемая в народе метлой. Главная и основная ее функция — очищать код от лишних тэгов, однако будьте осторожны — вы можете испортить «очищаемый» документ.

В Homesite можно узнать размер документа Сети с графикой или без, то есть вы сможете оценить, сколько времени он будет загружаться из Сети в зависимости от скорости доступа конечного пользователя. Подведя черту, скажу, что Homesite можно рекомендовать практически любому: от новичка, только что узнавшего значение тэга <a&qt;, до профессионального кодера, — ибо с его помощью можно не только научиться всем премудростям работы с HTML, но и эффективно управлять проектами с тысячами страниц.

Claris Home Page

Название этого редактора в полной мере отражает его предназначение: Claris Home Page — редактор для создания домашних страничек. Здесь он лучший. В нем нет ничего лишнего, только то, что может понадобиться вам для построения своей домашней страницы: набор популярных апплетов и скриптов, шаблоны для создания различного типа страниц, помощники, большая коллекция графики. Кроме того, в Claris Home Page встроен собственный ftp-клиент, который отслеживает изменения и синхронизирует их с версией на сайте. Достойно выполнена раскраска кода. Последняя версия включает в себя ряд функций, ориентированных скорее на профессионала, чем на владельца домашней странички. Среди них тесная интеграция с базами данных File Maker Pro, управление web-узлом на уровне структуры, глобальные поиск/замена, проверка ссылок и точек привязки, проверка орфографии (среди поддерживаемых языков русского, к сожалению, нет). В редакторе реализован целый ряд уникальных функций, незаслуженно забытых даже в профессиональных пакетах. Так, например, в редакторе предусмотрена возможность удаленного редактирования документов. Среди недостатков были отмечены слабая система помощи и бедная документация.

Hotdog Professional

Если вы когда-либо видели Homesite, то в Hotdog Professional вы вряд ли найдете что-то новое. Парадоксально, но Hotdog — практически точная копия Homesite. Конечно, некоторые отличия присутствуют, однако они в большинстве своем не в пользу Hotdog. Так, например, отсутствуют так любимые многими в Homesite расширенные поиск/замена, а в качестве стандартных поиска/замены использованы примитивные, встроенные в ОС. Еще один недостаток: отсутствие возможности проверки русской орфографии. Функция **Clear Sweep**, ставшая стандартом не только для HTML-редакторов, но и для пакетов визуальной разработки сайтов, также отсутствует.

У Hotdog Professional есть и преимущества перед своим братом-близнецом. Так, например, в отличие от Homesite, в Hotdog внутренний механизм просмотра результатов редактирования работает в режиме реального времени. Помочь сэкономить время сможет также поддержка макросов, выполненная на самом высоком уровне. Еще одна важная функция — clipboard с произвольным числом ячеек (стандартный clipboard имеет лишь одну ячейку), ограниченным лишь размерами памяти вашего компьютера.

Кроме того, для каждой из ячеек можно задать свое сочетание клавиш. Если работа над HTML-кодом ведется одновременно несколькими

людьми, то Hotdog будет просто незаменим — в нем реализовано разграничение доступа к частям проекта, определение задач для каждого из разработчиков, протоколирование производимых работ.

Несмотря на свое забавное название это серьезный продукт, обладающий массой полезных функций. А главное — он прост в применении, в чем и состоит его основное отличие от HomeSite, который в значительной степени ориентирован на специалистов. Редактор HotDog могут использовать все разработчики, от опытных программистов до новичков, для чего предусмотрено несколько пользовательских интерфейсов.

После первого запуска редактор спрашивает, насколько хорошо пользователь знаком с языком HTML. Выбрав один из трех уровней подготовки — **Beginner** (начальный уровень), **Intermediate** (промежуточный уровень) или **Hardcore** (верхний уровень), — можно сразу начинать работу. Например, интерфейс **Beginner** превращает HotDog в советчика, подробно и терпеливо объясняющего новичку суть многочисленных функций и режимов редактора. Это удобно, так как в пятой версии HotDog появилось немало новых функций, для освоения которых требуется время. Более опытные пользователи могут пропустить этап обучения, выбрав другой интерфейс.

Главное окно редактора HotDog похоже на окно текстового процессора, окруженное инструментальными линейками и встроенными функциями, которые предназначены для вызова программ-мастеров, макросов и других вспомогательных утилит, облегчающих разметку текста или создание кода. В главном окне имеется область предварительного просмотра **ROVER** (Real-time Output ViewER), в которой встроенный браузер динамически воспроизводит создаваемую Web-страницу в процессе ее редактирования.

Главное окно можно настраивать. Например, разработчик способен свернуть окно просмотра и работать только в окне редактирования, или, наоборот, раскрыть область просмотра на весь экран, или динамично переключаться между этими двумя режимами, причем редактор запоминает настройку главного окна, выбранную для каждого документа. Кроме того, предусмотрены средства настройки, которые позволяют строить страницы для экранов различного разрешения (640, 800, 1024 точек и др.), а также выводят линейки, помогающие выравнивать отдельные элементы страницы.

В левой части главного окна расположено поле, предназначенное для вывода номеров строк и свертывания выбранного сегмента кода. Чтобы удобнее переключаться между разными частями документа, можно использовать закладки. К сожалению, закладки и свернутые сегменты кода не сохраняются после окончания сеанса работы с редактором.

Уникальной особенностью HotDog является функция фильтрации тэгов. Нажав на кнопку Tag Filter, пользователь открывает окно, содержащее массу полезных сведений: список всех спецификаций языка HTML (от версии 2.0 до 4.0), специализированные тэги (тэги системы ColdFusion 3.0, события языка JavaScript, заданные пользователем тэги, элементы технологии WebTV), а также зависящие от браузера расширения языка HTML (поддерживаемые механизмом проверки правильности кода, который входит в состав HotDog). Достаточно выбрать мышью нужные технологии или версии HTML, и редактор подсветит все несовместимые с ними тэги красным цветом, указывая на синтаксические ошибки. Это очень удобная функция, которая облегчает работу с бесконечным количеством вариантов HTML.

Имеются и другие средства, упрощающие использование тэгов. Так, если выделить тэг мышью, появится всплывающая подсказка, которая содержит объяснение действий, выполняемых данным тэгом, а также окно с перечнем всех допустимых параметров и возможных окончаний тэга.

Для поддержки коллективной работы в HotDog предусмотрен модуль **Website** (аналог функции **Projects** редактора HomeSite). Если два разработчика попытаются одновременно открыть один файл, Website передаст им предупредительные сообщения. Кроме того, пользователь может работать с расположенными на различных узлах сети документами или другими элементами Web-страниц (например, графикой), не задумываясь об их объединении. Во время публикации Web-страницы на Web-узле HotDog автоматически интегрирует все использованные в ней элементы. Для работы с удаленным Web-узлом имеется встроенный FTP-клиент.

Чтобы ускорить редактирование атрибутов, следует воспользоваться функцией **Property Sheet**, открывающей на экране окно, в которое выводится описание атрибутов каждого тэга. С помощью данной функции можно также создавать макросы и клавиши быстрого вызова для любых тэгов. Имеются и средства оптимизации, ускоряющие загрузку Web-страниц. Например, функция **Bandwidth Buster** не только определяет, сколько времени будет загружаться какая-либо Web-страница, но и автоматически преобразует ее графику в формат, более удобный для использования в Web.

Предусмотрено великое множество различных средств, облегчающих подготовку Web-страниц. Так, редактор кнопок **Button Editor** позволяет накладывать текст на кнопки и добавлять к ним некоторые специальные эффекты. Удачно организован редактор таблиц и форм, имеется конвертер для преобразования звука и изображения в форматы потоко-

вого видео и аудио **RealAudio** и **RealVideo**, а утилита ICQ помогает организовать на Web-узле общение в режиме реального времени (чат).

Особого внимания заслуживает то, насколько тщательно создатели HotDog проработали функции, обеспечивающие выполнение даже самых мелких заданий в ходе подготовки Web-страниц. Например, выбирая цвет для Web-страницы, пользователь может выделить область в любой части экрана, увеличить ее и с помощью виртуальной пипетки подобрать нужный цвет. Редактор автоматически найдет наиболее близкий цвет в палитре браузера.

Пользовательский интерфейс HotDog отличается чрезвычайно высокой гибкостью. Можно настраивать почти все его элементы — установочные параметры, пиктограммы, инструментальные линейки, навигационные инструменты. Обратная сторона медали: интерфейс перегружен инструментальными линейками, поэтому открывается довольно медленно, а кроме того, каждая вызываемая функция открывает свое окно, и экран оказывается слишком переполненным. Пользователь должен очень тщательно настроить среду разработки, иначе главное окно редактирования окажется "погребенным" под грудой инструментальных линеек, пиктограмм и всплывающих окон.

Еще одним серьезным недостатком HotDog является его высокая цена. Однако данный недостаток компенсируется тем, что помимо HTML-редактора HotDog включает в себя ряд полезных дополнительных продуктов: графический редактор Paint Shop Pro компании JASC, программу Interactor фирмы mBed для работы с языком Dynamic HTML, утилиту анализа Web-узлов и проверки гиперсвязей Linkbot от Tetranet, а также интегрируемые модули (plug-in) SuperToolz, выполняющие различные дополнительные функции. Поэтому тем пользователям, которым требуются вспомогательные приложения, стоит приобрести HotDog, а те, кому нужен только редактор для HTML-кода, могут найти продукт и подешевле.

CoffeeCup HTML Editor ++

CoffeeCup отличает огромный набор различных помощников: визардов форм, изображений, таблиц. Благодаря этим визардам, из всех обозреваемых HTML-редакторов он наиболее близок к классу WYSIWYG-редакторов. В нем можно создать целый сайт, не прибегая к непосредственному редактированию исходного кода, вследствие чего для него свойственны недостатки визуальных средств разработки, такие как «лишний» код и слабые средства редактирования. Наряду с помощниками в CoffeeCup встроены также библиотеки CGI- и Java-скриптов, шаблоны DHTML, большая библиотека графики. Программа воспринимается как красивая игрушка — это, скорее, некий конструктор HTML, нежели HTML-редактор. CoffeeCup HTML Editor ++ будет интересен прежде всего тем, кто только начинает свое изучение HTML. Ему вряд ли найдется применение «на конвейере» — слишком слаб инструментарий редактирования.

WebEdit Pro

По разнообразию функциональных возможностей редактор WebEdit отстает от HomeSite и HotDog, однако содержит практически все основные средства, которые требуются профессиональному разработчику Web-узлов, и отличается удобным и простым интерфейсом. Дизайн пользовательского интерфейса очень похож на Microsoft Word — те же инструментальные линейки, расположенные между строкой меню и окном редактирования. Хотя инструментальные линейки нельзя настраивать, их можно перегруппировывать по своему усмотрению и даже буксировать в любое место экрана, превращая в "плавающие" палитры.

Одна из инструментальных линеек редактора представляет собой набор раскрывающихся списков, содержащих тэги и вспомогательные функции. Например, стоит нажать на кнопку **Font**, и появится список всех доступных шрифтов, а щелчок по кнопке **Table** позволяет запустить утилиту создания таблиц **Table Builder** или приступить к редактированию заголовков, строк и столбцов таблиц. С помощью кнопки **Custom** разработчик может добавлять собственные тэги, а редактор автоматически сопоставит с этими тэгами клавиши быстрого вызова. Нажатие клавиши на любом тэге вызывает диалоговое окно, содержащее подробную информацию о тэге, а также о браузерах и версиях HTML, которые его поддерживают.

WebEdit выполняет и подсветку синтаксических ошибок, причем пользователь имеет возможность настраивать этот процесс. Для проверки правильности кода служит встроенный механизм, который поддерживает все варианты HTML, включая версию 4.0. Многочисленные функции-мастера помогают создавать фреймы и формы в режиме WYSIWIG, проверять гиперсвязи, составлять оглавления, строить простые таблицы, импортировать в таблицы данные из ODBC-совместимых баз данных, добавлять к Web-страницам мультимедийные элементы (звук, видео, сценарии на языке JavaScript, анимацию и т. д.), а также составлять рейтинги Web-узлов по системам RSAC или SafeSurf. (Напомним, что RSAC и SafeSurf — это некоммерческие организации, которые разработали системы "рейтингования" Web-узлов с точки зрения их пригодности для детей и подростков).

Правда, если требуется создать более сложную таблицу или изменить существующую, приходится прибегать к ручному программированию. Кроме того, страницы с фреймами и страницы в каждом фрейме нужно открывать и редактировать в отдельных окнах, что не очень удобно. Однако это неудобство в некоторой степени компенсируется тем, что WebEdit — единственный из HTML-редакторов, который позволяет предварительно просматривать совокупности фреймов и страницы с фреймами в главном окне браузера.

Для коллективной работы предназначен модуль управления проектами **Project**, а для публикации Web-страниц на Web-сервере служит программа Web Publishing Wizard компании Microsoft. Кроме того, имеется встроенный FTP-клиент, облегчающий работу с удаленным Web-сервером.

WebEdit представляет собой компактный редактор, не перегруженный лишними функциями, который можно быстро загрузить по Интернет и начать с ним работу.

По единодушному мнению аналитиков и экспертов, WebEdit проигрывает по сравнению с HomeSite и HotDog. Например, он не обеспечивает возможности предварительного просмотра Web-страниц в разных браузерах, которая имеется в HomeSite, или развитых средств администрирования, предусмотренных в HotDog. Тем не менее опытным программистам, хорошо знакомым с HTML, может пригодиться этот полезный инструмент, который позволяет выполнять все основные задания при создании Web-узла и требует меньше системных ресурсов, чем конкурирующие продукты.

В целом, все рассмотренные редакторы можно отнести к двум группам: профессиональные и редакторы для начинающих. Среди первых бесспорными лидерами являются Homesite и Hotdog Professional. Трудно сказать, какой из них лучший — различия между ними незначительны; каждому стоит определиться самому, насколько для него важна та или иная функция. Среди непрофессиональных стоит отметить Arachnophilia. Эта небольшая, малотребовательная, к тому же бесплатная программа «умеет» очень и очень много. До уровня профессиональных пакетов она не дотягивает, однако для мелких работ ее более чем достаточно. Лучший HTML-редактор для начинающих — это, безусловно, CoffeeCup HTML Editor ++. С его помощью любой желающий сможет создать достаточно «навороченный» HTML-документ при минимуме знаний HTML, однако слабый инструментарий вряд ли удовлетворит создающих с его помощью серьезные коммерческие продукты.

Глава 2. Adobe ImageReady

Поначалу эту программу можно принять за «урезанную» версию Фотошопа, но в том что касается создания графики для Интернета, она гораздо удобнее, и обладает рядом уникальных возможностей.

Что добавлено по сравнению с Фотошопом:

- удобная, быстрая и гибкая оптимизация картинок для всех web-форматов, предварительный просмотр результатов сжатия «не отходя от кассы», прямо в процессе работы над изображением;
- отличный набор инструментов для создания гиф-анимации (пока это самый удобный и продвинутый гиф-аниматор); то, что он работает в одном формате с Фотошопом (PSD) делает возможным анимировать рисунки, до этого сделанные в нескольких слоях, при помощи его «старшего брата».
- мелкие, но приятные, дополнительные возможности: web-палитра; автоматическая резка рисунка по линейкам (задача, с которой часто сталкиваются web-дизайнеры, хотя что касается банеров, то их обычно резать необходимости нет); возможность обрезать размер холста, но при этом сохранить всю картинку, оставив часть ее временно вне пределов видимости (Clip вместо Crop). Или то, что шаг линеечек (Guidelines) в отличие от Фотошопа, привязан к размеру пиксела (это часто важно, чтобы изображения и линии были более четкими).

Часть возможностей, которых IR лишился по сравнению с Фотошопом, как например, поддержка цвета в форматах CMYK и др., действительно совершенна не нужна для Web. Других же, таких как работа с масками, модификация выбранных областей (**Select ⇨ Modify** или **Filer ⇨ Fade**), наверняка будет не хватать профессионалам, которые знают Фотошоп от и до.

Но все основные инструменты все же сохранились, как и привычный удобный отшлифованный интерфейс, так что для большинства повседневных задач IR вполне хватает. Зато он заметно меньше (упакованный дистрибутив 19 Мб), быстрее грузится и работает, и его можно скачать через Интернет. Таким образом, IR можно порекомендовать всем Adobe-ориентированным людям, как начинающим, так и профи.

Последним, возможно, будет удобнее использовать его в качестве инструмента для финального «web-тюнинга» своих работ.

ImageReady унаследовал от Фотошопа (хотя появился чуть раньше) и один гаденький глюк, в том что касается поддержки русских Unicode TrueType шрифтов (это те шрифты, что идут в комплекте с Win 98 — Arial Cyr и др.). Этот глюк лечится так же, как и у Фотошопа. Но хуже того, IR вообще сильно не дружит с русским текстом, с этим можно бороться, но не без потерь, да и удовольствия работе это, конечно, не добавляет.

Глава 3. Corel Xara

В начале отметим, что Corel Xara, это не совсем Корел, или даже совсем не Корел, хотя с виду и очень похоже, и любой человек, знакомый с Corel Draw, будет легко и свободно работать с этой программой. На самом же деле, несколько лет назад жила-была небольшая фирма Xara, которая написала такой хороший (удобный, быстрый и компактный) редактор векторной графики, что фирме Corel, не осталось ничего другого как просто купить опасного конкурента (кстати кое-что в Xar'е появилось раньше, чем в Corel Draw, например, такая удобная вещь, как регулируемая прозрачность объектов).

Не известно, что бы делали кореловцы дальше, и не наступи так быстро эпоха Интернета, вряд ли у Xar'ы было бы будущее как у отдельного графического пакета. Но Интернет наступил, и тут эти замечательные свойства Xar'ы оказались как нельзя кстати. Согласитесь, что программу, которая занимает всего около 8-ми мегабайт вместо Кореловских, как минимум, ста с гаком, распространять в сети гораздо проще. А возможности ее вполне сравнимы с возможностями Большого Корела (в смысле собственно Draw) или, скажем, Adobe Illustrator'а, или Macromedia FreeHand'а.

В общем программа многим нравится больше чем Corel Draw, пусть даже кое-чего делать и не может, но то, что она не может, по большей части, и не нужно. Зато работает феноменально быстро и освоить работу с ней гораздо проще. Опять же распространяется в Интернете.

И еще она может делать гиф-анимацию. Чтобы делать анимацию в Xar'е, нужно при создании файла задать ему соответствующий тип — Анимация. При этом активизируется несколько кнопочек на рабочей панели, позволяющие добавлять новые кадры, перемещаться взад-вперед, и т.п., плюс добавится несколько специальных пунктов в меню.

Каждый кадр, является по сути отдельным векторным изображением, аналогичным, к примеру файлу CDR. Набор средств для их рисования стандартный для векторного редактора. Окружности и эллипсы, прямоугольники, многоугольники и звездочки, кривые Безье, текст и различные инструменты по их слиянию, комбинированию, искажению (Edit envelope) и смешиванию (Blend). Различные линии, градиентные заливки, регулируемая прозрачность (тоже градиентная)... Текст вдоль кривой, правда без таких гибких настроек, как в Corel Draw.

Дополнительный плюс — неплохие средства для работы с растровыми изображениями, которые можно импортировать из популярных форматов, или конвертировать в растр нарисованные векторные объекты. После чего к ним можно к примеру применять похожие на Фотошоповские эффекты, типа Blur, корректировать цвета, яркость и контраст, разрезать и масштабировать, но уже как растровую графику.

Между собой различные кадры никак явно не связаны... Это, на первый взгляд, может показаться не очень удобным, но все необходимые средства для их синхронизации имеются. Объекты можно клонировать (дублировать без изменения позиции) и перебрасывать из одного кадра в другой. Плюс, есть линейки-указатели (guidelines), тоже общие для всех кадров. Можно копировать весь кадр целиком, и потом вносить во второй копии какие-то изменения. Есть опция, позволяющая видеть на экране и редактировать содержимое всех кадров одновременно.

Рабочее поле не ограничено, что позволяет работать с большим удобством, но создает и небольшие проблемы. Чтобы не запутаться, можно, к примеру создать рамку с необходимыми размерами (скажем, 468x60 для большого банера) и продублировать ее в одной и той же точной позиции, во всех кадрах. Можно ограничить рабочую область линеечками-указателями (guidelines).

Результат анимации можно посмотреть в отдельном окошке в самой Xare, а можно, что более полезно, сразу в браузере, в виде тут же сгенерированного гифа, уже в HTML-страничке с выбранным фоном. Причем чуть ниже, на той же страничке, будет указан размер файла и табличка времени его загрузки в зависимости от скорости модема.

Теперь о недостатках Corel Xara.

Основные недостатки Xara происходят от того, что это векторный редактор, и проявляются в принципе, и в других подобных программах, когда их начинают использовать для web-графики.

Суть в том, что рисунок в Xara, не зависит от разрешения, то есть от количества пикселов, и если в других областях графики это плюс, то

для Web это неожиданно оказывается минусом. Ведь теперешняя web-графика, к примеру тот же банер, напротив, сильно зависит от разрешения, особенно сильно, когда размер и количество пикселов, в которые нужно уложиться очень малы. Поэтому не советуем делать, скажем, мелкие банерики размера 88x31 в Corel Xara, иначе то, что вы получите в результате, может неожиданно разительно отличаться, от того, что вы видели и рисовали в редакторе.

Хотя уже для размера 468x60 использование Xara вполне возможно, хотя и с некоторыми своими тонкостями. Многие предпочитают GIFы, сделанные в Xar'e, дополнительно дорабатывать в растровом редакторе, по пикселам. Иначе, к примеру, нельзя поручиться, что обычная горизонтальная линия толщиной один пиксел не окажется размытой, попав на границу двух пикселов в финальной картинке.

Это бывает особенно полезным для анимированных изображений. Увы, сама Xara'a часто плохо их оптимизирует из-за того, что переводит одинаковые элементы в разных кадрах из векторного вида в растровый с небольшими различиями, а при этом сильно снижается качество сжатия.

Кто-то спросит, а стоит ли тогда вообще ей пользоваться?

Что ж, порой когда стоит цель сделать что-то быстрее, а стоит она очень часто, создать рисунок с нуля в векторном редакторе бывает на порядок быстрее, чем Фотошопе, особенно, когда нет возможности воспользоваться библиотекой готовых изображений.

Глава 4. Macromedia Fireworks

Программа Fireworks предлагает эффективное решение для профессиональных Web-дизайнеров, нуждающихся в надежных инструментах для создания и оптимизации Web-графики.

С ее помощью можно быстро создавать и редактировать растровые и векторные изображения, а также многокадровые GIF-анимации. Работа с Web-графикой в Fireworks оптимизирована так, чтобы пользователи могли легко и просто создавать интерактивные приложения. Эта программа позволяет в визуальной среде разрабатывать кнопки, rollover-эффекты и сложные навигационные схемы с многоуровневыми всплывающими меню. Кроме того, она поддерживает множество форматов графических файлов, а также позволяет быстро проектировать и создавать графику, HTML-коды и сценарии JavaScript.

Достоинства

Web-графика. Fireworks оптимизирует весть цикл работ по подготовке Web-графики. В уникальной рабочей среде Fireworks все элементы Web-дизайна постоянно остаются доступным для редактирования.

Интерфейс пользователя основан на знакомых и привычных свойствах известных программных продуктов Macromedia и других приложений для Web-дизайна.

Интерактивность

Программа позволяет добавлять интерактивные элементы, включая rollover-эффекты и всплывающие меню, не прибегая к программированию. При этом возможно пошаговое создание интерактивных элементов, подготовка rollover-эффектов при помощи мыши, а также визуальное создание и редактирование всплывающих меню благодаря инструменту Pop-up Menu Creator.

Оптимизация изображений

Создавайте компактную, высококачественную графику с минимальными усилиями. Превосходные средства оптимизации Fireworks 4 предоставляют полный контроль над процедурами цветовой оптимизации сегментированной графики.

Автоматизация рабочих процедур

Начинающие пользователи могут формировать команды с помощью панели History, которая записывает каждую операцию, а опытным дизайнерам пригодятся сценарии JavaScript для управления приложениями.

Анимация

Fireworks 4 позволяет создавать и просматривать многокадровые GIF-анимации с помощью нового интерфейса Live Animation. Возможна разработка и более сложных анимаций с автоматической прорисовкой промежуточных фаз.

Профессиональные инструментальные средства

Редактирование растровых изображений

В Fireworks 4 имеется все необходимое для редактирования и управления растровыми рисунками, включая поддержку цифровых камер и сканеров. При этом каждое изображение является независимым объектом, к которому применимы все функции редактирования.

Векторная графика

При обработке векторных изображений возможно быстрое создание кнопок и других объектов, размеры которых могут изменяться без потери качества. Можно копировать векторные объекты из других приложений через буфер обмена, а также импортировать файлы из программ FreeHand и Illustrator. При этом все слои будут корректно сохранены, а векторные объекты — доступны для редактирования в Fireworks .

Pop-Up Menu Creator

Этот инструмент позволяет начинающим дизайнерам формировать сложные многоуровневые всплывающие меню, не прибегая к программированию. Кнопки можно создавать либо рисованные, либо с помощью HTML-кода. Простая интуитивная визуальная среда позволит определить пункты и оформление всплывающего меню. Экспорт соответствующего сценария JavaScript обеспечит работу меню в Web-браузере.

Rollover-эффекты

Можно сконструировать сложные rollover-эффекты и сгенерировать для них сценарий JavaScript, даже не зная языка программирования — достаточно мышью переместить соответствующую пиктограмму, чтобы быстро создать кнопку и «оживить» ее при помощи rollover-эффектов.

Динамические анимации

Новые элементы управления позволяют создавать простейшие анимационные эффекты перемещения, масштабирования, вращения или затенения объектов. Достаточно выбрать объект и применить желаемый эффект. После создания анимации ее можно отредактировать с помощью экранных элементов управления, либо путем изменения параметров.

Маски объектов

Маскирование позволяет удалить нежелательные области изображений и добавить эффекты обрамления. Модернизированная панель Layers делает более удобным процесс создания и редактирования масок. В качестве масок могут применяться векторные объекты, оставаясь при этом полностью доступными для редактирования.

Промышленные стандарты и форматы файлов Интерфейс пользователя

Знакомый интерфейс пользователя позволяет работать со всеми программами для Web-дизайна компании Macromedia.

Основные элементы интерфейса включают Customizable Keyboard Shortcuts, Launcher Bar, структуру меню и управление панелями.

Импорт файлов Freehand 9

Можно быстро переориентировать традиционные проекты для использования в Web, импортируя и редактируя файлы FreeHand 9. Легко добавляются интерактивные элементы типа кнопок и rollover-эффектов.

Кривые Безье

Создание и редактирование кривых Безье знакомыми инструментальными средствами и способами, традиционными для программ Fireworks, Macromedia Flash и FreeHand.

Экспорт файлов Photoshop

Новая возможность экспорта файлов Photoshop позволяет организовать совместную работу дизайнеров и программистов. Можно экспортировать изображения Fireworks как файлы .psd и открывать их в программе Photoshop, в которой остаются доступными для редактирования текст, анимационные эффекты, слои и объекты.

Импорт EPS-файлов

Поддержка формата EPS позволяет использовать иллюстрации, ранее подготовленные для печати, и интегрировать библиотеки иллюстраций в Web-страницы.

Интеграция с приложениями Macromedia

Редактирование в Macromedia Flash

Из библиотеки символов достаточно загрузить исходный файл Fireworks или растровое изображение, а затем отредактировать графику, и она будет автоматически обновлена в Macromedia Flash.

Редактирование таблиц в Dreamweaver

HTML-таблицы и изображения могут быть отредактированы и изменены достаточно быстро — Fireworks 4 интегрирует новый код в существующий HTML-файл без потери внесенных вручную изменений.

Настраиваемые быстрые клавиши

Выберите привычные сочетания «горячих» клавиш. Не нужно помнить различные сочетания клавиш для выполнения сходных действий в разных программах — можно выбрать стандартные наборы клавиш для Fireworks, Photoshop, Illustrator или создать свои собственные.

Системные требования

- процессор Intel Pentium (рекомендуется Pentium II)
- операционная система Windows 98, Me, XP
- оперативная память 64 MB
- монитор разрешением 640 x 480 и поддержкой 256 цветов (рекомендуется разрешение 1024 x 768 и поддержка миллиона цветов)
- 100 Мб свободного места на жестком диске
- Adobe Type Manager со шрифтами Type 1

Многие профессионалы в области компьютерной графики, знают и используют графический редактор Macromedia Freehand, хотя он и уступает по популярности Corel Draw или Adobe Illustrator.

Для многих других хорошей рекомендацией будет, то что Macromedia же сделала такую программу как Dreamweaver, единственный пока WISIWIG HTML редактор, который смог добиться признания и уважения профессиональных web-мастеров. Или такой набор программ как Shockwave, Flash и Director, настолько удачных, что их формат стал уже стандартом для мультимедиа-информации в Интернете.

Короче, в фирме Macromedia явно хорошо чувствуют дух времени, и умеют идти с ним в ногу, делая качественные продукты, которые нравятся людям, и отвечают их потребностям. Поэтому стоит повнимательнее приглядеться к программе, которую они предлагают в качестве рабочего инструмента web-дизайнера для создания графики.

Те, кто работали с Freehand, сразу узнают в Fireworks («фейерверк») его младшего брата. Думаем, им тут будет легко и удобно. Действительно интерфейс, набор инструментов, и возможностей для рисования очень похож. Многим, в свое время, интерфейс Freehand'a не очень понравился, показался слегка громоздким, оставляющим слишком мало места для работы, при разрешении экрана 800x600. Поэтому сейчас мы не будем детально описывать, чем отличается Fireworks от своего предшественника, в этом смысле. Скажем только, что всяких примочек как и в случае с Corel Xara, тут конечно поменьше, но все основные средства рисования, и работы с объектами, стандартные для векторного редактора, в нем имеются.

Главное его отличие от других в том, что это не совсем векторный редактор. Помните, что было сказано про недостатки Xara? Так вот, у Fireworks их нет! (хотя конечно есть свои).

Так вот, недостатки векторного подхода были преодолены простым и элегантным способом — не смотря на то, что сами объекты и текст в FW векторные, для изображения в целом задается фиксированный размер в пикселах, и в процессе его редактирования все объекты сразу прорисовываются в том разрешении, в каком они будут потом экспортированы в gif (или jpeg).

То есть, простыми словами, если вы рассматриваете рисунок с увеличением в 200%, то пикселы на экране выглядят крупнее, совсем как в Фотошопе, но когда вы его по настоящему увеличиваете, потери качества не происходит!

Что касается набора средств для рисования, то он как и в Xare, довольно стандартный, хотя и кажется чуть победнее. Некоторые возможности, реализованы довольно нетипично и требуют достаточно сильного умственного напряжения, чтобы понять как ими пользоваться.

Но есть и свои плюсы, особенно для любителей делать кнопочки, и разные прочие штучки в одно касание. Помимо огромного выбора всевозможных линий и заливок, настраиваемых по многим параметрам (пожалуй даже побогаче многих больших редакторов), в FW каждый объект приобрел интересный атрибут — Effects. Среди эффектов — Inner и Outer Belevel, Glow, Emboss, которые благодаря, опять же, множеству настроек, позволяют быстро создавать тени, нимбы или добавлять объем любому тексту, или любой фигуре, или создавать пимпочки любой формы.

Удобно и просто реализована анимация. Имеется последовательность кадров, и небольшая панелька для управления ими — копирования, добавления, удаления. Любой объект в пару нажатий легко копируется в предыдущий-следующий, выборочные, или во все кадры сразу. А больше, в принципе, ничего и не нужно... Просмотреть анимацию, к сожалению, можно только в меню экспорта. Нужно только не забыть выбрать тип экспортируемого файла Animated GIF.

В том же меню мы получаем доступ ко множеству настроек оптимизации, и качество оптимизации, к примеру тех же анимированных ГИФов, очень высокое. Удобно реализована возможность порезать рисунок на кусочки, задав для каждого фрагмента свои настройки для оптимизации. Автоматизирован процесс создания Image Map'ов.

Как и в Xara, в FW можно конвертировать векторные объекты в растр, и применять к ним растровые спец-эффекты из большой галереи.

Теперь о недостатках.

Увы, есть и они. Самый большой — это следствие глюка в работе с русским текстом. Впрочем, в отличие от ImageReady, шрифты он видит

все и отображает буквы корректно. Беда в том, что у русских шрифтов полностью слетает кернинг, то есть настройки по ширине шага между буквами, который в большинстве шрифтов варьируется в зависимости от ширины самих букв. Короче говоря, текст просто слипается или, вообще, собирается в гармошку.

Приходится вручную растаскивать каждый раз все буковки, что иногда достает. Плюс надо более-менее понимать, что у буквы Ж, к примеру шаг отличается от буквы К. И даже более того, у буквы К отличается от буквы Л.

Хорошо, что текста в web-графике бывает обычно все же не так много. И доступ к настройкам кернинга FW дает практически полный для каждой буквы, и это в принципе, один из его плюсов.

Остальное, что можно причислить к недостатками, это уже упомянутая нестандартность в некоторых походах и идеях, и возможно все же нехватка каких-то инструментов. Но зато размер дистрибутива меньше 10 Мб, а все остальное преодолевается сообразительностью и смекалкой.

Итак, если вы готовы потратить немного времени на освоение возможно непривычного для вас интерфейса, и не спасуете перед мелкими заморочками и непонятками, то скорее всего Macromedia FireWorks вам понравится. И возможности, которые он предоставляет, сильно облегчат вам жизнь, а или даже откроют новые горизонты.

Глава 5. Macromedia Dreamweaver

Dreamweaver содержит все, что вам нужно как для визуальной компоновки web-страниц, так и для работы с HTML-кодом. Интуитивный интерфейс Dreamweaver настолько прост, что даже начинающий дизайнер сможет быстро создать профессиональный web-сайт. Непосредственно в Dreamweaver можно создавать анимации в формате Macromedia Flash, использовать данные из Microsoft Office, легко импортировать rollover-графику, меню и кнопки из Fireworks 4.

Достоинства

Полный контроль над исходным текстом

Представление Code View дает возможность получить доступ к новому интегрированному текстовому редактору, а с помощью представления Split View можно одновременно видеть исходный HTML-код страницы и ее изображение.

Быстрый редактор тэгов (Quick Tag Editor) позволяет отредактировать HTML-код любого объекта.

Проектирование сложных страниц

Новое уникальное средство компоновки **Layout View** позволяет вам рисовать непосредственно на странице и мышью перемещать ячейки или группы ячеек для создания вложенных таблиц. Независимо от того, ведется ли работа с готовыми таблицами или создается новая компоновка, **Layout View** всегда создает корректные таблицы, которые правильно отображаются в любом web-браузере.

Импорт из Microsoft Office

Команда **Clean Up Word HTML** удаляет специфические тэги, вставленные Microsoft Word, а объект **Table Object** позволяет импортировать в Dreamweaver файлы с данными, разделенными запятыми.

Отладка JavaScript

Новый отладчик JavaScript Debugger позволяет контролировать выполнение сценариев JavaScript в web-браузерах Netscape и Microsoft, помогая понять разницу в реализациях JavaScript. Справочная система. Быстрое получение справок по JavaScript, HTML, CSS и модели DOM (Document Object Model) благодаря новой функции Code Reference.

Создание элементов векторной графики непосредственно в Dreamweaver 4

Новые возможности **Flash Buttons** и **Flash Text** совместно с программой Macromedia Flash применяются для оформления кнопок и текста.

Управление ресурсами

С помощью новой панели **Asset** очень удобно вести учет всех ресурсов сайта. Эта панель позволяет управлять изображениями, цветами, внешними адресами URL, сценариями, видеоклипами Macromedia Flash, Shockwave и QuickTime, шаблонами и элементами библиотек.

Загрузка дополнений

С сайта Macromedia Exchange можно загрузить множество дополнений для Dreamweaver — от графики Flash и Fireworks до интерфейса к поисковым машинам. Сайт Macromedia Exchange также может оказаться полезным при совместном использовании объектов, сценариев и команд группой разработчиков.

Совместное редактирование HTML и графики в Dreamweaver и Fireworks

Средство **Roundtrip Graphic Editing** позволяет редактировать код и оптимизировать графику. При помощи соответствующих дополнений можно быстро добавить к web-сайту графические маркеры, кнопки и даже целый фотоальбом.

Управление информационным наполнением

Программа Dreamweaver интегрируется с Microsoft Visual SourceSafe, а также с любой из распространенных систем управления документами, которые используют протокол WebDAV. Компании Interwoven, Vignette, BroadVision, ATG и Documentum выбрали платформу Dreamweaver в качестве внешнего HTML-интерфейса для своих систем управления web-контентом.

Интегрированный текстовый редактор

В дополнение к Инспектору HTML, который представляет концепцию Roundtrip HTML, новый интегрированный текстовый редактор Dreamweaver делает более удобной обработку кода. Текстовый редактор вызывается с помощью Code View, а также из Инспектора HTML (Inspector HTML). Представление Split View позволяет видеть одновременно исходный HTML-код страницы и ее изображение, т. е. теперь можно просматривать проект при редактировании HTML-кода. В Code View также доступны системное меню Dreamweaver 4 и панель объектов (Object Panel).

Code View позволяет автоматически добавить отступы, выделить цветом HTML-код и JavaScript, проверить сбалансированность пунктуации, выровнять выделенную группу строк. Все это делается так же удобно, как и при работе в Design View.

Файлы, которые не содержат кода HTML, например, файлы JavaScript или XML, не изменяются при их открытии в Dreamweaver 4 — они автоматически открываются в представлении Code View.

Отладчик JavaScript

Новый отладчик JavaScript использует web-браузер для поиска ошибок в клиентской части кода JavaScript. Можно проследить, каким образом сценарий JavaScript выполняется в Netscape Navigator или Internet Explorer. Отладчик JavaScript может значительно сократить время поиска и устранения ошибок в программном коде.

Если отладчик находит ошибку, установите контрольную точку, затем проверьте объекты и свойства, которые были определены в

JavaScript. Пошаговое выполнение позволяет достаточно точно локализовать ошибку в программе.

При пошаговом выполнении программного кода можно проверять значения переменных, используя список переменных (watch list) в нижней половине окна отладчика. Это поможет обнаружить проблемы в сценарии JavaScript.

Режим компоновки

Для компоновки страницы дизайнеры всегда использовали HTML-таблицы, но это никогда не было простой задачей. Уникальное представление Layout View упрощает процесс компоновки, позволяя использовать таблицы как основу проекта, избегая при этом распространенных ошибок компоновки. Например, можно сначала нарисовать ячейки на странице, а затем переместить их в другое место и поменять их параметры. Можно задать фиксированную ширину ячеек или позволить ей изменяться при изменении размеров страницы. Независимо от того, ведется ли работа с готовыми таблицами или создается новая компоновка, Layout View всегда создает корректные таблицы, которые правильно отображаются в любом web-браузере.

Возможность применения вложенных таблиц поможет избежать некоторых проблем, связанных с табличной компоновкой. Так, если размеры ячейки увеличиваются, например, под влиянием изменения динамических данных, то это увеличение затронет и другие ячейки таблицы, нарушая параметры компоновки. Группируя ячейки при помощи вложенных таблиц, их можно защитить от влияния других ячеек и, тем самым, сохранить общую компоновку страницы.

Текст и кнопки Macromedia Flash

Программа Macromedia Flash проста в использовании, а Flash-графику способны отображать большинство браузеров и web-приложений. Теперь, используя средства Macromedia Flash Buttons и Macromedia Flash Text, можно достаточно просто добавить Flash-графику к web-страницам, созданным в Dreamweaver 4. Macromedia Flash Buttons и Macromedia Flash Text позволяют создавать доступную для редактирования векторную графику в Dreamweaver 4. Векторные объекты Macromedia Flash 5, заменяющие растровые изображения кнопок и текста, легко поддаются масштабированию, сокращают размеры файла и лучше выглядят при печати.

Эти новые возможности позволяют для создания графических текстовых объектов и кнопок в Dreamweaver 4 привлечь специалистов-дизайнеров, работающих с Macromedia Flash, или бесплатно загрузить

оформление кнопок с сайта Macromedia Exchange. Архитектура Macromedia Flash Objects является расширяемой, поэтому разработчики могут создавать новые объекты для Dreamweaver 4, например, график, диаграмму и даже "бегущую строку" новостей.

Глава 6. Metacreation Headline Studio

Фирма Metacreation, известная программами под маркой Kay Power (Tools, Kay Power Goo, Kay PhotoSoap), а также Metacreation Brice (генератор 3D ландшафтов) и недавно взявшая под свою эгиду графический редактор Fractal Design Painter (теперь Metacreatin Painter), наконец выпустила программу нацеленную и на рынок графики для Web.

Metacreation Headline Studio 1.0. многих заинтересует, так как все привыкли ждать от Кая Краузе (идеолога Metacreation) нетрадиционных интересных решений и новых возможностей в красивой стильной упаковке, и видеть в его программах альтернативу большим серым и скучным продуктам от монстров типа Adobe или Corel.

Хотя Headline Studio с виду тоже довольно серая, здесь вы наблюдаете все меню и окошки, которые в ней есть. Проведя мышкой над картинкой, можете узнать и их функции.

Остановимся лишь на основных особенностях этой программы и ее отличиях от других графических редакторов.

Специализация ее довольно узкая — она позволяет делать ГИФ-анимацию, и по замыслу ее авторов, наверное должна стать любимым инструментом создателей банеров.

Таким образом Headline Studio — это не полноценный графический редактор, а просто аниматор. Нарисовать вы здесь сможете только пару простейших фигур — прямоугольники и эллипсы, текст, или же экспортировать уже готовую картинку в формате GIF, TIFF или JPEG (для поддержки других форматов авторы рекомендуют пользоваться некими «плагинами»).

Все эти объекты вы можете разглядеть в меню инструментов. В том же меню видны трансформации, которым можно их повергать: масштабирование, сдвиг, поворот. Так же можно менять прозрачность, отбрасывать тень, размывать (Blur), и «размазывать» под разным углом, имитируя движение (Motion Blur), используя меню свойств объекта (меню с движками в центре).

Идеология создания анимации здесь совсем не такая, как скажем в Image Ready, Macromedia Fireworks или большинстве других программ такого типа. За основу здесь скорее взяты большие аниматоры из мира 3D, или возможно Flash, но конечно в очень урезанном виде.

Основная идея в том, что на этапе ее создания нам не нужно думать о кадрах, а только о времени и траекториях движения объектов. Для этого используется линия времени (timeline) и меню анимации. Для каждого объекта вы задаете контрольные точки на линии времени. Дальше вы просто изменяете положение объекта в этих точках (либо другие его свойства — прозрачность, размер, и т.п.), а программа сама генерирует все промежуточные кадры, в зависимости от траектории движения (изменения), которую вы можете задать в том же меню отдельно для каждого временного отрезка. К сожалению, вы не сможете сами нарисовать произвольную траекторию, как в серьезных аниматорах, а можете только выбрать ее из нескольких стандартных без каких-либо настроек. Вообще отсутствие или сильное ограничение свободы в задании многих параметров — это один из основных недостатков, который сильно испортил первое впечатление. Остается ждать следующей версии.

Второй серьезный недостаток — это размер файлов, получающихся на выходе... Причем, проблема не плохом качестве оптимизации, а самом подходе, при котором вы лишены доступа к детальной проработке каждого кадра. Можно, конечно, пытаться потом дорабатывать ГИФы в других редакторах, но трудно оценить насколько это будет эффективно. Скорее всего не слишком.

Headline Studio в ее теперешнем виде не слишком подходит в качестве полноценного web-инструмента (возможно и не претендуя на это). И может пока использоваться только как «генератор приколов». К сожалению, набор этих приколов сильно тоже ограничен небольшим количеством инструментов и отсутствием гибкости в настройках.

«Кинематографический» подход, используемый при создании анимации, пока недостаточно адаптирован к особенностям ГИФ-анимации, что тоже сильно ограничивает творческую свободу размерами файлов, допустимых в Интернете, особенно это касается банеров.

Из плюсов отметим малый размер дистрибутива — всего 3,5 мегабайта. Ну и довольно приятный интерфейс, который правда требует определенного привыкания. Правая клавиша никак не задействована, что немного обескураживает. Хелп в нормальном виде к сожалению отсутствует. Есть только Руководство в виде pdf. файла, который читать с экрана не очень удобно, но можно распечатать.

Глава 7. HomeSite

Редактор HomeSite быстро завоевал успех у разработчиков, создающих Web-страницы на платформе Windows. С самого начала этот редактор был ориентирован только на профессионалов, хорошо знакомых с Web-технологиями, что отличало его от большинства других средств разработки Web-страниц. Редактор HomeSite открывает разработчикам доступ к новейшим Web-технологиям, таким как Dynamic HTML (DHTML), SMIL, Cascading Style Sheets (CSS), ASP, Perl и JavaScript. Из ряда других профессиональных средств подготовки Web-страниц система выделяется наличием визуальной среды разработки, которая обеспечивает целостность кода, благодаря чему повышается производительность программистов, создающих и обновляющих Web-узлы. Разработчики могут быстро переключаться между тремя режимами:

- ручное кодирование на языках HTML, DHTML, SMIL, Perl и JavaScript (режим Edit View);
- работа в визуальной среде разработки (режим Design View);
- просмотр Web-страниц с помощью браузера (режим Browse View).

Процессы создания и редактирования HTML-кода и сценариев ускоряются за счет использования новых функций. Библиотека определений тэгов **Tag Definition Library** позволяет редактировать существующие тэги и добавлять новые. Когда разработчик вводит код, на экране возникает окно подсказки **Tag Completion**, содержащее допустимые атрибуты вводимого тэга (если это окно не требуется, его можно отключить). Очень удобна кнопка **Tag Insight**: нажав на нее, разработчик получает от редактора совет, какой тэг или атрибут ему может пригодиться (например, после ввода тэга, открывающего новый параграф, редактор рекомендует атрибут ALIGN).

Новая функция **Site View**, вызываемая через окно **Resource Bar**, графически показывает иерархическую структуру всего создаваемого Web-узла (включая Web-страницы и другие элементы), а опция **Tag Inspector** открывает быстрый доступ ко всем атрибутам тэга, если по нему щелкают мышью в окне редактирования. Новая объектная модель **Visual Tools Object Model** позволяет писать сценарии на языке VBScript или JavaScript, однако для этого требуется дополнительный механизм ActiveScripting компании Microsoft, который устанавливается вместе с браузером Internet Explorer.

В HomeSite предусмотрено множество клавиш быстрого вызова функций и тэгов, но если разработчику их не хватает, он может задавать свои собственные комбинации клавиш, что значительно повышает гибкость среды разработки. Средства настройки этим не исчерпываются. Существует дополнительная возможность приписывать кодам шаблоны, которые функционируют как клавиши быстрого вызова и автоматизируют работу программиста. Ему достаточно создать шаблоны, сопоставить их с требуемыми кодами (например, шаблон ftf можно сопоставить с кодом **<FONT FACE="Courier">**), ввести шаблоны в нужные места Web-страницы, а затем нажимать **Ctrl-J** — все шаблоны автоматически заменяются на соответствующие коды. Очевидно, что это существенно ускоряет разработку Web-страниц.

Для создания сложных элементов в HomeSite предусмотрены программы-мастера, которые помогают быстро строить таблицы со взаимосвязанными ячейками и наборы фреймов.

Пользовательский интерфейс редактора HomeSite также заслуживает внимания. Его можно настраивать в зависимости от потребностей, превращая то в простое окно редактирования, то в развитую среду разработки. Для этого предусмотрено множество вспомогательных панелей, таких как встроенное окно предварительного просмотра (снабженное линейками с делениями размером в пиксел), списки локальных файлов (со всплывающими списками часто используемых папок), перечни файлов удаленного доступа, к которым можно обращаться с помощью встроенного в систему FTP-клиента. Данный FTP-клиент позволяет считывать Web-страницы с удаленного сервера, редактировать их, а затем динамично публиковать обновленные страницы на Web-сервере. Кроме того, имеется панель, содержащая миниатюрные изображения создаваемых Web-страниц, которые можно буксировать в окно редактирования.

Для разработчиков, предпочитающих создавать Web-страницы не вручную, а автоматически, в четвертой версии HomeSite предусмотрен режим **Design View**. В нем можно буксировать объекты из одной ячейки таблицы в другую, выравнивать объекты, менять шрифты, готовить формы и выполнять другие действия визуального программирования, причем HTML-код формируется автоматически. Однако чтобы воспользоваться этим режимом, необходимо предварительно установить браузер IE. Если же режим **Design View** не нужен, его можно отключить.

С развитием Web-технологий растет сложность разработки Web-узлов. Над их созданием трудятся целые команды программистов, работу которых необходимо координировать и контролировать. Чтобы обеспечить интеграцию HomeSite со специальными средствами администрирования, встроенная в редактор функция управления проектами **Projects**

была полностью переработана. В результате четвертую версию редактора можно использовать вместе со средствами управления версиями кода (например, SourceSafe фирмы Microsoft). Это очень важно для координации действий большого коллектива программистов, совместно разрабатывающих и поддерживающих крупный Web-узел.

Что же касается встроенных средств администрирования, предусмотрена возможность удаленного управления проектом с помощью FTP-клиента. Кроме того, имеется программа-мастер, проверяющая состояние гиперссылок в одном или нескольких документах. Это необходимо для обеспечения надежности Web-узла в процессе его эксплуатации.

Дело в том, что со временем Web-страницы, на которые указывают гиперссылки, исчезают или перемещаются, а в результате появляются «висячие» ссылки, которые указывают вникуда. Пользователей чрезвычайно раздражают ложные указатели, поэтому создателям Web-узлов важно их вовремя находить и исправлять. Надежность кода обеспечивает и дополнительная программа проверки корректности HTML-кода CSE 3310 HTML Validator компании AI Internet Solutions, которая поставляется вместе с HomeSite.

Как добавить в IE HomeSite в качестве редактора страниц

Из IE, как вы знаете, можно отрывать страницы в различных HTML редакторах. По умолчанию в браузере установлены Notepad и Front Page. В данный список можно добавить в качестве редактора и HomeSite. Для этого необходимо:

- На панели IE должна быть кнопка «**Edit**», если ее нет, надо добавить.
- Загрузите с http://www.macromedia.com/v1/handlers/index.cfm? ID=10425&Method=Full файл для автоматического изменения реестра.
- Распакуйте файл и запустите вложенный файл, после чего появиться сообщение о добавлении изменений в реестр.
- Закройте и откройте IE. При нажатии кнопки «**Edit**», у вас в списке должен появиться Homesite.

Работа с изображениями в HomeSite

HomeSite предусматривает несколько способов добавления изображение в рабочий документ.

Вы можете добавить изображение используя диалоговое окно или просто перетащить файл изображения из окна Files List на web-страницу.

Добавление изображения с помощью диалогового окна

Установите курсов в то месте документа, куда вы хотите вставить изображение.

Щелкните на закладке **Common** на панели **QuickBar**.

Щелкните на пиктограмме **Image** (изображение). Откроется диалоговое окно в котором вы можете выбрать дополнительные варианты атрибутов изображения.

Выберете файл, который вам необходим используя при этом кнопку обзора. Установите дополнительные значения к атрибутам.

Щелкните **OK** для закрытия диалогового окна.

После закрытия диалогового окна, изображение вместе с кодом HTML будет добавлено в документ.

Добавление изображений из Files List с помощью перетаскивания

В окне **File List** выберете папку, где находиться требуемый файл. Выделив файл и удерживая правую кнопку мыши нажатой, перетащите его на web-страницу. Изображение вместе с кодом HTML будет добавлено в документ.

Просмотр и добавление изображений свернутых в пиктограмму

Данная функция позволяет быстро просматривать изображения в уменьшенном масштабе перед их добавлением в документ.

В окне **File List** выберете папку, где находиться требуемые файлы изображения.

Щелкните на панели инструментов пиктограмму **Thumbnails**.

Все изображения находящиеся в выбранной папке отобразятся в окне результатов в уменьшенном масштабе.

Щелкните на изображении правой кнопкой мыши, появится контекстное меню, в котором вы можете выбрать команду **Open** или **Properties** (Свойства). В новой версии HomeSite 5, данное контекстное меню пополнено новыми командами. Одна из них, а именно **Edit In Macromedia Fireworks** очень полезна. Удобно держать под «рукой» Fireworks для резки, оптимизации изображений и много другого.

Подцепив нужное изображение мышкой, перетащите его в рабочую область документа.

Проекты

Проект-это совокупность файлов используемых для создания WEB-сайта.

Проект позволяет разместить все необходимые файлы для создания WEB-сайта в легко доступном месте в рамках проекта HomeSite.

После завершения разработки сайта, вы можете загрузить весь проект на WEB-сервер и быть уверенным, что не один файл не будет потерян.

Обслуживающие операции, такие как поиск, перемещение и проверка линков можно осуществлять в проекте целиком, а не в каждом отдельном файле.

Папки Проекта

Папки служат основным средством организации файлов в Проекте. В HomeSite 4.5 существует два вида папок:

- **Physical folders** — Физические папки
- **Virtual folders** — Виртуальные папки

Физические папки относятся к определенной директории вашего компьютера.

Виртуальные папки служат репозиториями для файлов, не имеющих логический связи между собой. Вы можете использовать виртуальную папку как некий «контейнер», куда можно помещать любые файлы. Виртуальные папки позволяют организовать разбросанные файлы в одном месте HomeSite.

Физические Папки

Физические папки можно разделить еще на два вида:

- **Manual-inclusive** — Папка ручного включения
- **Auto-inclusive** — Папка автоматического включения

Папка ручного вложения требует, чтобы вы определили какие файлы в отображенной директории необходимо вложить в папку. Например, в вашей папке находиться 10 файлов и только три из них необходимы для WEB-сайта. Вы можете создать папку ручного вложения, поместив туда эти три файла.

Папка автоматического вложения включает каждый файл в отображенной директории по определению. Вы можете определить только тот формат файлов, который вы хотите включить (например, все файлы

HTML файлы). Папки автоматического включения обновляются автоматически при добавлении или удалении файлов из данной директории.

Виртуальная папка

Используйте, когда хотите организовать файлы из разных мест в одном месте HomeSite.

Папка ручного вложения (физическая)

Используйте, если вы хотите чтобы избранные файлы были включены в проект в специальной директории.

Папка автоматического включения (физическая)

Используйте, если вы хотите чтобы все файлы (или файлы определенного формата) были включены в проект в специальной директории.

Работа с папками и файлами Проекта

Как только вы создали свой проект, вам необходимо заполнить его папками и файлами. Процедуры, описанные в данном разделе, посвящены тому, как:

- Добавлять, править и удалять все типы папок (виртуальные, ручного вложения, автоматического вложения)
- Добавлять и удалять файлы из папок
- Открывать документы в выбранной вами папке

Работа с папками

При добавлении папки к проекту, вам необходимо решить, куда разместить данную папку. Вы можете добавить папку к корню проекта или к уже существующей папке. При добавлении к существующей папке, новая папка обозначается, как «дочерняя» (порожденная) папка, а существующая папка — как «родительская» (порождающая). Существуют некоторые ограничения, связанные с добавлением виртуальных папок. Виртуальные папки могут быть добавлены только к корню проекта или к другой виртуальной папке. Вы не можете добавлять виртуальные папки к папкам ручного или автоматического вложения, однако вы можете добавлять папки ручного и автоматического вложения к виртуальным папкам.

Добавление виртуальной папки:

- Кликните правой кнопкой мышки на «родительскую» папку для создания новой виртуальной папки (или же на корень проекта или на другую виртуальную папку),

выберите из меню команду **Add Folder** — Добавить Папку, перед вами появится диалоговое окно «**Добавить папку проекта**».

- Введите название папки в окно «**Название папки**».
- Кликните на значок «**Виртуальная папка**», а затем **OK**.

Добавление остальных типов папок осуществляется аналогичным образом, однако, с указанием директории.

Правка папки:

- Щелкните правой кнопкой мышки на папку и выберите из меню команду свойства. Перед вами появится диалоговое окно «**Правка свойств папки**».
- Внесите необходимые изменения и щелкните **OK**.

Открытие документов в папке:

- Щелкните правой кнопкой мышки на папку и выберите из меню команду **Open All Documents In Folder** — Открыть Все Документы. При этом все текстовые файлы будут открыты в окне редактора. Для нетекстовых файлов предлагается на выбор:
  - Открыть файл в сопутствующей программе
  - Открыть файл в окне редактора, если возможно
  - Пропустить файл.

Работа с файлами

Теперь, когда у вас уже есть представление о том, как работать с папками проекта, вы можете приступить к работе с файлами. Методы работы с файлами могут различаться в зависимости от типа папки, содержащие данный файл.

Файлы в папках ручного вложения и виртуальных папках

Процедура добавления и удаления файлов одинакова для папок ручного вложения и виртуальных папок.

Добавление файлов в виртуальную папку и папку ручного вложения:

- Щелкните правой кнопкой мышки на папку и выберите команду **Add Files to Folder** — Добавить файлы в папку. Появится диалоговое окно.

- Разместите и выберите файлы, которые вы хотите добавить в вашу папку проекта. Для выбора более, чем одного файла используйте клавиши **Shift** и **Ctrl**.
- Щелкните на команду **Open** — Открыть для добавления файлов в папку.

Удаление файлов из папки ручного вложения или из виртуальной папки:

- Выберите папку. Все файлы, содержащиеся в выбранной папке, отображены в окне **Files List** — Список файлов.
- Щелкните правой кнопкой мышки на файл, который вы хотите удалить и выберите команду **File ⇨ Delete ⇨ Файл ⇨ Удалить**.
- Щелкните **Yes**.

Файл удален.

Файлы в папках автоматического вложения

Вы можете контролировать файлы, хранящиеся в папке автоматического вложения, определяя типы файлов. Вы можете изменить типы файла.

Изменение типов файлов в папках автоматического вложения:

- Щелкните правой кнопкой мышки на папку и выберите **Properties** — Свойства из меню.
- Измените расширение для тех типов файлов, которые вы хотите вложить в папку. Вы можете изменить расширение с помощью «выпадающего» списка или вручную.
- Щелкните **OK**.

Список вложенных файлов будет отфильтрован в соответствии с расширениями, установленными вами.

Работа с ресурсами

Древо ресурсов позволяет видеть файлы вашего проекта, распределенные по типам. Например, кликнув на **HTML Documents** вы можете видеть список всех HTML файлов вашего проекта.

Древо ресурсов состоит из двух типов ресурсов **HTML** и **Images** (Изображения). Вы также можете создать ваши собственные ресурсы и определить типы расширения файлов для них.

Просмотр файлов определенного ресурса:

- Кликнете на ресурс, файлы которого вы хотите увидеть (например, если вы хотите увидеть все файлы HTML, кликнете на **HTML Documents**).

Добавление ресурса:

- Правой кнопкой мышки кликнете на **Resources** (Ресурсы) и выберете из меню команду Add **Resources** — Добавить Ресурс. Появиться диалоговое окно **Add Resource Folder**.
- Введите название ресурса в **Resource Name**.
- Введите расширения файла для данного типа ресурса в окно **Resource Filter**, разделяя их с помощью точки с запятой, если их более одного (например, для фильтра ресурса Каскадных листов стилей нужно внести — CSS).
- Щелкните **OK**

Новый ресурс добавлен в список ресурсов. Кликнув этот ресурс вы увидите список файлов с расширениями, которые вы установили.

Правка ресурса:

- Правой кнопкой мышки кликнете на **Resources** (Ресурс) и выберете в меню **Properties** (Свойства). Появиться диалоговое окно **Edit Resourse Folder**.
- Сделайте необходимые изменения названия или фильтров ресурса и щелкните **OK**.

Удаление Ресурса:

- Правой кнопкой мышки кликнете на **Resources** (Ресурс) и выберете из меню команду **Remove Resource** — Удалить Ресурс. Подтвердите, что вы действительно хотите удалить ресурс.
- Кликнете **Yes**.

Ресурс удален из списка Ресурсов.

Добавление, Правку и Удаление ресурсов можно осуществить другим способом а именно, выберете **Options ⇨ Settings** (Опции ⇨ Настройки) или (**F8**). Появится диалоговое окно **Settings** (Настройки). Щелкните на **Projects**, появиться окно **Project**.

Конфигурация внутреннего браузера

Для просмотра документов в HomeSite можно использовать три типа браузеров. Но в качестве внутреннего браузера можно установить только один по вашему выбору. Для установки опций, откройте панель **Options ⇨ Setting ⇨ Browse**.

Что вы можете выбрать из данных опций:

- Если у вас установлен Internet Explorer то вы можете использовать его в качестве внутреннего браузера.
- Если у вас установлен браузер Mozilla и NGLayout/Gecko контроль установлен, вы можете выбрать опцию использовать Netscape в качестве внутреннего браузера.
- Имеется также свой, встроенный браузер HomeSite, если вы захотели использовать его имейте в виду, что он имеет ограниченную поддержку HTML и расширений браузера.

Согласование страниц

По умолчанию при обзоре документа во внутреннем или внешнем браузере, документ открывается в локальной системе файлов или через FTP из удаленного сервера. Этого вполне достаточно для проверки содержания или форматирования страниц. Однако для разработки Web-сайта вам необходимо видеть динамические страницы так, как их будут видеть посетители.

Для этого вы можете переправить документы через Web-сервер. Программное обеспечение сервера может стоять как на локальной машине, так и на удаленной системе. Таким образом, вместо простого открытие файла, HTTP запрос посылается на сервер. При необходимости дополнительной обработки данных сервером, Web-сервер перепроверит страничку на соответствующий сервер для дополнительной обработки и затем вернет ее на браузер. Это особенно важно для предварительного просмотра приложений и элементов сайта в режиме испытания до развертывания сайта.

Вы устанавливаете данную маршрутизацию путем, ассоциирования физических директорий, где хранятся ваши файлы с сервером, где эти файлы отображаются. Такое ассоциирование называется согласованием (mapping). Большое количество Web-серверов имеют поддержку и вы можете создавать многочисленные отображения и выбирать сервер для обработки. Проконсультируйтесь у вашего провайдера или просмотрите описание к серверу на предмет каких-либо особенностей доступа на директории сервера.

Добавление согласования:

- Откройте панель **Setting (F8)** выберете **Browse** и выделите опцию **Enable Server**
- Щелкните на кнопку **Add**
- Введите путь согласуемого документа
- Введите URL сервера для согласования
- Щелкните **OK**

Если вы хотите обрабатывать документы с другого Web-сервера, вы должны прописать путь файла в Web-сервер.Нижеследующая процедура основана на типичной инсталляции IIS, процедура для вашей инсталляцией может отличаться. Если вы используете другой Web-сервер, ознакомьтесь с документацией сервера.

Конфигурация IIS для согласования сервера:

- Откройте **IIS Management Console** и разверните **IIS узел**
- Правой кнопкой мыши щелкните на **Default Web Server** (Web-сервер по умолчанию)
- Следуйте шагам указанными в **New Virtual Directory Wizard** для добавления директорий
- Следуйте шагам, указанным в **To Add** (Добавить)
- Не забудьте прописать полностью путь виртуальной директории в URL

Установка согласования по умолчанию:

- Откройте панель **Setting (F8)** ⇨ **Browse**
- Выберите элемент, который вы хотите переместить в список для согласования
- Щелкните на кнопку **Up Arrow** (Стрелка вверх) для перемещения элемента в начало списка
- Щелкните **OK**

Конфигурация внешнего браузера

После установки HomeSite, программа сформирует список Web-браузеров, которые были обнаружены в вашей системе. Список браузеров отобразиться в диалоговом окне, которое вызывается из меню командой **Options ⇨ Configure External Browsers**.

Вы можете легко добавлять, удалять, редактировать и управлять данным списком.

Браузер, стоящий первый в списке, установлен по умолчанию. Чтобы установить другой браузер по умолчанию, выберите из списка необходимый браузер, щелкните на кнопку со стрелкой вверх для перемещения его в начало списка.

Сохранение

Данная настройка позволяет определить, как обрабатывается текущий документ при загрузке во внешнем браузере. В HomeSite предусмотрено три возможности:

- **Prompt to save changes to the current document** — сохранение изменений текущего документа после вашего подтверждения
- **Automatically save changes to the current document** — автоматическое сохранение изменений текущего документа
- **Browse using a temporary copy (no need to save)** — использование временной копии (без сохранения)

Вы можете использовать любой из ниже перечисленных способов просмотра страниц во внешнем браузере:

- Для загрузки активного документа во внешнем браузере, установленном по умолчанию, нажмите **F11**. Затем вы можете обновить страницу после редактирования, нажав на кнопку **Refresh** (обновить) или еще раз нажать **F11**.
- Для того, чтобы переключиться на другой браузер, щелкните на кнопку **View External Browser** (Показать список внешних браузеров) на панели инструментов **View** и выберите браузер из списка.

Подсоединение к FTP-серверу

HomeSite предоставляет возможность удаленного доступа к FTP серверам для передачи файлов и других задач, связанных с управлением сайтом.

FTP (Протокол передачи файлов) — это стандартное средство передачи файлов между узлами в сети Интернет. Функция **Allaire FTP& RDS** предоставляет интерфейс для подсоединения к FTP серверу. После подсоединения вы можете получить доступ к файлам на удаленных серверах и управлять сайтом.

Условия подключения к FTP серверу могут сильно различаться. Данное руководство подходит в большинстве случаев, тем не менее, вы можете попробовать поменять настройки. Для подсоединение к удаленному серверу вам может потребоваться дополнительная информация, такая как имя хоста и другие требования для пользовательского доступа.

Для подсоединения к FTP серверу:

- Откройте список, размещенный вверху **Files Tab** и выберите **Allaire FTP&RDS**. Вы также можете сделать из это Проводника Windows.
- Правой кнопкой мыши щелкните на **Allaire FTP&RDS** и выберите **Add FTP Server** для открытия диалогового окна **Configure FTP Server**.

Настройки FTP сервера

- **Description** (Описание) — введите имя для показа в **Allaire FTP&RDS**
- **Host Home** (Имя Хоста) — введите доменное имя сервера или IP адрес
- **Initial Directory** (Начальная Директория) — введите имя начальной директории, если необходимо
- **Username** (Имя пользователя) — введите имя (**login**) или укажите «anonymous» для анонимного FTP Servera
- **Password** (Паспорт) — введите пароль
- **Root URL** (Корень URL) — введите IP адрес сервера вашего сайта
- **Remote Port** (Удаленный порт) — выберите порт сервера 21 указанный по умолчанию, если администратор сервера или Провайдер Интернет услуг не указал какой-либо другой
- **Request Timeout** (Запрос) — установите время ожидания соединения с сервером
- **Assume UTC file times ()** — выберите данную настройку, если вы обнаружили не верные дату или время на FTP Сервере. Это позволит настроить сервер, использующий универсальный формат времени

- **Disable passive mode** (Отключение пассивного режима) — выберете данную настройку, если сервер не использует пассивное соединение
- Нажмите **OK** завершения конфигурации. Теперь вы можете с файлами на сервере.

Для просмотра и редактирования конфигурации подключения к удаленному серверу, щелкните на имя сервера в списке **Allaire FTP&RDS** и выберете **Properties** (Свойства) из меню. Сделайте необходимые изменения, нажмите OK для сохранения свойств Сервера.

Редактирование страниц

HomeSite предоставляет огромное количество инструментов для редактирования кода и содержания.

Настройка опций редактирования

Данные свойства позволяют контролировать изображение тэгов и текста в редакторе:

- **Color-coding tags** (Тэги кодирования цвета) — HomeSite имеет возможность комбинировать цвета кодов. Используйте панель **Color Coding** (Кодирования Цвета) в диалоговом окне настройки (**F8**) для необходимых изменений.
- **Word wrap** (Заворачивание текста) — Контролирует длину строки текста в редакторе. Щелкните на кнопку **Word wrap** в панели инструментов.
- **Gutter** (Межстолбцовый промежуток) — Поле с левой стороны документа предназначено для закладок, номера строк и отлаживания точек прерывания. Щелкните на кнопку **ShowCulter** в панели инструментов редактора. Нажмите **Ctrl+K** для установки или удаления закладок и **Ctrl+Shift+K** для того, чтобы перейти к следующей закладке.
- **Convert tag case** (Перестройка регистра тэгов) — Изменяет по верхнему или нижнему регистру все тэги в текущем документе. Используйте команду меню **Edit ⇨ Convert Tag Case** для полного изменения тэгов в документе.
- **Indenting code** (Кодирование отступов в тексте) — Для установки отступа в тексте используйте кнопки с изображением стрелок направленных влево и вправо в

панели инструментов Редактора. Для автоматического отступа строки на тот же уровень, что и предыдущая строка, выберите команду **Auto Indent** в панели **Editor** (Правка) диалогового окна настроек (**F8**).

Работа в режиме правка

В режиме Правки вы выполняете большую часть вашего редактирования и написания кодов.

С помощью мышки или стандартных команд с клавиатуры вы можете выделить части документа, а с помощью ниже перечисленных сокращенных клавишных команд — быстро выбрать и определить блоки тэгов:

- Чтобы выбрать целый блок кода, нажмите **Shift+Ctrl** и дважды щелкните на первый или последний тэг.
- Чтобы выделить часть текста, щелкните на начало и удерживая **Shift** щелкните там где вы ходите закончить выделение.
- Нажмите **Ctrl** + двойной щелчок для выделения тэга.

Сохранение текста в буфере

Данная функция позволяет копировать множество текстовых блоков и сохранять их в качестве отдельных записей в буфере.

Скопировав необходимый блок в буфер его можно позднее по необходимости использовать. В буфер можно помещать до 36 фрагментов записей. При необходимости эту настройку можно изменить, выполнив **Setting ⇨ Editor ⇨ Maximun clipboard entries**. Когда количество записей в буфере начинает превышать количество записей установленных в настройках, последнее копирование удаляет старейшую запись и добавляет новую в конец буферного окна.

На панели инструментов есть три кнопки для данной функции.

- **Показать буфер** — показывает окно прежде с скопированными записями (клипы). Если мышкой навести на пиктограмму клипа, появиться первые строки содержания. Щелкните на пиктограмму клипа для того, чтобы вставить клип в редактор, указав при этом место с помощью курсора.
- **Вставить все** — все записи (клипы) при щелчке на пиктограмму вставляются в редактор в том же порядке, в котором они были скопированы.

- **Очистить буфер** — удаляет все текущие клипы из буфера.

Настройка команд буфера.

Команды буфера могут быть добавлены в панель инструментов с помощью команд **Options ⇨ Customize**. Кнопки расположены в **Toolbuttons** разделе **Edit**.

В Keyboard shortcuts можно назначить сокращенный клавишный набор для функций:

- **Show clipboard** — Показать буфер
- **Paste All** — Вставить все
- **Clear clipboard** — Очистить буфер

Сворачиваемый текст

При редактировании длинных документов и сложных приложений вы можете прятать текстовые и кодовые блоки с тем, чтобы сосредоточиться на какой-либо части контекста. Маркер показывает первые несколько символов свернутой выборки.

Для настройки опций откройте панель **Editor ⇨ Collapsed Text** в диалоговом окне **Settings** (**F8**) для установки опций. Используя настройки, вы можете:

- Настроить маркер свернутого текста включая шрифт, цвет и количество текста отображенного внутри маркера.
- Запустить выпадающее окно показа целого текста при движении мыши по маркеру свернутого текста. Вы можете установить максимальное число строк для отображения в выпадающем окне.
- Запустить автоматическую выборку текста при разворачивании свернутого текста. Это позволит развернуть свернутый участок текста для обзора содержания и затем снова его можно свернуть.
- Установить опции сохранить/открыть файл для свернутого текста.

Для сворачивания текста в текущем документе, используйте одну из ниже перечисленных команд:

- Выделите часть текста в и щелкните одну из кнопок (-) находящихся на панели редактирования. Правой кнопкой мышки щелкните на текст и выберите команду **Collapse** из выпадающего меню.

- Щелкните правой кнопкой мыши на начало или конец тэга и выберите **Collapse Tag**. Эта команда сворачивает тэг полностью.
- Щелкните правой кнопкой мыши на начало или конец тэга и выберите **Collapse All IndTag**. Это позволит свернуть все тэги схожие с текущим.
- Щелкните на знак плюс голубого цвета в верхней части панели **Tag Inspector**. Таким образом свернется текст, основанный на текущем инспекторе тэгов.

Работа со свернутым текстом:

- Двойным щелчком на маркер разверните текст.
- Для разворачивания всего текста щелкните правой кнопкой в любом месте документа и выберите команду **Expand All** из выпадающего меню.
- Вы можете перемещать мышкой свернутый текст, так же как и любой другой текстовый блок.
- Блоки свернутого текста могут быть размещены один в одном.
- При запуске поиска или проверки орфографии свернутый текст разворачивается в случае обнаружения результата поиска или ошибки.

Инструменты редактирования

Панель редактирования предоставляет все необходимые инструменты, которые могут понадобиться для работы:

- **Tag Editors** (Редактор тэгов) — специальное диалоговое окно для редактирования атрибутов
- **Tag Tree** (Древо Тэга) — конфигурируемый навигационный помощник, для обзора и навигации иерархии тэга документов
- **Tag Inspector** (Инспектор Тэга) — тэговый редактор листа стилей
- **Design Mode** (Режим Дизайн) — режим, позволяющий визуально редактировать элементы страниц
- **CodeSweeper** (Метла) — конфигурируемый форматизатор кода

- **TopeStyle** — позволяет создавать, редактировать и присоединять листы стилей к документам

Редакторы тэгов

Редакторы — это специальные диалоговые окна, которое содержат атрибуты и значения выделенного тэга. Каждый редактор содержит встроенную справку (Help).

Отрыть окно редактора можно любым из ниже перечисленных способов:

- Кликните в тэге правой кнопкой мышки и выберите **Edit Tag** (Редактор Тэга).
- Поставьте курсор внутрь тэга и нажмите **Ctrl+F4**.
- В окне редактора нажмите на кнопку **Toggle Embedded Help** и вы увидите синтаксис и пользовательскую справку.

Древо Тэгов

Закладка **Tag Inspector**, находящаяся на панели **Resource** содержит два инструмента: **Tag Tree** и **Tag Inspector**. Используя эти инструменты, вам будут доступны только коды HTML без текста документа, что упрощает создание и редактирование тэгов.

Дерево Тэгов, расположенное в верхней части панели, позволяет легко перемещаться и проверять структуру кода документа. Так же с его помощью можно отображать только те тэги, которые вам нужны.

Для использования Древа тэгов пользуйтесь «выпадающим» списком для фильтрации показов кодов древа:

- Щелкните на знаках плюс и минус для раскрытия и сокращения узлов (блоков) древа тэгов.
- Щелкните на тэг в древе для быстрого перехода к нему в документе. Используйте команду **Shift + Ctrl + double-clik** (двойной щелчок мыши) для выделения блока тэга.
- Кликните на кнопку **Refresh** для обновления документа после его правки.

Визуальное редактирование в режиме Desing View (Дизайн)

HomeSite предоставляет дополнительную опцию по визуальному редактированию кода, так называемый **Desing View**. Данная опция работает по принципу WYSIWYG («what-you-see-is-what-you-get» — «что-видите-то-получаете»). Страница создается так, как она будет выглядеть на экране браузера, а программа записывает ее в виде HTML-кода.

Desing View может быть полезен при размещении изображений на странице, и для создания комплексных таблиц и форм. Код для этих элементов генерируется автоматически и обновляется по мере появления каких-либо изменений.

Функция **Desing View** предполагает наличие браузера IE 4 или более позднею версию. Если у вас не установлен данный браузер или вы не не намерены использовать данную функцию, вы можете отключить ее, для этого зайдите в меню **Options ⇨ Settings ⇨ Desing** и поставьте галку напротив Disable (hide) desing tab.

Редактирование страниц в режиме Desing View:

- Если документ открыт в режиме **Editor** (Редактор), сохраните ваши изменения и затем щелкните на закладку **Desing**. Откроется окно **Desing view** позволяющий просматривать вид документа, а так же текст, стили и графику.
- Используйте команды панели инструментов **Desing view** для редактирования текста, изменения стилей, добавления цветов или создания таблиц и формирования элементов форм.
- Если вы открыли **Desing view** и хотите отменить все сделанные изменения, нажмите на **Cancel Desing View** и вернитесь в режим **Edit view**.
- Используйте закладки **Edit** и **Browse**, для перехода в эти режимы.

Преобразование содержания Word и Excel:

- Вы можете преобразовывать содержание, то есть листы, таблицы и ячейки рабочего листа в эквивалентный формат HTML, путем копирования из Word и Excel.
- Выделите содержание в Word(e) или Excel(e).
- Вставьте выделенный фрагмент в окно **Desing** в то место, где вы хотите, чтобы этот фрагмент размещался на странице.
- Отредактируйте новое содержание. Например вы можете щелкнуть на границу таблицу таблицы и потянув изменить ее размер.
- Для работы непосредственно с кодом перейдите в режим **Editor**.

Для изучения данной функции попробуйте поэкспериментировать, вставляя разные типы содержания из документов Office или других программ Windows.

Сохранение форматов кода с помощью CodeSweepers

Функция CodeSweeper позволяет автоматизировать процесс форматирования HTML-кода, что может быть полезным в ряде случаев:

- Визуальное редактирование элементов страницы в режиме **Design view** может изменить форматирование кода. Вы можете настроить CodeSweeper так, чтобы форматирование кода производилось при выходе из режима **Design**.
- Вы легко можете усилить стили кодирования для многочисленных developers, просто установив для них такие же настройки.
- Вы можете легко подчистить форматирование кода при просмотре документов существующих проектов.

HomeSite включает в себя несколько видов CodeSweeper. Вы можете также создавать свои собственные CodeSweeper или править уже существующие. В HomeSite имеется следующие CodeSweeper:

- HTML и CFML тэги (по умолчанию) — форматирует прикладные программы GoldFusion.
- HTML тэги — оптимально подходит для стандартных Web-страниц.
- My CodeSweeper — копия CodeSweeper (установленного по умолчанию), которую можно использовать для тестирования.
- WDDX Sweeper — копия CodeSweeper форматирует код написанный в WDDX, например, .apd файлы используемые для хранения информации проекта.
- WEB-XML CodeSweeper — только в HomeSite v 4.5.2.
- JSP CodeSweeper — только в HomeSite v 4.5.2.
- HTML Tidy — независимый верификатор и форматизатор кода HTML, который можно использовать как альтернативный CodeSweeper. Он так же позволяет преобразовывать HTML в XHTML сочетающийся с XML.

Для настройки и использования CodeSweeper по умолчанию щелкните на кнопку **CodeSweeper** из режима **Edit** и **Design** с тем, чтобы запустить CodeSweeper в текущем документе. Вы так же можете настроить CodeSweeper на автоматический запуск при выходе из режима **Design**.

Создание нового CodeSweeper:

- Откройте диалоговое окно **Setting** (**F8**).
- Щелкните **CodeSweeper**, появиться окно **CodeSweepers**.
- Выберите **CodeSweeper**, который вы бы хотели использовать по умолчанию из списка и щелкните на команду **Set As Default** (Установить по умолчанию).
- Щелкните **OK**.

Автоматический запуск CodeSweeper после выхода из режима **Design**:

- В диалоговом окне **Settings** (Настройки) выберете **Design**.
- Щелкните на команду **Apply CodeSweeper** при возвращении из просмотра в режиме **Desing**.
- Щелкните **OK**.

Использования какого-либо другого CodeSweeper отличного от CodeSweeper установленного по умолчанию

При использовании CodeSweeper отличного от CodeSweeper установленного по умолчанию изменения сохраняются только в текущем документе.

Использование другого CodeSweeper:

- Щелкните на стрелку указывающую вниз на кнопке **CodeSweeper** или выберите **Tools ⇨ CodeSweeper** и из списка выберите необходимый вам CodeSweeper.
- Имейте в виду, что CodeSweeper форматирует целый документ и произведенные действия уже нельзя отменить.
- Щелкните **Run CodeSweeper**. Документ отформатирован.

Создание, редактирование и удаление CodeSweeper

Вы можете создавать новые CodeSweepers, редактировать существующие CodeSweepers и удалять устаревшие CodeSweepers, используя при этом диалоговое окно **Setting** (**F8**).

Создание нового CodeSweeper:

- Откройте диалоговое окно **Setting** (**F8**).
- Щелкните CodeSweeper, появиться окно CodeSweepers.
- Щелкните **New Profile** (Новый Профиль), появиться диалоговое окно **New CodeSweeper Profile**.
- Введите имя **CodeSweeper** и выберите тип: **Allaire CodeSweeper** или **HTML Tidy CodeSweeper**.
- Щелкните **OK**. CodeSweeper добавлен в список.

Следуйте нижеперечисленному руководству для редактирования CodeSweeper.

Редактирование:

- В диалоговом окне **Setting** (**F8**) разверните CodeSweeper.
- Разверните тот тип CodeSweeper, который вы хотите отредактировать: **Allaire CodeSweeper** или **HTML Tidy CodeSweeper**.
- Выберите CodeSweeper, который вы хотите отредактировать и сделайте необходимые изменения в настройках.
- Щелкните **OK**.

Удаление:

- В диалоговом окне **Setting** (**F8**) щелкните на CodeSweeper.
- Выберете CodeSweeper, который вы хотите удалить из списка и щелкните **Remove Profile**. CodeSweeper будет удален без предупреждения.

Настройки CodeSweeper

Существует три вида настроек для CodeSweeper:

- Общие настройки (**General Settings**) и Специальные настройки (**Tag-Specific Settings**) могут быть установлены для каждого Allaire CodeSweeper.
- **HTML Tidy** настройки относятся только к **HTML Tidy CodeSweeper**.

Общие настройки (General Settings)

Вы можете установить ниже следующие правила форматирования для каждого Allaire CodeSweeper:

- Установите регистр для тэга и атрибутивные имена. Вы можете выбрать как верхний или нижний регистр, так и сохранить регистр без изменений.
- Настройка имен событий формата включает дополнительную опцию, а именно смешанный регистр, который применим для имен событий JavaScript, таких как **OnMouseOver**. Если данный код в вашем документе правильный, выберете команду **Preserve Case**.
- Установите кавычки для значений.
- Пробелы между кодами убираются с помощью инструментов кода, установленных по умолчанию. Рекомендуется оставлять их как есть, отключая только при необходимости для отдельных тэгов.
- Работайте в режиме «**Silent Mode**» во избежание предупреждений, которые может генерировать CodeSweeper. Вы можете использовать эту опцию для работы CodeSweeper в автоматическом режиме.
- Проверьте **Write Errors** в системном журнале **Log Fife** с целью обнаружения каких-либо ошибок, найденных CodeSweeper. Вы можете сами определить место системного журнала.
- Щелкните **OK** для сохранения настроек.

Специальные настройки (Tag-Specific Settings)

Широкий выбор опций существует для индивидуальных тэгов, содержащихся в каждом **Allaire CodeSweeper**. Если вы устанавливаете форматирование «**All Other Tags**», то это правило будет действовать в отношении каждого тэга, появляющегося в документе и не состоящего в списке.

- Вставка команды с новой строки для первого и последнего тэгов.
- Установка отступа с помощью Tab или пробелов — разделитель строк вставиться автоматически для каждой новой строки.
- Запуск разделителя строк для вложенных под — тэгов (sub tags).

- Оставляет выбранный тэг неизмененным при активизации CodeSweeper.
- Отделение, удаление тэга из документа — это удобно для избавления от ненужных тэгов, вставленных генератором кода.

Установка правил для тэга:

- Выберите тэг из списка **Tag Specific Settings** в диалоговом окне **CodeSweeper Settings**.
- Измените специальные настройки.
- Щелкните **Update Tag** (Обновить Тэг) для сохранения настроек данного тэга.

Добавление тэга:

- Щелкните кнопку **Add Tag** в диалоговом окне **CodeSweeper Settings**.
- Введите имя тэга и щелкните **OK**.
- Измените настройки тэга и щелкните **Update Tag** для сохранения настроек.

Удаление тэга:

- Выберите тэг из списка **Tag Specific Settings** в диалоговом окне **CodeSweeper Settings**.
- Щелкните **Remove Tag** (Удалить Тэг).
- Щелкните **OK** для закрытия диалогового окна.

Создание собственного профиля

Вы можете создать ваш собственный настроечный конфигурационный файл и сохранить его с расширением .tdy в вашу папку \extensions\Codesweepers.

Форматирование страниц с помощью CSS

Каскадные листы стилей (CSS) становятся все более важным компонентом web-сайтов, поскольку они предоставляют много возможностей для дизайна и управления содержанием. Спецификация HTML 4.0 постоянно поддерживает их использование не смотря на ограничения поддержки Каскадных листов стилей браузерами.

Листы стилей могут быть добавлены на любом этапе разработки сайта, но в идеале спецификация дизайна для сайта вводиться посредст-

вом листов стилей при этом отпадает необходимость кодирования и форматирования индивидуальных тэгов. Еще более важным является то, что каскадные листы стилей позволяют разработчикам и авторам содержания, сконцентрироваться на проблемах организации и навигации путем отделения форматирования от содержания. По мере разработки сайта стили могут быть коренным образом изменены, добавлены и удалены, давая разработчику больше свободы (и времени) для достижения более высоких результатов в дизайне.

Терминология CSS

Если вы новичок, то первое, что вам необходимо узнать — это термины используемые в работе со стилями. Правила стилей совпадают с HTML элементами, но между ними не всегда есть полное соответствие.

HTML тэги содержат атрибуты со значениями, например:

```
<h1 align="center"><font size="+2" color="green">Some
Text</font></h1>
```

Тэги h1и font содержат атрибуты, чьи значения определяют место размещения, размер шрифта и цвет текста.

Листы стилей содержат правила, которые применяются к селекторам (selectors), использующим описания (declarations) для определения форматирования:

```
h1 { text-align : center; font-size : larger; color : Green; }
```

В данном примере селектор h1 содержит описание, которое определяет расположение, размер шрифта и цвет текста. Если это единственное правило для h1, то каждый тэг <h1>;будет отображать данное форматирование.

Каскадные листы стилей — сложный инструмент для четкого контроля за видом web-страниц, однако их правильное применение требует определенных навыков.

Вы можете начать с простых правил для общих элементов страниц и постепенно перейти к созданию более сложных стилей, по мере овладения техническими приемами. HomeSite Style Editor дает предварительный просмотр правил и их определения перед тем, как вставить их в документ. Вы можете посмотреть страницы целиком в браузере с тем, чтобы наглядно увидеть как взаимодействуют стили.

Использование редактора стилей (Style Editor)

Редактор стилей HomeSite — **TopStyle** предоставляет полный интерфейс для дизайна и применения каскадных листов стилей. TopStyle имеет свою справочную систему, которая охватывает основы CSS интеграцию HomeSite, и такие свойства (функции) как **Style Insight** и **Style Inspector**.

Запустить TopStyle можно нажав кнопку **StyleEditor** на панели инструментов.

Проекты развертывания

Развертывание — это процесс копирования всех файлов вашего проекта на один или несколько хост серверов. Вы можете указать путь развертывания как для всего проекта так и для индивидуальных папок.

Установка Уровня Развертывания Проекта

Вам необходимо только ввести путь для установки развертывания проекта.

Установить путь развертывания для проекта:

- Откройте проект в панели Проектов
- Правой кнопкой мышки кликните на корень проекта и выберите Свойства (**Properties**). Появится диалоговое окно **Edit Project Properties**
- Введите путь в текстовом окне (**Deployment Path**)
- Нажмите **OK**, чтобы применить настройки

Установка развертывания папки

Развертывание папки позволяет получить определенную гибкость, но требует прохождения ряда этапов. При открытии диалогового окна **Properties** (Свойства) через папку проекта, вы можете выбрать одну из следующих опций в разделе **Deployment** (Развертывание).

Опция развертывания относительно родительской папки

При установке данной опции HomeSite развертывает папку относительно ее родительской папки проекта. Например:

```
Parent Folder: MyDirectory/ParentFolder
Child Folder: MyDirectory/ParentFolder/ ChildFolder
```

Если развертывание родительской папки имеет следующий путь **Server Directory/ParentFolder**, то развертывание дочерней папки будет следующим **Directory/ParentFolder/ChildFolder**. Путь дочерней папки просчитывается автоматически.

Данная опция не распространяется на уровень проекта, так как папка проекта является корневой и не имеет родительской папки.

Специальная опция развертывания

Вы можете использовать данную опцию для введения точного пути развертывания папки.

Allaire рекомендует использовать команду **Browse** для установки специального пути развертывания. Используйте нижеследующее руководство, если вы вводите путь вручную:

- Прописывая путь на локальный или сетевой диск необходимо включать букву диска и обратные косые черты, например:
  `C:\MyDirectory\MySubdirectory`
  или
  `\\MyDirectory\MySubdirectory`
- Прописывая путь на удаленный сервер, используйте левые косые черты.

Как правило, путь, который вы прописываете не включает имя сервера, таким образом, программа автоматически добавляет имя сервера в путь, при выборе сервера.

Опция «без загрузки» (do not upload)

Данная опция отличается тем, что при развертывании игнорируется папка и ее содержимое. На уровне проекта данная опция отсутствует.

Настройка развертывания папки:

- Правой кнопкой щелкните на папку и выберите Свойства (**Properties**). Появиться диалоговое окно **Edit Folder Properties**
- Щелкните на кнопку **Deployment** и выберите нужную опцию развертывания
- При выборе Специальной Опции Развертывания (**Special Deployment Properties**) введите место размещения в окне **Deployment Path** (Путь Развертывания)
- Щелкните **OK**

Добавление Серверов Развертывания

Процесс добавления сервера развертывания подобен конфигурации отдаленного сервера через Файл панель (**Files panel**).

Основным отличием является то, что информация о сервере развертывания вноситься в файл проекта, тогда как информация отдаленного сервера хранится в **Windows Registry**.

Если вы уже добавляли отдаленный сервер с помощью **Allaire FTP&RDS**, вам необходимо ввести такую же информацию о сервере для конфигурации сервера развертывания.

Добавление сервера развертывания:

- Щелкните правой кнопкой на **Deployment Server** в вашем проекте
- Выберите тип сервера который вы хотите добавить
- Заполните диалоговое окно **Server Configuration**
- Щелкните на **OK** для сохранения информации Сервера

Сервер появиться в списке **Deployment Server**. При развертывания проекта выберите нужный вам сервер из данного списка.

Глава 8. Microsoft FrontPage Express

Microsoft FrontPage Express один из тех редакторов Web-страниц, которые не требуют от вас знаний основ HTML. Забудьте о программировании! Просто размечайте и оформляйте!

В законченной среде разработки Web в режиме непосредственного отображения вы можете профессионально создавать web-документы и даже организовывать небольшие web-сервера.

В Microsoft FrontPage Express вы можете подготавливать к публикации в Internet web-страницы, используя JavaScript, VB Script, ActiveX и даже выборки к вашим базам данных. Кроме этого, в Microsoft FrontPage Express встроены весьма неплохие графические эффекты, которые оживят ваш дизайн.

Вы можете управлять созданным в среде Microsoft FrontPage Express web-сервером из локальной сети или используя удаленный доступ к основному компьютеру.

Забудьте о контроле за ложными ссылками! В Microsoft FrontPage Express встроено средство, позволяющее автоматически фиксировать подобные ссылки.

Вы можете:

- создавать и сохранять web-страницы
- сохранять web-страницы непосредственно в Web
- загружать из Internet и редактировать web-страницы
- использовать в оформлении вашей web-страницы подложки
- просматривать и администрировать web-страницы
- создавать сложный дизайн web-страниц
- использовать готовые тэги HTML
- использовать готовые изображения из комплекта поставки программы
- использовать компоненты WebBot для придания вашей web-страничке динамики
- использовать в оформлении web-страниц элементы управления ActiveX

Если вы хотите стать профессионалом в области дизайна web-страниц, вам необходимо освоить некоторые специальные HTML-редакторы с полной поддержкой тэгов HTML.

Интерфейс пользователя

Microsoft FrontPage Express считается одним из лучших средств подготовки документов для их публикации Web. Вы просто создаете web-страницу в том виде, в каком она будет выглядеть в окне обозревателя.

Ниже приведено краткое описание набора средств, с помощью которого вы можете взаимодействовать с редактором web-страниц Microsoft FrontPage Express.

С помощью стрелок **полос прокрутки** или посредством клавиш **PgUp** и **PgDn** вы можете перемещать текущую страницу в вертикальном или горизонтальном направлении.

Вверху экрана находятся **панели инструментов**, с помощью которых вы можете быстро выполнить то или иное действие.

Доступ к командам редактора осуществляется через щелчок мышью на соответствующем имени меню.

Команды меню

В редакторе гипертекстовых документов Microsoft FrontPage Express все наборы команд и опций, использующиеся в процессе работы с Интернет содержатся в:

- меню **Файл**
- меню **Правка**
- меню **Вид**
- меню **Переход**
- меню **Вставка**
- меню **Формат**
- меню **Сервис**
- меню **Таблица**
- меню **Окно**
- меню **Справка**

Панели инструментов

Редактор Microsoft FrontPage Express содержит три панели инструментов. Каждая кнопка той или иной панели соответствует команде меню.

Панель форматирования

С помощью **Панели форматирования** вы можете форматировать абзацы, определять параграфу тот или иной шрифтовой стиль, раскрашивать текст и выравнивать параграфы.

Панель редактирования

С помощью **Панели редактирования** вы можете:

- создавать, открывать и сохранять web-страницы
- выводить на печать и предварительно просматривать web-публикации перед выводом на печать
- выполнять операции копирования и вставки
- создавать или изменять ссылки
- вставлять компоненты WebBot, изображения, горизонтальные строки и таблицы

- показывать или скрывать метки параграфа
- возвратиться на одну страницу назад
- перейти на одну страницу вперед
- вызвать справочную систему редактора Microsoft FrontPage Express
- прервать загрузку текущей страницы

Панель форм

С помощь **Панели форм** вы можете создать поле формы из:

- однострочного текстового поля
- прокручиваемого текстового поля
- флажка
- кнопки-переключателя
- простой кнопки

Всплывающее меню

Всплывающее меню позволяет более удобно обратиться к той или иной команде. Просто поместите курсор на интересующий элемент вашего документа и нажмите правую кнопку мыши.

С помощью элементов всплывающего меню вы можете выполнять следующие команды и функции:

- **Вырезать** (выполнить операцию удаления элемента и помещения его в буфер промежуточного хранения)
- **Копировать** (поместить элемент в буфер промежуточного хранения)
- **Вставить** (вставить элемент из буфера промежуточного хранения)
- **Свойства страницы** (обратиться к диалоговому окну свойств **Свойства страницы**)
- **Свойства таблицы** (обратиться к диалоговому окну свойств **Свойства таблицы**)
- **Свойства ячейки** (обратиться к диалоговому окну **Свойства ячейки**)

- **Свойства абзаца** (обратиться к диалоговому окну **Свойства абзаца**)
- **Свойства шрифта** (обратиться к диалоговому окну **Шрифт**)
- **Свойства изображения** (обратиться к диалоговому окну свойств **Свойства изображения**)
- **Проверка достоверности поля формы** (обратиться к диалоговому окну **Проверка достоверности текстового поля**)
- **Свойства поля формы** (обратиться к диалоговому окну **Свойства прокручиваемого текстового поля**)
- **Свойства компонента WebBot** (обратиться к диалоговому окну **Свойства компонента WebBot**)

Буксировка

Вы можете перемещать один или несколько элементов вашего документа из одного окна программы в другое, или между элементами основного окна программы, или из Проводника. Просто выделите тот или иной элемент и отбуксируйте его в нужное место редактируемого документа.

При буксировке файлов изображений Microsoft FrontPage Express автоматически преобразует файлы формата BMP в формат JPEG.

Создание новой web-страницы

Создание новой web-страницы осуществляется через выбор команды **Создать** в меню **Файл** (вы также можете выбрать эту команду через комбинацию клавиш **Ctrl+N**). В появившемся диалоговом окне **Новая страница** просто выберите из списка **Шаблон или мастер** параметр **Нормальная страница**.

Вы можете также создать web-страницу с помощью **Мастера личной основной страницы** или **Мастера страниц форм**. Для этого выберите из списка **Шаблон или мастер** необходимый параметр и далее следуйте указаниям **Мастера страниц**.

Сохранение web-страницы

Сохранение web-страницы осуществляется через выбор команды **Сохранить** в меню **Файл** (вы также можете выбрать эту команду через комбинацию клавиш **Ctrl+S**). В появившемся диалоговом окне введите название страницы. Если же вы хотите сохранить страницу в виде файла, нажмите кнопку **Как файл** и в появившемся диалоговом окне укажите

имя файла вашего документа HTML. Если вы хотите сохранить страницу прямо в Web, определите ее местоположение.

Открытие файла

Выбранная команда **Файл ⇨ Открыть** открывает доступ к диалоговому окну **Открыть файл**, с помощью которого можно загрузить в программу:

- файлы HTML
- файлы HTML после предварительной обработки
- текстовые файлы
- шаблоны гипертекста
- документы Microsoft Word
- документы текстового процессора Windows Write
- документы текстового процессора Windows Write
- документы WordPerfec
- книги Microsoft Excel

Чтобы открыть файл нужно выбрать его имя из списка и нажать кнопку **Открыть**.

Выход из программы

Выйти из редактора Microsoft FrontPage Express можно, выбрав команду меню **Файл ⇨ Выход**. Если документ, находящийся на экране был изменен, то Microsoft FrontPage Express выведет запрос о том, сохранять ли внесенные изменения. В качестве ответа можно воспользоваться кнопкой **Да**, **Нет** или **Отмена**.

Редактирование текста web-страницы

Microsoft FrontPage Express позволяет редактировать также, как вы это делаете в текстовых процессорах Windows.

Ввод текста осуществляется через щелчок левой клавишей мыши в том месте вашего документа, где вы хотите ввести текст.

Помните, что четыре символа — знак меньше (<), знак больше (>), амперсанд (&) и двойные кавычки (“) имеют специальное значение внутри HTML и следовательно не могут быть использованы в тексте вашего документа в своем обычном значении.

Отмена и восстановление выполненной команды

Отмена выполненной команды осуществляется через выбор команды **Отменить** в меню **Правка**.

Microsoft FrontPage Express позволяет отменить только последнее выполненное действие. Например, если удален текст, а затем набран новый, то нельзя отменить удаление текста.

Выбранная команда **Правка ⇨ Восстановить** восстанавливает последнее отмененное действие.

Выделение слова

Если вы хотите выбрать слово, дважды щелкните по нему левой клавишей мыши.

Выделение строки

Выделение строки осуществляется простым щелчком левой клавишей мыши при нажатой клавише **Alt** в любом месте строки.

Выделение параграфа

Выделение параграфа осуществляется простым щелчком левой клавишей мыши при нажатых клавишах **Alt** и **Ctrl** в любом месте параграфа.

Выделение всех элементов вашего документа

Команда **Правка ⇨ Выделить все** используется для выделения текстовых и графических элементов на текущей странице.

Отображение абзацных меток

Отображение абзацных меток осуществляется через выбор команды **Маркеры форматов** в меню **Вид**.

Операции вырезания, копирования и вставки

Выбранная команда **Правка ⇨ Вырезать** удаляет активный элемент из документа Microsoft FrontPage Express и помещает его в буфер промежуточного хранения. Вы можете удалить из документа и поместить в буфер обмена любой выбранный текстовой или графический элемент как Microsoft FrontPage Express, так и приложения Windows.

Команда **Файл ⇨ Скопировать** используется для вставки активного элемента в буфер промежуточного хранения. Вы можете поместить в буфер обмена любой выбранный текстовой или графический элемент как Microsoft FrontPage Express так и приложения Windows.

Выбранная команда **Правка ⇨ Вставить** помещает копию элемента из буфера промежуточного хранения в документ Microsoft FrontPage Express. Вы можете поместить в документ Microsoft FrontPage Express текстовой или графический элемент из любого приложения Windows.

Поиск и замена текста

Выбранная команда **Правка ⇨ Заменить** открывает доступ к встроенному в Microsoft FrontPage Express инструменту редактирования, который используется для поиска и замены символов (включая прописные и строчные буквы), целых слов или частей слов. Поиск и замена спецсимволов осуществляется так же, как поиск и замена простого текста — достаточно ввести в поля данных **Образец** или **Заменить** на ключевые комбинации для специальных символов или ANSI (ASCII) коды символов.

Вывод на печать

Выбранная команда **Файл ⇨ Напечатать** открывает доступ к диалоговому окну **Печать**, с помощью которого можно выбрать принтер для печати текущего документа, установить параметры работы этого принтера и вывести на печать текущий документ.

Оперирование открытыми окнами

С помощью команд меню **Окно** вы можете:

Рядом

Расположить все окна без взаимного перекрытия по горизонтали.

Каскадом

Упорядочить все окна с перекрытием так, чтобы название каждого окна было видимо.

Упорядочить значки

Упорядочить пиктограммы минимизированных окон.

Установив курсор мыши на кнопке со стрелкой и буксируя мышью один из указателей прокрутки, можно перемещать страницу до тех пор, пока в документе не покажется нужное место. Вертикальный указатель предназначен для передвижения страницы по вертикали, горизонтальный — по горизонтали.

Доступ к системе оперативной подсказки

С помощью команд меню **Справка** можно получить доступ к системе оперативной подсказки редактора web-страниц Microsoft FrontPage Express.

Стили форматирования

В языке описания гипертекстовых документов слова и строки кодируются логическими и физическими стилями. Поэтому в редакторе web-страниц Microsoft FrontPage Express доступны только два стиля оформления параграфов:

Стили оформления символов (физические стили) влияют на выравнивание параграфов, тип, размер, гарнитуру и цвет шрифта.

Стили оформления параграфов (логические стили) влияют на все параграфы выделенного текста. Эти виды стиля позволяют вести форматирование через указания заголовков того или иного уровня и игнорировать информацию о размере шрифта и гарнитуре. Поэтому, чтобы изменить форматирование заголовка, вы должны модифицировать заголовок первого уровня. Через стили оформления вы можете сформировать согласованный гипертекстовый документ, то есть определить заголовок первого уровня в качестве только **<H1>** (без информации о гарнитуре шрифта и его кегле).

Физические стили

Физические стили или стили оформления параграфов доступны через **Панель форматирования** или через команду **Абзац** меню **Формат**.

Вы можете оформить доступным физическим стилем не только выбранный абзац, но и несколько выделенных абзацев.

К каждому из этих стилей вы можете применить выравнивание по центру, по левому или по правому краю. Для этого в меню **Формат** выберите команду **Абзац** и в появившемся диалоговом окне **Свойства абзаца** обратитесь к списку **Выравнивание абзаца**.

Просто определите точку вставки и выберите необходимый стиль абзаца из списка следующих стилей:

Адрес

Подпись автора web-страницы. Обычно располагается в конце документа.

Заголовки с 1 по 6 уровень

Разделение текста документа на разделы, каждый из которых оформлен текстом определенного размера.

Маркированный список

Список, оформленный маркерами.

Обычный

Стиль абзаца по умолчанию.

Список определений

Оформление глоссариев или других списков.

Нумерованный список

Иерархия с использованием цифр.

Список каталогов

Специальный стиль для оформления списков файлов или каталогов.

Список меню

Специальный стиль для оформления небольших параграфов.

Форматированный

Специальный стиль для оформления листингов программ и другой технической документации.

Логические стили

Microsoft FrontPage Express поддерживает следующие типы логических стилей:

- Полужирный
- Курсивный
- Полужирный курсив
- Машинописный шрифт
- Верхний индекс
- Нижний индекс
- Подчеркнутый
- Зачеркнутый

Вы можете оформить доступным логическим стилем не только выбранный символ, но и несколько выделенных символов. Просто выделите текст, обратитесь к диалоговому окну **Шрифт** (доступ: **Формат ⇨ Шрифт**) и выберите необходимый атрибут шрифта.

Раскрашивание текста

Вы можете оформить цветом не только выбранный символ, но и несколько выделенных символов. Просто выделите текст, обратитесь к диалоговому окну **Шрифт** (доступ: **Формат ⇨ Шрифт**) и выберите из списка **Цвет** понравившийся вам цвет шрифта.

Многоплатформенные шрифты

В принципе с помощью Microsoft FrontPage Express вы можете оформлять свои web-страницы всеми доступными шрифтами вашей системы. С другой стороны, если вы хотите, чтобы вашу страницу могли просматривать пользователи других операционных систем, вы должны использовать в оформлении так называемые многоплатформенные шрифты. К ним относятся следующие шрифты:

- Arial
- Courier
- Times

Горизонтальные линии

Вы можете украсить дизайн вашей web-страницы горизонтальной линией. Для этого определите место в вашем документе, в котором должна быть линия, и выберите из меню **Вставка** команду **Горизонтальная линия**.

Просмотр и правка исходного кода HTML

Microsoft FrontPage Express позволяет вам просмотреть текст исходного кода HTML и, в случае необходимости, произвести редактирование. Для этого выберите из меню **Вид** команду **HTML**.

Вставка в документ тэгов HTML

Вы можете вставить в ваш документ любой тэг HTML, включая Java Script. Для этого выберите из меню **Вид** команду **Разметка HTML**, обратитесь к диалоговому окну **Разметка HTML** и поместите ваш тэг в поле данных **Вставляемая метка HTML**.

Одновременно вы можете вставить только один тэг.

Дизайн web-страниц с помощью таблиц

Мощнейшее средство HTML — таблицы — позволяют создавать достаточно сложный дизайн web-страницы, например, вы можете расположить текст вашего документа в нескольких колонках.

Для того, чтобы разместить в вашем документе таблицу, определите точку вставки и обратитесь к диалоговому окну **Добавить таблицу** через команду **Вставить таблицу** меню **Таблица**.

Опции диалогового окна **Добавить таблицу** позволяют:

Размер строки

Указать количество строк таблицы.

Размер колонки

Указать количество колонок таблицы.

Выравнивание

Определить тип выравнивания элементов таблицы.

Размер рамки

Определить размер линий границ ячеек в пикселах (до 100 пикселей). Если размер рамки будет иметь нулевое значение, то линия границы таблицы отобразится пунктирами, а при просмотре документа в обозревателе станет невидимой.

Промежуток между столбцами

Определить расстояние между столбцами в пикселах.

Ширина

Указать ширину таблицы в пикселах или в процентах от ширины основного окна соответствующего обозревателя.

Независимо от того или иного обозревателя, таблица, ширина которой определена в пикселах, будет иметь постоянный размер.

Таблица, ширина которой определена в процентах, будет подстраиваться к тому или иному обозревателю.

Дополнительно

Обратиться к диалоговому окну **Дополнительные атрибуты**, с помощью которого можно вставить в таблицу тэг HTML или сценарий Java.

Как добавить в таблицу строку или столбец

В созданную таблицу вы можете добавить строку или столбец. Для этого выберите в таблице место вставки и обратитесь к диалоговому окну **Вставить строки или столбцы** (доступ: **Таблица ⇨ Вставить строки или столбцы**).

Совет: Ничто вам не мешает вставить таблицу в другую таблицу. Этим часто пользуются опытные дизайнеры web-страниц.

Как добавить в таблицу ячейку

В созданную таблицу вы можете добавить ячейку. Для этого выберите в таблице место вставки и дайте команду **Вставить ячейку** (доступ: **Таблица ⇨ Вставить ячейку**).

Как выделить таблицу

С помощью меню **Таблица** вы можете выделить всю таблицу или только одну ячейку, строку или столбец, просто дав одну из следующих команд: **Выделить таблицу**, **Выделить ячейку**, **Выделить строку**, **Выделить столбец**.

К выделенной таблице вы можете применить стандартные операции копирования и вставки.

Как разбить ячейку на столбцы или строки

Вы можете разбить ячейку на столбцы или строки с помощью диалогового окна **Разбить ячейки** (доступ: **Таблица ⇨ Разбить ячейки**). В поле данных **Число колонок** определите количество столбцов вашей новой ячейки, а в поле данных **Число строк** укажите количество строк.

Свойства таблицы

Созданной таблице вы можете назначить те или иные свойства (ширина линии границы, поля и отступы ячеек, ширина, цвет). Для этого в меню **Таблица** выберите команду **Свойства таблицы** и обратитесь к диалоговому окну **Свойства таблицы**.

Опции диалогового окна **Свойства таблицы** позволяют:

Выравнивание

Определить тип выравнивания элементов таблицы.

Размер рамки

Определить размер линий границ ячеек в пикселах (до 100 пикселей). Если размер рамки будет иметь нулевое значение, то линия границы таблицы отобразится пунктирами, а при просмотре документа в обозревателе станет невидимой.

Заполнение ячеек

Определить максимальный размер заполнения ячейки в пикселах (до 100 пикселей).

Промежуток между столбцами

Определить расстояние между столбцами в пикселах.

Ширина

Указать ширину таблицы в пикселах или в процентах от ширины основного окна соответствующего обозревателя. Независимо от того или иного обозревателя, таблица, ширина которой определена в пикселах будет иметь постоянный размер.

Таблица, ширина которой определена в процентах, будет подстраиваться к тому или иному обозревателю.

Специальный фон

Использовать в качестве фона таблицы картинку. Просто выберите флажок **Использовать фоновое изображение** и нажмите кнопку **Обзор** для поиска файла картинки в форматах GIF, JPG, BMP, TIF, WMF, RAS, EPS, PCX или TGA.

Цвет фона

Раскрасить подложку таблицы.

Дополнительные цвета

В этом разделе вы можете определить цвет для рамки вашей таблицы. Если вы выберите только параметр **Граница**, т.е. параметры **Светлая** и **Темная рамка** останутся используемыми по умолчанию, то через ниспадающий список цветов можно присвоить цвет всей рамке вашей таблице.

Имеющие приоритет параметры **Светлая** и **Темная рамка** позволяют присвоить цвет верхней и нижней рамкам таблицы.

Манипулирование ячейками таблицы

Вы можете назначить те или иные свойства не всей таблице, а только ее ячейкам. Для этого в меню **Таблица** выберите команду **Свойства ячейки** и обратитесь к диалоговому окну **Свойства ячейки**.

Опции диалогового окна **Свойства ячейки** позволяют:

Выравнивание по горизонтали

Определить горизонтальную выключку элементов таблицы (по горизонтали, влево и вправо).

Вертикальное выравнивание

Определить вертикальную выключку элементов таблицы (сверху, снизу и посередине).

Ячейка с заголовком

Присвоить выбранной ячейке таблицы заголовок.

Без разбиения

Не допускать переноса текста ячейки на следующую строку.

Минимальная ширина

Указать минимальную ширину ячейки в пикселах или в процентах от ширины таблицы. Ячейка, ширина которой определена в процентах, будет подстраиваться к тому или иному обозревателю.

Специальный фон

Использовать в качестве фона ячейки картинку. Просто выберите флажок **Использовать фоновое изображение** и нажмите кнопку **Обзор** для поиска файла картинки в форматах GIF, JPG, BMP, TIF, WMF, RAS, EPS, PCX или TGA.

Цвет фона

Раскрасить подложку ячейки.

Дополнительные цвета

В этом разделе вы можете определить цвет для рамки ячейки. Если вы выберите только параметр **Граница**, т.е. параметры **Светлая** и **Темная рамка** останутся используемыми по умолчанию, то через ниспадающий список цветов можно присвоить цвет всей рамке ячейки.

Имеющие приоритет параметры **Светлая** и **Темная рамка** позволяют присвоить цвет верхней и нижней рамкам ячейки.

Дизайн web-страниц с использованием графики

Редактор web-страниц Microsoft FrontPage Express поддерживает графические изображения двумя форматами GIF (CompuServe Graphics Interchange Format Расширение .GIF) и JPEG (Joint Photographic Experts Group Расширение .JPG). Это означает, что любой файл формата BMP, TIF, WMF, RAS, EPS, PCX и TGA при вставке в рабочую среду Microsoft FrontPage Express будет преобразован в GIF-картинку.

Несколько слов о форматах GIF и JPEG. GIF-формат хорош тем, что он маленький и имеет всего 256 цветов. Формат JPEG используется только для картинок, имеющих более 256 цветов. С другой стороны, вы можете создать JPEG-файл с потерей качества, но без потери количества цветов, это может значительно уменьшить размер файла. Поэтому, если вы хотите иметь хорошую картинку, просто преобразуйте ее в формат JPG.

Вы можете поместить в Microsoft FrontPage Express изображение посредством выбора из меню **Вставка** команды **Изображение**.

Изображение Microsoft FrontPage Express может быть помещено в любое место вашего документа, определенное точкой вставки. Просто поместите точку вставки в нужное место, выберите через диалоговое окно **Изображение** (доступ: **Вставка ⇨ Изображение**) имя графического файла и нажмите кнопку **OK**.

Вы можете помещать в документ Microsoft FrontPage Express только растровые (двоичные отображения) картинки. Растровые картинки строятся из отдельных точек и хранятся в виде групп точек, а экран представляется в виде прямоугольного массива точек, компьютеру указывается цвет каждого элемента изображения. Если документ с помещенной картинкой был сохранен, а затем картинка подверглась редактированию в какой-либо программе, то Microsoft FrontPage Express не внесет эти изменения в ваш документ.

Если вы переместите файл изображения в другую папку, то само изображение не отобразится в вашем документе. Помните, что ваше изображение должно храниться в том же каталоге, в каком находится файл исходного документа. Если формат графического файла не поддерживается программой, то стоит попытаться вставить картинку в Microsoft FrontPage Express из буфера обмена, или открыть картинку в исходном приложении, или сохранить ее в другом формате. Если же файл загружается в исходное приложение, но неправильно отображает картинку, то необходимо записать файл в другом графическом формате или распрощаться с этим файлом раз и навсегда.

При импортировании картинки в Microsoft FrontPage Express размер файла документа практически не увеличивается.

Библиотека рисунков Microsoft FrontPage Express

С помощью вкладки **Картинки** диалогового окна **Изображение** вы можете воспользоваться библиотекой рисунков Microsoft FrontPage Express.

Альтернативное отображение картинок

Microsoft FrontPage Express отображает изображения посредством заменяющего текста и так называемой превью-картинки. Превью-картинка имеет низкое разрешение (при загрузке исходного изображения сначала проявится изображение с низким разрешением), а заменяющий текст используется при просмотре web-страниц в текстовом режиме.

Чтобы отобразить картинку посредством заменяющего текста, выберите ее, обратитесь к вкладке **Общие** диалогового окна **Свойства изображения** (доступ: **Правка ⇨ Свойства изображения**) и в поле данных **Текст** введите информацию, тем или иным образом описывающую вашу картинку. Чтобы отобразить картинку посредством превью-картинки, выберите ее, обратитесь к вкладке **Общие** диалогового окна **Свойства изображения** (доступ: **Правка ⇨ Свойства изображения**) и с помощью кнопки **Обзор** раздела **Другие представления** выберите картинку с низким разрешением.

Как присвоить изображению ссылку

Вы можете присвоить помещенной в рабочую среду Microsoft FrontPage Express картинке любую ссылку. Для этого выберите картинку, обратитесь к вкладке **Общие** диалогового окна **Свойства изображения** (доступ: **Правка ⇨ Свойства изображения**) и с помощью кнопки **Обзор** раздела **Ссылка по умолчанию** выберите документ HTML, на который должен быть сделан переход.

Позиционирование изображений

Вы можете позиционировать изображениями, помещенными в рабочую среду Microsoft FrontPage Express. Для этого выберите изображение и обратитесь к вкладке **Внешний вид** диалогового окна **Свойства изображения** (доступ: **Правка ⇨ Свойства изображения ⇨ Внешний вид**).

С помощью ниспадающего списка **Выравнивание** укажите тип позиционирования вашего изображения относительно страницы.

С помощью опции **Интервал по горизонтали** вы можете сместить выбранное изображение относительно страницы по горизонтали. Параметр смещения одновременно воздействует на правую и левую части выбранного изображения. С помощью опции **Интервал по вертикали** вы можете сместить выбранное изображение относительно страницы по вертикали. Параметр смещения одновременно воздействует на верхнюю и нижнюю части выбранного изображения.

Основные свойства web-страницы

Свойства созданной страницы представлены диалоговым окном **Свойства страницы** (доступ: **Файл ⇨ Свойства страницы**).

Папка

Укажите путь к папке, в которой хранятся все файлы вашей web-страницы.

Заголовок

Текст, который вы впишите в это поле данных отобразится в заголовке окна обозревателя. Практически все поисковые машины Интернет начинают сканирование web-страниц с заголовка.

Как озвучить web-страницу

Вы можете озвучить вашу web-страницу, просто поместив в нее файлы звуков WAV, AIFF, AU MIDI. Для этого определите точку вставки и обратитесь к разделу **Фоновый звук** вкладки **Общие** диалогового окна **Свойства страницы** (доступ: **Файл ⇨ Свойства страницы**).

С помощью кнопки **Обзор** выберите звуковой файл на вашем диске. Если вы хотите, чтобы ваш звук все время играл в окне обозревателя, поставьте флажок **Непрерывно**. Если же вы хотите, чтобы ваш звук воспроизводился в течение определенного цикла, снимите флажок **Непрерывно** и укажите количество циклов воспроизведения с помощью счетчика **Цикл**.

Кодирование web-страниц

С помощью Microsoft FrontPage Express вы можете изменить кодировку символов web-страницы. Для этого обратитесь к разделу **Кодировка HTML** вкладки **Общие** диалогового окна **Свойства страницы** (доступ: **Файл ⇨ Свойства страницы**).

Отображение русских букв зависит от вашей кодировки. Поэтому для того, чтобы в обозревателе нормально отображались русские буквы, вам необходимо установить в систему многоязыковую поддержку и шрифты с соответствующей кодировкой, например KOI8. Как правило, при работе в Windows 98 для кириллизации используется кодировка CP1251, а на непонятных компьютерах с UNIX — кодировка KOI-8. Стандарт KOI8 является кириллической кодовой таблицей, позволяющей вам обмениваться некоторыми сообщениями электронной почты и просматривать web-страницы. Для того, чтобы Windows 98 понимала KOI8, необходимо установить многоязыковую поддержку и шрифты с кодировкой KOI8.

Помните, что российские серверы Web поддерживают русский язык либо через кодовую страницу 1251 (Windows 95), либо через таблицу КОИ-8 (Unix).

Как присвоить цвет или фон вашей web-странице

Вы можете указать цвет фона для вашего документа HTML. Для этого обратитесь к вкладке **Фон** диалогового окна **Свойства страницы** (доступ: **Файл ⇨ Свойства страницы**), поставьте флажок **Фоновое изображение** и с помощью кнопки **Обзор** выберите файл изображения, который будет использоваться в качестве фона. Вы можете указать дополнительный цвет фона для текущего документа. Для этого обратитесь к вкладке **Фон** диалогового окна **Свойства страницы** (доступ: **Файл ⇨ Свойства страницы**) и выберите из ниспадающего списка **Фон** понравившийся вам цвет.

Как присвоить цвет тексту и ссылкам

Вы можете присвоить цвет любому выделенному фрагменту текста, а также гиперссылке, просмотренной и активной ссылкам. Для этого обратитесь к вкладке **Фон** диалогового окна **Свойства страницы** (доступ: **Файл ⇨ Свойства страницы**) и выберите из ниспадающего списка **Текст** цвет для выделенного фрагмента текста. Цвет ссылки вы можете изменить с помощью ниспадающих списков **Гиперссылк**а, **Просмотренная ссылка** и **Активная ссылка**.

Мета-тэги

С помощью вкладки **Специальный** диалогового окна **Свойства страницы** (доступ: **Файл ⇨ Свойства страницы**) вы можете включить в ваш документ HTML дополнительную информацию, которая не отобразится в окне обозревателя, но помогает той или иной поисковой машине Интернет более легко обнаружить вашу web-страницу. Речь идет о так называемых мета-тэгах. Используя мета-тэги вы можете включить в свой документ, например, ключевые слова и короткое описание вашей web-странички:

```
<META name="description" content="Хороший сайт">
<META name="keywords" content="автомобиль, стоп">
```

Использование гипертекстовых ссылок

Гипертекстовая ссылка считается активной частью документа HTML. Это означает, что любому пользователю Интернет достаточно щелкнуть по ссылке для перехода к другой web-странице или другому узлу Интернет. Для того, чтобы создать ссылку, выделите текст или изображение и обратитесь к диалоговому окну **Создать гиперссылку** (доступ: **Правка ⇨ Гиперссылка**). С помощью ниспадающего списка **Тип гиперссылки** укажите необходимый тип гиперссылки, а в поле данных **Адрес (URL)** введите необходимый адрес URL.

Абсолютные ссылки указывают путь к месту вашего диска, где находится документ HTML. Такие ссылки совершенно не пригодны для публикации вашего документа в Web. Относительные ссылки не имеют пути, поэтому, если вы переместите ваши файлы, ссылки все равно будут работать.

Часть 5. Создание Интернет-портала

Глава 1. Что такое портал?

Для начала — нужно понимать, что «портал» это понятие не техническое и не технологическое, а маркетингово-конъюнктурное. Соответственно, точного определения у него быть не может. Поэтому понимать, что означает это слово, это значит:

- знать, в каких случаях и зачем оно употребляется
- представлять себе генезис термина (откуда он взялся и как эволюционировал)

С маркетингово-конъюнктурной точки зрения «портал» — это большой, сложный, крутой и дорогой сайт. Считается, что бюджет порталов должен на порядок превосходить бюджет «просто сайтов». Так что если нужно задрать или оправдать бюджет, то, что ты делаешь, нужно называть не «сайт», а «портал». Насколько он будет действительно сложен и крут, особого значения при этом не имеет — с этим будут разбираться потомки, споря «портал это или не портал».

Сам термин изначально является архитектурным и обозначает «парадный фасад здания». Его также лет тридцать назад полюбили фантасты, обозвав порталами приспособления для мгновенных перемещений между мирами. В любом случае, первичный смысл термина «Интернет-портал» означал сайт, с которого значительное количество пользователей начинало свое путешествие по Сети.

Первые сайты, удостоившиеся этого названия, были либо сайтами провайдеров (например AOL), либо каталогами (Yahoo) или поисковиками (Lycos, AltaVista). Предполагалось, что зайдя на этот сайт пользователь получает легкий доступ ко всей необходимой ему в Сети информации. Пользователю предоставлялась всякая общеполезная информация (погода, основные новости, индексы и курсы валют, программы телепередач и др.), возможность понять — куда ему надо, быстро и удобно туда попасть.

Причем, пока не развилась Интернет-реклама, никто особенно не стремился как можно дольше удерживать пользователя на своем ресурсе, и даже бытовала такая концепция как «Jump station» — станция перехода, с которой пользователь перепрыгивает туда, куда ему надо. Именно таковым был ранний Yahoo.

Затем, когда коммерческая ценность сайта стала измеряться количеством показанных банеров, все порталы стали прирастать собственным контентом, чтобы как можно дольше удержать пользователя у себя, а выпускать только в крайнем случае, и то — открывая целевой сайт в отдельном окне... Так в числе общепринятых атрибутов портала появилась масса собственного контента, а крайне симпатичная концепция jump stations почила в бозе.

Огромные массивы информации, накапливающиеся на порталах, привели к тому, что для обеспечения юзабильности (как это по-русски — возможности полезного использования?) пришлось придумывать механизмы настройки страниц под конкретных пользователей. Так в число атрибутов портала попала возможность кастомизации (индивидуальной настройки).

По мере разбухания порталов стало ясно, что кастомизация не спасает. Поэтому лет пять назад появилось понятие «вертикального портала», т.е. портала для определенного вертикального рынка (целевой группы). Соответственно, всеобъемлющие порталы предыдущего поколения переименовали в «горизонтальные». Вертикальные порталы сохранили претензию на всеобъемлющность, но уже в отдельно взятой предметной области.

Относительно недавно появилось понятие «корпоративного портала», т.е. сайта (обычно закрытого — интранетного), на котором предоставлен доступ ко всей корпоративной информации отдельно взятой компании. Сейчас это направление на подъеме, на рынке предлагается масса инструментов для разработки корпоративных порталов. Основная проблематика здесь — это разграничение доступа и извлечение и интеграция информации из различных корпоративных систем, которых только промышленных в мире многие тысячи наименований, а уж самодельных не счесть. Так что развивать это направление можно неограниченно долго...

Итак, уже очевидно, что само по себе слово «портал», не снабженное уточнениями, сохранило только конъюнктурный смысл.

Портал — это такой сайт, где интегрировано много всякой разной информации.

А все споры «портал или не портал» сводятся к субъективным трактовкам того — достаточно ли информации много, достаточно ли она разная и достаточно ли хорошо она интегрирована...

Глава 2. Что такое современный корпоративный портал?

Довольно трудно описать корпоративный портал (сайт). Сегодня не только сотрудники фирмы и их партнеры пользуются порталом. Старинное деление корпоративных сайтов на интранет (только для сотрудников), экстранет (только для партнеров) и Интернет (для всех желающих) безвозвратно устарело, так как появилась технологическая возможность интеграции всех этих частей в единое целое, разграничивающе информацию правами доступа. Можно указать несколько точек зрения на портал.

Сайт как корпоративное зеркало

Идеальный корпоративный портал представляет собой информационное зеркало всех аспектов деятельности фирмы. Все, что происходит в корпорации, отражается на ее портале, в открытом или закрытом доступе. Если объединить формальную (документы) и неформальную (записки, обсуждения, мнения, отчеты) информацию, то получается корпоративная база знаний, позволяющая отделить необходимую для работы фирмы информацию от ее живых носителей, и обеспечивающая возможность координации действий различных сотрудников.

Таким образом современный корпоративный портал — это информационная модель фирмы, и чем больше аспектов деятельности отражено в этой модели, тем более она полезна. Корпоративный портал отражает список текущих заказов, прайс-лист, состояние склада, список претензий заказчика X по проекту Y, проект технического задания для заказчика X, чем занимается сотрудник S, что висит на контроле у менеджера M и так далее.

В прошлом большое распространение получили две основные модели построения корпоративных порталов или сайтов — сайт с текстами и форумами и клиент-серверное приложение. Довольно редко они перемешивались на одном сайте, а тем более на одной странице. А ведь как

удобно было бы иметь прайс-лист, одним нажатие кнопки просмотреть всю информацию о реализации той или иной его позиции, список всех заданий, выполненных сотрудником S, список всех договоров, заключенных с клиентом C и так далее.

Сегодня оба метода основательно перемешаны: в рамках корпоративного портала и даже одной web-страницы. Информационное моделирование корпорации ведется всеми доступными методами одновременно: там, где это возможно, применяется классическое ИТ-приложение с web-интерфейсом, а где это трудно сделать — просто хранятся тексты и проводится их обсуждение. Например, наличие товаров на складе и рабочие операции по их отпуску проводятся в web-формочках ИТ-приложения, а вот обсуждение качества этих товаров и претензии к работе самого склада оформляются в виде структурированной дискуссии.

Отношение заказчик-подрядчик

Внутрифирменная работа на практике сводится к отношениям «заказчик-подрядчик» между конкретными людьми внутри фирмы. Кто-то заказывает работу, кто-то назначает исполнителя, а кто-то исполняет — и практически нет разницы, внутри фирменный заказчик или внешний. Так что основное, что должен поддерживать корпоративный портал — рабочее общение между заказчиками, менеджерами и исполнителями по поводу выполнения работы в непрерывно изменяющихся условиях. Поэтому корпоративный портал обязан поддерживать подобную технологию работы. В частности, портал должен понимать разницу между заказчиками, менеджерами, исполнителями, текущими проектами. Для всей этой информации нужно вводить ограничения по доступу и редактированию. Корпоративный портал позволяет естественным образом организовать навигацию и ввести систему разграничения доступа для сотрудников. Важно, что заказчик, менеджер и исполнитель имеют на портале разделяемую информационную модель выполнения заказа. В любой момент можно проконтролировать выполнение заказа, получить требуемую статистику.

Средство корпоративной коммуникации

Гибкость современного web-инструментария позволяет решать задачи, прежде малодоступные в архитектурах «клиент-сервер». Например, обеспечивать управление различными видами корпоративных коммуникаций. Обычно в больших корпорациях существует множество служб, отделов и задача организации их координации и взаимодействия весьма нетривиальна. Конечно, поддержка корпоративных коммуникаций не является единственной существенной инновацией, возможной в современных порталах по сравнению со старыми приложениями. Web-

технологии дистанционного обучения, клиентской поддержки, электронной коммерции уже давно появились на страницах корпоративных порталов. Вопрос только в том, когда все эти и еще многие другие необходимые для работы web-технологии станут такими же стандартными атрибутами компаний, как учрежденческие мини-АТС сегодня.

Возможность быстрого поиска необходимой информации и быстрого принятия решений

Одной из возможностей портала является предоставление сотрудникам, партнерам и просто всем желающим максимально быстрого поиска необходимой им информации. При этом необходимо четкое разделение прав доступа к той или иной информации. Поиск может осуществляться по всем видам информации, доступной порталу — как находящейся в реляционных базах данных, так и в виде web-страничек. Одно из преимуществ — нет необходимости задумываться, какие приложения вызывать, все сосредоточено в одном прозрачном, интуитивно понятном для пользователя интерфейсе. После того, как обеспечен доступ к требуемой информации, пользователь может действовать в соответствии с полученными данными. Например, сотрудник, просматривая полученные заказы, замечает, что один из заказов необычно большой. Его интересует, изменилось ли что-либо у заказчика на рынке готовой продукции за последнее время. Без использования портала ему бы пришлось просмотреть множество различных источников информации, что бы понять причины изменения объема заказа. Через портал вся эта информация была бы доступна по одному запросу.

Высокие технологии

Можно с уверенностью сказать, что портальные технологии в ближайшие два года будут востребованы на не менее чем 60% крупных российских предприятий (речь идет о компаниях из рейтинга top-200 или top-100). При этом, как показывает практика, наиболее эффективным внедрение портала оказывается для предприятий с географически распределенной структурой, а также для фирм, которым важно повысить эффективность взаимодействия между сотрудниками и автоматизировать процесс сбора и анализа информации для принятия правильных решений.

Для того, чтобы рассказанное выше не осталось сказкой, а становилось явью, требуется использование не высоких, а очень высоких технологий. Если на Западе стратегическим считается портальное решение, связанное с ERP и, желательно, исходящее от того же разработчика, то в России, за малой распространенностью ERP-систем, имеет место обратное движение — от портала к информационным системам предприятия.

Сегодня мы вплотную приблизились к тому, чтобы превратить портал из инструмента, обеспечивающего взаимодействие между сотрудниками и доступ к базам данных, в полноценное ядро информационной системы компании, интегрирующее различные процессы и системы (то есть, интеграционный портал). Кроме того, создаются новые бизнес-приложения, построенные на единой портальной платформе. По мере внедрения на рынок модели web-сервисов этот процесс будет развиваться существенно быстрее. Однако надо понимать, что внедрение корпоративного портала не может заменить традиционные системы автоматизации, такие как ERP. Корпоративный портал особенно эффективно будет функционировать в сочетании с другими системами автоматизации.

По сути корпоративный портал — лишь средство доступа сотрудников к необходимой информации, способное интенсифицировать определенные бизнес-процессы. Цена корпоративного портала несопоставима с ценой обычного интернет-сайта. Корпоративный портал — сложный инструмент, заточенный под процессы компании. Он требует тщательной проработки бизнес-процессов компании, интеграции с различными системами автоматизации (ERP, CRM), разделения прав доступа, особых средств защиты информации и многого другого.

Глава 3. Для чего нужен корпоративный портал?

Сначала разберемся, в чем разница между порталом и сайтом? Традиционный корпоративный сайт, как правило, существует отдельно от самой компании и не является повседневным рабочим инструментом сотрудников, тогда как портал прочно интегрирован в информационную систему компании, включает в себя различные бизнес-приложения, обладает развитой системой авторизации и разграничения прав доступа. Корпоративные порталы относятся к категории B2E (Bussiness To Employee), обозначающей взаимодействие компании с сотрудниками. Решения, направленные на улучшение доступа и обработки структурированной или неструктурированной информации обозначают как EIP (enterprise information portals — корпоративные информационные порталы). Они позволяют упорядочить бизнес-процессы, наладить документооборот, хранение корпоративных данных, их обработку.

Создание простого сайта может позволить себе каждый, у кого в штате есть более или менее грамотный технический специалист. Настраивается компьютер под управлением, например, Linux, устанавливается MySQL или Postgress, Apache, PHP, готовые форумы, новостные ленты, отправка SMS, «грабилка» ресурсов, и простенький сайт готов.

Сложнее обстоит дело с корпоративным порталом. На Западе наиболее перспективным считается решение, связанное с ERP-системами, причем желательно исходящее от того же разработчика. В России, наоборот, из-за малой на сегодняшний день распространенности ERP-систем, преобладает противоположное движение — от портала к информационным системам предприятия. Однако необходимо четко понимать, что нельзя смешивать понятия ERP и корпоративного портала. Внедрение корпоративного портала не может заменить традиционные системы автоматизации, такие как ERP. Корпоративный портал наиболее эффективно будет функционировать в сочетании с другими системами автоматизации на предприятии. По сути, корпоративный портал лишь средство доступа сотрудников к необходимой информации, способное интенсифицировать бизнес-процессы.

Корпоративный портал наиболее эффективен для рассредоточенных компаний, когда портал превращается в мощную информационную систему, связывающую все удаленные подразделения компании. Он особенно необходим, если компания имеет разветвленную региональную структуру: в этом случае корпоративный портал, построенный на базе интернет-технологий, является практически единственным экономически целесообразным решением проблемы прохождения информации и доступа к корпоративной базе знаний. Портал также эффективен в компаниях, где хорошо отлажены бизнес-процессы и взаимодействия подразделений. В противном случае, автоматизируя бардак, мы получим просто автоматизированный бардак.

Слово «портал» уже довольно давно вошло в лексикон российских пользователей Интернет, и благодаря восторженно-праздничной мишуре вокруг интернет-рынка его довольно часто воспринимают совсем не в том смысле, в котором это слово появилось первоначально. На деле понятию «портал» больше всего соответствует простое русское слово «вход». Таким образом, интернет-портал – это точка входа в Интернет, то есть, место, где для пользователя сведены ссылки на интересующую его информацию. Порталы принято делить на *горизонтальные* и *вертикальные*. Вертикальные порталы посвящены какой-то одной теме, например, компьютерному бизнесу или домашним животным. Горизонтальные же подобной специализацией не страдают и предлагают сведения о чем угодно, упорядочивая их по многочисленным категориям.

Функции корпоративного портала

- Ориентированность на работу с корпоративной БД и помощь в принятии правильных решений на основе найденной информации.

- Может быть интерфейсом к таким корпоративным приложениям, как ERP, приложения отдела продаж и маркетинга, управление поставками.
- Создание виртуальных рабочих групп, управления знаниями, управления контентом, документооборотом.
- Обращенность наружу, к партнерам и клиентам компании, что обозначается уже привычной аббревиатурой B2C (когда компания со своего портала предоставляет некие услуги клиентам — по заказам, биллингу, обслуживанию и так далее) и B2B решения, ориентированные на поставщиков, подрядчиков.
- Стыковка с уже имеющимися решениями по автоматизации управления компанией. Портал может взаимодействовать с унаследованными приложениями через промежуточное ПО интеграции приложений, называемое часто AIS. Оно выступает в роли общего программного шлюза, позволяет снизить число необходимых схем преобразования документов, контролировать права доступа к информации.
- Портал — «свалка» информации, из которой читатель/посетитель выбирает наиболее интересные ему сообщения. Портал не может нести ответственность перед читателем за достоверность, так как не является информационным агентством, портал несет ответственность перед агентством, которое предоставило ему данную информацию, портал обязуется не искажать предоставленную информацию и делать ссылку на первоисточник.

Ознакомившись с функциями, которые может выполнять портал, вы вполне сможете определиться, что именно вам необходимо: простенький сайт или корпоративный портал, а если портал, то выполнение каких функции и бизнес-процессов можно переложить на него.

Еще недавно наполнение сайтов содержанием не доставляло больших проблем. Сейчас, с появлением интернет-магазинов и крупных интернет-порталов это стало намного сложнее. Web-мастер, который раньше вполне справлялся с этим в одиночку, теперь не в состоянии следить за всем обилием информации.

Глава 4.
Использование пакета Cold Fusion

Пакет Cold Fusion фирмы Allaire — это средство для быстрой разработки интерактивных, динамических документов для Web основанное на обработке информации из баз данных, в основе которого лежит следующий набор технологий:

- HTML (Hyper-Text Markup Language)
- CGI (Common Gateway Interface)
- SQL (Structured Query Language)
- ODBC (Open Database Connectivity)

Разработка приложений с использованием Cold Fusion не требует программирования на таких языках как Perl, C/C++, Visual Basic или Delphi. Вместо этого вы создаете приложение, встраивая в обычный (стандартный) HTML файл специальные тэги для работы с базами данных.

Cold Fusion запускается как CGI приложение на различных Web-серверах под Windows и должен быть совместим с любым сервером, поддерживающим CGI.

Cold Fusion тестировался на совместимость со следующими серверами:

- O'Reilly WebSite
- Microsoft Internet Server
- EMWAC HTTPS
- Process Software Purveyor
- Netscape Communications/Commerce Server
- Internet Factory Communications/Commerce Builder
- Spry Safety Web Server
- CSM Alibaba

Для связи с различными СУБД Cold Fusion использует 32-разрядные ODBC-драйвера. Для корректной работы с Cold Fusion ODBC-драйвер должен удовлетворять следующим требованиям:

- Это должен быть 32-разрядный драйвер.

- Он должен поддерживать Уровень 1 ODBC API.
- Должна поддерживаться базовая грамматика SQL.
- Для совместимости с функциями ввода даты/времени, драйвер должен поддерживать соответствующие типы данных.

Для установки и использования Cold Fusion система должна удовлетворять следующим требованиям:

- Операционная система Windows.
- Микропроцессор Pentium и выше.
- 10Mb свободного дискового пространства.
- 24Mb RAM.
- Установленная сетевая поддержка TCP/IP.
- Установленный WWW сервер.

Для установки Cold Fusion нужно запустить программу **SETUP.EXE**, которая должна находится на инсталляционном диске 1.

Помимо копирования файлов, необходимых для работы Cold Fusion, в процессе установки, в корневой директории с документами Web-сервера создается директория с именем **CFPRO**. Эта директория содержит:

- Тест, для проверки правильности установки системы.
- Мини-учебник, в формате HTML, помогающий освоить азы Cold Fusion.
- Примеры приложений, демонстрирующие различные способы использования Cold Fusion.

Чтобы проверить правильность установки Cold Fusion, нужно открыть документ, URL до которого имеет вид **http://myserver/cfpro/getstart.htm**, где **myserver** — имя или IP-адрес вашего Web-сервера.

Администрирование

Для администрирования в Cold Fusion предусмотрен специальный интерфейс администратора. Этот интерфейс позволяет изменять различные параметры настройки Cold Fusion по четырем категориям:

- Data Sources — используется для настройки источников данных ODBC, для использования их с Cold Fusion. Чтобы

добавить источник данных, нужно нажать на кнопку **Add...**, выбрать один из установленных в системе драйверов ODBC и задать определенные для него настройки. Переопределить эти настройки можно воспользовавшись кнопкой **Setup...**, предварительно выделив конкретный источник данных. Если нужно определить способы взаимодействия Cold Fusion с источником данных, то нужно нажать на кнопку **Preferences...** и определить такие атрибуты как имя пользователя, пароль, допустимые операции с базой данных.

- **Templates** — используется для настройки логических путей до директорий, в которых расположены файлы с шаблонами Cold Fusion.
- **Debugging** — используется для настройки отладочных сообщений и сообщений об ошибках.
- **Mail** — используется для настройки параметров SMTP mail и позволяет просматривать журнал сообщений и ошибок.

Взаимодействие с базами данных

Cold Fusion позволяет динамически генерировать HTML документы основанные на запросах пользователя. Эти запросы передаются в Cold Fusion CGI-скрипт (**DBML.EXE**), который пересылает данные в Cold Fusion Engine, обрабатывающий эти данные в соответствии с заданным шаблоном, выполняя необходимые запросы и генерируя HTML документ, который отправляется пользователю.

Основой динамического создания документов являются специальные тэги, входящие в язык разметки DBML, ориентированные на работу с базами данных. Почти все основные возможности Cold Fusion сосредоточены в четырех тэгах:

- **DBQUERY** — выполнение SQL-запроса к базе данных;
- **DBINSERT & DBUPDATE** — создание и модификация записей в базе данных;
- **DBOUTPUT** — отображение результата запроса, допускающее его произвольное размещение среди HTML-тэгов.

Шаблон, на основе которого генерируется HTML-документ, представляет собой комбинацию тэгов HTML и DBML:

- **HTML**-тэги используются для форматирования как постоянной части документа, так и результатов запросов. Например, можно определить полужирный шрифт для каждого поля и разделительные линии между записями.
- **DBML**-тэги используются для формирования запроса к базе данных, а также определяют где и как будут отображены результаты запросов.

Когда пользователь нажимает кнопку типа **Submit** в форме или выбирает гипертекстную ссылку в документе, Web-браузер отправляет запрос на Web-сервер.

Web-сервер, если в запросе указан DBML-шаблон, запускает процесс Cold Fusion, отправляя ему данные, полученные от клиента.

Cold Fusion принимает данные, полученные от клиента, обрабатывает тэги DBML в шаблоне, включая подготовку запроса к базе данных и форматирование, которое будет использоваться в результирующем документе.

Cold Fusion взаимодействует с базой данных используя ODBC.

Cold Fusion динамически генерирует HTML-документ, содержащий результат выполнения запросов к базе данных, и возвращает его Web-серверу. Cold Fusion может также динамически генерировать почтовое сообщение и отправлять его через почтовый SMTP-сервер.

Web-сервер возвращает сгенерированный HTML-документ Web-клиенту.

Передача параметров в DBML-шаблон

Существует несколько способов передачи параметров между шаблонами. Можно передавать параметры непосредственно в URL, использовав для этого форму либо **cookie**.

Если параметры передаются через URL, то они добавляются к адресу вызываемого шаблона через символ **&** (амперсанд) в виде **параметр = значение**. Например, гипертекстовая ссылка, приведенная ниже, отправляет параметр с именем **user_id** и значением 5 в шаблон **example.dbm**:

```
<A HREF="cgi-shl/dbml.exe? Template=example.dbm&user_id=5">
```

При передаче параметров через форму используются поля формы, которые должны иметь имена, совпадающие с именами параметров, которые требуется передать. Ниже приведен пример передачи параметра из предыдущего примера, используя форму:

```
<FORM ACTION="cgi-shl/dbml.exe?Template=example.dbm">
```

```
<INPUT TYPE="HIDDEN" NAME="user_id" VALUE="5">
<INPUT TYPE="SUBMIT" VALUE="Enter">
</FORM>
```

Заметим, что при обращении к CGI-программе **DBML.EXE** должен быть определен стандартный параметр **Template**, указывающий на конкретный шаблон.

Переменные, занесенные в **cookie** браузера и переменные окружения CGI доступны в любом шаблоне.

Занесение и модификация данных с использованием тэгов DBINSERT и DBUPDATE

При использовании тэгов **DBINSERT** и **DBUPDATE** для занесения или модификации данных, параметры должны быть переданы в шаблон обязательно из формы, используя метод POST.

Для создания новой записи в базе данных используется тэг **DBINSERT**, а для модификации существующей записи используется тэг **DBUPDATE**. При использовании этих тэгов необходимо определить атрибуты **DATASOURCE** и **TABLENAME**.

DATASOURCE — это название источника данных ODBC, содержащего редактируемую таблицу, а **TABLENAME** — имя этой таблицы.

Например, если источник данных ODBC называется **Person DB**, а таблица, в которой требуется создать запись — **Person**, то тэг **DBINSERT** в шаблоне будет иметь следующий вид:

```
<DBINSERT DATASOURCE="Person DB" TABLENAME="Person">
```

Параметры, переданные в шаблон должны совпадать с именами полей таблицы, в которой создается (модифицируется) запись. В том случае, если не все передаваемые параметры должны участвовать в этой процедуре, используется атрибут **FORMFIELDS**, в котором через запятую перечисляются имена полей таблицы, для которых должны существовать одноименные параметры.

Особо следует отметить, что для того, чтобы использовать тэг **DBUPDATE** для модификации записи, в таблице должен быть определен первичный ключ, а для всех полей, включенных в первичный ключ из обрабатываемой HTML-формы, обязательно должны быть переданы соответствующие параметры.

Тэги **DBINSERT** и **DBUPDATE** могут иметь также еще два необязательных атрибута:

- **TABLEOWNER** — для тех источников данных, которые поддерживают права собственности на таблицы (например, SQL Server, Oracle), этот атрибут можно использовать, чтобы указать собственника таблицы.
- **TABLEQUALIFIER** — для различных источников данных этот атрибут может иметь разный смысл. Так, для SQL Server и Oracle — это имя базы данных, в которой содержится таблица, а для Intersolv dBase — это директория, в которой расположены DBF-файлы.

Пример.

Пусть определена HTML-форма для ввода данных:

```
<HTML>
<HEAD>
<TITLE>Пример ввода данных для создания записи</TITLE>
</HEAD>
<BODY>
<FORM ACTION="/cgi-shl/dbml.exe?Template=example.dbm"
METHOD="POST">
ФИО : <INPUT TYPE="Text" NAME="FullName">
Телефон : <INPUT TYPE="Text" NAME="Phone">
Дата рождения : <INPUT TYPE="Text" NAME="Birthday">
</FORM>
</BODY>
</HTML>
```

Следующий шаблон, **example.dbm**, которому будут переданы данные из формы, создает запись в таблице и выдает подтверждающее сообщение:

```
<DBINSERT DATASOURCE="Person DB" TableName="Persons"
FORMFIELDS="FullName,Phone,Birthday">
<HTML>
<HEAD><TITLE>Подтверждение</TITLE></HEAD>
<BODY>
<H1>Запись создана!</H1>
</BODY>
</HTML>
```

Выполнение запросов к базам данных

Для выполнения запросов к базе данных используется тэг **DBQUERY**. Этот тэг имеет следующий синтаксис:

```
<DBQUERY NAME="имя запроса"
```

```
DATASOURCE="имя источника данных odbc"
SQL="sql выражение" TIMEOUT=n MAXROWS=n DEBUG>
```

Атрибут **NAME** определяет имя запроса, которое используется далее для отображения результата выполнения запроса. Имя запроса должно начинаться с буквы и может содержать буквы и цифры (пробелов быть не должно).

Атрибут **DATASOURCE** задает имя источника данных ODBC, который должен быть создан с помощью интерфейса администратора Cold Fusion.

Ключевым атрибутом тэга **DBQUERY**, является атрибут **SQL**, который собственно и определяет запрос к базе данных на языке SQL (для улучшения читабельности, допускается расположение значения атрибута SQL на нескольких строках).

Создавая SQL запрос, следует помнить, что конкретная база данных может иметь свои особенности в синтаксисе SQL, использование которых ограничивается этой базой данных. Чтобы проверить, является ли конкретное SQL выражение совместимым с ODBC и независимым от конкретной базы данных, лучше всего использовать Microsoft Query, входящий в состав Microsoft Office. Для этого нужно в меню Microsoft Query выбрать **Файл ⇨ Выполнить SQL**, в появившемся окне диалога ввести предложение SQL, выбрать источник данных ODBC, нажав на кнопку **Источники...**, после чего нажать на кнопку **Выполнить**. Этот продукт можно также использовать и для создания SQL-выражений, используя для этого визуальные средства создания запросов. Получить SQL-выражение созданного таким образом запроса можно, нажав на кнопку **SQL** в панели инструментов.

Атрибут **MAXROWS** является необязательным и определяет максимальное количество записей, которые могут быть возвращены в результате выполнения запроса.

Атрибут **TIMEOUT** также является необязательным и определяет максимальное количество миллисекунд для выполнения запроса, до выдачи сообщения об ошибке. Заметим, что этот атрибут поддерживается только некоторыми ODBC-драйверами (например, драйвером для MS SQL Server 6.0).

Атрибут **DEBUG** используется для отладки запросов. При наличии этого атрибута пользователю отправляется дополнительная информация о выполнении этого запроса, такая как текст выполненного SQL-запроса, число возвращенных записей.

Приведем пример запроса с именем **AllPersons**, который возвращает все записи таблицы **Persons** из базы данных, с которой связан источник данных ODBC с именем **Person DB**:

```
<DBQUERY NAME="AllPersons" DATASOURCE="Person DB"
SQL="select * from Persons">
```

Для динамической настройки SQL-выражения можно использовать параметры, переданные в шаблон. Это могут быть параметры, переданные из формы, URL, а также переменные CGI. Параметры, используемые внутри SQL-выражения, должны быть обрамлены символом # (например, **#Name#**). При обработке запроса Cold Fusion ищет параметр с таким именем среди параметров, полученных из формы, в URL или среди переменных CGI. При нахождении подходящего параметра, его значение подставляется вместо соответствующей ссылки на параметр.

Примеры использования параметров в SQL-выражении.

Пример 1

Предположим что обрабатывается URL

```
/cgi-shl/dbml.exe?Template=prs.dbm&Id=22,
```

а атрибут **SQL** в **DBQUERY** имеет вид

```
SQL="select * from Persons where Id = #Id#",
```

тогда в базу данных будет передано следующее SQL-выражение:

```
select * from Persons where Id = 22.
```

Пример 2

Предположим, что в шаблон передан параметр **FirstLetters** и нужно найти в таблице **Persons** записи, в которых первые буквы в поле **FullName** совпадают со значением этого параметра. Значение атрибута **SQL** в этом случае будет следующим:

```
SQL="select * from Persons
where FullName like '#FirstLetters#%'"
```

Следует обратить внимание на то, что маска, состоящая из параметра и символа **%**, в отличие от предыдущего примера, обрамлена одинарными кавычками. Это связано с тем, что поле **Id** из примера 1 имеет числовой тип, а поле **FullName** — текстовый тип (синтаксис SQL требует, чтобы текстовые значения всегда были обрамлены одинарными кавычками). Для того чтобы задать маску, в примере использовался символ **%**, который в SQL-запросах соответствует произвольной последовательности символов. Также для определения маски может использоваться символ **_** (подчеркивание), соответствующий одному произвольному символу.

Использование результатов запроса для динамического создания HTML-документа

Для вывода данных, возвращаемых в результате выполнения запроса, определенного в **DBQUERY**, применяется тэг **DBOUTPUT**. Внутри этого тэга, связанного с конкретным запросом, может находиться обычный текст, тэги HTML, ссылки на поля, определенные в запросе. При обработке шаблона, содержимое тэга **DBOUTPUT** отправляется клиенту для каждой записи, возвращаемой в результате выполнения запроса, с подстановкой соответствующих значений параметров и полей.

Тэг **DBOUTPUT** имеет следующий синтаксис:

```
<DBOUTPUT QUERY="имя запроса" MAXROWS=n>
Текст, тэги HTML, ссылки на поля и параметры (то есть, #Name#)
</DBOUTPUT>
```

Атрибут **QUERY** применяется для указания имени запроса **DBQUERY**, результат выполнения которого будет использоваться, а атрибут **MAXROWS** определяет максимальное количество записей этого запроса, которые будут переданы для вывода в тэг **DBOUTPUT**.

Пример

Для вывода результата выполнения запроса с именем **AllPersons**, отображая имя персоны и телефон, и разделяя записи горизонтальной линией, может использоваться следующая конструкция:

```
<DBOUTPUT QUERY="AllPersons" MAXROWS=50>
<HR>
#FullName# (Телефон: #Phone#) <BR>
</DBOUTPUT>
```

Результат обработки этого тэга будет иметь вид:

```
<HR>
Иванов Иван Иванович (Телефон: 222-22-22) <BR>
<HR>
Петров Петр Петрович (Телефон: 444-44-44) <BR>
```

Вывод результата выполнения запроса в виде таблицы

Тэги **DBTABLE** и **DBCOL** всегда употребляются вместе для отображения результата выполнения запроса в виде таблицы.

Атрибуты тэга **DBTABLE**:

- **QUERY** — имя **DBQUERY**, для которого нужно отобразить данные;

- **MAXROWS** — максимальное количество записей, которое может быть отображено в таблице;
- **COLSPACING** — количество пробелов, которые будут вставлены между колонками (по умолчанию 2);
- **HEADERLINES** — количество строк, которые будут отведены для заголовка (по умолчанию 2);
- **HTMLTABLE** — при наличие этого тэга результат запроса будет отображен в виде HTML-таблицы, в противном случае будет использован тэг HTML **<PRE>**.
- **BORDER** — используется только вместе с атрибутом **HTMLTABLE** для отображения рамки в таблице.

Атрибуты тэга **DBTABLE**:

- **HEADER** — текст, который будет выводиться как заголовок колонки;
- **WIDTH** — ширина колонки в символах (по умолчанию 20);
- **ALIGN** — выравнивание содержимого колонки (**LEFT**, **RIGHT** и **CENTER**);
- **TEXT** — заключенный в кавычки текст, определяющий содержимое колонки, в котором могут находиться те же тэги, ссылки на параметры, что и в тэге **DBOUTPUT**.

Приведем пример использования тэгов **DBTABLE** и **DBCOL**:

```
<DBTABLE QUERY "AllPersons" MAXROWS=20>
<DBCOL HEADER="Фамилия Имя Отчество" WIDTH="30" TEXT="#FullName#">
<DBCOL HEADER="Телефон" WIDTH="10" TEXT="#Phone#">
<DBCOL HEADER="Дата рождения" WIDTH="9" TEXT="#
DateFormat(Birthday)#">
</DBTABLE>
```

Дополнительные замечания по созданию DBML-шаблонов

В шаблонах DBML для комментариев используется три тире (**<!---**), в отличие от двух в HTML (**<!--**). Эта специальная форма синтаксиса для комментариев позволяет Cold Fusion игнорировать тэги и текст, содержащиеся внутри этого комментария.

Ссылки на другие файлы (графические, HTML и CGI-программы), содержащиеся в DBML шаблоне, должны использовать полный путь, начиная с корневого каталога сервера.

Так как символ # является специальным символом в Cold Fusion, то чтобы включить его в область вывода, определяемую тэгом **DBOUTPUT**, нужно в шаблоне использовать два символа # вместо одного. Это же правило относится и к двойной кавычке, если этот символ нужно вставить, например, в атрибут **SQL**.

Cold Fusion не поддерживает имена полей, содержащих пробелы, внутри тэга **DBOUTPUT**. Если в имени поля все же встречаются пробелы, то при определении SQL-выражения в тэге **DBQUERY**, для каждого такого поля следует задать псевдоним. Например,

```
SQL="select ""Full Name"" as FullName from Persons"
```

Псевдонимы бывает также полезно применять для удобства, в случаях, если имя поля велико.

Использование параметров и переменных в шаблонах

Поля формы и параметры URL

Если в шаблон с помощью полей формы или в URL, были переданы параметры, то внутри любого тэга DBML к этим параметрам можно обращаться, используя следующий синтаксис: **#Form.Name#**, **#URL.Name#**. На самом деле, префиксы **Form.** и **URL.** могут опускаться, если заранее известно, что не может быть параметров других типов с такими же именами. Это правило относится ко всем типам параметров и переменных.

Переменные окружения CGI

Каждый сеанс связи, вызывающий CGI-программу, имеет конкретные переменные окружения. Доступ к ним из шаблона осуществляется, также как и к другим параметрам, только используется префикс **CGI.**, например, **#CGI.REMOTE_ADDR#**.

Применение тэга DBSET для создания переменных

С помощью тэга **DBSET** можно создавать переменные непосредственно в самом шаблоне и использовать их. Приведем пример, в котором создается переменная **#UserId#** и ей присваивается значение 10.

```
<DBSET #UserId#=10>
```

В правой части операции присваивания в **DBSET**, может находиться как число, текст (заключенный в кавычки), так и любые параметры, доступные в шаблоне, например, **#CGI.SCRIPT_NAME#**. Обращаясь к этим переменным, следует использовать префикс **Variable**, например, **#Variable.UserId#**.

HTTP Cookies

Cookies — это механизм, позволяющий приложениям со стороны сервера сохранять и использовать параметры на стороне клиента. Этот механизм поддерживается всеми версиями Netscape Navigator, MS Internet Explorer, начиная с версии 2.0, и будет поддерживаться остальными Web-браузерами в ближайшем будущем.

Для сохранения параметров в **Cookies** используется тэг **DBCOOKIE**, имеющий следующий синтаксис:

```
<DBCOOKIE NAME="Имя_параметра" VALUE="Значение параметра"
EXPIRES="Срок действия"
SECURE>
```

В атрибутах **NAME** и **VALUE** определяются имя и значение параметра соответственно. Атрибут **EXPIRES** определяет, когда закончится срок действия этого параметра. Этот атрибут может быть задан как дата, то есть, **10/09/97**, количество дней (то есть, 10, 100), **NOW** (удаляет параметр) или **NEVER**.

Наличие необязательного атрибута **SECURE** запрещает отправлять параметр браузеру, если тот не поддерживает стандарт SSL. К параметрам, хранимым в **cookies**, можно обращаться внутри любого тэга DBML, добавляя префикс **Cookies.**, например:

```
<DBOUTPUT>
#Cookies.User_Id#
</DBOUTPUT>
```

Использование результатов выполнения запросов

После выполнения запроса, результат его выполнения может быть использован в качестве динамического параметра для спецификации другого запроса.

Например, если создан запрос с именем **FindUser**, который возвращает идентификатор записи, расположенный в поле **USER_ID**, то можно использовать этот идентификатор в другом запросе, используя имя запроса как префикс к имени поля, разделяя их точкой (то есть, **#FindUser.UserId#**).

Каждый запрос, описанный тэгом **DBQUERY**, после выполнения имеет два специальных атрибута, **RecordCount** и **CurrentRow**, содержащих информацию о количестве возвращенных в результате выполнения запроса записей и о текущей записи, обрабатываемой тэгом **DBOUTPUT**, соответственно. Используются эти атрибуты так же, как и поля запроса (**#FindUser.RecordCount#**).

Проверка корректности данных и форматирование вывода

Проверка корректности данных в полях формы

В Cold Fusion предусмотрен механизм проверки корректности заполнения полей формы. Этот механизм основан на добавлении в форму дополнительных полей типа **HIDDEN** (скрытые), с именем, составленным из имени поля, значение которого нужно проверить, и одного из допустимых в Cold Fusion суффиксов, задающих контекст проверки. Ниже приведен список всех суффиксов, используемых для проверки:

_required

Текст сообщения об ошибке. Проверяется, произведен ли ввод в поле формы.

_integer

Текст сообщения об ошибке. Проверяется, является ли значение, введенное пользователем, целым числом.

_float

Текст сообщения об ошибке. Проверяется, является ли значение, введенное пользователем, числом.

_range

```
MIN = Минимальное_Значение
MAX = Максимальное_Значение
```

Проверяется, находится ли введенное число в определенных границах.

_date

Текст сообщения об ошибке. Проверяется, находится ли введенная пользователем дата в одном из следующих форматов: **DD/MM/YY**, **DD/MM/YYYY**, **DD/MM** (используется текущий год). В качестве разделителя можно, также, использовать тире (то есть, **DD-MM-YY**).

Пример

Ниже приведен фрагмент описания формы, состоящий из двух текстовых полей: обязательное для заполнения поле **FullName** и поле типа дата **Birthday**, и для каждого из этих полей описано поле типа **HIDDEN**, задающее контекст проверки.

```
Фамилия Имя Отчество : <INPUT TYPE="TEXT" NAME="FullName">
<INPUT TYPE="HIDDEN" NAME="FullName_required"
VALUE="Фамилия Имя Отчество должны быть заданы!">
```



```
Дата рождения : <INPUT TYPE="TEXT" NAME="Birthday">
<INPUT TYPE="HIDDEN" NAME="Birtday_date"
VALUE="Дата рождения должна быть в формате DD/MM/YYYY (например,
20.08.1968)">
```

Функции вывода в DBML

Для отображения данных в нужном формате в Cold Fusion предусмотрены специальные функции. При использовании такой функции, примененной к конкретному параметру, она заключается в символ #, например, **#DateFormat(Form.LastUpdate)#**.

Ниже приведен список основных функций, ее описание и пример использования.

DateFormat

Отображает поле базы данных типа **дата/время** или **дата** в формате **DD/MM/YY**.

```
12/01/96
```

TimeFormat

Отображает поле базы данных типа **дата/время** в формате **HH:MI AM/PM**.

```
10:22 AM
```

NumberFormat

Отображает числовые значения как целые числа, разделяя разряды запятой.

```
10,256
```

DecimalFormat

То же, что и **NumberFormat**, плюс отображаются два знака после десятичной точки.

```
10,256.3
```

DollarFormat

То же, что и **DecimalFormat**, плюс добавляется символ **$** и вместо знака минус перед отрицательным значением оно помещается в скобки.

```
$10,256.73
```

YesNoFormat

Отображает данные логического типа как **Yes** или **No**. Все ненулевые значения интерпретируются как **Yes**, нуль — как **No**.

```
Yes
```

ParagraphFormat

Применяется при отображении данных, введенных в поле **TEXTAREA**. Преобразует символ перевода строки в пробел, два перевода строки подряд — в тэг параграфа HTML (**<P>**).

HTMLCodeFormat

Удаляет символ перевода строки и пропускает все специальные символы (**>**, **<**, **"**, **&**), применяя к тексту тэг переформатирования HTML (**<PRE>**).

HTMLEditFormat

То же, что и **HTMLCodeFormat**, только без добавления тэга **<PRE>**.

Кроме вышеперечисленных, есть еще несколько функций, обеспечивающих дополнительные возможности манипулирования параметрами в шаблоне:

ParameterExists

Эта функция проверяет, доступен ли в шаблоне параметр с заданным именем, возвращая **Yes** или **No**. Например, чтобы проверить был ли отправлен из формы параметр **UserId**, используется следующее выражение:

```
<DBIF #ParameterExists(Form.UserId)# is Yes>
```

PreserveSingleQuotes

Эта функция обычно используется в SQL выражениях для устранения из значений параметров одиночных кавычек, которые являются специальным символом в SQL. Приведем пример использования этой функции:

```
SELECT * FROM Persons WHERE
FullName Like '#PreserveSingleQuotes(Form.FullName)#%'
```

URLEncodedFormat

Функция заменяет пробелы на символ + и все не латинские символы и цифры — шестнадцатеричным эквивалентом, что позволяет использовать результат в строке URL.

IncrementValue и DecrementValue

Эти функции возвращают переданный им параметр, прибавив или отняв от него единицу соответственно. Например, чтобы увеличить параметр **OrderCount** можно воспользоваться следующим выражением:

```
<DBSET #OrderCount# = #IncrementValue(OrderCount)#
```


ValueList и QuotedValueList

Используя в качестве аргумента имя поля конкретного запроса, эти функции возвращают разделенный запятыми список значений этого поля для каждой записи, возвращенной в результате выполнения запроса.

Например, если запрос возвращает четыре записи, то результат функции **ValueList** будет иметь вид **11,22,33,44**, а результат функции **QuotedValueList**, примененной к этим же данным, будет возвращать **'11','22','33','44'**.

Эти функции могут применяться для использования результата одного запроса в операции **IN** последующего запроса, например:

```
<DBQUERY NAME="Customers" ...определение запроса...>
<DBQUERY NAME="CustomerOrders" DATASOURCE="EXAMPLE"
SQL="SELECT * FROM Orders WHERE Customer_ID
IN ( #ValueList(Customer.CustomerID)# )">
```

Динамическое изменение содержимого документа

Условный оператор (DBIF & DBELSE)

Основным средством динамического определения содержимого документа являются тэги **DBIF** и **DBELSE**, позволяющие производить проверку некоторого условия и в зависимости от этого формировать результат.

Тэг **DBIF** имеет следующий синтаксис (тэг **DBELSE** может не использоваться):

```
<DBIF значение оператор значение>
тэги HTML и DBML
<DBELSE>
тэги HTML и DBML
</DBIF>
```

В качестве элемента тэга **DBIF** — «значение», могут использоваться любой параметр или переменная Cold Fusion (например, **#Form.Name#**, **#CGI.User_Agent#**), числовое значение, произвольная последовательность символов (заключенная в кавычки).

Элемент тэга **DBIF** — «оператор» ограничивается следующим списком:

- **is** — сравнивает два значения, с учетом регистра, и возвращает значение **True** (истина), если эти значения совпадают;

- **is not** — оператор, обратный оператору **is**;
- **Contains** — проверяет, содержится ли значение находящееся слева от оператора в значении справа и возвращает **True**, если да;
- **does not contain** — оператор, обратный оператору **contains**;
- **great then** — проверяет, что значение слева от оператора больше чем справа и возвращает **True**, если да;
- **less then** — проверяет, что значение слева от оператора меньше, чем справа и возвращает **True**, если да;
- **greater then or equal to** — оператор, обратный оператору **less then**;
- **less then or equal to** — оператор, обратный оператору **great then**;

Пример

```
<DBIF #PersonSearch.RecordCount()# is 0>
<P>Лиц, удовлетворяющих заданным критериям поиска,
в базе данных не обнаружено!
<DBELSE>
<DBOUTPUT QUERY="PersonSearch">
<HR>
Фамилия Имя Отчество: #FullName# <BR>
<DBIF #Phone# is "">
Телефон: #Phone# <BR>
<DBIF>
</DBOUTPUT>
</DBIF>
```

Перенаправление на другой URL (DBLOCATION & DBABORT)

Для перенаправления пользователя на другой URL предназначен тэг **DBLOCATION**.

Этот тэг обычно применяется, если в шаблоне выполняется один или более запросов, а затем нужно сразу же перейти к другому документу, либо URL, на который нужно отправить пользователя, зависит от параметра. Приведем пример использования тэга, иллюстрирующий его синтаксис:

```
<DBIF #NewPassword# is not #PasswordConfirmation#>
<DBLOCATION URL="/login/invalidpassword.htm">
</DBIF>
```


В качестве значения атрибута URL тэга **DBLOCATION** можно использовать параметры и переменные, допустимые в шаблоне. Например, **<DBLOCATION URL=#Page#>**.

Для прерывания обработки шаблона в Cold Fusion используется тэг **DBABORT**.

```
<P>Этот текст будет отправлен клиенту
<DBABORT>
<P>Этот текст не будет отправлен клиенту
```

Обычно этот тэг используется при неправильной аутентификации.

Включение в шаблон других шаблонов

По мере усложнения приложений, разрабатываемых с использованием Cold Fusion, появляется необходимость упростить используемые шаблоны. Одним из способов решения этой проблемы, предлагаемых в Cold Fusion, является выделение из шаблонов часто используемых блоков, таких как запросы и области вывода, и их многократное использование в других шаблонах. Для включения часто используемых шаблонов в другой шаблон, используется тэг **DBINCLUDE**.

Тэг **DBINCLUDE** может быть расположен в любом месте шаблона, кроме как в тэгах **DBQUERY**, **DBOUTPUT** и **DBTABLE**. Тэг **DBINCLUDE** имеет атрибут **TEMPLATE**, который задает путь до файла с шаблоном. Этот шаблон будет обработан Cold Fusion как часть основного шаблона (то есть, в нем могут использоваться запросы, уже запущенные в основном шаблоне, а также ссылки на параметры формы, URL и CGI).

Приведем пример включения в шаблон шаблона с именем **test.dbm**:

```
<DBINCLUDE TEMPLATE="test.dbm">
```

Определение типа данных MIME для содержимого документа

Cold Fusion позволяет задавать тип MIME для данных, которые будут отправлены пользователю из текущего шаблона (по умолчанию используется **text/html**). Для этого используется тэг **DBCONTENT**, имеющий единственный атрибут **TYPE**, который, собственно, и задает тип данных. Например, чтобы отправить клиенту VRML-документ может использоваться следующий шаблон:

```
<DBCONTENT TYPE="x-world/x-vrml">
<DBQUERY NAME="GetCyberRoom"
SQL="SELECT VRML_Script FROM CyberRooms WHERE
RoomNumber=#URL.RoomNumber#">
<DBOUTPUT QUERY="GetCyberRoom">
```

```
#VRML_Script#
</DBOUTPUT>
```

Заметим, что Cold Fusion не будет отправлять клиенту текст, расположенный до тэга **DBOUTPUT**.

Расширенные возможности

Динамическое определение SQL выражения

В некоторых сложных приложениях может потребоваться, в зависимости от значений параметров, определять не только содержание запроса, но и его структуру. В Cold Fusion предусмотрен тэг **DBSQL**, который может употребляться внутри тэга **DBQUERY**, доопределяя SQL-выражение, в зависимости от значений параметров. Тэг **DBSQL** имеет единственный атрибут SQL, значение которого будет добавляться к основному SQL-выражению.

Пример

```
<DBQUERY NAME="SiteSearch" DATASOURCE="Sites Database"
SQL="SELECT * FROM SITES WHERE SiteType = #SiteType# ">
<DBIF #Form.City# is not "">
<DBSQL SQL=" AND City = '#Form.City#' ">
</DBIF>
<DBIF #Form.SortOrder# is not "">
<DBSQL SQL=" ORDER BY #Form.SortOrder# ">
</DBIF>
</DBQUERY>
```

Поддержка транзакций

Для объединения нескольких запросов в одну транзакцию может быть использован тэг **DBTRANSACTION**. Все запросы, содержащиеся внутри этого тэга, будут интерпретироваться как одна транзакция. То есть все изменения, сделанные в базе данных, либо будут одновременно сохранены, либо не будет сохранено ни одно из них.

Приведем пример, в котором денежная сумма переводится с одного банковского счета на другой:

```
<DBTRANSACTION>
<DBQUERY NAME="WithdrawCash" DATASOURCE="Bank Accounts"
SQL = "UPDATE Accounts SET Balance = Balance - #Amount#
WHERE Account_ID = #AccountFrom# ">
<DBQUERY NAME="DepositCash" DATASOURCE="Bank Accounts"
SQL = "UPDATE Accounts SET Balance = Balance + #Amount#
WHERE Account_ID = #AccountTo# ">
</DBTRANSACTION>
```


Заметим, что не все драйверы ODBC поддерживают транзакции. Например, драйверы для Oracle, SQL Server и Access поддерживают транзакции, а драйверы для FoxPro, dBase и Paradox — нет.

Вложенные области вывода и группирования

Тэги **DBOUTPUT** могут вкладываться друг в друга, с целью сгруппировать области вывода. Группирование достигается с помощью использования атрибута **GROUP** в тэге **DBOUTPUT**, который содержит другой тэг **DBOUTPUT**. Этот атрибут определяет поле, по которому будет производится группирование. Внешний тэг **DBOUTPUT** обычно используется для вывода заголовка группы, а внутренний — для вывода записей, содержащихся в группе.

Приведем пример вывода сотрудников организации, сгруппированных по отделам:

```
<DBQUERY NAME="ListEmployees" DataSource="Employees"
SQL="SELECT * FROM Emp ORDER BY Department">
<DBOUTPUT QUERY="ListEmployees" GROUP="Department">
<P> <H2>#ListEmployees.Department#</H2>
<UL>
<DBOUTPUT>
<LI> #FullName# ##
</DBOUTPUT>
</UL>
</DBOUTPUT>
```

Обратим внимание на то, что если поле используется для группирования, то результат запроса должен быть отсортирован по этому полю. Если используется многоуровневое группирование (ограничений на количество уровней вложенности тэгов **DBOUTPUT** нет), то соответственно в SQL-выражении должна быть задана многоуровневая сортировка (например, **ORDER By Country, Region**).

Использование списочных полей с множественным выбором

Если HTML-форма содержит поле типа **SELECT** с множественным выбором, либо поля с одинаковыми именами (например, поля типа **checkbox**), то данные будут переданы в шаблон в виде списка значений, разделенных запятыми. Такая форма представления наиболее удобна для использования в операторе **IN** языка SQL.

Пример

Предположим, что в форме содержится поле:

```
<SELECT NAME="SelectedPersons" MULTIPLE SIZE="3">
<OPTION VALUE="1">Иванов И.И.
<OPTION VALUE="2">Петров П.П.
```

```
<OPTION VALUE="3" SELECTED>Сидоров С.С.
</SELECT>
```

Этот параметр, переданный в шаблон, может быть использован в следующем SQL-выражении:

```
SQL="SELECT * FROM Persons
WHERE Person_ID IN ( #SelectedPersons# )
```

Следует обратить внимание на то, чтобы параметр, который используется в операторе **IN**, был не пуст. Для этого можно, пользуясь стандартными средствами Cold Fusion, описать поле как требуемое, либо использовать поле типа **HIDDEN** с тем же именем и с заведомо неверным значением. Например, предыдущий пример можно дополнить следующим полем:

```
<INPUT TYPE="HIDDEN" NAME="SelectedPersons" VALUE="-1">
```

Дополнительные команды SQL

В тэге **DBOUTPUT**, кроме выражения **SELECT** языка SQL, допускается использование и любых других, допустимых для конкретного источника данных, SQL-команд, включая:

- **INSERT** — добавление записи в таблицу.
- **UPDATE** — модификация записи в таблице.
- **DELETE** — удаление записи из таблицы.

Использование непосредственно команд SQL вместо тэгов **DBINSERT** и **DBUPDATE** в некоторых случаях может обеспечить большую гибкость и эффективность. Например, при модификации или создании новой записи, появляется возможность использовать все параметры и переменные, доступные в шаблоне, в том числе и результаты определенных в шаблоне запросов.

Глава 5. Использование пакета Web-Oracle-Web (WOW)

Пакет WOW предназначен для использования под ОС Unix. Пакет предназначен для обработки запросов от WWW-сервера (Web) к SQL-серверу Oracle (Oracle) с генерацией динамических HTML-документов (Web). Разработчик приложений, использующий WOW оперирует только с родным языком Oracle — PL/SQL, являющимся процедурным расширением языка SQL. Это обуславливает высокую эффективность разработки приложений. Обработка созданных приложений непосредственно в сервере Oracle определяет высокую скорость исполнения приложений.

Основная идея пакета WOW — преобразование запроса к WWW-серверу в вызов определенной процедуры PL/SQL. В качестве параметров процедуры, используются данные из запроса к WWW-серверу. Кроме этого, язык PL/SQL дополняется функциями вывода различных данных в формате HTML.

Состав

Структурно, WOW состоит из ряда исполняемых программ, соответствующих спецификации CGI и набора пакетов PL/SQL.

Пакет **htp** содержит процедуры и функции, облегчающие формирование HTML-документа. Пакет **htf** содержит описание различных констант и функций, используемых при формировании HTML-документов.

Установка

Для корректной работы пакета необходимо правильно провести процедуру установки. Пакет WOW требует около 2Mb дискового пространства. На базовом сервере должны быть установлены:

- операционная система семейства Unix;
- WWW-сервер;
- сервер баз данных Oracle или сетевой стек Oracle SQL*Net с возможностью доступа к удаленному серверу баз данных Oracle.

Этап I

В случае, когда пакет поставляется в виде исходных текстов, необходимо произвести компиляцию и сборку исполняемого модуля **wowstub**. При сборке **wowstub** необходимо использовать библиотеки установленного сервера Oracle или сетевого стека SQL*Net. Компиляция и сборка производится утилитой **make** на основании данных файла **Makefile**. Вам необходимо изменить ряд параметров **Makefile** для настройки на вашу конфигурацию Oracle и Unix:

- **ORACLE_HOME** — должен совпадать с каталогом, определенным переменной **ORACLE_HOME** сервера Oracle.
- **DEST_DIR** — должен указывать на каталог, хранящий CGI-модули вашего WWW-сервера.

- **DOC_ROOT** — должен указывать на каталог в котором будет размещена документация по WOW. Каталог должен быть доступен для WWW-сервера.

Этап II

Независимо от сборки **wowstub**, необходимо создать ряд структур данных в БД Oracle.

1. Создать пользователя, через которого WOW будет осуществлять доступ к данным и исполнение рабочих процедур. Обычно используется имя WWW.

2. Создать из под вышеупомянутого пользователя все необходимые структуры данных и примеры. Для этого необходимо исполнить следующие SQL файлы, идущие в дистрибутиве WOW:

- **wow.sql**
- **ht.sql**
- **math.sql**
- **emp.sql**
- **showemp.sql**
- **hanoi.sql**.

Исполнить эти файлы можно с использованием одной из следующих утилит:

- **SQL*Plus**
- **SQL*DBA**
- **Server Manager**
- **Enterprize Manager**.

Этап III

Необходимо поместить модуль **wowstub** в каталог CGI программ вашего WWW-сервера. Необходимо переопределить ряд параметров файла **wow**, представляющего собой скрипт **sh**:

- **ORACLE_HOME** — в соответствии с параметром **ORACLE_HOME** вашего сервера Oracle или стека SQL*Net;
- **TWO_TASK** — в соответствии с параметром **TWO_TASK** клиентской части вашего сервера Oracle;

- **WOW_UID** — в соответствии с именем пользователя и его паролем, созданными на этапе II установки.

Отредактированный файл wow необходимо поместить в каталог для CGI-программ вашего WWW-сервера.

Использование

Рассмотрим простейший пример с использованием пакета WOW. При обращении к WWW-серверу **www.cnit.nsu.ru** по URL:

```
http://www.cnit.nsu.ru/cgi-bin/wow.win /example.test?answer=no
```

происходит следующая цепочка действий:

- WWW-сервер интерпретирует это обращение как запуск CGI-программы **wow.win**.
- Программа **wow.win** интерпретирует параметры как вызов процедуры **test** пакета **example** с параметром **answer**, имеющим значение **no**, созданной в схеме WWW сервера Oracle.
- Сервер Oracle исполняет эту процедуру и все процедуры и функции, вызываемые из нее. Выходные данные, представляющие динамически созданный HTML-документ, передаются программе **wow.win**.
- Программа **wow.win** перекодирует выходной документ в кодировку Microsoft CodePage 1251, используемую в Windows-приложениях, и передает его WWW-серверу.
- WWW-сервер возвращает созданный документ, как результат запроса, WWW-клиенту.

URL, обращающийся к процедуре PL/SQL должен быть построен по определенным правилам и содержать ряд элементов:

- Указатель на модуль пакета WOW, находящийся в каталоге CGI-программ.

Расширение программы **wow** — **.koi8**, **.win**, **.iso**, **.alt** определяет кодировку WWW-клиента:

- КОИ-8
- Microsoft Code Page 1251
- ISO 8859-5
- Microsoft Code Page 866

Например:

```
http://www.cnit.nsu.ru/cgi-bin/wow.win
```

Имя процедуры PL/SQL, к которой происходит обращение. Модули пакета используют схему и регистрационные данные пользователя www БД Oracle. Таким образом, вызываемая PL/SQL-процедура должна быть доступна пользователю www на исполнение. Если процедура (test) создана прямо в схеме www, необходимо просто указать ее имя:

```
http://www.cnit.nsu.ru/cgi-bin/wow.win/test
```

Если процедура входит в состав пакета (example), созданного в схеме www, необходимо добавить имя пакета и имя процедуры:

```
http://www.cnit.nsu.ru/cgi-bin/wow.win/example.test
```

Когда пакет создан в другой схеме Oracle, необходимо указывать и имя схемы. Например, для схемы **fancy** получим:

```
http://www.cnit.nsu.ru/cgi-bin/wow.win /fancy.example.test
```

Пользователь www должен иметь права на исполнение этой процедуры, явно предоставленные оператором **GRANT** языка SQL.

После имени процедуры, через разделитель **?**, начинают перечисляться параметры процедуры и их значения в виде пар:

```
<название_параметра> = <значение_параметра>
```

Между собой различные параметры разделяются амперсандом (**&**):

```
<название_параметра1>=<значение_параметра1>&
<название_параметра2>= <значение_параметра2>
```

Название параметра должно совпадать с названием параметра вызываемой процедуры. Число параметров должно в точности соответствовать числу параметров процедуры. Если хотя бы одно из этих требований не соблюдено, вы получите сообщение об ошибке. Порядок указания параметров значения не имеет. Большие и маленькие буквы в названии параметров равнозначны.

Число реально передаваемых параметров может и не совпадать с числом параметров, указанных в спецификации процедуры. В этом случае, все опускаемые параметры должны иметь значения по умолчанию.

Пример:

```
http://www.cnit.nsu.ru/cgi-bin/wow.win /example.test?answer=no
```

Исходя из описанного механизма работы пакета WOW можно сформулировать основные требования к PL/SQL-процедурам, обрабатывающим запросы от WWW-сервера.

Все входные переменные, передающиеся через WOW в процедуру, всегда имеют тип **varchar2**. Если вы хотите использовать какой-либо другой тип данных, необходимо использовать функции преобразования из **varchar2**.

В пакете **htp** отсутствуют функции вывода начала и конца HTML-документа. Поскольку многие современные браузеры интерпретируют текст без обрамляющих тэгов **<HTML> ... </HTML>** как переформатированный, необходимо прямо задавать эти тэги в начале и в конце документа.

Пример пакета **example**:

```
Create or Replace package example is
procedure test(answer in Varchar2);
end;
/
Create or Replace package body example is
procedure test(answer in Varchar2) is
ответ varchar2(3);
cursor c_man(ans in varchar2) is select Фамилия from
Результаты_опроса where Ответ=ans order by Фамилия;
begin
-- Начало документа
htp.p('<HTML>');
-- Вывод названия страницы и заголовка
if answer = 'no'
then
ответ:='НЕТ';
htp.htitle('Фамилии людей, ответивших отрицательно');
else
ответ:='ДА';
htp.htitle('Фамилии людей, ответивших положительно');
end if;
htp.olistopen;
-- Начало нумерованного списка
for man in c_man(Ответ) loop
-- Элемент списка
htp.item(man.Фамилия);
end loop;
-- Конец нумерованного списка
htp.olistclose;
-- Конец документа
htp.p('</HTML>');
end;
```

```
end;
/
```

При этом подразумевается что в схеме WWW Oracle находится таблица примерно следующей структуры:

```
Create table Результаты_опроса(Фамилия varchar2(30),
Имя varchar2(14),
Отчество varchar2(20),
Ответ varchar2(3));
```

Обращаться к пакету WOW можно и из форм HTML. Ниже приведен пример обращения к тому же пакету **example** из простейшей формы.

```
<HTML>
<HEAD>
<TITLE>Тестовая форма</TITLE>
</HEAD>
<BODY>
<H1>Тестовая форма</H1>
<FORM ACTION="http://www.cnit.nsu.ru/cgi-
bin/wow.win/example.test">
Введите ответ:<INPUT NAME="answer">
<INPUT VALUE="Найти" TYPE="SUBMIT">
</FORM>
</BODY>
</HTML>
```

Спецификация процедур пакета htp

- **procedure title(ctitle in varchar2)** — выводит название документа (тэги **<TITLE>**).
- **procedure htitle(ctitle in varchar2)** — выводит название документа и повторяет его в заголовке первого уровня (тэги **<TITLE>**, **<H1>**).
- **procedure header(nsize in integer, cheader in varchar2)** — выводит заголовок уровня **nsize** (тэги **<H1> ... <H6>**).
- **procedure url(curl in varchar2, cname in varchar2)** — формирует **cname** как гипертекстовую связь, указывающую на **curl** (тэги **<A HREF>**).
- **procedure gif(curl in varchar2)** — включает в документ картинку, путь до которой **curl** (тэги **<IMG>**).
- **procedure gif(curl in varchar2, calign in varchar2)** — включает в документ картинку, путь до которой **curl** с выравниванием, определяемым параметром **calign** (тэги **<IMG>**).

- **procedure bold(ctext in varchar2)** — выводит текст **ctext** жирным шрифтом (тэги **<B>**).
- **procedure italic(ctext in varchar2)** — выводит текст **ctext** шрифтом **italic** (тэги **<I>**).
- **procedure item(cval in varchar2)** — выводит **cval** как элемент списка (тэги **<ITEM>**).
- **procedure formOpen(curl in varchar2)** — создает форму с действием **curl** (тэги **<FORM>**).
- **procedure formHidden(cname in varchar2, cvalue in varchar2)** — создает скрытое поле формы для хранения значения **cvalue** переменной с именем **cname**.
- **procedure formPassword(cname in varchar2)**, **procedure formPassword(cname in varchar2, cvalue in varchar2)** — создает поле формы для ввода значения переменной — пароля с именем **cname** и значением по умолчанию **cvalue**.
- **procedure formField(cname in varchar2, nsize in integer), procedure formField(cname in varchar2), procedure formField(cname in varchar2, cvalue in varchar2)** — создает поле формы для ввода значения переменной с именем **cname** длиной **nsize** со значением по умолчанию **cvalue**.
- **procedure formText(cname in varchar2, nrow in integer, ncol in integer)** — создает многострочное поле формы (длиной **ncol**, высотой **nrow**) для ввода значения переменной с именем **cname**.
- **procedure formCheckbox(cname in varchar2)** — создает элемент **checkbox** для ввода значения логической переменной **cname**.
- **procedure formRadio(cname in varchar2, cval in varchar2)** — создает элемент **radiobutton** для ввода одного из значений **cval** переменной **cname**.
- **procedure formSelectOpen(cname in varchar2)** — создает список значений для переменной с именем **cname**.
- **procedure formSelectOption(cval in varchar2)** — добавляет значение **cval** в список значений переменной, описанной в **formSelectOpen**.
- **procedure formSelectClose** — заканчивает список значений, открытый **formSelectOpen**.

- **procedure formDo(cname in varchar2)** — создает кнопку типа **SUBMIT** текущей формы с именем **cname**.
- **procedure formDo** — создает кнопку типа **SUBMIT** текущей формы с именем **Submit**.
- **procedure formUndo(cname in varchar2)** — создает кнопку типа **RESET** текущей формы с именем **cname**.
- **procedure formUndo** — создает кнопку типа **RESET** текущей формы с именем **Reset**.
- **procedure formClose** — закрывает текущую форму.

Процедуры вывода

- **procedure print (cbuf in varchar2), procedure print (dbuf in date), procedure print (nbuf in number)** — выводят значение различных типов.

Синонимы для процедуры print — p

- **procedure p (cbuf in varchar2)**
- **procedure p (dbuf in date)**
- **procedure p (nbuf in number).**

Процедуры, выводящие постоянные значения

- **procedure line** — разделительная линия (тэг <**HR**>).
- **procedure para** — начало параграфа (тэг <**P**>).
- **procedure nl** — перевод строки (тэг <**BR**>).
- **procedure item** — элемент списка (тэг <**LI**>).
- **procedure ulistOpen** — начало ненумерованного списка (тэг <**UL**>).
- **procedure ulistClose** — окончание ненумерованного списка (тэг </**UL**>).
- **procedure olistOpen** — начало нумерованного списка (тэг <**OL**>).
- **procedure olistClose** — окончание нумерованного списка (тэг </**OL**>).
- **procedure dlistOpen** — начало списка определений (тэг <**DL**>).

- **procedure dlistClose** — окончание списка определений (тэг </**DL**>).
- **procedure dterm** — термин списка определений (тэг <**DT**>).
- **procedure ddef** — определение термина (тэг <**DD**>).
- **procedure preOpen** — начало форматированного текста.
- **procedure preClose** — окончание форматированного текста.

Часть 6. CGI, PHP, Perl, MySQL и CMS системы

Глава 1. CGI

Большое количество World Wide Web приложений основано на использовании внешних программ, управляемых Web сервером. Использование данных программ позволяет строить Web приложения с динамически обновляемой информацией, хранящейся в базах данных или генерирующейся в зависимости от бизнес-правил решаемых задач. Для связи между Web сервером и вызываемыми программами широко используется Common Gateway Interface (CGI), имеющий реализации как для Windows-ориентированных программ, так и для приложений, функционирующих в среде Unix.

Windows CGI требует, чтобы Web сервер декодировал данные из HTML форм, если они переданы при помощи POST метода запроса. Он не требует от сервера декодирования параметров, если они переданы в качестве строки запроса (**query string**), являющейся частью URL. Существует два способа, которыми данные из форм могут быть переданы серверу браузером:

URL-Encoded

Это наиболее используемый формат данных, передаваемых из форм. Содержимое полей формы выделяются из формы и передаются согласно спецификации HTML 1.0, а затем собираются в одну строку, где отделяются друг от друга символом амперсанда. Тип содержания сообщения устанавливается браузером в **application/ x — www — form — urlencoded**.

Multipart Form Data

Данный формат разработан для эффективной загрузки файлов на сервер с использованием форм. Содержимое полей формы передается как многостраничное MIME сообщение. Каждое поле содержится в одной странице. Тип содержания, устанавливается браузером в **multipart/**

form — data. «Грамотные» серверы должны уметь обрабатывать оба типа данных из форм.

Вызов CGI программ

Сервер использует функцию **CreateProcess()** для вызова CGI программ. Сервер синхронизируется с CGI программой, поскольку он должен определить момент завершения CGI программы. Это достигается использованием функции Win32 WaitForSingleObject(), ожидающей получения сигнала завершения CGI программы.

Командная строка

Сервер должен вызывать CGI программу выполняя функцию **CreateProcess()** с командной строкой следующего формата:

```
WinCGI-exe cgi-data-file
```

WinCGI-exe — полный путь к исполняемой CGI программе. Сервер не зависит от «текущего каталога» или переменной окружения PATH. Примите к сведению, что «исполняемая» не обязательно означает .EXE файл. Это может быть документ, ассоциирующийся с реально исполняемой программой, описанной в WIN.INI или System Registry.

cgi-data-file — полный путь к CGI файлу данных.

Метод вызова

Сервер использует **CreateProcess()** для запуска процесса, не имеющего главного окна. Вызванный процесс не будет отображаться каким либо образом на мониторе сервера.

Некоторые сервера поддерживают режим отладки CGI программ и скриптов, что позволяет серверу запускать CGI программу как обычный процесс с созданием главного окна и отображением информации на мониторе сервера. Данный способ весьма удобен на стадии отладки CGI программ.

CGI файл данных

Сервер передает данные CGI программам через Windows «private profile» afqk, в формате «параметр-значение» (windows INI файл). CGI программа может прочитать данный файл и получит все данные, передаваемые ей из формы, а также автоматически генерируемые браузером данные.

CGI файл данных состоит из следующих секций:

- CGI

- Accept
- System
- Extra Headers
- Form Literal
- Form External
- Form Huge
- Form File

Секция (CGI)

Данная секция содержит большинство специфических CGI параметров (тип доступа, тип запроса, дополнительные заголовки, определенные в других секциях и т.п.). Каждое значение представлено в виде символьной строки. Если значение является пустой строкой, значит данный параметр был опущен. Список параметров данной секции представлен ниже:

Request Protocol

Название и модификация информационного протокола, использованного для передачи данного запроса. Формат: протокол/модификация.

Пример:

```
HTTP/1.0
```

Request Method

Метод, который использовался для данного запроса. Для HTTP это **GET**, **HEAD**, **POST** и т.д.

Executable Path

Логический путь к исполняемой CGI программе, необходимый для ссылки CGI программе на саму себя.

Logical Path

Запрос также может указывать к ресурсам, необходимым для выполнения данного запроса. Данный параметр содержит путь в том виде, который был получен сервером без мэпирования его на физический путь на диске.

Physical Path

Если запрос содержит информацию о логическом пути, сервер преобразует его к физическому пути (например, к пути к файлу на диске) доступа согласно синтаксическим правилам операционной системы.

Query String

Информация, размещающаяся после ? в URL вызываемой CGI программы. Сервер оставляет эту информацию без изменений в том виде, в котором она была помещена в URL.

Request Range

Byte-range спецификация получаемая вместе с запросом (если есть). Смотри текущий Internet Draft (или RFC), описывающий расширение HTTP для получения более полной информации. Сервер должен поддерживать работу CGI программ в byte-ranging.

Referer

URL документа, содержащего ссылку на данную CGI программу. Надо заметить, что некоторые браузеры закрывают данную возможность и не дают ее использовать.

From

E-mail адрес пользователя браузера. Надо заметить, что данный параметр присутствует с спецификации HTTP, но не используется большинством браузером из соображений секретности.

User Agent

Строка, описывающая программное обеспечение браузера. Не генерируется большинством браузеров.

Content Type

Данный параметр содержит MIME-тип данных, посланных клиентом вместе с полями из формы, если эти данные были посланы. Формат:

```
type/subtype
```

Content Length

Для запросов, с которыми посланы дополнительные данные в это поле заносится длина посланных данных в байтах.

Content File

Для запросов, содержащих дополнительные данные, посланные пользователем, этот параметр содержит имя файла, в которое WEB-сер-

вер записывает эти данные. В дальнейшем, пользовательская программа может считать эти данные. Параметр содержит полный путь к файлу данных.

Server Software

Название и версия серверного программного обеспечения, обработавшего запрос и вызвавшего CGI-программу. Формат:

```
name/version
```

Server Name

Сетевое имя сервера или псевдоним, необходимый для ссылающихся на себя URL Этот параметр (в комбинации с параметром Server-Port) может быть использован для вычисления полного URL к серверу.

Server Port

Номер порта, по которому работает сервер.

Server Admin

E-mail адрес администратора сервера. Данный параметр необходим для генерации сообщений об ошибках и отправки данных сообщений администратору сервера или для генерации форм с URL «**mailto:**».

CGI Version

Версия спецификации CGI. Формат:

```
CGI/версия
```

Remote Host

Сетевое имя хоста клиента, если доступно. Данный параметр может быть использован для опознавание клиента.

Remote Address

Сетевой (IP) адрес клиента. Данный параметр может быть использован для проверки пользователя если отсутствует сетевое имя.

Authentication Method

Если используется защищенный вызов CGI программы, это протокол-зависимый метод аутентификации, используемый для аутентификации пользователя.

Authentication Realm

Если используется защищенный вызов CGI программы, это протокол-зависимый сервис, используемый для аутентификации пользователя. Список пользователей для полученного вида сервиса проверяется для аутентификации пользователя.

Authenticated Username

Если используется защищенный вызов CGI программы, это имя пользователя, которое клиент использует для аутентификации при доступе к CGI-программе.

Секция (Accept)

Данная секция содержит типы данных, посылаемых клиентом, найденные в заголовке запроса в виде

```
Accept: type/subtype {parameters}
```

Если данные параметры присутствуют (например, q=0.100), они передаются как значения параметра **Accept**. Для каждого типа передаваемых данных заводится свой параметр **Accept**.

Секция (System)

Данная секция содержит параметры, специфические для Windows реализации CGI:

GMT Offset

Количество секунд, которое необходимо добавить к времени по Гринвичу для вычисления локального времени клиента.

Debug Mode

Данный параметр имеет значение «Yes» если включен режим «CGI/script tracing» на сервере.

Output File

Полный путь к файлу, в который необходимо поместить данные, отсылаемые сервером клиенту после завершения работы программы.

Content File

Полный путь к файл у в котором содержится дополнительная информация, поступающая вместе с запросом.

Секция (Extra Headers)

Данная секция содержит «дополнительные» заголовки, которые включены в запрос в виде «параметр=значение». Сервер должен раскодировать как параметр, так и его значение прежде чем они будут помещены в файл данных CGI.

Секция (Form Literal)

Если запрос от клиента пришел в виде HTTP POST из HTML формы (с типом содержимого **application/x-www-form-urlencoded** или **multipart/form-data**), то сервер раскодирует данные из формы и поместит их в секцию [**Form Literal**].

Для URL-кодированных данных формы, строка передаваемых параметров выглядит как «параметр=значение&параметр= значение&...», где значения находятся в url-кодированном формате. Сервер разделяет «параметр=значение» по символу «&», затем разделяет собственно «параметр» и «значение», декодирует «значение» и помещает результат в виде «параметр=раскодированное_значение» в секцию [**Form Literal**].

Для многостраничных данных строка данных представляется в многостраничном MIME формате, где каждое поле представлено как отдельная часть (файл). Сервер декодирует имена и значение каждой части и размещает их в формате «параметр=значение» в секции [**Form Literal**].

Если форма содержит какие-либо элементы SELECT MULTIPLE, то будет создано несколько строк с вида «параметр=значение» с одинаковым именем «параметра». В этом случае генерирует нормальную строку «параметр=значение» для первого встречающегося элемента, а каждый следующий представляет в виде «параметр_X=значение», где «X» — увеличивающийся счетчик.

Секция (Form External)

Если размер декодированной строки превышает 254 символа или декодированная строка содержит управляющие символы, такие, как перевод строки, возврат каретки, двойные кавычки и т.д., то сервер помещает данное значение в отдельный временный файл, а в секцию [**Form External**] помещает строку в виде:

```
параметр=путь длина
```

где путь — это полный путь и имя временного файла, содержащего декодированное значение параметра, а длина — длина в байтах этого файла.

Секция (Form Huge)

Если общая длина строки с кодированными параметрами превышает 65,535 байт, то сервер не выполняет декодирование, а оставляет данный в **Content File**, а в секцию [**Form Huge**] помещает строки в виде:

```
параметр=смещение длина
```


Где смещение — это смещение от начала **Content File** по которому находится требуемый параметр, а длина — длина в байтах значения выбранного параметра. Вы можете использовать смещение для выполнения поиска начала значения выбранного вами параметра и использовать длину для чтения значения выбранного параметра. Не забывайте, что если параметр закодирован, то вам необходимо раскодировать его перед использованием.

Секция (Form File)

Если запрос пришел в виде **multipart/form-data**, то он может содержать один или несколько загруженных с клиента файлов. В этом случае каждый загруженный файл размещается в специальном временном файле, а в секции [**Form File**] строки имеют тот же формат, что и секции [**Form External**]. Каждая строка параметра в этом случае выглядит так:

```
параметр=[полный_путь_к_файлу] длина тип ссылка [имя_файла]
```

где полный_путь_к_файлу — это путь к временному файлу, содержащему загруженный файл, длина — длина в байтах загруженного файла, тип — тип MIME загруженного файла, ссылка — способ кодировки загруженного файла и имя_файла — исходное название загруженного файла. Использование квадратных скобок обязательно, поскольку имя файла и путь могут содержать символы пробела.

Пример декодированных значений формы

В данном примере форма содержит небольшое поле, SELECT MULTIPLE с 2-я небольшими секциями, поле длиной 300 символов, поле, содержащее специальные символы и поле длиной 230KB.

```
[Form Literal]
smallfield=123 Main St. #122
multiple=first selection
multiple_1=second selection
[Form External]
field300chars=C:\TEMP\HS19AF6C.000 300
fieldwithlinebreaks=C:\TEMP\HS19AF6C.001 43
[Form Huge]
field230K=C:\TEMP\HS19AF6C.002 276920
```

Обработка результата

CGI программа возвращает результат работы, отвечающий (явно или неявно) целям запроса. Сервер кодирует результат работы в соответствии со стандартом HTTP и использует HTTP для отправки результата клиенту.

Это означает, что сервер добавляет необходимый HTTP заголовки в сообщение, формируемое CGI программой.

Результат работы CGI программы состоит из двух частей: заголовка и тела сообщения. Заголовок состоит из одной или более строк текста, отделенных от тела пустой строкой. Тело сообщения содержит данные, представленные в MIME формате, указанном в заголовке.

Сервер не изменяет тело документа, что означает, что сервер передает сформированный CGI программой ответ «как он есть».

Специальные строки заголовка

Сервер распознает следующие строки заголовка в выходном потоке:

Content-Type:

Указывает на MIME тип тела сообщения. Значение этого параметра должно быть в формате **type/subtype**.

```
URI: <value> (value enclosed in angle brackets)
```

Данное значение указывает на полный URL или ссылку на локальный файл, сообщение из которого будет возвращено клиенту в теле сообщения. Если значение является локальным файлом, сервер отсылает его как результат запроса, как будто клиент воспользовался методом GET при генерации запроса. Если значение является полным URL, то сервер возвращает сообщение «401 redirect» для обеспечения прямой загрузки указанного объекта.

Location:

То же самое, что и URI, но данная форма сейчас не используется. Параметр value НЕ должен быть взят в угловые скобки.

Другие заголовки

Другие заголовки передаются клиенту в том виде, в котором они представлены.

Прямой возврат

Сервер позволяет конечному приложению осуществлять прямой возврат результата запроса клиенту. Это осуществляется посредством включение в заголовок возвращаемого сообщения его информационного протокола. Это позволяет CGI программам формировать непосредственный ответ клиенту с указанием HTTP заголовка без предварительной обработки его сервером.

Сервер анализирует результат запроса, помещаемый CGI программой в выходной файл (Output File), и, если первая строка «HTTP/1.0», он предполагает, что сообщение содержит полный HTTP ответ и отсылает его клиенту без упаковки.

Что такое CGI-скрипты?

Начнем с того, что ваш браузер (когда вы набрали **URL**) соединяется по протоколу **HTTP** с указанным сервером и просит у него нужный файл, примерно так:

```
GET /~paaa/cgi-bin/guestbbok.cgi HTTP/1.0
```

Вот это самое главное в запросе.

Дальше идет посылаемая браузером информация о себе и о том, что более подробно ему надо (например: **Accept:** */*).

Если запрошен простой файл: например **.html,** то если такой файл есть, сервер отошлет браузеру ответ:

```
HTTP/1.0 200 Okay
Content-Type: text/html
<HTML>
<BODY>
.......
</BODY></HTML>
```

В ответе, состоящем из заголовка и тела, в заголовке содержится код возврата и информация о типе содержимого.

Далее после пустой строки (она нужна чтобы отделить **заголовок** от **тела**) идет информация из самого документа, по заданному **URL:**

```
<HTML><BODY>...
```

Вот в принципе и весь **WWW** — ходишь от ссылки к ссылке...

А что если нужно внести в этот унылый процесс что-нибудь по-настоящему интерактивное, динамическое, прекрасное и великолепное? Что ж, есть ответ и на этот вопрос. Если в запрашиваемом **URL** указать специальную программу (**CGI**, программа **Common Gateway Inteface — Общего Шлюзового Интерфейса**) то, что эта программа выдаст, то и отправится браузеру...

Сервер запускает **.cgi**-программу, и она, например обработав данные формы, заносит вас куда-нибудь в свою базу данных, а вам сообщит, что вы большой молодец.

Что надо знать, чтобы писать CGI скрипты

Во-первых надо знать что такое Интернет и как он работает. Ну и чуть-чуть умения программировать (это самое главное).

Кроме того, если вы хотите изучать **CGI,** то вам нужен нормальный, полнофункциональный сервер. Если же вы сами являетесь системным администратором на своем сервере, то для вас, естественно нет проблем, ведь включить CGI для какой-нибудь директории — это просто подправить одну строчку в файле конфигурации сервера.

Простой скрипт

Сначала в своем домашнем каталоге создайте директорию **cgi-bin**:

```
cd public_html
mkdir cgi-bin
chmod 0777 cgi-bin
```

Последняя строчка будет очень важна.

Возьмите редактор и наберите:

```
#!/usr/bin/perl
#first.cgi
print "Content-Type: text/html\n\n";
print "<HTML><BODY>";
print "<H1>Hello you!!!</H1>";
print "</BODY></HTML>";
```

Сохраните его в директории **cgi-bin** под именем **first.cgi**. Ну как, сохранили?

А теперь сделайте его исполняемым (ведь это — программа):

```
chmod +x first.cgi
```

Итак, подходим к торжественному моменту — наберите в строке браузера

```
http://www.ваш.сервер.ru/~ваш_логин/cgi-bin/first.cgi
```

и посмотрите что будет. Будет одно из двух: либо скрипт заработает, и вы увидите сгенерированную им страничку (*поздравляю, в нашем полку прибыло!*), либо **Internal Server Error** — тогда не расстраивайтесь, вы что-то сделали не так.

Вам тогда пригодится пособие по ловле блох. Во-первых проверку синтаксиса можно осуществить следующим образом:

```
perl -c first.cgi
```

Perl вам сразу выдаст либо сообщения об ошибках (ну бывает, точку с запятой пропустили, скобочки или кавычки забыли закрыть...) это

по ходу дела поправимо. Более грубо с логической точки зрения — это пропустить вывод пустой строки, которая отделяет заголовок от тела:

```
print "Content-Type: text/html\n\n"; #Все Правильно
print "Content-Type: text/html\n"; #ОШИБКА!!!
```

Разберем скрипт.

Первая строка **#!/usr/bin/perl** просто указывает, где в системе расположен компилятор **Perl**.

Обычно он находится **/usr/bin/perl** или **/usr/local/bin/perl**, выяснить это можно одной из команд **which perl** или **whereis perl,** ну или (что очень долго) запустить полный поиск: **find / -name perl -print**.

Вторая строка — это просто комментарий — вы можете тыкать что угодно после знака **#**, однако обычно во второй строке пишут название скрипта, что очень удобно.

Затем идет **print «Content-Type: text/html\n\n»;** Это заголовок, указывающий тип содержимого.

Все, что скрипт печатает в свой стандартный вывод **STDOUT** идет на обработку к серверу. Пустая строка отделяет заголовок от тела, которое в нашем случае представляет собой:

```
<HTML><BODY>
<H1>Hello you!!!</H1>
</BODY></HTML>
```

Сервер обработает ответ скрипта и на базе него сформирует и пошлет браузеру ответ (сервер обычно не изменяет тела сообщения, он только дополняет заголовок нужными для работы протокола **HTTP** полями).

Ну вот, азы уже освоены, все не так трудно и удручающе, как могло показаться на первый раз.

Вы теперь можете сами потренироваться в написании таких вот простеньких скриптов, чтобы набить руку.

Переменные среды CGI

Предыдущий скрипт не содержал ничего особенно замечательного, так просто вываливал HTML текст, который благополучно отображался на экране браузера. Но по-настоящему мощь придает **CGI** возможность обработки параметров, которые переданы скрипту. Например вы можете набрать:

```
http://www.somehost.ru/somedir/cgi-bin/my_cgi.cgi?param=value
```

То есть вы хотите, чтобы скрипт **my_cgi.cgi** обработал для вас параметр **param** со значением **value**, или когда вы заполнили запрос, например в форме **yahoo** или **altavista**.

Ну это с точки зрения пользователя...

А на сервере при запуске **CGI**-скрипта сервер формирует среду окружения, в которой скрипт может найти всю доступную информацию о **HTTP**-соединении и о запросе.

Вот эти переменные:

REQUEST_METHOD

Это одно из самых главных полей, используемое для определения метода запроса **HTTP** Протокол **HTTP** использует методы **GET** и **POST** для запроса к серверу.

Они отличаются тем, что при методе **GET** запрос является как бы частью **URL,** т.е.

```
http://www..../myscript.cgi?request
```

А при методе **POST** данные передаются в теле **HTTP**-запроса (при **GET** тело запроса пусто), и следовательно для **CGI** тоже есть различие: при **GET** запрос идет в переменную **QUERY_STRING,** а при **POST** подается на **STDIN** скрипта.

Пример:

```
REQUEST_METHOD=GET
```

QUERY_STRING

Это строка запроса при методе **GET**.

Вам всем известно, что запрос из формы кодируется браузером, поскольку не все символы разрешены в **URL** некоторые имеют специальное назначение.

Теперь о методе **urlencode**. Неплохо бы чисто формально напомнить, что все пробелы заменяются в **URL** на знак «+», а все специальные и непечатные символы на последовательность **%hh**, где **hh** — шестнадцатеричный код символа, разделитель полей формы знак «**&**», так что при обработке форм надо произвести декодирование.

Пример:

```
QUERY_STRING= name=quake+doomer&age=20&hobby=games
```


CONTENT_LENGTH

Длина в байтах тела запроса. При методе запроса **POST** необходимо считать со стандартного входа **STDIN CONTENT_LENGTH** байт, а потом производить их обработку.

Обычно методом **POST** пользуются для передачи форм, содержащих потенциально большие области ввода текста **TEXTAREA**.

При этом методе нет никаких ограничений, а при методе **GET** существуют ограничения на длину **URL**.

Пример:

```
CONTENT_LENGTH=31
```

CONTENT_TYPE

Тип тела запроса. Для форм, кодированных выше указанным образом он

```
application/x-www-form-urlencoded
```

GATEWAY_INTERFACE

Версия протокола **CGI.**

Пример:

```
GATEWAY_INTERFACE=CGI/1.1
```

REMOTE_ADDR

IP-Адрес удаленного хоста, делающего данный запрос.

Пример:

```
REMOTE_ADDR=139.142.24.157
```

REMOTE_HOST

Если запрашивающий хост имеет **доменное** имя, то эта переменная содержит его, в противном случае — тот же самый **IP-адрес,** что и **REMOTE_ADDR.**

Пример:

```
REMOTE_HOST=idsoftware.com
```

SCRIPT_NAME

Имя скрипта, использованное в запросе. Для получения реального пути на сервере используйте **SCRIPT_FILENAME.**

Пример:

```
SCRIPT_NAME=/~paaa/guestbook.cgi
```

SCRIPT_FILENAME

Имя файла скрипта на сервере.

Пример:

```
SCRIPT_FILENAME=/home/p/paaa/public_html/cgi-bin/ guestbook.cgi
```

SERVER_NAME

Имя сервера. Чаще всего доменное, как **www.microsoft.com**, но в редких случаях за неимением такового может быть **IP-адресом,** как **157.151.74.254.**

Пример:

```
SERVER_NAME=www.uic.nnov.ru
```

SERVER_PORT

TCP-Порт сервера. Используется для соединения. По умолчанию **HTTP-порт 80**, хотя может быть и другим.

Пример:

```
SERVER_PORT=80
```

SERVER_PROTOCOL

Версия протокола сервера.

Пример:

```
SERVER_PROTOCOL=HTTP/1.1
```

SERVER_SOFTWARE

Программное обеспечение сервера.

Пример:

```
Apache/1.0
```

AUTH_TYPE, REMOTE_USER

Эти переменные определены в том случае, когда запрошенный ресурс требует аутентификации пользователя.

Переменные заголовка HTTP-запроса

За исключением тех строк из заголовка **HTTP-запроса,** которые были включены в другие переменные, сервер приделывает строкам префикс **HTTP_** и заменяет знаки «-» на «_»:

```
HTTP_ACCEPT
```

Давая запрос на сервер браузер обычно рассчитывает получить информацию определенного формата, и для этого он в заголовке запроса указывает поле **Accept.**

Отсюда скрипту поступает список тех **MIME**, которые браузер готов принять в качестве ответа от сервера.

Пример:

```
HTTP_ACCEPT=text/html,text/plain,image/gif
```

HTTP_USER_AGENT

Браузер обычно посылает на сервер и информацию о себе, чтобы базируясь на знании особенностей и недостатков конкретных браузеров **CGI**-скрипт мог выдать информацию с учетом этого.

Например, разные браузеры могут поддерживать или не поддерживать какие-то **HTML** тэги.

Пример:

```
HTTP_USER_AGENT=Mozila/2.01 Gold(Win95;I)
```

HTTP_HOST

Имя хоста к которому обращается браузер. Так как физически на одном сервере может находиться сразу много серверов (Виртуальные Хосты), то должен быть способ сообщить серверу к какому именно идет обращение.

Скрипт же может в зависимости от этой переменной производить различные действия, если он используется на сайтах сразу нескольких виртуальных хостов.

Пример:

```
HTTP_HOST=www.nnov.city.ru
```

Начнем применять на практике усвоенные уроки.

```
#!/usr/bin/perl
#vars.cgi
sub urldecode{  #очень полезная функция декодирования
local($val)=@_; #запроса,будет почти в каждой вашей CGI-программе
 $val=~s/\+/ /g;
 $val=~s/%([0-9A-H]{2})/pack('C',hex($1))/ge;
 return $val;
 }
print "Content-Type: text/html\n\n";
print "<HTML><HEAD><TITLE>CGI-Variables</TITLE></HEAD>\n";
print "<BODY>\n";
print "Enter here something:<ISINDEX><BR>\n";
print "Your request is:$ENV{'REQUEST_STRING'}<BR>\n";
print "Decoded request is:urldecode($ENV{'REQUEST_STRING'})
<BR>\n";
```

```
print "<HR>\n";
print "Variables:<BR>\n";
print "<I><B>REQUEST_METHOD</B></I>=$ENV{'REQUEST_METHOD'}
<BR>\n";
print "<I><B>QUERY_STRING</B></I>=$ENV{'QUERY_STRING'} <BR>\n";
print "<I><B>CONTENT_LENGTH</B></I>=$ENV{'CONTENT_LENGTH'}<BR>\n";
print "<I><B>CONTENT_TYPE</B></I>=$ENV{'CONTENT_TYPE'}<BR>\n";
print "<I><B>GATEWAY_INTERFACE</B></I>=$ENV{'GATEWAY_INTERFACE'}
<BR>\n";
print "<I><B>REMOTE_ADDR</B></I>=$ENV{'REMOTE_ADDR'}<BR>\n";
print "<I><B>REMOTE_HOST</B></I>=$ENV{'REMOTE_HOST'}<BR>\n";
print "<I><B>SCRIPT_NAME</B></I>=$ENV{'SCRIPT_NAME'}<BR>\n";
print "<I><B>SCRIPT_FILENAME</B></I>=$ENV{'SCRIPT_FILENAME'}
<BR>\n";
print "<I><B>SERVER_NAME</B></I>=$ENV{'SERVER_NAME'}<BR>\n";
print "<I><B>SERVER_PORT</B></I>=$ENV{'SERVER_PORT'}<BR>\n";
print
"<I><B>SERVER_PROTOCOL</B></I>=$ENV{'SERVER_PROTOCOL'}<BR>\n";
print
"<I><B>SERVER_SOFTWARE</B></I>=$ENV{'SERVER_SOFTWARE'}<BR>\n";
print "<I><B>HTTP_ACCEPT</B></I>=$ENV{'HTTP_ACCEPT'}<BR>\n";
print
"<I><B>HTTP_USER_AGENT</B></I>=$ENV{'HTTP_USER_AGENT'}<BR>\n";
print "<I><B>HTTP_HOST</B></I>=$ENV{'HTTP_HOST'}<BR>\n";
print "<HR>\n";
print "All enviroment:<BR>\n";
foreach $env_var (keys %ENV){
print "<I>$env_var=$ENV{$env_var}</I><BR>\n";
 }
print "</BODY></HTML>\n";
```

Так как все ваши **.cgi-файлы** должны быть исполняемыми, то чтобы облегчить себе жизнь заведите в директории **cgi-bin** командный файл **mkcgi**, содержащий:

```
#!/bin/sh
#mkcgi
chmod +x *.cgi
```

Сделайте его в свою очередь исполняемым **chmod +x mkcgi** — он сильно упростит вам жизнь.

Ну, а теперь запускайте скрипт...

Изучив информацию, выдаваемую данным скриптом, вы сможете лучше ориентироваться в переменных окружения **CGI**.

Заголовки запросов и ответов

Даже если вы и знаете кое-что о **HTTP,** все равно не лишнее будет вспомнить о том, как это все работает, тем более на эту информацию придется ориентироваться при написании **CGI** скриптов.

Этапы соединения

Первый этап — это когда **HTTP** — клиент (браузер) соединяется с сервером. Для этого он использует протокол **TCP/IP.** Соединение происходит к известному клиенту **TCP**-порту (**80** — номер порта **HTTP** (другие сервисы сидят на других портах, например **FTP** и **SMTP** на **21** и **25**)).

На втором этапе идет запрос клиента: клиент передает заголовок запроса и возможно (в зависимости от метода) тело сообщения запроса.

В заголовке обязательно указывается метод **URI** и версия **HTTP** и может быть еще несколько необязательных полей.

Третий этап — ответ сервера, который опять таки состоит из заголовка, в котором сервер указывает версию **HTTP** и код статуса, который может говорить об успешном или неуспешном результате и его причинах. Далее идет тело ответа.

На четвертым этапе происходит разрыв **TCP/IP**-соединения.

HTTP-запрос

Запрос состоит из **Строки запроса** (она обязательна) и **остальных полей**.

Синтаксис строки:

```
МЕТОД <SP> URI <SP> HTTP/версия <CRLF>
```

Где **<SP>** — пробел, **<CRLF>** — переход на новую строку.

Методы HTTP

GET

Самый часто применяемый метод, в протоколе **HTTP/0.9** был единственным методом и применялся для извлечения информации по заданному **URI.** Может быть условным, если в заголовке указано поле **If-Modified-Since:**

HEAD

Почти идентичен **GET,** но отличается тем, что сервер не возвращает тело объекта, а только его заголовок (метаинформацию).

Программы могут применять его для проверки гиперссылок на правильность, доступность и изменения.

POST

Передает данные для обработки их программой, указанной в **URI.** Здесь обязательно указывается поле **Content-Length:** Существуют и другие, реже применяемые методы, например **PUT** — для сохранения передаваемых данных в указанном **URI** и **DELETE** для удаления ресурса.

Поля заголовка запроса

После строки запроса идут **поля заголовка запроса**. Поля общего (**general-header**) заголовка (он общий как для запросов, так и для ответов):

Date:

Указывает дату запроса, например:

```
Date: Sun, 20 Nov 1994 08:12:31 GMT
```

MIME-version:

Указывает версию **MIME** (по умолчанию **1.0**):

```
MIME-version: 1.0
```

Pragma:

Содержит указания для таких промежуточных агентов, как прокси и шлюзы:

```
Pragma: no-cache
```

Поля, относящиеся к запросу **(Request-Header):**

Authorization:

Содержит информацию аутентификации:

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

From:

Браузер может посылать адрес пользователя серверу:

```
From: quake@doom.ru
```

If-Modified-Since:

Используется при методе **GET**, ресурс возвращается, если он был изменен с указанного момента, может использоваться при кэшировании.

```
If-Modified-Since:Mon 15 Jul 1997 00:15:24 GMT
```


Referer:

Содержит **URL** предшествующего ресурса.

```
Referer: http://www.uic.nnov.ru/~paaa/index.html
```

User-Agent:

Программное обеспечение клиента.

```
User-Agent: Mozilla/3.0
```

Заголовок информации сообщения (**Entity-Header**) применяется как в запросах, так и в ответах (при этом некоторые поля только в ответах).

Allow: (в ответе сервера)

Список методов, поддерживаемых ресурсом.

```
Allow: GET, HEAD
```

Content-Encoding:

Идентифицирует метод кодировки, которым был закодирован ресурс.

```
Content-Encoding: x-gzip
```

Content-Length:

Длина тела сообщения.

```
Content-Length: 102
```

Content-Type:

Содержит тип ресурса (**MIME**), для текстовых и кодировку символов (необязательно).

```
Content-Type: text/html; charset=windows-1251
```

Expires: (в ответе сервера)

Дата окончания действия ресурса применяется в кэшировании для запрета кэширования устаревших ресурсов (в ответе).

```
Expires: Tue, 24 Sep 1998 23:00:15 GMT
```

Last-Modified: (в ответе сервера)

Время последнего обновления ресурса.

```
Last-Modified: Tue, 23 sep 1998 13:48:40 GMT
```

Другие поля:

Поля Accept:

Указывают серверу выдавать только указанные форматы данных, которые клиент может распознать.

```
Accept: text/html
Accept: text/plain
Accept: image/gif
```

Поле Host:

Служит для того, чтобы указать, к какому хосту идет обращение. Данное поле не входит в число обязательных. Однако оно является необходимым в тех случаях, когда одному физическому серверу соответствует несколько виртуальных хостов. Тогда в этом поле указывается какой из виртуальных хостов имеется в виду.

```
Host: www.nnov.city.ru
```

Примеры запросов

Простейший запрос:

```
GET /index.html HTTP/1.0
```

Посложнее:

```
GET /somedir/somedoc.html HTTP/1.0
User-Agent: Mozilla/2.0
Accept: text/html
Accept: text/plain
Accept: image/gif
```

Передача данных CGI-скрипту через метод GET:

```
GET /~paaa/cgi-bin/test.cgi?
name=Dmitry&organization=%D3%ED%E8%E2%E5%FO%F1%E8%F2%E5%F2+%CD%E8%
E6%ED%E5%E3%EE+%CD%EE%E2%E3%EE%FO%EE%E4%EO &Name=&email=&comment=
HTTP/1.0
User-Agent: Mozila/2.0
Accept: text/html
Accept: image/gif
```

Используя метод **POST**, данные передаются в теле сообщения запроса:

```
POST /~paaa/cgi-bin/test.cgi HTTP/1.0
User-Agent: Mozila/2.0
Accept: text/html
Accept: image/gif
Content-Type: application/x-www-form-urlencoded
Content-Length: 131
name=Lesha
&organization=%D3%ED%E8%E2%E5%FO%F1%E8%F2%E5%F2+%CD%E8
%E6%ED%E5%E3%EE+%CD%EE%E2%E3%EE%FO%EE%E4%EO&Name=
&email=
&comment=
```


Ответ HTTP-сервера

Ответ идет от сервера. Состоит он из строки состояния и поля ответа. Общий заголовок (**General-Header**), заголовок тела сообщения (**Entity-Header**), которые уже описаны при обсуждении запроса и заголовок ответа (**Response-Header**).

Строка состояния имеет следующий формат:

```
HTTP/version <SP> Status-Code <SP> Status-Phrase
```

Где **HTTP/version** — версия, **Status-Code** — 3-х значный код, и **Status-Phrase** — текстовая фраза, поясняющая код.

Например:

```
HTTP/1.0 200 Ok
```

200 — код, означающий успешную обработку запроса, что и поясняет **«Ok»**

Заголовок ответа состоит из полей:

Location:

Содержит **URI** ресурса, может быть использован для переключения клиента в другое место, если например ресурс был перемещен в другое место или на другой сервер.

```
Location: http://www.uic.nnov.ru/newlocation/index.html
```

Server:

Информация о программном обеспечении сервера.

```
Server: Apache/1.1
```

WWW-Autenticate:

Параметры аутентификации.

```
WWW-Autenticate: Basic realm="doomsday"
```

Коды ответов HTTP

Более подробное описание всех кодов можно найти в **RFC-1945.**

Несколько примеров:

```
HTTP/1.0 200 Ok
Date: Wed, 25 Sep 1998 23:00:00 GMT
Server: Apache/1.1
MIME-version: 1.0
Last-Modified: Mon 15 Nov 1996 15:20:12 GMT
Content-Type: text/html
Content-Length: 2000
```

```
<HTML><HEAD><TITLE>Hello</TITLE></HEAD>
<BODY bgcolor="green" text="yellow">
......
</HTML>
```

А вот такое сервер выдаст в неудачном случае:

```
HTTP/1.0 404 Not Found
```

CGI-заголовок

В том случае, когда запрашиваемый **URI** есть **CGI**-скрипт, сервер, базируясь на данных запроса создает среду **переменных CGI** и передает управление скрипту. Скрипт должен выдать **CGI-заголовок**, после которого и идет тело ответа, сгенерированное скриптом.

Заголовок (**CGI-Header**) состоит из полей:

Content-Type:

Должно обязательно присутствовать, если есть тело.

```
Content-Type: text/html
```

Location:

Содержит **URL** ресурса, на который скрипт перенаправляет запрос. Как правило, если присутствует это поле, больше ничего не указывается.

```
Location: http://www.idsoftware.com/index.html
```

Status:

Позволяет **CGI** скрипту вернуть статус обработки, если это поле не задано, то сервер подразумевает **«200 Ok»**

```
Status: 404 Not found
```

На базе этой информации сервер и формирует окончательный заголовок, который и передается клиенту.

Примеры:

Обычно такое выдает скрипт:

```
Content-Type: text/html
<HTML><HEAD>.......
```

Но иногда такое (когда он служит для перенаправления):

```
Location: http://www.mustdie.ru/
```

А вот пример возврата статуса:

```
Content-Type: image/gif
Status: 190 Its seems great like a playing doom! WOW!
GIF89a........
```


Nph-скрипты

Иногда возникает необходимость чтобы **CGI**-скрипт сам отвечал напрямую клиенту, минуя разбор заголовка. Это во-первых уменьшает нагрузку на сервер и во-вторых, что самое главное, такой прямой ответ клиенту позволяет скрипту полностью контролировать транзакцию.

Для этого существуют **nph-**скрипты **(Not Parse Header)**, имя скрипта должно начинаться с префикса **«nph-»**.

Например: **«nph-animate.cgi»**.

Такие скрипты сами формируют **HTTP**-ответ клиенту, что полезно при анимации:

```
#!/usr/bin/perl
#nph-animate.cgi
$times = 20;
#Заготовьте несколько небольших gif-файлов для этой программы
@files = qw(img0.gif img1.gif img2.gif img3.gif);
select (STDOUT);
$|=1; #autoflush mode on
#Generate header
print "HTTP/1.0 200 Okay\n";
print "Content-Type: multipart/x-mixed-replace; boundary=myboundary\n\n";
print "--myboundary\n";
for ($num=1;$num<=$times;$num++) {
 foreach $file (@files) {
  print "Content-Type: image/gif\n\n";
  open(PIC,"$file");
  print <PIC>;
  close(PIC);
  print "\n--myboundary\n";
  sleep(3);
  }
 }
print "\n--myboundary--\n";
```

Этот пример выдаст вам анимацию, составленную из нескольких **.gif**-файлов.

Права доступа

Сидя в пределах своей домашней директории и качая с Интернета всякую фигню, вы возможно и не задавались некоторыми вопросами, а зря. Ведь немного надо, чтобы попортить нервы начинающему **CGI**-про-

граммисту. Одна из таких вещей — это права доступа. Начнем с того, что в системе Unix каждый пользователь имеет свой идентификатор — число, уникально идентифицирующее его в этой системе (например, логин рааа, а ему соответствует число 1818). Это число внутреннее для операционной системы, для пользования оно представлено как логин, который и соответствует пользователю.

Только не надо думать о пользователе как о конкретном человеке, сидящим за клавиатурой, пользователем может быть и какой-нибудь процесс. Важно отметить что пользователь — это определенная область прав доступа, которая ему соответствует.

Вы, например, не можете обычно писать и удалять файлы из каталога другого пользователя. Это и дает возможность стабильной работы всей системы.

Итак, есть идентификатор пользователя. Также имеется идентификатор группы. Группа служит для выделения пользователей по группам. Например, у пользователей группы users (обычные пользователи) не такие права, как у группы wheels (административная группа). Каждый процесс, который вами запущен (будь то Netscape, терминал, или текстовый редактор), получают ваши идентификаторы пользователя и группы. Таким образом они исполняются от вашего имени.

Теперь рассмотрим повнимательнее файловую систему.

В Unix с файлом связано много характеристик. Во-первых — в системе нет «ничьих» файлов, все файлы имеют владельца-пользователя и владельца-группу.

Любой файл, который вы создаете автоматически получает ваш идентификатор. Поэтому система очень легко отслеживает, чьи это файлы и каталоги.

Следующее новшество, по сравнению с DOS — это права доступа к файлу. Их может сменить только тот пользователь, которому принадлежит файл, или супервизор (это в отличии от DOS, где каждая дрянь типа вируса может снять атрибут readonly читать и писать все файлы).

Права доступа задаются обычно числом в восьмеричной записи и разбиты на 3 части по 3 бита. Каждая часть задает права доступа для конкретной группы.

- 1-я — права доступа для пользователя, которому принадлежит файл;
- 2-я — для группы, которой принадлежит файл;

- 3-я — для всех остальных.

В каждой такой категории выделяются 3 права:

- право на чтение,
- право на запись,
- право на исполнение

Все права и атрибуты очень наглядно показываются командой ls с ключом l.

Так как исполнять каталоги бессмысленно, то право на исполнение для них означает право обращаться к файлам из этого каталога.

Изменяются права командой chmod, ее синтаксис такой:

```
chmod [u|g|o]{+|-}{r|w|x} file
chmod number file
```

Где:

- **u** — user
- **g** — group
- **o** — other
- **r** — read
- **w** — write
- **x** — execute
- «**-**» — удалить
- «**+**» — установить.

Примеры:

```
chmod +r file.txt #разрешает всем право на чтения файла
chmod u+w file.txt #устанавливает для владельца файла право на
запись в него
chmod +x gbook.cgi #право на исполнение для всех, как для
пользователя, группы и для других
chmod 0777 cgi-bin #Разрешает самые широкие права доступа для
cgi-bin
```

При открытии файла программой, операционная система сравнивает идентификатор пользователя с идентификатором пользователя владельца файла, если они равны, то действуют права пользователя, если не равны, то сравниваются идентификаторы группы, если и они не равны, то действуют права доступа для остальных остальных.

В том случае, если у процесса нет достаточных прав, система возвращает ошибку. Следует заметить, что для супервизора **root** права доступа не проверяются.

Можно выполнить скрипт, только если есть права на его исполнение. Вот почему следует давать chmod +x *.cgi, иначе ваши скрипты станут просто недоступными.

Но и это еще не все.

Ваш скрипт может обращаться к вашим файлам (например ведет базу данных гостевой книги). Все выглядит нормально, но только ничего не работает, файл в который вы намеревались писать, не открывается. Знакомая проблема?

Так вот, чтобы вы не мучились в догадках ваш скрипт не может получить доступ к вашим файлам, потому что он выполняется не вами (не с вашим идентификатором), а от имени nobody (непривилегированный пользователь).

Эта мера предосторожности направлена на то, чтобы скрипты, взбесившись из-за неправильно переданных параметров (или вообще от глюков), не могли ничего повредить ценного и важного на сервере.

Поэтому тем файлам, к которым скрипт по смыслу должен обращаться, нужно присвоить самые широкие права доступа 0777.

Например в случае гостевой книги:

```
chmod 0777 guestbook.dat
```

Если также важно чтобы скрипты могли заводить новые файлы в **cgi-bin,** то нужно дать также права на это:

```
chmod 0777 cgi-bin
```

Если вы видите, что ваш скрипт не может обратиться к какому-то файлу, то это в 99% случаях происходит из-за вашей забывчивости!

На самый крайний случай воспользуйтесь **setuid**-скриптами (к этому делу, если вы на это решились, отнеситесь очень серьезно, так как целые тома по безопасности в **Unix** посвящены именно **setuid**-программам). Для общего развития имейте в виду следующую информацию: «Кроме указания прав доступа существуют специальные биты у файла. Это биты установки пользователя и группы. Когда процесс выполняется (простой процесс), то его реальный и эффективный идентификаторы пользователей совпадают, идентификаторы групп тоже».

На самом деле значение имеют как раз эффективные значения пользователя и группы, они участвуют в сравнении прав доступа.

Нельзя ли их как-то изменить, когда уж совсем нужда заставит? Можно!

На этот вопрос дают ответ программы с установленным битом пользователя. Когда система запускает такую программу, она присваивает новому процессу не идентификатор того пользователя, что запустил ее, а идентификатор пользователя-владельца исполняемого файла.

Самый классический пример **setuid**-программ — это программа **passwd**, предназначенная для смены пароля пользователя.

Такие данные, как пароль и прочие характеристики пользователей, хранятся в специальном файле, который имеет огромное значение при входе в систему.

Так как это системный файл, то открыть к нему доступ на запись всем — значит подвергнуть всю систему риску, ведь любое неправильное изменение его повлечет катастрофические последствия (в конце концов бывает просто хулиганство).

Поэтому доступ к этому файлу закрыт для всех пользователей. А что, если надо сменить пароль?

Запускаем программу **passwd**!

Если глянуть на ее атрибуты, то видно, что она принадлежит **root-супервизору**, и еще имеет установленный бит **setuid**. Так корректно обходится эта проблема.

Если вы все же решили попытаться, то знайте, что сделать программу **setuid** можно командой:

```
chmod +s myprogramm
```

Примерчик напоследок: эта программа выдает содержимое вашей директории **public_html** в том случае, если она доступна для чтения и для каждого файла указывает: можно ли его читать, писать и исполнять. Попробуйте сделать ее **setuid** и посмотрите как изменится результат.

```
#!/usr/bin/perl
#listmydir.cgi
print "Content-Type: text/html\n\n";
if(!(-r '..')){
 print ".. is not allowed for reading ;)))))\n";
 }
else{
 @list=glob('../*');
 foreach(@list){
 print "<A href=\"$_\">$_</A>";
```

```
print " readable" if -r;
print " writable" if -w;
print " executable" if -x;
print "<BR>\n";
}
}
```

Генерация ответа

Стандарт MIME

Стандарт **MIME** появился в электронной почте (**e-mail**) потому, что остро стала проблема пересылки по **e-mail** различных данных в различных форматах.

Так как **HTTP** тоже работает с различными типами данных, то тоже использует **MIME** для своих нужд.

Типы **MIME** состоят из Типа и подтипа.

Например:

```
text/plain
```

где **text** — указывает на наличие текстового содержимого, а **plain** — уточняет его, как простой текст.

Приведенный ниже список (он далеко не полон, типов **MIME** огромное количество), описывает некоторые часто встречающиеся типы:

- **text/html**
- **text/plain**
- **text/richtext**
- **image/gif**
- **image/jpeg**
- **image/tiff**
- **audio/basic**
- **audio/32kadpcm**
- **audio/**
- **video/mpeg**
- **video/quicktime**
- **multipart/mixed**

- **multipart/alternate**
- **multipart/**
- **application/octet-stream**
- **application/msword**
- **application/postscript**
- **message/digest**

Информация о **MIME** возможно пригодится вам и в том случае, если вы собираетесь работать из ваших скриптов с электронной почтой, но и для **WWW** она не повредит. Особенно знание **Content-Type:**

Content-Type:

Состоит из типа и подтипа. Типы могут быть как стандартные так и экспериментальные, начинающиеся с префикса «**x-**»:

text — текстовые данные. Первый подтип, который включен сюда — это plain, что значит простой текст. Сюда же включен самый ходовой формат html.

У типа **text** как и у многих типов могут быть параметры, главным из них является **charset.** Он как раз и указывает на раскладку символов, которая применена в тексте, так что если вы хотите указать браузеру какую раскладку применять, то просто укажите **charset:**

```
Content-Type: text/plain; charset=us-ascii
Content-Type: text/html; charset=iso-8859-1
Content-Type: text/html; charset=koi8-r
```

multipart — данные, которые могут состоять из нескольких частей различных типов данных. Поэтому параметром **multipart** служит **boundary**, позволяющий указать разделитель.

Каждый фрагмент в многочастевом сообщении имеет свой **Content-Type:** (он может быть также **multipart**, т.е. допускаются вложенные **multipart**, главное чтобы **boundary** были разными).

В электронной почте применяется больше **multipart/mixed** (основной подтип) и **multipart/alternative** (он отличается тем, что показывается одна из альтернатив, например сообщение шлется в простом и **HTML** форматах, и почтовая программа показывает часть, которую она способна отобразить).

В **WWW**-программировании распространен **x-mixed-replace**, который означает, что следующая часть должна заменить предыдущую после подгрузки, что применяется для анимации.

Теперь о разделителе. Его надо выбирать так, чтобы он не встретился где-то в данных (т.е. что-то вроде «**diUr344rnmvforgefvrg923rghyj2**»).

Когда вы задали разделитель, например **boundary=«boundary»,** то когда закончилась одна часть, вы должны выдать строку **— boundary**, последняя часть **— boundary**, причем эти разделители должны быть на отдельной строке, а не сливаться с текстом.

Пример:

```
MIME-Version: 1.0
Content-Type: multipart/alternative;
boundary="w23renff491nc4rth56u34-9449"
-w23renff491nc4rth56u34-9449
Content-Type: text/plain; charset="koi8-r"
 Hello,World!!
 -w23renff491nc4rth56u34-9449
 Content-Type: text/html; charset="us-ascii"
 <H1>Hello,Word!!</H1>
 <HR>
 <FONT size=+1 color=red>Hello people!</FONT>
 -w23renff491nc4rth56u34-9449-
```

message — представляет инкапсулированное почтовое сообщение. Используется в e-mail, а не в WWW.

image — некоторое графическое изображение (чаще всего image/gif или image/jpeg).

audio — аудиоданные.

video — видеоданные.

application — бинарные данные какого-нибудь приложения. В том случае, если данное приложение может быть запущено, браузер запускает его.

Например при поступлении данных **application/msword** браузер спросит: нужно ли запустить **Word** для просмотра документа. При отсутствии нужного приложения браузер спросит: в каком файле сохранить данные.

Подтип **octet-stream** как раз и означает поток байт информации, который и используется по умолчанию (к сожалению не все так гладко, известен глюк в **Netscape Navigator**'е, который вместо того, чтобы сохранить **application/octet-stream,** пытается его показать как **text/plain,** и тогда если это сгенерировано из **CGI** — ни к чему хорошему не приводит)

Что касается **application**, то вы можете тут смело извращаться, используя **x-**типы данных. Например:

```
application/x-fuck-to-netscape-navigator
```

Часто используемый параметр **name** позволяет указать имя файла.

Например:

```
Content-Type: application/msword; name="readme.doc"
```

Что полезно при получении файлов через **HTTP**, причем этот параметр может применяться и для других типов, таких как **image** или **audio**.

Например:

```
Content-Type: image/gif; name="myfoto.gif"
```

Content-Transfer-Encoding:

Применяется больше в системе электронной почты и обозначает метод кодирования, которым были закодированы данные при передаче сообщения.

Например:

```
7bit 8bit quoted-printable base64 binary x-типы
```

MIME-Version:

Указывает версию **MIME**.

Браузеры

Теперь поговорим о разных браузерах. Вы знаете, что браузеры бывают разные: разных версий, на разных платформах, поддерживают разные тэги и глюки у них тоже разные.

Это могло попортить много нервов **WEB**-дизайнерам и конечно же — **CGI**-программистам.

Профессионально написанный сайт от просто хорошего отличается тем, что хорошо выглядит не только на экране того браузера, которым пользуется сам его автор, а и на других тоже.

Если вы используете **JavaScript** для своих страничек, то вы уже наверно использовали (или хотя бы вам в голову приходила мысль использовать) свойства **navigator.AppName navigator.AppCodeName navigator.app Version navigator.userAgent:**

```
<SCRIPT language="JavaScript">
 if(navigator.AppName=="Netscape"){
 /*Сделать что-нибудь специфичное для Netscape*/
 }
```

```
 else if(navigator.AppName=="Microsoft Internet Explorer"){
 /*Сделать что-нибудь специфичное для Explorer*/
 }
 else{
 /*Не делаем специфичных вещей-хрен его знает с каким браузером
мы имеем дело*/
 }
</SCRIPT>
```

или

```
<SCRIPT language="JavaScript">
if((navigator.AppName=="Netscape")&&(parseFloat
(navigator.appVersion)<3.0)){
 document.writeln("Пользуетесь слишком старым браузером");
 }
</SCRIPT>
```

Ну не волнуйтесь вы так!

CGI-программисты не в самых худших условиях на этот счет. Вспомните о том, что браузер сам при запросе посылает вам данные о себе и о своей версии. И делает он это для того, чтобы эту информацию можно было учесть.

В запросе он указывает **User-Agent:,** которое и попадает на сервере в переменную среду **HTTP_USER_AGENT**, которую и можно использовать.

Например, если в ней содержится **Mozilla/3.01Gold (Win95;I),** то значит вы имеете дело с **Netscape** (**Mozilla** — кодовое название **Netscape Navigator**'а), версии **3.01Gold**.

Далее после имени и версии может следовать необязательная информация, как в приведенном примере о платформе **Win95** и о том, является ли версия **U** — для США (**USA**) или **I** — международной (**International**).

Такая информация необязательна (то есть, если браузер послал информацию **User-Agent:,** то гарантировано рассчитывать вы можете только на **Название/Версия**).

Допустим, ваш скрипт генерирует какие-то **тэги**, которые слишком старые браузеры не поддерживают, причем без них не обойдешься, они составляют всю «изюминку» сайта.

```
#!/usr/bin/perl
#oldbrowser.cgi
print "Content-Type: text/html\n\n";
if(defined ($ENV{'HTTP_USER_AGENT'})){
```



```
 $browser=$ENV{'HTTP_USER_AGENT'};
 ($vers)=($browser=~/\/(\d+\.\d+)/);
 if(($browser=~/mozilla/i)&&($vers<=2.0)){
 print "<HTML><HEAD><TITLE>Too old!</TITLE></HEAD>";
 print "<BODY bgcolor=\"red\" text=\"black\">";
 print "<CENTER><H1>Ваш Netscape Слишком старый для этого сайта";
 print "(старость не радость;))</H1></CENTER>";
 print "</BODY></HTML>";
 exit;
 }
 if(($browser=~/msie/i)&&($vers<=3.0)){
 print "<HTML><HEAD><TITLE>Too old!</TITLE></HEAD>";
 print "<BODY bgcolor=\"red\" text=\"black\">";
 print "<CENTER><H1>Ваш Explorer устарел";
 print "(а не пора ли сделать апгрейд хотя бы до 4.0
версии)</H1></CENTER>";
 print "</BODY></HTML>";
 exit;
 }
 }
print "<HTML><HEAD>.........";
```

Ну, уже почувствовали, насколько это здорово?!

А вот еще примерчик. Это из разряда того, что тэги бывают разные. Например, в **Explorer** есть тэг **BGSOUND,** предназначенный для проигрывания музыки на страничке (в **Netscape** этого тэга нет, поэтому для втыкания музыки приходится использовать подключаемые модули **plugin**).

Только не забывайте, что если вы не получили информацию о клиенте (так может быть, если например ваш скрипт вызвала какая-нибудь поисковая машина), то в этом случае не надо делать никаких предположений, а просто пусть ваш скрипт продолжает делать то, что должен был делать.

Этот пример позволит выбирать из списка файлов и загружать то, что пользователь хочет.

```
#!/usr/bin/perl
#download.cgi
sub urldecode{
 local($val)=@_;
 $val=~s/\+/ /g;
 $val=~s/%([0-9a-hA-H]{2})/pack('C',hex($1))/ge;
 return $val;
```

```
 }
@Filelist=qw(index.html readme.txt jmj00.mid gunshot.wav
foto.gif);
@Sel_list=();
if($ENV{'REQUEST_METHOD'} eq 'GET'){$query=$ENV{'QUERY_STRING'};}
elsif($ENV{'REQUEST_METHOD'} eq
'POST'){sysread(STDIN,$query,$ENV{'CONTENT_LENGTH'});}
if($query eq ''){
 #Если никаких данных не подано на обработку,то сгенерируем
форму, которую и предложим заполнить пользователю.
 print "Content-Type: text/html\n\n";
 print "<HTML><HEAD><TITLE>File Downloading</TITLE></HEAD>";
 print "<BODY bgcolor=\"white\">";
 print "Выберите файлы которые вы хотите загрузить:<BR>";
 print "<FORM METHOD=\"POST\">";
 print "<SELECT NAME=\"file\" size=4 multiple>";
 foreach(@Filelist){
 print "<OPTION value=\"$_\">$_";
 }
 print "</SELECT><BR>";
 print "<INPUT TYPE=\"Submit\" value=\"Download!\">";
 print "</FORM>";
 print "</BODY></HTML>"
 }
else{
 @formfields=split(/&/,$query);
 foreach(@formfields){
 if(/^file=(.*)/){push(@Sel_list,urldecode($1));}
 }
 unless(@Sel_list){
 print "Content-Type: text/html\n\n";
 print "<HTML><BODY><CENTER><H1>Вы должны выбрать что-то из
списка";
 print "</H1></CENTER></BODY></HTML>";
 }
 else{
 print "Content-Type: multipart/mixed;boundary= \"bhy3e23r4t34tne-
htpo7678nneu4232y213vdg\"\n\n";
 print "-bhy3e23r4t34tnehtpo7678nneu4232y213vdg\n";
 foreach(@Sel_list){
  print "Content-Type: application/x-qwerty; name=\"$_\"\n\n";
  open(F,"$_");
  print <F>;
```



```
  close(F);
  print "\n-bhy3e23r4t34tnehtpo7678nneu4232y213vdg\n";
  }
print "Content-Type: text/html\n\n";
print "<HTML><H1>Thats all folks!</H1></HTML>";
print "\n-bhy3e23r4t34tnehtpo7678nneu4232y213 vdg-\n";
}
}
```

Обработка форм

Пришло время перейти к очень важной теме — обработке форм.

При всей простоте (кажущейся) — это едва ли не самое главное предназначение всего стандарта **CGI**. Куда бы вы не зашли, на любой уважающий себя сайт, везде вы встретите **формы**, которые вам предложат заполнить.

В этом деле можно положится только на CGI, так как Java и JavaScript, выполняющиеся на страничке у клиента не имеют доступа к серверу, на котором находится сайт.

Коротко вспомним о том, что происходит при рассматриваемом процессе поближе.

Итак, браузер требует у сервера определенный URL (это может быть как простой документ, так и сгенерированный CGI). В этом документе может содержаться форма. Отображая такой документ браузер также выводит элементы формы (кнопки, поля ввода, поля ввода пароля, переключатели, радио-кнопки, списки, текстовые области, скрытые поля).

И со всем этим добром пользователь может взаимодействовать. К форме естественно имеет доступ и встроенный язык программирования JavaScript — он может как использовать форму для своих нужд, не передавая CGI, так и помогать пользователю в заполнении формы.

После того, как пользователь заполнил форму он нажимает кнопку **Submit**, которая говорит, что форму надо отправить на сервер.

Браузер собирает все имена и значения элементов формы, кодирует их методом **urlencode** и в зависимости от указанного в тэге **FORM** метода вызывает **GET** или **POST** с указанным **URL**, передавая ему данные.

На сервере **CGI**-скрипту это попадает (в зависимости от метода) либо в переменную **QUERY_STRING,** либо на **STDIN**.

Скрипт может проверить данные, занести их в какую-нибудь базу данных, может как **yahoo** выполнить какой-нибудь поиск, может что-нибудь вычислить... Да мало ли что он может, все зависит только от нашей фантазии.

В конце концов скрипт выдает браузеру ответ, который он и отображает.

В этом ответе может содержаться все, что вашей душе угодно — от сообщения об удачном или неправильном запросе, до таких ответов, по сравнению с которыми **yahoo** и **altavista** повиснут от зависти, главное чтобы вам и тем, кто посещает ваш сайт, это нравилось.

А теперь немного о синтаксисе элементов форм, их описании и самое главное особенностях при обработке **CGI**-скриптом.

Итак небольшой экскурс в **HTML**.

FORM

```
<FORM action="http:// ......cgi" method="GET"|"POST"
enctype="encodingType"
  name="formName" target="windowName" onSubmit="Handler">
</FORM>
```

Атрибуты:

action

Как раз и задает тот **URL**, который будет обрабатывать форму, если он опущен, то текущий **URL** документа (а он-то может быть сгенерирован нашим скриптом).

method

Задает метод **GET** или **POST.**

enctype

Обычно не задается, для форм он **application/x-www-form- urlencoded** — по умолчанию, и поддерживается всеми **CGI** скриптами.

Но если вы уж очень хотите, чтобы браузер послал вам данные в другом формате (например **text/plain**), то можете указать этот тип кодировки, только потом не жалуйтесь, что ваш скрипт не может разделить поля, или вообще начинает глючить, когда пользователь ввел какой-то спецсимвол.

name

Задается для **JavaScript**, чтобы обращаться к форме по имени, а не по номеру. Для **CGI** не играет никакой роли.

target

Может определять в какой фрейм отправить полученную информацию. Имеет значение во фреймосодержащих документах. Прозрачен для **CGI** обработки данных.

onSubmit

Определяет **Java Script** — обработчик активизации формы. Применяется для проверки **Java Script**'ом правильности заполнения. Опять таки прозрачен для **CGI**.

Пример типичной формы:

```
<FORM action="http://www.uic.nnov.ru/~paaa/ cgi-bin/test.cgi"
method="POST">
 .........Поля формы.........
</FORM>
```

Форма может содержать элементы. Элементы имеют имена, которые используются для кодирования пар **имя=значение**. Некоторые элементы не передаются CGI, а используются Java Script для управления, например кнопки. Некоторые поля передаются только в тех случаях, когда в них что-то выбрано, например списки и переключатели. Остальные поля передаются всегда, даже когда они пустые.

Например:

```
<FORM action="http://www.doom/cgi-bin/test.cgi">
Your Name:<INPUT name="Name"><BR>
E-Mail:<INPUT name="Email"><BR>
Are you doomer:<INPUT type="checkbox" name="doomer" value="Yes">
<INPUT type="submit" value="Send Form!">
</FORM>
```

Допустим, вы ввели имя **lesha** и адрес **paaa@uic.nnov.ru**, при этом выбрали переключатель. После нажатия кнопки будет отправлен вот такой запрос:

```
http://www.doom/cgi-bin/test.cgi?
Name=lesha&Email=paaa@uic.nnov.ru&doomer=Yes
```

Если же вы не выбрали переключатель, то запрос будет таким:

```
http://www.doom/cgi-bin/test.cgi?
Name=lesha&Email=paaa@uic.nnov.ru
```

Как видите, элемент **doomer** не вошел в строку запроса

Теперь попробуйте оставить поля редактирования пустыми:

```
http://www.doom/cgi-bin/test.cgi?Name=&Email=
```

Эти элементы (**Name** и **Email**) присутствуют и сообщают что они пустые.

Кнопка (button)

```
<INPUT type="button" name="buttname" value="Текст На Кнопке"
onClick="Handler">
```

В форме изображается кнопка, при нажатии которой вызывается **JavaScript-обработчик,** заданный атрибутом **onClick.** Атрибут **name** служит для **JavaScript-именования** кнопки, а не для передачи **CGI**. Так как значение кнопки не передается **CGI**, **value** задает текст, изображаемый на кнопке.

```
<FORM onSubmit="return false;">
<INPUT type="button" value="Просто Кнопочка"
 onClick="alert('Нажали на кнопку!');">
</FORM>
Начало формы
Конец формы
```

Submit

```
<INPUT type="submit" name="submitName" value="Отправить Форму"
onClick="Handler">
```

Кнопка, предназначенная для передачи формы. Опять же сама не передается, а служит только для управления. Текст на ней задается атрибутом **value**.

```
<FORM onSubmit="alert('Нечего Посылать!'); return false;">
<INPUT type="Submit" value="Послать!">
</FORM>
Начало формы
Конец формы
```

Reset

```
<INPUT type="reset" name="resetName" value="Очистить"
onClick="Handler">
```

Кнопка очистки формы. При ее нажатии всем измененным элементам возвращается значение по умолчанию.

```
<FORM onSubmit="return false;">
<INPUT name="something"><BR>
<INPUT type="reset" value="Очистить!">
</FORM>
Начало формы
Конец формы
```


Поле ввода (text)

```
<INPUT [type="text"] name="textName" value="textValue" size=число
[обработчики]>
```

Применяется очень часто, поэтому тип **«text»** служит для **INPUT** по умолчанию, его не надо каждый раз указывать. Имя поля, задаваемое **name** является **обязательным** для **CGI** (в отличии от **JavaScript,** где элементы формы можно индексировать по номерам, а имена служат для удобства и читабельности кода).

Можно задать значение по умолчанию атрибутом **value**, которое будет после загрузки документа. Атрибут **size** позволяет задать размер поля. Также может содержать обработчики **onBlur, onChange, onFocus, onSelect**.

```
<FORM onSubmit="return false;">
<INPUT name="something" size=30
 value="Введите что-нибудь">
</FORM>
Начало формы
Конец формы
```

Текстовая Область (textarea)

```
<TEXTAREA name="textareaName" rows="число" cols="число"
wrap="hard"|"soft">
 TextToEdit
</TEXTAREA>
```

Область многострочного редактирования. Размеры в строках и столбцах задаются атрибутами **rows** и **cols**. Значения атрибута **wrap «hard»** и **«soft»** означают соответственно — мягкую или жесткую разбивку на строки (в большинстве случаев это не существенно).

На что следует действительно обратить внимание, так это на символ, используемый для указания перехода на новую строку.

В Windows — это **«\r\n»**, а в **Unix — «\n»**, так что если это для вас существенно, то приводите преобразование. Например так:

```
$my_text =~ s/\r\n/\n/g;
 <FORM onSubmit="return false;">
 <TEXTAREA name="MyText" rows=7 cols=30>
 Тут можно что-нибудь написать
 </TEXTAREA>
 </FORM>
```

```
Начало формы
Конец формы
```

Поле ввода пароля (password)

```
<INPUT type="password" name="passName" size=число
value="passValue">
```

Очень похоже на поле ввода, отличается тем, что вместо символов в нем отображаются символы «*». Служит для ввода пользователем пароля.

```
<FORM onSubmit="return false;">
Пароль:
<INPUT type="password"
  name="yourpass" size=30>
</FORM>
Начало формы
Пароль:
Конец формы
```

Скрытое поле (hidden)

```
<INPUT type="hidden" name="hiddName" value="hidValue">
```

Поле не отображаемое на экране. Но оно имеет имя и значение, и следовательно передается в форму. Служит для того (и очень часто программисты его применяют), чтобы передавать скрипту какую-нибудь информацию.

Например, если ваш скрипт обрабатывает несколько форм разных типов, то в скрытом поле каждой формы можно указать с какой формой конкретно вы имеете дело. Так как это ваша внутренняя кухня, то нечего пользователю мозолить глаза этой информацией.

```
<FORM onSubmit="return false;">
Этого здесь вам не видно, поле-скрытое.
<INPUT type="hidden" name="formNum" value="3">
</FORM>
Начало формы
Этого здесь вам не видно, поле-скрытое.
Конец формы
```

Переключатель (checkbox)

```
<INPUT type="checkbox" name="checkboxname" value="checkboxValue"
[checked] onClick="Handler">Text
```

В отличии от кнопки, атрибут **value** здесь задает не надпись на переключателе, а его значение (внутреннее).

Поэтому, если надо что-то подписать, пишите рядом с ним. Может быть сразу выбранным, если указан атрибут **checked.** Если **value** не

указано, то значение по умолчанию «**on**». Передается только в том случае, когда выбран.

```
<FORM onSubmit="return false;">
<INPUT type="checkbox" name="inet" value="Yes"
 checked>Доступ к Интернет
</FORM>
Начало формы
Доступ к Интернет
Конец формы
```

Радио-кнопка (radio)

```
<INPUT type="radio" name="radioName" value="radioVal1" [checked]
onClick="Handler">Text
```

В отличие от **checkbox** может быть несколько радио-кнопок с одинаковым параметром **name**, но с разными **value**, из них передается только та, что выбрана.

Одна из них может быть изначально выбрана по умолчанию **checked**.

Например:

```
<FORM onSubmit="return false;">
Вы уверены?<BR>
<INPUT type="radio" name="Radbut" checked>Yes
<INPUT type="radio" name="Radbut">No
</FORM>
Начало формы
Вы уверены?
Yes
No
Конец формы
```

Список (select)

```
<SELECT name="SelectName" size=число [multiple] [обработчики] >
<OPTION value="optionValue1" [selected]>Опция 1
<OPTION value="optionValue2" [selected]>Опция 2
<OPTION value="optionValue3" [selected]>Опция 3
.....
<OPTION value="optionValueN" [selected]>Опция N
</SELECT>
```

Задает список, позволяющий выбрать одну (или несколько) опций из списка. Если атрибут **multiple** не указан, то создается простой выпадающий список, в котором можно выбрать только одну из опций. Его

значение всегда передается, т.к. всегда хоть одно выбрано. Если указан атрибут **multiple**, то во-первых можно указать размер видимой части списка атрибутом **size** (если опций больше, появится скролинг). Во-вторых передаются только выбранные опции, т.е. он может передаться несколько раз

```
?SelectName=opt1&SelectName=opt2&SelectName=opt9
```

если выбраны скажем несколько опций. А может и не разу, если ничего не выбрано из списка.

Можно задавать обработчики **onBlur, onChange, onFocus**.

```
<FORM onSubmit="return false;">
Ваш цвет:<BR>
<SELECT name="singleSel">
<OPTION value="white">Белый
<OPTION value="black">Черный
<OPTION value="magenta">Фиолетовый
<OPTION value="green">Зеленый
<OPTION value="red">Красный
</FORM>
Начало формы
Ваш цвет:
Конец формы
<FORM onSubmit="return false;">
Какие сорта пива вы пили:<BR>
<SELECT name="miltiSel" multiple size=4>
<OPTION value="Балтика">Балтика
<OPTION value="Толстяк">Толстяк
<OPTION value="Премьер">Премьер
<OPTION value="Хольстен">Хольстен
<OPTION value="Бавария">Бавария
<OPTION value="Coca-Cola">Coca-Cola
</SELECT>
</FORM>
Начало формы
Какие сорта пива вы пили:
Конец формы
```

Изображения ismap

Хотя с приходом новых веяний в **HTML** (особенно **Java-апплетов**), метод **ismap** стал настоящей редкостью. И хотя в **80%** случаях возможно найти ему более быструю замену, вы можете именно **ismap** предпочесть всем остальным.

Синтаксис очень простой, почти не отличается от того, если бы вы решили оформить рисунок для гиперссылки:

```
<A href="cgi-bin/somescript.cgi"><IMG src="somepic.gif"
border=0 ismap></A>
```

Заметьте, что все отличие заключается в том, что в тэге **IMG** добавлен атрибут **ismap**. Он говорит браузеру о том, что когда пользователь щелкнет на картинке, то нужно перейти не просто к **URL,** указанному в **<A href=URL>,** а к этому **URL** необходимо добавить координаты той точки, по которой пользователь щелкнул мышью.

В нашем примере, если пользователь щелкнул по точке **x=10, y=15,** то браузер перейдет на **URL**:

```
http://www.somehost.ru/cgi-bin/somescript.cgi?10,15
```

Т.е. координаты идут на скрипте в переменную **QUERY_STRING**. Как их оттуда извлечь? Нет ничего проще:

```
($x,$y)=split(/,/,$ENV{'QUERY_STRING'});
```

Вот скрипт, который просто показывает координаты точки щелчка:

```
#!/usr/bin/perl
#ismap_xy.cgi
($x,$y)=split(/,/,$ENV{'QUERY_STRING'});
print "Content-Type: text/html\n\n";
print "<HTML><HEAD><TITLE>Ismap X Y</TITLE></HEAD>";
print "<BODY><H1>Вы щелкнули в точке: x=$x,
y=$y</H1></BODY></HTML>";
```

А что с ними делать дальше — это уже зависит только от вашей фантазии. Дайте ей ход — и все у вас получится!

Очень часто **ismap** применяют для графического оглавления сайта. Когда щелкают на разные части рисунка, то переходят к разным страничкам сайта.

Это легко реализуется, если скрипт выдаст нужный **URL** в **Location:** (вспомните заголовок ответа **CGI**).

Вот пример и покажет это. Заготовьте файл **urlmap.txt,** в котором будет информация из строк в таком формате:

```
minx miny maxx maxy URL
```

Где **minx miny maxx maxy** задают участок рисунка, а следующее за ними поле задает **URL**, которому этот участок соответствует.

Пример:

```
1 1 20 50 http://www.uic.nnov.ru/~paaa/index_p.html
1 50 20 100 http://www.uic.nnov.ru/~paaa/projects.html
20 1 100 100 http://www.uic.nnov.ru/~paaa/ cgi-bin/guestbook.cgi
```

Где-нибудь на своей страничке воткните что-то вроде:

```
<A href="cgi-bin/testismap.cgi"><IMG src="gifs/doom2.jpg" border=0
ismap></A>
```

А сам скрипт **testismap.cgi** будет иметь вот такой простенький вид:

```
#!/usr/bin/perl
#testismap.cgi
$default_url="http://www.uic.nnov.ru/~paaa/"; #URL по умолча-
нию,переходим к нему когда щелкнули
#в участок,которому не сопоставлен URL
$url_map_file="urlmap.txt"; #файл с информацией об URL
($x,$y)=split(/,/,$ENV{'QUERY_STRING'});
open(F,"$url_map_file")|| print "Location: $default_url\n\n";
$url=$default_url;
foreach(<F>){
 chomp;
 ($minx,$miny,$maxx,$maxy,$URL)=split(/\s+/);
 if(($x>=$minx)&&($x<$maxx)&&
 ($y>=$miny)&&($x<$maxy)){$url=$URL;}
 }
close(F);
print "Location: $url\n\n";
```

Анимация

Когда говорят о каком-то популярном сайте, то частенько к преимуществам относят и **анимацию**. Действительно, когда изображение изменяется (и особенно к месту), то это смотрится и пользователю нравится.

Говоря об анимации, нужно сразу отметить, что нет лучшего способа. Анимацию можно сделать десятками способов, каждый из которых хорош в своей области применения. Перечислим только некоторые из них, которые чаще всего применяются.

Самый простой, но наименее функциональный способ — это **GIF** с анимацией. Потом можно воткнуть анимационный файл **MPEG** или **AVI** — они больше отражают суть анимации, но имеют недостаток — для проигрывания их на некоторых браузерах нужны специальные подключаемые модули. К тому же они не интерактивны.

Можно реализовать анимацию в рамках **Java-апплета**, когда апплет находясь на страничке сам перерисовывается со временем.

Таким же интерактивным средством служит обращение к массиву **document.images[]** из **JavaScript**. Достоинство — помимо интерактивности — **полная** интегрированность с **HTML**-страничкой. Оно может использоваться как и предыдущее, но только с относительно новыми браузерами, которые поддерживают **Java** и **JavaScript**.

В каждом случае выбор остается за вами. Вам решать насколько тот или иной способ хорош в вашей ситуации. Познакомимся еще с одним.

Вы уже были знакомы с этим способом, когда шел рассказ о **nph**-скриптах. Теперь, когда вы уже так много знаете, можно модифицировать тот пример, добавив в него вызов картинки по случайному принципу:

```
#!/usr/bin/perl
#nph-animate2.cgi
$delay=3;
@files = qw(img0.gif img1.gif img2.gif img3.gif);
select (STDOUT);
$|=1; #autoflush mode on
#Generate header
print "HTTP/1.0 200 Okay\n";
print "Content-Type: multipart/x-mixed-replace; boundary=mybound-
ary\n\n";
srand;
print "--myboundary\n";
while(1){
 $file=$files[int(rand($#files))]; #random file
 print "Content-Type: image/gif\n\n";
 open(PIC,"$file");
 print <PIC>;
 close(PIC);
 print "\n--myboundary\n";
 sleep($delay);
 }
```

Конечно, это одно из самых примитивных применений такой системы. Более мощным примером могло бы послужить отслеживание на сервере какого-нибудь периодически изменяющегося файла и пересылка пользователю обновленной версии.

Такая система применяется например в Чате, при появлении новых сообщений.

Отладка

CGI-программы — не самые простые в отладке, по сложности они способны сравниться лишь с отладкой драйверов. Вся сложность заключается в том, что скрипт выполняется не как обычная программа. Он выполняется в специальной среде сервера, которая создается при клиентском запросе, к тому же он исполняется не из под вашего аккаунта, а на непривилегированном уровне.

Если скрипт не исполняется потому, что вы допустили синтаксические ошибки, то самих этих ошибок вы не увидите, на экране будет только «Internal Server Error». Из-за чего она произошла, вы можете только гадать.

Также если вы забыли задать к какому-то файлу нужные права доступа, то тоже будет трудно выяснить в чем причина ошибки (если конечно к этому вы не готовы).

Начнем с того, что у нас есть скрипт **test.cgi,** мы уже сделали его исполняемым **chmod +x test.cgi.** Простейший способ проверить его на ошибки — это команда **perl-c test.cgi.**

Ключ **-c** говорит Perl, что надо только проверить синтаксис. Все сообщения об ошибках вы можете видеть и поправить. Более тяжелый случай состоит в том, когда Perl встроен в Web-Сервер, причем версии разные. Так до недавнего времени было на uic'e.

Тот Perl, с которым работаем в командной строке — 4-й версии, а на сервере стоит 5-й версии. Если ваша CGI-программа использует при этом какие-нибудь преимущества 5-й версии (например, объектно-ориентированные модули), то вы думаете отладить ее нельзя? Ошибаетесь!

Закомментируйте всю вашу программу, т.е. перед каждой строчкой поставьте символ «#». Затем добавьте вот такие строчки:

```
print "Content-Type: text/html\n\n"; print "<HTML>Test</HTML>";
exit;
Должно получиться так:
#!/usr/bin/perl
#test.cgi
print "Content-Type: text/html\n\n";
print "<HTML>Test</HTML>";
 exit; #Программа как вы понимаете выполняется только до этого
места
#
#if($ENV{'REQUEST_METHOD'} eq 'GET'){$query=$ENV{'QUERY_STRING'}}
#else{sysread STDIN,$query,$ENV{'CONTENT_LENGTH'};}
```



```
#if($query eq ''){
# @formfields=split /&/,$query;
# .......
# ........
```

А теперь запускайте скрипт. Естественно, он выдаст одно слово «Test». Разкомментируйте несколько строчек. Еще раз запустите скрипт. Он опять выдаст «Test». Значит синтаксически эти разкомментированные строчки были правильными. И так далее...

Если в очередной раз после разкомментирования вы запустили скрипт и получили «Internal Server Error» — значит в этих строках содержалась какая-то синтаксическая ошибка.

Этот способ отловки синтаксических ошибок трудоемок, но к нему придется прибегнуть, если ваш скрипт написан под ту версию Perl, что на сервере, а не под ту, что у вас.

Узнать версию Perl можно при помощи команды: **perl -v**

Когда мы отловили в нашем скрипте все синтаксические ошибки, он заработал, но это не значит, что он работает правильно.

Что еще можно посоветовать при отладке ошибок CGI-скриптов, возникающих во время выполнения программы. Допустим, какой-то файл не открылся. Конечно, показывать перепуганному пользователю эти технические подробности ни к чему, поэтому заведите себе специальный файл debug.txt, и пусть ваши скрипты пишут в этот файл причины своих ошибок и сбоев, да и вообще о всех непредвиденных событиях.

Это можно реализовать так:

```
sub debug_err{
 open(DEBUGFILE,">>debug.txt");
 print DEBUGFILE $ENV{'SCRIPT_NAME'}.' '.scalar localtime.'
'.@_."\n";
 close(DEBUGFILE);
 }
```

Примеры использования (напомню, что встроенная переменная Perl $! содержит сообщение о причине последней ошибки, поэтому включайте ее всегда в свои сообщения):

```
open(F,"+<$myfile") || debug_err("Cannot open $myfile $!");
seek(F,0,0) || debug_err("Cannot seek $myfile $!");
connect(SOCKET,$paddr)|| debug_err("Cannot connect to $remote
$!");
......
```

Потом можно периодически заглядывать в этот файл **debug.txt** и смотреть, какие ошибки встречались при работе ваших скриптов. Таким образом, ваши скрипты сами помогают в своей отладке.

Также может оказать помощь (может и не оказать) просмотр http'шных логов. Все обращения к URL на сервере, и все возникающие при этом ошибки идут в логи сервера httpd.

Необходимо сразу предупредить — размеры этих логов (даже на средних размеров сервере) достигает десятков мегабайт. Поэтому даже не пытайтесь их вот так просто посмотреть. Лучше воспользуйтесь такими утилитами, как **grep, head, tail, more, less.**

Кстати, необходимо сказать о причине еще одной (совсем не очевидной) ошибки.

Если вы набрали скрипт у себя дома на компьютере, то полученный скрипт состоит из текста в DOS'ом формате, а не в Unix'ом, так что имейте это ввиду. Запускать вам его придется в системе Unix, так что следует перевести программный текст в нужный формат.

Дело в том, что в системах DOS и Windows для разделения строк используется не один — «символ новой строки» («\n»), а пара символов — «возврат каретки» («\r») и «символ новой строки» («\n»).

Для простых HTML-файлов это не типично — браузеры игнорируют такие символы при выводе. Но скрипт является программой. А в программе никаких символов возврата каретки быть не должно! Особенно в первой строке. Потому что, когда операционная система запускает скрипт, она определяет какое приложение запустить для обработки данного скрипта именно по первой строке.

В первой строке, как вы знаете содержится **#!/usr/bin/perl** или **#!/usr/local/bin/perl**. Поэтому, при запуске скрипта **mysrcipt** (это кстати вы можете сами увидеть с помощью команд **top** и **ps**), система запускает команду **/usr/bin/perl mysript**.

А теперь представьте — что будет если не убрать символ **возврата каретки** из Windows-файла.

Получится

```
#!/usr/bin/perl<возврат каретки>
```

и естественно, что такого приложения нет, и, следовательно, попытка выполнить команду

```
/usr/bin/perl<возврат каретки> mysript
```

ни к чему не приведет!

Глава 2. Выбор CMS

Для выбора того или иного CMS, вы должны определиться и спросить себя, что вы ожидаете получить. Существует такое огромное количество content management systems, что выбрать подходящую очень сложно. У всех систем управления контентом существуют свои плюсы и минусы. Лучше всего сразу определиться, какие функции должен нести ваш портал.

Вот пять основных критериев выбора CMS:

1. Сперва определите, какой сайт вы желаете видеть. Это портал? Это BLOG? Это сайт газеты или online ленты новостей? Это сайт вашего проекта? Это каталог или рейтинг сайтов? Это представительство компании? Может это каталог эротики? Или же все вместе?

Это самый простой и ответственный шаг в выборе площадки для ваших дальнейших действий.

2. Далее определитесь с тем, какие функции будет выполнять портал. Будет ли он представлять собой статическую или динамическую информацию? Нужен ли будет календарь дат и событий? Потребуется ли возможность загрузки или скачивания файлов (upload/download)? Как на счет фото галереи? Или вам потребуется голосование и рейтинг? Все это называется модули (modules) или плагины (plug-ins) и часто являются дополнениями к основной системе (иногда они интегрированы в саму систему). Более стабильная CMS имеют больше поддержки, которая выполняет перевод, создает модули и плагины, следит за обнаруженными ошибками и дырами в content management system. Другим может уделяться мало внимания, и многие модули могут быть недоработанными и так и не выйти релиз.

3. После выбора базовой конфигурации и модулей, которые вы добавите, нужно остановиться на внешнем виде (дизайне) вашего сайта. Большинство систем управления контентом сейчас используют CSS (cascading style sheet) и так называемые темы (themes), позволяющие быстро сменить дизайн вашего сайта. Иногда это называют скином (skin) сайта. Именно темы сделают ваш сайт профессионально, глупо, драматически или же по детски выглядящим. Все зависит от вашего желания и фантазии.

Если вы сами не можете или не хотите создать свою оригинальную тему, то существует множество ресурсов, способных вам помочь. На этих ресурсах вы сможете скачать тему, созданную более креативными людь-

ми. Некоторые продвинутые CMS позволяют зарегистрированному пользователю выбрать скин, которые ему больше нравится (конечно, если у вас загружены эти скины). Вообще большинство движков динамических сайтов дают вам практически неограниченные возможности по персонализации своего портала. Ваши изменения могут относиться как к оформлению сайта, так и к приветствию пользователя попавшего на сайт или же размещение блоков, модулей и т.д. на сайте.

4. Теперь после изменения сайта под свой вкус или нужду, стоит задуматься о «мясе» вашего сайта — информации (контент, content). На самом деле, без информации внешний вид сайта не имеет никакого значения. Если кроме привета всех зашедшим к вам, на сайте больше ничего нет, то вряд ли к вам зайдут еще раз. Изложение информации может иметь разных характер: шутливое, серьезное, деловое, познавательное и т.д., и т.п. Важно то, что информация сможет заставить посетителя вернутся к вам еще раз. Старайтесь писать интересные статьи, истории, новости, добавляйте картинки, предлагайте свои исследования. Возможность добавлять свои комментарии, сообщения на форуме и т.д. всегда привлекает посетителей, и они вернутся к вам снова и снова (лучшая бесплатная раскрутка вашего сайта).

5. Ну и наконец, последний шаг в выборе CMS. При выборе подумайте, насколько удобно вам администрировать эту систему, что с ней случится через определенное время? Будет ли контент добавляться в архив и доступен online? Сможете ли вы восстановить портал после переезда на новый хостинг или же после крушения старого? Эти не самые благоприятные мысли лучше обдумать заранее. Многие CMS имеют возможности сохранения своей базы данных автоматически или же вам самим придется это все делать. Продумайте самый плохой исход дел заранее, чтобы, столкнувшись с проблемой, не было слишком поздно. Просто не забывайте регулярно сохранять необходимые файлы и БД.

Глава 3. Серверные скрипты

В свое время перед любым web-мастером встают вопросы такого типа: «как сохранить информацию на сервере», «как отправить результаты заполнения формы на e-мейл».

Сделать это невозможно только средствами браузера. Для реализации подобных вещей существует множество специальных языков. Самой большой популярностью пользуются perl и PHP. В чем их сходство и различие?

Часто путают понятие CGI и perl. CGI (Common Gateway Interface) — протокол обмена данными с программами. А perl — специальный язык высокого уровня, на котором и реализуются необходимые функции взаимодействия с операционной системой на сервере. В общем случае с помощью CGI можно запустить любое приложение на сервере и все, что будет из него (приложения) выведено на стандартный поток вывода, попадет в браузер. Параллельно приложение может произвести вывод данных в файл на сервере, послать на e-мейл или поместить (извлечь) что-то в базу данных.

Коренное отличие PHP от CGI заключается в том, что PHP является препроцессором HTML. Т.е. его работа построена по следующей схеме:

```
.phtml(.php3) --> php.exe --> браузер
```

Т.е. до того, как сервер «отдаст» файл браузеру, его просматривает препроцессор-интерпретатор. Что это значит? Файлы, которые подвергаются обработке препроцессором, должны иметь определенное расширение (обычно это .phtml или .php3, но эти значения можно поменять) и содержать (хотя это не обязательное требование) код для препроцессора. Код этот может быть оформлен следующими способами:

```
<?php инструкции ?>
```

Или:

```
<SCRIPT LANGUAGE="PHP">
инструкции
<SCRIPT>
```

Если нам необходимо вставить в обычную HTML страницу результат работы несложной функции, то это удобнее сделать с PHP, так как код может содержаться прямо в HTML коде страницы. Для CGI, в такой ситуации, нам придется либо выводить всю страницу из скрипта, либо использовать технологию Server Side Include.

Для функционирования PHP сам препроцессор должен быть установлен на сервере. В настоящее время считается хорошим тоном предлагать эту услугу на всех серверах бесплатного доступа. На всякий случай, проконсультируйтесь с системным администратором своего сервера.

И так что мы можем сделать с помощью PHP? Самый простейший пример:

```
<html>
<head>
<title>Тест</title>
</head>
<body>
```

```
<?php echo "Hello, World!"; ?>
</body>
</html>
```

Исходный текст документа в браузере выглядит так:

```
<html>
<head>
<title>Тест</title>
</head>
<body>
Hello, World!</body>
</html>
```

С помощью функции echo мы можем вывести все, что угодно в исходный текст документа. Теперь рассмотрим, что-то более полезное.

Как вы могли заметить, страницы на большинстве серверов содержат неизменную часть: навигационные панели, логотипы, кнопки и пр. Довольно трудно переписывать эти вещи каждый раз, когда что-то добавляешь на свой сайт. Часто раньше приходилось создавать новый «опыт» из шаблона, который, зачастую, занимал больше места, чем сам опыт. К счастью, в PHP есть функция подключения внешних файлов. Таких функций две (на самом деле, их три, но функция «virtual» используется только для сервера Апач, и является заменой стандартной директивы **<!--virtual ...): include()** и **require()**. Основное отличие этих функций состоит в том, что вторая включает текст файла в любом случае, а первая — только, если он еще не был включен. Например следующий текст мы поместим в файл header.inc.php3:

```
<html>
<head>
<title>Тест</title>
</head>
<body>
```

А следующий в файл footer.inc.php3:

```
</body>
</html>
```

А в основном файле поместим вот это:

```
<?php
include("./header.inc.php3");
echo "Hello, World!";
include("./footer.inc.php3");
?>
```


Данный пример, конечно, не показателен. Но файлы заголовка и завершения, которые используются на этих страницах, в общей сложности, занимают почти 10 килобайт.

Глава 4. Вступление в PHP и MySQL

PHP

PHP – это скрипт-язык (scripting language), встраиваемый в HTML, который интерпретируется и выполняется на сервере. Проще всего это показать на примере:

```
<html>
<head>
<title>Example</title>
</head>
<body>
<?php echo "Hi, I'm a PHP script!"; ?>
</body>
</html>
```

После выполнения этого скрипта мы получим страничку, в которой будет написано

```
Hi, I'm a PHP script!
```

Весьма просто и бесполезно.

Основное отличие от CGI-скриптов, написанных на других языках, типа Perl или C – это то, что в CGI-программах вы сами пишете выводимый HTML-код, а, используя PHP – вы встраиваете свою программу в готовую HTML-страницу, используя открывающий и закрывающий тэги (в примере **<?php** и **?>**).

Отличие PHP от JavaScript, состоит в том, что PHP-скрипт выполняется на сервере, а клиенту передается результат работы, тогда как в JavaScript-код полностью передается на клиентскую машину и только там выполняется. Любители Internet Information Server найдут, что PHP очень похож на Active Server Pages (ASP), а энтузиасты Java скажут, что PHP похож на Java Server Pages (JSP). Все три языка позволяют размещать код, выполняемый на Web-сервере, внутри HTML страниц.

Возможности PHP

В нескольких словах – на PHP можно сделать все, что можно сделать с помощью CGI–программ. Например: обрабатывать данные из

форм, генерировать динамические страницы, получать и посылать куки (cookies).

Кроме этого в PHP включена поддержка многих баз данных (databases), что делает написание Web-приложений с использованием БД до невозможности простым.

Вот неполный перечень поддерживаемых БД:

- Adabas D
- InterBase
- Solid
- dBase
- mSQL
- Sybase
- Empress
- MySQL
- Velocis
- FilePro
- Oracle
- Unix dbm
- Informix
- PostgreSQL

Вдобавок ко всему PHP понимает протоколы IMAP, SNMP, NNTP, POP3 и даже HTTP, а также имеет возможность работать с сокетами (sockets) и общаться по другим протоколам.

Краткая история PHP

Началом PHP можно считать осень 1994 года, когда Rasmus Lerdorf решил расширить возможности своей Home-page и написать небольшой движок для выполнения простейших задач. Такой движок был готов к началу 1995 года и назывался Personal Home Page Tools. Умел он не очень много – понимал простейший язык и всего несколько макросов.

К середине 1995 года появилась вторая версия, которая называлась PHP/FI Version 2. Приставка FI – присоединилась из другого пакета Rasmusa, который умел обрабатывать формы (Form Interpritator).

PHP/FI компилировался внутрь Apache и использовал стандартный API Apache. PHP скрипты оказались быстрее аналогичных CGI – скриптов, так как серверу не было необходимости порождать новый процесс. Язык PHP по возможностям приблизился к Perl, самому популярному языку для написания CGI-программ. Была добавлена поддержка множества известных баз данных (например, MySQL и Oracle). Интерфейс к GD – библиотеке, позволял генерировать картинки на лету. С этого момента началось широкое распространение PHP/FI.

В конце 1997 Zeev Suraski и Andi Gutmans решили переписать внутренний движок, с целью исправить ошибки интерпретатора и повысить скорость выполнения скриптов. Через полгода, 6 июня 1998 года вышла новая версия, которая была названа PHP 3.

К лету 1999 года PHP 3 был включен в несколько коммерческих продуктов. По данным NetCraft на ноябрь 1999 PHP использовался в более чем 1 млн. доменах.

2000 год — переписанный движок PHP4.

На сегодняшний день используется новая версия PHP 5, ее производительность в десятки раз выше чем у предыдущей.

Почему нужно выбирать PHP

Разработчикам Web-приложений нет необходимости говорить, что web-страницы — это не только текст и картинки. Достойный внимания сайт должен поддерживать некоторый уровень интерактивности с пользователем: поиск информации, продажа продуктов, конференции и т.п. Традиционно все это реализовалось CGI-скриптами, написанными на Perl. Но CGI- скрипты очень плохо масштабируемы. Каждый новый вызов CGI, требует от ядра порождения нового процесса, а это занимает процессорное время и тратит оперативную память. PHP предлагает другой вариант – он работает как часть Web-сервера, и этим самым похож на ASP от Microsoft.

Синтаксис PHP очень похож на синтаксис C или Perl. Люди, знакомые с программированием, очень быстро смогут начать писать программы на PHP. В этом языке нет строгой типизации данных и нет необходимости в действиях по выделению/освобождению памяти. Программы, написанные на PHP, достаточно легкочитаемы. Написанный PHP – код легко зрительно прочитать и понять, в отличие от Perl-программ.

Недостатки PHP

PHP является интерпретируемым языком, и, вследствие этого, не может сравниться по скорости с компилируемым C. Однако при написа-

нии небольших программ, что, в общем-то, присуще проектам на PHP, когда весь проект состоит из многих небольших страниц с кодом, вступают в силу накладные расходы на загрузку в память и вызов CGI-программы, написанной на C.

Не такая большая база готовых модулей, как, например, CPAN у Perl. С этим ничего нельзя поделать – это дело времени. В PHP 4 разработчики предусмотрели специальный репозиторий, аналогичный CPAN, и я думаю, очень скоро будет написано достаточное количество модулей для его наполнения.

MySQL

MySQL – компактный многопоточный сервер баз данных. MySQL характеризуется большой скоростью, устойчивостью и легкостью в использовании.

MySQL был разработан компанией TcX для внутренних нужд, которые заключались в быстрой обработке очень больших баз данных. Компания утверждает, что использует MySQL с 1996 года на сервере с более чем 40 БД, которые содержат 10,000 таблиц, из которых более чем 500 имеют более 7 миллионов строк.

MySQL является идеальным решением для малых и средних приложений. Исходники сервера компилируются на множестве платформ. Наиболее полно возможности сервера проявляются на Unix-серверах, где есть поддержка многопоточности, что дает значительный прирост производительности.

На текущий момент MySQL все еще в стадии разработки, хотя версии 3.22 полностью работоспособны.

MySQL-сервер является бесплатным для некоммерческого использования. Иначе необходимо приобретение лицензии.

Возможности MySQL

MySQL поддерживает язык запросов SQL в стандарте ANSI 92, и кроме этого имеет множество расширений к этому стандарту, которых нет ни в одной другой СУБД.

Краткий перечень возможностей MySQL:

- Поддерживается неограниченное количество пользователей, одновременно работающих с базой данных.
- Количество строк в таблицах может достигать 50 млн.

- Быстрое выполнение команд. Возможно MySQL самый быстрый сервер из существующих.
- Простая и эффективная система безопасности.

MySQL действительно очень быстрый сервер, но для достижения этого разработчикам пришлось пожертвовать некоторыми требованиями к реляционным СУБД. В MySQL отсутствуют:

- Поддержка вложенных запросов, типа SELECT * FROM table1 WHERE id IN (SELECT id FROM table2). Утверждается, что такая возможность будет в версии 3.23.
- Не реализована поддержка транзакций. Взамен предлагается использовать LOCK/UNLOCK TABLE.
- Нет поддержки внешних (foreign) ключей.
- Нет поддержки триггеров и хранимых процедур.
- Нет поддержки представлений (VIEW). В версии 3.23 планируется возможность создавать представления.

Примеры использования PHP

Работа с формами

В этом примере показано как в PHP легко обрабатывать данные с HTML – форм.

Создадим простой HTML файл.

```
<HTML>
<HEAD>
<TITLE>Запрос информации</TITLE>
<BODY>
<CENTER>
Хотите больше знать о наших товарах?
<P>
<TABLE WIDTH = 400><TR><TD align = right>
<FORM ACTION="email.php3" METHOD="POST">
Ваше имя:<BR>
<INPUT TYPE="text" NAME="name" SIZE="20" MAXLENGTH="30">
<P>
Ваш email:<BR>
<INPUT TYPE="text" NAME="email" SIZE="20" MAXLENGTH="30">
<P>
Меня интересуют:
<SELECT NAME="preference">
<OPTION value = "Яблоки">Яблоки
```

```
<OPTION value = "Апельсины">Апельсины
</SELECT>
<P>
<INPUT TYPE="submit" VALUE="Отправить запрос!">
</FORM>
</TD></TR></TABLE></CENTER>
</BODY>
</HTML>
```

Назовем этот файл request.html. В нем мы указали, что данные формы будут обрабатываться файлом email.php3. Приведем его содержание:

```
<?
/* Этот скрипт получает переменные из request.html */
PRINT "<CENTER>";
PRINT "Привет, $name.";
PRINT "<BR><BR>";
PRINT "Спасибо за ваш интерес.<BR><BR>";
PRINT "Вас интересуют $preference. Информацию о них мы пошлем вам
на email: $email.";
PRINT "</CENTER>";
?>
```

Теперь, если пользователь вызовет request.html и наберет в форме имя «Вася», email: vasya@pupkin.com и скажет, что его интересуют «Яблоки», а после этого нажмет «Отправить запрос!», то в ответ вызовется email.php3, который выведет на экран примерно следующее:

```
Привет, Вася
Спасибо за ваш интерес.
Вас интересуют Яблоки. Информацию о них мы пошлем вам на email:
vasya@pupkin.com
```

Теперь мы должны сдержать обещание и выслать email.

Для этого в PHP есть функция **MAIL**.

Синтаксис:

```
void mail(string to, string subject, string message, string
add_headers);
```

to – email адрес получателя.

subject – тема письма.

message – собственно текст сообщения.

add_headers – другие параметры заголовка письма (необязательный параметр).

Допишем в конец файла email.php3 следующий код:

```
<?
mail($email, "Запрос на информацию", "$name\\n
Спасибо за ваш интерес!\\n
Вас интересуют $preference\\n
Мы их распространяем бесплатно. Обратитесь в ближайший филиал на-
шей компании и получите ящик этого продукта.\\n
");
mail("administration@me.com",
"Был запрос на информацию.",
"$name интересовали $preference\\n
email-адрес: $email. \\n");
?>
```

Вот теперь пользователь будет получать письмо с более подробной информацией о наших товарах. Также письмо получит и администратор сайта.

Когда интересующихся нашими товарами станет очень много, мы захотим их как-то упорядочить и хранить информацию о них в базе данных.

Работа с MySQL: сохранение данных в базе данных

Для начала создаем базу данных и таблицу. Входим в MySQL, и выполняем команды:

```
>CREATE DATABASE products;
>CREATE TABLE clients (name VARCHAR(25), email VARCHAR(25),
choise VARCHAR(8));
```

Для общения с MySQL из PHP понадобятся следующие функции.

```
int mysql_connect(string hostname, string username, string
password);
```

Создать соединение с MySQL.

Параметры:

- **Hostname** – имя хоста, на котором находится база данных.
- **Username** – имя пользователя.
- **Password** – пароль пользователя.

Функция возвращает параметр типа int, который больше 0, если соединение прошло успешно, и равен 0 в противном случае.

```
int mysql_select_db(string database_name, int link_identifier);
```

Выбрать базу данных для работы.

Параметры:

Database_name – имя базы данных.

link_identifier – ID соединения, которое получено в функции **mysql_connect** (параметр необязательный, если он не указывается, то используется ID от последнего вызова **mysql_connect**).

Функция возвращает значение **true** или **false**.

```
int mysql_query(string query, int link_identifier);
```

Функция выполняет запрос к базе данных.

Параметры:

Query – строка, содержащая запрос.

link_identifier – см. предыдущую функцию.

Функция возвращает ID результата или 0, если произошла ошибка.

```
int mysql_close(int link_identifier);
```

Функция закрывает соединение с MySQL.

Параметры:

link_identifier – см. выше.

Функция возвращает значение true или false.

Теперь наш файл email.php3 будет иметь следующий вид:

```
<?
/* Этот скрипт получает переменные из request.html */

/* Некоторые переменные */

$hostname = "localhost";
$username = "myusername";
$password = "mypassword";
$dbName = "products";

/* Таблица MySQL, в которой хранятся данные */
$userstable = "clients";

/* email администратора */
$adminaddress = "administration@me.com";
```



```
/* создать соединение */
MYSQL_CONNECT($hostname,$username,$password) OR DIE("Не могу
создать соединение ");

@mysql_select_db("$dbName") or die("Не могу выбрать базу данных
");

PRINT "<CENTER>";
PRINT "Привет, $name.";
PRINT "<BR><BR>";
PRINT "Спасибо за ваш интерес.<BR><BR>";
PRINT "Вас интересуют $preference. Информацию о них мы пошлем вам
на email: $email.";
PRINT "</CENTER>";

/* Отправляем email */
mail($email, "Запрос на информацию", "$namen\\n
Спасибо за ваш интерес!\\n
Вас интересуют $preference\\n
Мы их распространяем бесплатно. Обратитесь в ближайший филиал на-
шей компании и получите ящик этого продукта.\\n
");

mail("administration@me.com",
"Был запрос на информацию.",
"$name интересовали $preference\\n
email-адрес: $email. \\n");

/* Вставить информацию о клиенте в таблицу */
$query = "INSERT INTO $userstable VALUES('$name','$email',
'$preference')";

$result = MYSQL_QUERY($query);

PRINT "Информация о вас занесена в базу данных.";

/* Закрыть соединение */
MYSQL_CLOSE();
?>
```

Вот так легко можно работать с базой данных в PHP. Теперь кроме письменных уведомлений, информация о клиенте и его интересах будет заносится в таблицу MySQL.

Работа с MySQL: получение данных из базы данных

После занесения данных, нас иногда будет интересовать вопрос так кого же из наших клиентов интересует товар «Яблоки».

Напишем скрипт apple.php3:

```
<?/* Скрипт показывает клиентов, которые яблоки любят больше чем
апельсины */
$hostname = "localhost";
$username = "myusername";
$password = "mypassword";
$dbName = "products";

/* Таблица MySQL, в которой хранятся данные */
$userstable = "clients";

/* создать соединение */
MYSQL_CONNECT($hostname,$username,$password) OR DIE("Не могу со-
здать соединение ");

@mysql_select_db("$dbName") or die("Не могу выбрать базу данных
");

/* Выбрать всех клиентов - яблочников */
$query = "SELECT * FROM $userstable WHERE choice = 'Яблоки'";

$result = MYSQL_QUERY($query);

/* Как много нашлось таких */
$number = MYSQL_NUMROWS($result);

/* Напечатать всех в красивом виде*/
$i = 0;

IF ($number == 0) {
PRINT "<CENTER><P>Любителей яблок нет</CENTER>";
} ELSEIF ($number > 0) {
PRINT "<CENTER><P>Количество любителей яблок: $number<BR><BR>";
WHILE ($i < $number){
$name = mysql_result($result,$i,"name");
$email = mysql_result($result,$i,"email");
PRINT "Клиент $name любит Яблоки.<BR>";
PRINT "Его Email: $email.";
```



```
PRINT "<BR><BR>";
$i++;
}
PRINT "</CENTER>";
}
?>
```

Здесь мы использовали две новых функции:

```
int mysql_num_rows(int result);
```

Функция возвращает количество строк в результате запроса.

Параметр **result** – содержит ID результата запроса.

```
int mysql_result(int result, int i, column);
```

Функция возвращает значение поля в столбце column и в строке i.

Вот и все, коммерческий продукт практически готов.

Глава 5. Написание Гостевой книги на PHP

То, что знать мнения других о себе и о своем детище иногда полезно, никто не спорит. Когда сайт крепко становится на ноги, и выходит из стадии младенчества, web-мастер задается вопросом – как бы мне узнать, что именно хотят мои посетители, что волнует их и что не нравится на сайте? Все эти вопросы легко исследовать, установив гостевую книгу.

Для начала вам нужен сайт, размещенный на хостинге, где поддерживается гипертекстовый препроцессор PHP. Это – очень простой и несложный язык программирования, и именно на нем и будет написана наша гостевая книга.

Все, что понадобится, это два файла. В первом будут размещены форма для ввода данных и сам скрипт гостевой, а во втором – храниться результаты введенных данных в специальном формате. Давайте обзовем их соответственно guest.phtml и guest.txt.

Не забывайте, что после закачки на сервер этих двух файлов на файл с результатами (guest.txt) нужно будет установить атрибуты, разрешающие запись в файл. Сделать это можно практически в любом из FTP-менеджеров, проверив атрибуты уже закачанного на сервер файла, и установив в настройках все галочки. Теперь перейдем непосредственно к коду. Он должен быть введен в файл guest.phtml.

Для начала, делаем форму для ввода данных. Оформить вы сможете ее сами по собственному вкусу, так что сосредоточимся на сути.

```
<h2>Гостевая книга</h2>
<form action=guest.phtml method=post>
Введите email: <INPUT TYPE=TEXT NAME=email><BR>
Ваше имя: <INPUT TYPE=TEXT NAME=name><BR>
Сообщение: <br><textarea name="msg" rows=10 cols=40></textarea>
<br><input type=submit value="Отправить ">
<br><br>
```

Только что мы определили форму с тремя полями – адрес (переменная email), имя (переменная name) и сообщение (переменная msg). После того, как посетитель введет данные, и нажмет на кнопку Отправить, все эти переменные будут доступны нашему скрипту, причем значение переменных будет соответствовать введенным данным. Теперь нужно все это обработать.

```
<?
$files = "guest.txt";
$qq=50;
```

Мы определились с именем файла, куда будем писать данные и с максимальным количеством сообщений, которое может быть выведено на экран.

```
if ($email == "") { $email = "нет"; }
$msg=substr($msg,0,999);
$email=substr($email,0,39);
$name=substr($name,0,39);
```

Здесь введенные данные обработаны таким образом, что бы переменная адреса не была пустой (то есть если туда ничего не ввели, она была равна строке «нет»). А так же каждая введенная переменная обрезается, что бы ограничить количество вводимых символов. Это нужно для защиты от баловства, когда в книгу начинают вводить массу информации, которая никому не нужна.

```
if ($msg != "" && $name != "") {
```

Очень важный момент: если сообщение или имя не указаны вообще, скрипт ничего никуда не записывает, а просто продолжит обработку дальше, где выведет сообщения гостевой книги на экран. Но если и имя и сообщение введены, скрипт прежде чем вывести данные на экран, должен сделать запись отформатированных данных в файл для сообщений.

```
$time = Date("h:i:M:d");
$soo = "\\n<b>$time $name (<a href=\\"mailto: $email \\">
$email </a>)</b><br> $msg<hr>";
$fp = fopen($files, "a+");
$fw = fwrite($fp, $soo);
fclose($fp); }
```


Вначале определяется и форматируется время, когда вводится сообщение. Потом формируется строка для записи в файл. Она представляет из себя последовательность нужных переменных, отформатированных тэгами HTML. За счет этого нам дальше будет очень легко просматривать архив сообщений и выводить на экран нужный промежуток (если количество сообщений превысит сотню, вы это оцените – очень удобно поставить ссылку, и смотреть сообщения с 50-го по 80-е, например). После того, как строка для записи подготовлена, она записывается в файл. Дальше – вывод результатов записи.

Обратите внимание, что в первый раз, когда посетитель попадает на страничку гостевой, идет обработка сразу этого кода, так как переменные имени и сообщения пусты.

```
$lines = file($files);
$a = count($lines);
$u = $a - $qq;
for($i = $a; $i >= $u ;$i--) { echo $lines[$i]; }
?>
```

Тут все очень просто. В массив считывается файл сообщений, и в цикле выводятся на экран его содержимое. Если количество сообщений превысило наше ограничение, они просто не показываются. Причем – новые сообщения всегда вверху, около формы для вода, так как вывод идет снизу вверх по индексу массива. Это очень удобно, но при желании может быть изменено.

Вот и все! В файл для сообщений ничего записывать не нужно – он будет заполняться по мере ввода данных. Архив сообщений всегда будет доступен, если вы поставите ссылку **архив**. Нужно заметить, что код не учитывает ввод посетителем тэгов HTML. Но это уже несколько иная задача, и сделать это легко, если только нужно. Использоваться этот скрипт может в любом месте, где нужно узнать мнение посетителей например, о статье, о новостях и т.д. Так же это – простейший форум.

Глава 6. Гостевая книга на PHP — еще один вариант

Обычно гостевая книга состоит из двух частей: первая часть выводит записи, а вторая добавляет их.

В рассмотренном ниже примере гостевой книги каждая запись хранится в отдельном файле. Имя файла создается автоматически, и состоит из двух частей — префикса и уникального идентификатора. Пре-

фикс нам необходим для того, чтобы отличить наш файл, от любого другого находящегося в том же каталоге (Честно говоря, по хорошему, в том же каталоге больше ничего не должно находиться), а уникальный идентификатор необходим для того, чтобы отличить одну запись от другой, и предоставить возможность сортировки записей.

В нашем случае префикс будет — «rec», а уникальный идентификатор мы получим с помощью функции **time()**. Функция **time()** возвращает текущее время, измеренное в числе секунд с эпохи Unix (1 Января 1970 00:00:00 GMT). Конечно, абсолютно уникальное число мы с помощью этой функции мы получить не сможем, но она нас устроит.

Сначала мы рассмотрим исходный текст модуля добавления записи в гостевую книгу.

Файл add.php

В переменной $err мы будем хранить сообщения о произошедших ошибках. Если переменная пуста, то ошибок не происходило.

```
<?
$err="";
```

Если переменная **$action** не пуста, то значит происходит обращение при котором необходимо добавить запись, иначе необходимо просто вывести форму добавления записи. Переменная **$action** задается в форме с помощью тэга **<input type=hidden>**

```
$action=$HTTP_POST_VARS["action"];
if (!empty($action)) {
```

Первым делом необходимо проверить введено ли сообщение и указан ли его автор.

```
$name=trim($name);
$msg=trim($msg);
if (empty($msg)) {$action="";$err=$err."<LI>Вы не ввели сообще-
ние\\n";}
if (empty($name)) {$action="";$err=$err."<LI>Вы не ввели имя\\n";}
```

После этого мы должны осуществить преобразование введенных данных, проверить их длину. Слишком длинные записи могут сильно испортить внешний вид страницы. Необходимо убрать слэши «\\» перед кавычками и заменить специальные символы HTML.

```
$name=substr($HTTP_POST_VARS["name"],0,32);
$name=htmlspecialchars(stripslashes($name));
$email=substr($HTTP_POST_VARS["email"],0,64);
$email=htmlspecialchars(stripslashes($email));
$www=substr($HTTP_POST_VARS["www"],0,64);
```



```
$www=htmlspecialchars(stripslashes($www));
$city=substr($HTTP_POST_VARS["city"],0,64);
$city=htmlspecialchars(stripslashes($city));
$msg=substr($HTTP_POST_VARS["msg"],0,1024);
$msg=htmlspecialchars(stripslashes($msg));
```

Если не произошло ошибок, то можно заменить специальные символы (такие как **[b]**, **[i]**, **[u]**) на их HTML аналоги (**<b>**, **<i>**, **<u>**):

```
if (!empty($err)) {
$msg=nl2br($msg);
$msg=str_replace("[u]","<u>",$msg);
$msg=str_replace("[i]","<i>",$msg);
$msg=str_replace("[b]","<b>",$msg);
$msg=str_replace("[/u]","</u>",$msg);
$msg=str_replace("[/i]","</i>",$msg);
$msg=str_replace("[/b]","</b>",$msg);
```

С помощью регулярного выражения заменим запись типа **[url] http://www.codenet.ru/[/url]** на **http://www.codenet.ru/**:

```
$msg=eregi_replace("(.*)\\\\[url\\\\](.*)\\\\[/url\\\\](.*)",
"\\\\1<a href=\\\\2>\\\\2</a>\\\\3",$msg);
$msg=str_replace("\\n"," ",$msg);
$msg=str_replace("\\r"," ",$msg);
```

Теперь данные готовы к сохранению в файл. Файлы с записями у нас будут храниться в каталоге ./records.

```
$fp=fopen("records/rec.".time(),"w");
fputs($fp,$name."\\n");
fputs($fp,$email."\\n");
fputs($fp,$city."\\n");
fputs($fp,$www."\\n");
fputs($fp,$msg."\\n");
fclose($fp);
```

Все действия произведены, и мы можем смело отправить пользователя на главную страницу гостевой книги, где он сможет увидеть свою запись.

```
print "<HTML><HEAD>\\n";
print "<META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php'>\\n";
print "</HEAD></HTML>\\n";
}
}
```

Если переменная **$action** пуста, то выводим форму добавления записи:

```
if (empty($action)) {
?>
<HTML>
<HEAD>
<TITLE>Гостевая книга – добавить запись</TITLE>
</HEAD>
<BODY>
<?
if (!empty($errro)) {
print "<P>Во время добавления записи произошли следующие ошибки:
</P>\\n";
print "<UL>\\n";
print $err;
print "</UL>\\n";
}
?>
<H3>Добавление записи.</H3>
<center>
<table width=1 border=0>
<form action=add.php method=post>
<input type=hidden name=action value=post>
<tr><td width=50%>Имя<font color=red><sup>*</sup></font>:</td>
<td align=right>
<input type=text name=name maxlength=32 value='<? echo $name;
?>'></td></tr>
<tr><td width=50%>E-Mail:</td>
<td align=right>
<input type=text name=email maxlength=64 value='<? echo $email;
?>'></td></tr>
<tr><td width=50%>Домашняя страница (WWW):</td>
<td align=right>
<input type=text name=www maxlength=64 value='<? echo $www;
?>'></td></tr>
<tr><td width=50%>Город:</td>
<td align=right>
<input type=text name=city maxlength=64 value='<? echo $city;
?>'></td></tr>
<tr><td colspan=2>Сообщение<font
color=red><sup>*</sup></font>:<br>
<textarea cols=50 rows=8 name=msg><? echo $msg;
?></textarea></td></tr>
```



```
<tr><td colspan=2><input type=submit value='Добавить'></td></tr>
</form>
</table>
</center>
<P>Используйте размету для [b]<B>выделения текста</B>[/b]
и вставки [url]гиперссылок[/url] </P>
</BODY>
</HTML>
<?
}
?>
```

Теперь рассмотрим модуль, ответственный за вывод записей гостевой книги.

Файл index.php

```
<HTML>
<HEAD>
<TITLE>Гостевая книга</TITLE>
</HEAD>
<BODY>
<?
```

Первым делом, с помощью объекта dir, считаем содержимое каталога, в котором у нас хранятся записи. Все идентификаторы записей мы сохраним в массив, для его последующей сортировки.

```
$d = dir("records");
$i=0;
while($entry=$d->read()) {
if (substr($entry,0,3)=="rec") {
$names[$i]=substr($entry,4);
$i++;
}
}
$d->close();
```

Сортируем массив:

```
@rsort($names);
$count=$i;
$cnt=$count;
if (empty($start)) $start=0;
$start=intval($start);
if ($start<0) $start=0;
```

Выводим ссылки навигации по гостевой книге «Предыдущие» и «Следующие»:

```
print "<center>";
if ($count>$start+10) $count=$start+10;
if ($start!=0)
print "[ <A href=index.php?start=".($start-10).">Предыдущие</A>
]";
print " [ <a href=add.php>Добавить запись</A>] ";
if ($cnt>$start+10) {
print "[ <A href=index.php?start=".($start+10).">Следующие</A>
]\\n";
print "</center><br>";
```

Теперь самое главное — считываем нужные нам записи и выводим их.

```
for ($i=$start;$i<$count;$i++) {
$entry=$names[$i];
$data=file("records/rec.".$entry);
$date=$entry;
$name=trim($data[0]);
$email=trim($data[1]);
$city=trim($data[2]);
$www=trim($data[3]);
$question=trim($data[4]);
$answer=trim($data[5]);

print "<table border=0 cellspacing=0 cellpadding=2 width=100%>";
print "<tr bgcolor=#F0F0F0><td> ";
if (!empty($email)) print "<a href=mailto:$email>$name</A>\\n";
else print $name;
if (!empty($www)) print "[<a href=$www>$www</A>]";
print "</td><td align=right>".date("H-i-s <b>d-m-Y</b>",$date);
print "</td></tr>\\n<tr><td colspan=2>\\n";
print "<P>".$question."</P>\\n";
if (!empty($answer)) print "<P><B><I>$answer</I></B></P>\\n";
print "</td></tr></table>\\n<br><br>\\n";
}
```

Опять выводим ссылки навигации по гостевой книге «Предыдущие» и «Следующие»:

```
print "<center>";
if ($start!=0)
print "[ <A href=index.php?start=".($start-10).">Предыдущие</A>
]";
print " [ <a href=add.php>Добавить запись</A>] ";
if ($cnt>$start+10)
```



```
print "[ <A href=index.php?start=".($start+10).">Следующие</A>
]\\n";
print "</center>";
?>
</BODY>
</HTML>
```

Глава 7. Графический счетчик на PHP

Создавая текстовый счетчик, мы ограничены свойствами текста в браузере. Если же вы хотите чего-то из ряда вон выходящего, удовлетворяющего вашему полету фантазии, то вам подойдет как раз описываемая тема.

Данный пример демонстрирует работу простого графического счетчика. По функциональности он совершенно идентичен текстовому счетчику.

Для работы этого счетчика необходимо создать графическое изображение, которое послужит базой для счетчика.

Это изображение надо сохранить под именем counter.png

В скрипте использована библиотека GD, перед тем как пользоваться этой библиотекой, узнайте у хостера, подключена ли эта библиотека. Теперь сам скрипт:

```
<?php
$dat_file="counter.dat"; // Файл счетчика
$log_file="counter.log"; // Файл списка IP
// Открываем файл счетчика и считываем текущий счет
// в переменную $count
$f=fopen($dat_file,"r");
$count=fgets($f,100);
fclose($f);

$count=ereg_replace(" ","",$count); // Удаляем символ конца
строки
$count++; // Увеличиваем счетчик
// Записываем данные обратно в файл
$f=fopen($dat_file,"w");
fputs($f,"$count ");
fclose($f);
```

```
// Создаем новое изображение из файла
$im = ImageCreateFromPNG('counter.png');
// Назначаем черный цвет
$black = ImagecolorAllocate($im,0,0,0);
// Выводим счет на изображение
Imagestring($im,1,5,20,$count,$black);
// Выводим изображение в стандартный поток вывода
Header("Content-type: image/png");
ImagePng($im);

// Записываем IP посетителя
$f=fopen($log_file,"a+");
$ip=getenv("REMOTE_ADDR");
fputs($f,"$ip ");
fclose($f);
?>
```

Для работы этого скрипта необходимо создать два файла, для ведения счета и для списка IP. В файле счета необходимо установить начальное значение счетчика, сделать это можно в любом текстовом редакторе.

Для вывода счетчика в html используйте:

```
<IMG SRC="counter.php" WIDTH="88" HEIGHT="31" BORDER=0>
```


Часть 7. Life Site CMS — система создания и развития сайтов

Глава 1. Введение

Life Site — это система, призванная минимизировать временные и финансовые затраты на создание и управление сайтами.

Система включает в себя все типовые функции системы управления контентом (Content Management System или сокращенно CMS).

Life Site CMS предоставляется в аренду вместе с хостингом. Благодаря этому, пользователи не занимаются решением технических проблем со своими сайтами. За них это делает Life Site. Все что остается делать это создавать дизайн и наполнять сайт материалами. При этом совсем не обязательно быть специалистом в программировании.

Основные преимущества перед другими CMS:

- **Доступность**. Стоимость аренды Life Site чуть больше стоимости профессионального хостинга. Аналогичные продукты конкурентов стоят тысячи долларов. Плюс ко всему, Life Site не требует привлечения специалистов для внедрения. Вы просто присылаете заявку и в течении нескольких дней ваш сайт начнет работать.
- **Русский интерфейс**. Система полностью русифицирована, имеет полную документацию на русском языке и примеры решения типовых задач.
- **Полная предварительная компиляция**. Все страницы создаются во время создания сайта, а не во время обращения посетителя к странице. Что позволяет значительно увеличить скорость работы с сайтом.

- **Нет ограничений на дизайн**. Система не накладывает никаких ограничений на внешний вид сайта. Вам предоставляется полная свобода!
- **Скорость обучения**. Система очень проста и понятна в использовании. Вам не потребуется больших временных затрат на обучение работе с системой.

Система постоянно совершенствуется. Работая с Life Site, вы всегда будете обладать не устаревшей программой, а новейшей CMS. И это не потребует никаких действий с вашей стороны.

Глава 2. Что такое CMS?

CMS — это аббревиатура от Content Management System, что в дословном переводе — Система Управления Содержимым (сайта). Иногда CMS называют «движок» сайта (site engine).

Основная функция CMS — автоматизация процесса модернизации сайта, управления его содержимым, добавления, удаления и редактирования страниц. Некоторые CMS, такие как Life Site, еще и упрощают процесс создания сайта, позволяют легко расширять его функциональность, менять дизайн, добавлять новые интерактивные модули (ленты новостей и гостевые книги и т.д.).

А зачем, собственно, сайт?

Сайт это самый эффективный и удобный способ донести до клиента информацию о предприятии, его деятельности, услугах и товарах. Если мы используем сайт для этого, нам нужно чтобы информация на нем была полной, удобно представленной, а главное, актуальной. Понятно, что потенциальный клиент, зайдя на сайт и увидев явно неактуальную информацию, постарается не иметь дел с этой фирмой. Поэтому, если вы хотите, чтобы сайт приносил вашей компании отдачу, система управления контентом вам просто необходима.

Некоторые компании нанимают внештатных сотрудников или регулярно платят Web-Студиям за обновление сайта. Кроме того, что это обходится им в несколько сотен долларов, получается, что сайтом управляет человек, плохо знакомый с деятельностью фирмы. Курировать вопросы, связанные с web-представительством, должен сотрудник кампании хорошо знакомый с ее деятельностью и маркетинговой политикой.

Часто ли используются CMS?

Практически все западные компании используют CMS-ы для управления своими Web-Ресурсами. В России же ситуация несколько другая. Основных причин несколько:

- Во первых, многие компании до сих пор не считают Web-Сайт значимой составляющей бизнеса. Выделяют деньги на сайт по остаточному принципу и совсем не занимаются его обновлением и улучшением.
- Во вторых, отсутствие выгодных предложений на российском рынке CMS. Основная часть систем является собственными наработками web-студий и, часто, требует квалифицированного технического персонала для поддержки. Таким образом, Web-Студии навсегда привязывают к себе клиентов. Остальные CMS либо не являются продуктами вообще, либо стоят тысячи, а то и десятки тысяч долларов, что делает их использование экономически нецелесообразным для многих предприятий.
- В третьих, невозможность использования западных CMS из-за отсутствия документации и поддержки на русском языке и все той же стоимости.

Life Site призвана устранить последние две причины, а первая, со временем, станет не актуальной.

Зачем CMS Web-студиям?

Использование таких CMS, как Life Site, выгодно не только компаниям, занимающимся только собственным web сайтом, но и web-студиям, так как позволяет значительно снизить издержки на создание сайтов. Опытный web-мастер может создавать сайты на базе Life Site за 2-3 дня. И после этого не надо будет возится с верстской 1000 и 1-й страницы. Клиент сам будет добавлять и редактировать материалы на своем сайте. Вы же будете заниматься тем, что у вас получается лучше всего — дизайном, анализом удобства, рекламой и др. И даже, если добавлением и версткой всего материала будете заниматься вы, делать это будет гораздо удобнее.

Кроме того, использование общедоступной и известной CMS, а не продукта собственной разработки, предоставляет web-студии дополнительное конкурентное преимущество.

Наглядный пример удобства CMS: допустим, у вас есть сайт, работающий на CMS Life Site, и вы хотите добавить на сайт новую страницу. Заходим в административный модуль. Сразу перед вами структура сайта, а внизу форма добавления новой страницы.

- Вводим имя будущей страницы.
- Нажимаем «добавить».

Все! Страница готова! Ее внешний вид полностью соответствует дизайну сайта и ссылка на нее добавлена в меню. Вам только останется зайти в редактирование этой страницы и написать текст. В последствии вы сможете его также легко поменять, сможете легко поменять ее внешний вид, настроить элементы HTML заголовка и многое другое. Все это за пару нажатий мышкой!

Глава 3. Функциональность системы

Структура сайта

Полная структура сайта удобно представлена, в виде дерева. Можно добавлять, удалять и изменять страницы сайта. Для каждой страницы можно: изменить содержание, добавить один из интерактивных модулей, изменить внешний вид, настроить элементы HTML заголовка. Все делается в 1-2 нажатия мышкой.

Интерактивные модули

На данный момент доступны следующие модули:

- Система публикации новостей
- Система размещения банеров
- Форум
- Гостевая книга
- Система поиска по сайту
- Карта сайта

В ближайшем будущем будут внедрены:

- Каталог продукции
- Каталог ссылок

- Картинная галерея
- Форма отсылки сообщения на E-mail (SMS, ICQ...)
- Система публикации вакансий
- и другие интерактивные компоненты

Порядок выпуска и функциональность модулей может быть изменена по просьбам пользователей.

Дизайн сайта

Удобная система шаблонов позволяет настраивать внешний вид любой части сайта. При этом не потребуется никаких специальных знаний. Если же вы не предъявляете особых требований к дизайну, вам будет достаточно сделать всего 1 шаблон!

Меню сайта

Генерируется автоматически по структуре сайта. Для каждой страницы можно указать, отображать ее в меню или нет. С помощью гибкой системы шаблонов можно добавить на сайт меню любого типа, а также легко изменить его внешний вид.

Для удобства в примерах доступны следующие типы меню:

- Полностью раскрытое.
- Список разделов верхнего уровня.
- Выпадающие меню (реализовано с помощь Java Script)

Система публикации новостей

Позволяет гибко планировать размещение новостей на сайте. При отображении, производится сортировка новостей по дате и по приоритету. Для каждой новости можно указать период на который она будет отображаться. Можно создавать несколько новостных лент. Поддерживается архив новостей.

Система размещения банеров

Позволяет размещать банеры(любые куски HTML кода) в произвольных местах сайта. Для каждого банера можно задавать различные значения, регулировать частоту показа и ограничивать число показов этих значений. У вас будут самые объективные цифры о количестве показов банеров.

Статистика

Ведется полная статистика посещаемости ресурса. Регистрируется дата и время посещения, IP адрес и Proxy адрес, ссылающиеся страницы и пути по сайту, информация о браузере и операционной системе клиента. Вся информация просматривается с помощью удобной системы фильтров. В отличии от бесплатных счетчиков, сохраняется информация о каждом хите. Вы можете проследить каждый шаг каждого посетителя.

Система авторизации

Позволяет настраивать доступ к различным компонентам управления сайтом и к самому сайту.

Примеры сайтов работающих на базе Life Site

LifeSite.ru

Презентационный сайт системы Life Site. Полностью сделан на базе технологии LifeSite.

YourPage Solutions

Компания разработчик программного обеспечения для Интернет.

Free Mind

Официальный сайт Центра Медицинской Профилактики Наркологических Заболеваний. Санкт-Петербург.

SexyLife.ru

Популярный сексуально-развлекательный портал.

Глава 4.
Часто задаваемые вопросы

Что такое CMS?

CMS – Content Management System – это система управления сайтом.

Life Site CMS – это движок?

Да. Слово «движок» — это популярный синоним для CMS систем. В английском варианте используется слово engine.

Чем отличается от хостинга?

Хостинг – это услуга размещения сайта в сети Интернет. Life Site CMS – программный комплекс, предназначенный для удобного создания и управления сайтом. Для удобства наших пользователей, арендующих Life Site CMS, система поставляется вместе с хостингом. Хостинг нам предоставляет одна из лучших серверных площадок Рунета, отобранная на основе жесткого тендера.

Чем Life Site CMS отличается от PHP-nuke?

PHP-nuke — это Content Management System, реализованная на зыке программирования PHP, также как и Life Site CMS. Основные отличия:

1. Система поставляется в виде набора PHP скриптов, для установки которых требуется привлечение программиста со знанием PHP и MySQL.

2. Вам придется решать все проблемы с хостингом (размещением сайта в Интернет).

3. У системы PHP-nuke полностью отсутствует служба поддержки и документация на русском языке.

Систему PHP-nuke можно порекомендовать тем, у кого есть масса свободного времени, желание углубить знания PHP и MySQL и, в дальнейшем, заняться разработкой собственной CMS.

Как будет выглядеть мой сайт?

Так, как вы захотите. Система Life Site не накладывает никаких ограничений на внешний вид сайта. Вы можете перенести уже готовый сайт на Life Site CMS, самостоятельно сделать дизайн сайта, или заказать его в любой web-студии.

У меня уже есть работающий сайт, но мне понравилась ваша система, и я хотел бы перенести на нее свой сайт. Что для этого требуется?

Перенести работающий сайт на Life Site CMS очень просто, если вы владеете минимальным знанием HTML. Если нет, это можем сделать мы или любая web-студия. Исключение составляют сайты, с нестандартной функциональностью.

Часть 8. «Раскрутка» сайта

Глава 1. Выгодность хорошей «раскрутки»

Нехорошая получается ситуация, когда сайт создан и, при том, очень не плохой сайт, но никто на него не заходит. С подобной ситуацией надо бороться всеми доступными способами. Но, прежде, чем перечислять возможные варианты, давайте разберемся: «А что именно скрывается под словом «раскрутка»? Раскрутка (в широком смысле этого слова) есть распространение информации о данном предмете в те слои, категории потенциальных покупателей/посетителей, которые ни сном, ни духом не знают ничего о вашем товаре/сайте/фирмы. Раскрутка — это нечто большее, чем реклама, как говорится, антиреклама — тоже реклама. А поэтому надо определиться, что именно раскручивать: сам товар или брэнд.

Возьмем, к примеру, такую ситуацию: у вас есть какой-либо эксклюзивный товар или товар самый обыкновенный, но он обладает рядом преимуществ (высокое качество, низкая цена и т.д.) — тут все очевидно, раскручивать необходимо товар со всей его интригующей информацией. Вторая ситуация: есть небольшая фирма/сайт, занимающаяся чем-то обыденным и, следовательно, имеющая немало конкурентов. Для начала вам нужно найти изюминку в вашем сайте или в деятельности вашей фирмы, которой позавидовали бы конкуренты. Затем придумайте завлекающий слоган (а лучше два: один короткий, а второй подлиннее) и красочный логотип. Именно такая смесь и будет вашим брэндом.

«В чем же выгодность раскручивания брэнда?» — спросите вы. А вот в чем: разместили вы, например, всю информацию на выделенное рекламное место (банер или текстовый блок), и просмотрело ее 1000 человек. Про эффективность самой банерной рекламы говорить не надо — в лучшем случае 2-3 клика вам «обеспечено». А в данном случае ваша деятельность направлена на запоминание слогана, логотипа, стиля и, конечно же, название сайта. Таких людей уже будет 20-30, а около половины из них впоследствии наберет ваш брэнд в Яndex или в адресной строке. Выгода очевидна.

Исходя из вышесказанного можно подвести черту и выделить золотое правило в поиске своего оригинального брэнда: находите свою изюминку и хватайтесь за нее руками и ногами, только желание победы приведет вас к искомому результату.

Что же касается способов рекламной раскрутки, то первый и самый, наверное, дотошный способ — это банерная реклама. Не самый эффективный и новый способ раскрутки. Второй способ проявляется покупке рекламного места на каком-нибудь сайте.

Способы раскрутки сайта

Раскрутка — занятие довольно долгое и утомительное, но необходимое. Раскручивайте полностью сделанный сайт, иначе только навредите.

Во-первых, выберите себе звучное, интригующее название из 3-4 слов, чтобы красиво смотрелось в браузере. Выберите короткий URL (если вы такого не имеете) с легко запоминающимся названием и отвечающий содержанию с помощью служб переименования. Думаю, адрес webs.web.com легче запоминается, чем адрес типа http://www.fortunecity.com/business/fax/339.

Во-вторых, необходимо подготовить свои странички для их успешного индексирования поисковыми машинами. Что это значит? Дело в том, что большинство поисковиков далеко не полнотекстовые, как это рекламируется. И какие слова с вашей страницы он отберет в свою базу данных — это еще вопрос. Потому-то и надо заранее готовить документ, чтобы на соответствующий запрос поисковой машине ссылка на вашу страницу находилась если не первой, то хотя бы в начале списка ссылок. Ничего сложного здесь нет. Просто в вашем документе необходимо указать «говорящий» заголовок (title), ключевые слова (keywords) и описание (description). Рассмотрим по порядку:

- **Title** — заголовок документа. Обязательно выводится при ответе поисковой машины на запрос. Хороший осмысленный заголовок может заставить пользователя из множества других выбрать именно вашу ссылку.
- **Keywords** — ключевые слова. Содержимое этого контейнера точно попадет в базу данных поисковика. Пример:

```
<META name="keywords" content="hosting, sponsor, money, free, proxy, promotion, home, script, java, cgi, page, site, without, banner, спонсор, прокси, анонимный, обман спонсоров, халява, раскрутка, сайт, бесплатно, скрипт, скрипты">
```

Советы по заполнению этого контейнера:

- в поле «**content**» не должно быть знаков конца строки, кавычек и других специальных символов, регистр символов роли не играет. Слова должны набираться в одну строку через запятую;
- длина поля «**content**» не должна превышать 1000 символов. Большее количество может быть при индексировании отброшено поисковым сервером;
- не рекомендуется повторять одни и те же ключевые слова по нескольку раз. Это может быть воспринято как спам, и ваша страница рискует вообще не попасть в индекс поисковой машины;
- можно использовать фразы. Это повысит ваши шансы попасть в самое начало списка, выданного поисковиком, в случае совпадения фразы с запросом пользователя;
- не стоит делать одинаковые keywords для разных страниц вашего сайта — содержание страниц-то разное;
- используйте в описании терминов как можно больше синонимов.

- **Description** — описание документа. Содержимое этого контейнера используется как краткое описание документа в ответе поисковой машины. Если этого контейнера нет, то выдается некоторое количество строк с начала документа. Соответственно, не редкая картина, когда в самом начале документа расположен Java Script, и вместо нормального описания выдается абракадабра в виде куска скрипта. Пример:

```
<META name="description" content="Практические рекомендации
по размещению вашего сайта на бесплатных серверах без
рекламы. Максимальная раскрутка вашего сайта. Обман
спонсоров. Постоянно обновляемый список анонимных прокси.">
```

Советы по заполнению этого контейнера:

- в поле «content» не должно быть знаков конца строки, кавычек и других специальных символов;
- длина поля «content» не должна превышать 200 символов. Лишнее может не попасть в индекс поисковой машины. А будет потерян конец описания, начало или все описание полностью — неизвестно.

Необходимо вставить свой вариант тэгов **META** в вашу страничку между </**TITLE**> и </**HEAD**>.

Некоторые поисковые серверы при индексации игнорируют содержимое тэга **META "keywords"**, но зато индексируют комментарии в HTML коде (это то, что находится между **<!--** и **-->**). Поэтому имеет смысл продублировать ключевые слова в комментарии:

```
<!-- hosting, sponsor, money, free, proxy, promotion, home,
script, java, cgi, page, site, without, banner, спонсор, прокси,
анонимный, обман спонсоров, халява, раскрутка, сайт, бесплатно,
скрипт, скрипты -->
```

Многие поисковики просматривают содержимое атрибута ALT тэга IMG (надпись, замещающая картинку на страничке). Поэтому этим атрибутом пренебрегать не стоит.

В наиболее распространенных поисковых машинах надо регистрироваться вручную, тот же Yahoo вообще не принимает регистрации автоматически от роботов:

- Infoseek — указываете только адрес сайта, все остальное сделает робот.
- AltaVista — робот сам обходит вашу страничку после регистрации заглавной. От вас требуется указать только адрес. Ключевые слова обязательно будут включены в индекс.
- Excite — сервер интересует только содержимое странички, не смотрит на ключевые слова. Время регистрации — 2 недели.
- HotBot — только адрес странички, никакого описания вводить не надо.
- Lycos — время регистрации 2-3 недели.
- Webcrawler — обещают зарегистрировать в течение 2 недель.
- Русские поисковики Rambler и Rambler Top100 дают счетчик. Кстати, Rambler не регистрирует страницы, размещенные на зарубежных бесплатных серверах (типа FortuneCity, GeoCities, Xoom и т.п.).

- List.ru — популярный поисковик и тематический каталог. При регистрации дают счетчик с подробной статистикой. Рейтинг по категориям. В отличие от Rambler принимают сайты с любых серверов.
- Yandex, Апорт — просто указываете URL своей странички и даете ее краткое описание.
- @rus (бывший Ау!) — процесс регистрации сопровождается подробнейшими инструкциями. Так что затруднений здесь возникнуть не должно. Регистрация будет длиться около месяца — ваш сайт будет просмотрен администрацией @rus. В случае положительного результата вас уведомят по e-mail.
- Stars.ru — каталог «Созвездие Интернет». Зарегистрировать здесь свой сайт очень престижно, и это будет говорить о его интересном содержании. Перед регистрацией ваш сайт обязательно инспектируется администрацией каталога.

На всех других поисковиках в целях экономии времени лучше регистрироваться при помощи специальных сайтов: Submit-It, Add Url, @Submit, Free Url Submission — в 13 поисковиках, Add Me!, ABS Easy Submit, Self Promotion, Add4Free — Your free submission, Submission Wizard, Broadcaster, 123Launch, Simple Submit — в 9 поисковиках, Swift Submit — в 24 поисковиках, URL Submit — в 12 поисковиках (интересно то, что поисковики считают регистрацию через данный сервис, как регистрацию вручную). Для русских поисковиков — TAU.

Следует заметить, что после регистрации на таких сайтах желательно через пару дней вручную пройтись по основным поисковикам. Например, AltaVista может попросту удалить ваш сайт из своего индекса, если он уже был зарегистрирован (это оговорено в «условиях регистрации на AltaVista», допускается лишь регистрация через Submit-It и Netcreations). Что, кстати, и случилось однажды с данным сайтом.

Совет (надеюсь, что полезный): при регистрации сайта в различных каталогах и поисковиках вам неоднократно придется набирать полный URL сайта, ключевые слова, краткое описание сайта. Чтобы не тратить напрасно время и силы, подготовьте текстовый файл, содержащий следующее:

- полный адрес вашего сайта, начинающийся с http://;
- название сайта на русском языке;

- название сайта на английском;
- ключевые слова, разделенные запятой (не более 1000 символов);
- те же ключевые слова, но разделенные пробелом (кое-где такие требования);
- краткое описание сайта на русском (не более 255 символов);
- краткое описание на английском (тоже не более 255 символов).

При регистрации вы можете переносить необходимое через Clipboard из этого файла в заполняемые формы!

Размещайте сообщения об открытии нового сайта на досках объявлений. Лучшим вариантом будет, если вы найдете доски объявлений близкие по тематике вашему сайту (автомобили, компьютеры и т.п.). А вообще, практически каждая доска имеет раздел «Разное». Мне лично очень понравилась доска объявлений «Барахолка на Куличках». Другие легко найти любым поисковиком — лишь наберите запрос «доска объявлений».

Существуют различные каталоги ссылок. Omen — популярный русский тематический каталог. Здесь можно разместить ссылку и на свой сайт. От вас потребуется только название сайта, его адрес и описание. Попросят разместить на сайте кнопку.

Еще каталоги:

- WebList.ru
- Куда Пойти
- Diamond Team.

Значительно повысить посещаемость сайта может участие в банерообменных сетях — вы показываете у себя банеры других сайтов, в свою очередь у них крутятся ваши банеры. При этом соотношение примерно следующее: на 100 показов на вашем сайте приходится 50-85 показов вашего банера на других (за счет этой разницы и существуют сами сети). Посещаемость вашего сайта во многом будет зависеть от банера, который вы себе сделаете. Вот пример — банерообменная сеть HITEXchange Banner Network. Поддерживаются всевозможные размеры банеров, различные форматы (CGI, ASP и JAVA скрипты, Shockwave FLASH, JAVA Applets, META Stream), статистика, таргетинг, каждому новому участнику дают 10 тысяч показов.

Еще банерные сети (русскоязычные):

- Russian LinkExchange Service (RLE)
- InterReklama
- Reklama
- BannerGold.

Обмен ссылками просто необходим. Ищите, договаривайтесь и меняйтесь — это взаимовыгодно. Если боитесь конкуренции, то выбирайте сайты, которые, по вашему мнению, не имеют того, что есть у вас (но похожей тематики). Можно сделать отдельную страничку для ссылок, но лучше договариваться размещать на заглавной. Будьте предельно вежливы — и возможно обменяетесь с сайтом, у которого трафик намного больше.

Глава 2. Как создать вирусный трафик с помощью бесплатных электронных книг

Нет-нет! Речь пойдет не о изрядно надоевших всем компьютерных вирусах. Вирусах, которые способны испортить настроение тысячам владельцам персональных компьютеров, и доставить извращенное удовлетворение какому-нибудь прыщаво-очкастому юнцу — создателю подобного творения.

Речь пойдет о совершенно другом виде вируса, от распространения которого в выигрыше остаются как «инфицированные» пользователи Интернета, так и создатели вируса. И имя этому вирусу — информация.

Для чего люди используют Интернет? Ответ очевиден. Для получения информации. Изо дня в день вы садитесь за компьютер, открываете окно браузера и... с головой погружаетесь в безграничное пространство Интернета в поисках контента, способного удовлетворить ваш информационный голод.

И как вы радуетесь, когда, наконец-то, удается обнаружить нечто по настоящему ценное, сайт, где содержатся ответы на вопросы, которые мучили вас на протяжении долгого времени. Охота удалась на славу!

А какое раздражение вас охватывает, когда следуя за рекламным банером, или текстовой ссылкой, вы попадаете на ресурс, на котором, как вам казалось, есть то, что вы искали, но вместо этого видите низкопробную подборку статей, скопированных с чужих сайтов, или еще чего хуже...

Успех любого сайта в Интернете во многом зависит от того, насколько качественный контент предлагает своим посетителям web-мастер. И одним из таких видов контента являются электронные книги.

Надо сразу же определиться. Под понятием электронные книги подразумевается не то безобразие, которое «творится» в библиотеке Мошкова — сборище текстовых файлов, называемые «книгами». Текстовый файл — это не более, чем текстовый файл. Он предназначен для редактирования документа, для чего угодно, только не для чтения!

Вы пробовали когда-либо читать книги в текстовом формате? Тогда вы очень хорошо понимаете, о чем здесь говорится. Настоящие электронные книги делаются в удобном для чтения и навигации формате. Удобном во всех смыслах: легкость навигации, комфортный и регулируемый размер шрифтов, приятный дизайн...

Конечно же, ни одна электронная книга не заменит столь привычный и милый сердцу бумажный вариант! Но ведь за бумажную версию надо платить, а электронные книги часто можно скачать себе на компьютер совершенно бесплатно!

То есть, еще одной фундаментальной предпосылкой является то, что в Интернете люди ищут бесплатную информацию. Слова «халява» и «бесплатно» способны привлечь к вашим сайтам сотни, тысячи посетителей! И опытные web-мастера с большим успехом используют эту идею.

Что, если на вашем сайте вы предлагаете своему посетителю совершенно бесплатно скачать удобную, интересную, и, самое главное, бесплатную электронную книгу? Уверен, что большинство посетителей воспользуются вашим предложением. Даже из любопытства. Особенно, если вы дали хорошее описание книги.

Предположим, они скачивают книгу, читают, она им нравится, и они начинают рекомендовать ее другим людям. Может быть, размещают ее на своем сайте, может быть, сообщают о ней в своих почтовых рассылках, на форумах, в конце концов, просто пересылают ее своим друзьям по электронной почте.

Другие люди читают эту книгу, она им нравится, они начинают рекомендовать ее своим друзьям и знакомым... снежный ком начинает стремительно расти в своих размерах. Геометрическая прогрессия в действии! Хорошая электронная книга способна в считанные недели распространиться в Интернете тысячами экземпляров.

Ну и что, спросите вы, а мне то какой с этого прок? Ну, помогу я автору «раскрутиться», а где же увеличение количества посетителей моего сайта?

Отвечу вопросом на вопрос. А что, если в этой электронной книге будет реклама вашего сайта?

Что если на каждой странице данной книги, читатели будут видеть название и описание вашего любимого ресурса, а также его web-адрес? Хорошие электронные книги не очень то хочется удалять со своего компьютера.

И их можно даже перечитывать, время от времени. И каждый раз, когда очередной читатель открывает свой экземпляр электронной книги, он видит вашу рекламу.

А что если в сети распространится сотни экземпляров е-книги с рекламой вашего сайта? А если тысячи, десятки тысяч? Уловили идею?

С помощью бесплатных электронных книг вы можете создать постоянно растущий трафик на ваш сайт. Вирусный трафик!

Программы для создания электронных книг

Хотим порекомендовать две программы для создания электронных книг, которые компилируют HTML файлы в один EXE-файл:

- Бесплатная: **SbookBuilder**. Лучшая программа для создания электронных книг из всех бесплатных, когда-либо виденных мною. Удобный и понятный интерфейс. Поддержка фреймов, возможность установить защиту от копирования и печати книги, возможность установления пароля для открытия книги. Поддержка CSS...
- Платная: **EBookCompiler**. А эта программа, в свою очередь, лучшая из всех платных. Она обладает невероятно широким спектром возможностей для создания электронных книг.

Рекомендация: Если вы хотите создавать электронные книги для бесплатного распространения по сети, вам вполне может хватить SbookBuilder. Если же вы хотите создавать информационные продукты для последующей продажи их через Интернет, то настоятельно рекомендуется воспользоваться EBookCompiler. Лучшее соотношение цены и качества!

Глава 3.
Правильная раскрутка проекта и привлечение нужных посетителей

Любой владелец любого Интернет-ресурса всегда мечтает об одном – привлечь внимание к своему проекту. Причем совершенно не важно, что это за проект – личная web-страничка, on-line магазин, бизнес-сайт, или крупный коммерческий проект.

Важно только одно – как привлечь, а впоследствии удержать посетителей.

На данный момент российская часть Интернета просто кишит предложениями «раскрутить до небес» абсолютно любой, даже самый безнадежный ресурс. И почти бесплатно. Более того, многие предлагают заработать на этом просто фантастические суммы (до 10000$). Почти бесплатно – это потому, что в большинстве случаев после регистрации выясняется, что вы должны активно поработать, поскольку именно от вашей активности будет зависеть продвижение вашего сайта. Некоторые «фирмы» назначают сумму взноса, который вы непременно должны сделать, чтобы начать рекламную кампанию. И web-мастера стараются, какое-то время активно работают на систему, которая не занимается абсолютно ни чем, кроме саморекламы. К сожалению, таких систем сейчас очень и очень много, а с каждым днем их становится все больше. И все больше сеть заваливается выпадающими окнами, рекламой «халявы» (сразу вспоминается «бесплатный сыр»), дешевой порнографией и прочим мусором, от которого в конечном итоге начинают уставать даже те, кто активно его создавал.

Для начала давайте разделим процесс проведения рекламной кампании на 4 основных этапа (этап создания самого сайта и его оптимизации сознательно пропустим, назвав его этапом номер 0).

Этап первый «Регистрация»

Необходимо зарегистрировать сайт в максимальном количестве каталогов, рейтингов и поисковых системах.

Что это даст:

- Бесплатные рекламные площадки, пусть маленькие, но действенные.
- Когда через месяц поисковые роботы обойдут каталоги и рейтинги (особенно небольшие), в которых «увидят» и вашу ссылку, рейтинг вашего проекта значительно

повысится, причем ровно настолько, насколько много различных ресурсов будут ссылаться на вас.

Для простоты и удобства, а также в целях экономии времени, рекомендуем воспользоваться системой авторегистрации 1ps. Ее отличительная черта от большинства подобных проектов – абсолютная «бесплатность», т.е. для работы в системе вам не придется платить, регистрироваться у спонсора, или привлекать рефералов. Достаточно только зарегистрироваться, заполнить анкету, выбрать нужные каталоги, рейтинги и поисковики и дождаться, пока система автоматически отправить вашу анкету по нужным адресам. На следующий день ваш почтовый ящик будет просто забит сообщениями об успешной регистрации вашего ресурса.

Этап второй «Взлет»

Второй этап желательно (хотя и необязательно) проводить сразу после первого, выждав 1-2 дня. Суть его состоит в том, чтобы привлечь максимальное количество посетителей (для первой недели достаточно «видимых»).

Что это даст:

- Если в первый же день после регистрации в каталогах и рейтингах у вас будет довольно большое количество посетителей (50-150 человек), ваш сайт автоматически попадет в категорию «лучшие новички недели», а это значительно прибавит его «вес» при оценке поисковиками. Да и само по себе очень хорошо попасть на одну из верхних строчек в каталог, поскольку новички с высоким рейтингом заставляют обращать на себя внимание большинства посетителей.

Из множества систем по привлечению посетителей мы смогли выбрать только одну – Rasrutim.ru. Проста в использовании, не требует привлечения рефералов, установки на ваш сайт своей рекламы, не берет комиссию с вашего счета (других подобных систем, не берущих комиссию за свою работу мы в Рунете так и не нашли). Предлагает два варианта своего использования:

- Приобрести, т.е. купить показы (читай — посетителей)
- Заработать показы, путем просмотра чужих страниц. Причем вам не придется самому лазить по Интернету со списком сайтов, которые нужно посетить, система сама будет вам их показывать. Для ленивых есть режим автораскрутки, позволяющий заниматься своими делами в

то время, как робот будет сам просматривать нужные страницы, начисляя вам тем самым показы.

Очевидно, что первый вариант больше подходит для крупных проектов, тогда как сайты поменьше (точнее их владельцы) скорее воспользуются вариантом номер два. В любом случае, накопить, или купить показы лучше заблаговременно.

Этап третий «Активная пропаганда»

В третий этап мы включили следующее:

- Обмен рекламой банерных и текстовых форматов (рекомендуем сети TBN, TBN Text, LBE)
- Реклама на форумах, в гостевых книгах, на досках объявлений и т.п. Для успешной рекламы подобного рода иногда бывает достаточно оригинальной темы сообщения, которая заставит обратить внимание на все сообщение или объявление целиком. Подобный вид рекламы, помимо прямого своего назначения, сильно увеличит индекс цитируемости и повысит авторитетность вашего ресурса в поисковых системах.

Для облегчения задачи и опять-таки значительной экономии времени, предлагаем воспользоваться программой Espanadir Clasific, позволяющей рассылать ваше объявление на почти полторы тысячи российских досок объявлений. Программа не бесплатна (~1500 руб.), но оправдывает вложенные в нее средства очень и очень быстро. Кстати, поиски подобных программ freeweare результатов не принесли.

Отличительные черты Espanadir Clasific – максимально простой и удобный интерфейс, файл справки на русском языке (что редкость), возможность добавления своих адресов, последующее бесплатное обновление базы данных. Просто незаменима при рекламе какого-то конкретного продукта, а заодно и ресурса, который его предлагает.

Этап четвертый «Целевая аудитория»

Любому крупному коммерческому проекту необходима аудитория, посещающая их сайт целенаправленно (купить, обновить, узнать точную информацию). Если первые три этапа были ориентированы на раскрутку и рекламу, то целью четвертого этапа является привлечение именно целевой аудитории, т.е. посетителей, интересующихся строго определенными товарами или услугами. Для этого нам понадобится целевая реклама, т.е. та, которая будет показываться только тем людям, которых она может заинтересовать. Для этого существует довольно много компаний, которые занимаются рекламой в тематических рассылках, на

сайтах конкретной тематики и т.п. Тот же Яndex, например, предлагает публиковать ваше объявление на страницах поиска. Но у всех подобных проектов есть ряд недостатков, главным из которых является ограниченное пространство для вашей рекламы. Второй по важности недостаток – это, как правило, показ вашего банера или объявления просто всем подряд. Это сильно увеличивает затраты на рекламу, поскольку по статистике, примерно 2-3, максимум 5 человек из ста вашей рекламой заинтересуются вообще, не говоря уж о том, что они захотят что-то приобрести. Коэффициент отдачи от такой рекламы составляет в лучшем случае 5%. Но, как оказалось, выход есть.

В российском Интернете существует единственная в своем роде система по привлечению целевой аудитории. Система Бегун. Принцип ее действия основан на том, что ваше рекламное объявление показывается на различных крупных сайтах и (самое интересное) на страницах поискового запроса, в самом верху списка найденных ресурсов (т.н. нулевая ссылка). Причем показывается ваша реклама не просто всем подряд, а исключительно людям, которые в строке поиска вписывают нужные вам слова. Т.е., если фирма по производству спортивных тренажеров при создании рекламной кампании указывает в качестве ключевого слова «велотренажер», то их реклама впоследствии будет показана всем, кто ищет в поисковых системах велотренажеры. Причем оплачивать фирма будет не саму рекламу, а переход по ней на сайт данной фирмы. Более того, цену за переход по каждой ссылке будет устанавливать опять-таки сама фирма, исходя из собственных возможностей и потребностей.

Бегун прошел весьма тщательную проверку (такой проверки не было ни у одной системы, которая подвергалась нашему тщательному исследованию) и доказал свое лидерство среди подобных проектов.

Совет: не копируйте чужую рекламную кампанию, создайте свою, неповторимую и оригинальную. В этом случае у вас будет гораздо больше шансов обратить внимание на свой проект.

Глава 4. Экономика проекта

Как сделать свой проект прибыльным? Ответ на этот вопрос пытаются получить большинство участников рынка Интернет-ресурсов.

Сложившийся факт: основная масса начинающих web-мастеров плохо представляет себе перспективы не только получения дохода, но и самоокупаемости проекта. Проект на бесплатном сервере (как показывает опыт полностью бесплатных серверов не существует в природе, — есть

лишь временно бесплатные) часто оказывается лишь утешением собственного самолюбия. Тем не менее, такие проекты необходимы. Это своеобразный полигон, на котором обкатывается как мастерство начинающих, так и дальнейшая перспективность самого ресурса. И более 80% интернет-сайтов проверки не выдерживают.

Можно сделать следующее заявление: большинство проектов в Рунете — убыточны. Авторы сайтов зачастую не обладают даже основами экономических знаний. Технические аспекты дизайна — вещь безусловно необходимая, но отнюдь не достаточная, — без экономики также не обойтись. И влияние экономических категорий для большинства проектов оказывается первостепенным. Итак, первое, с чего должны начинать авторы проекта, так это с вопроса: Какие цели преследует сайт? Только не нужно давать следующий ответ — «Сначала раскрутимся, а уж затем...». После «затем» вариантов может быть несколько — от «продадим...» до «будем зарабатывать на показах банеров...». Можно смело утверждать — такой сайт честными способами раскрутиться не сможет.

Далее, если вы не хотите нести убытки, необходимо выделить пути получения прибыли. Только не нужно опять про банеры. С их падающим CTR (своеобразная Интернет-инфляция), рассчитывать на то, что проект хотя бы окупится, не приходится.

Уменьшение CTR также делает невыгодным членство в банерных сетях. При CTR=1% (по нынешним меркам это высокий CTR), если вы не имеете другого дохода, работает следующая арифметика: для того, чтобы накопить 1000 банерных показов, вы должны показать примерно 1100 банеров на своем сайте. После размещения банеров в Интернете, привлекается 11 человек. Чтобы окупить свою рекламную компанию, нужно показать им каждому по 100 банеров.

То, что 100 банеров на посетителя — недостижимый показатель, это, надеюсь, вы понимаете. В этом примере не учитывалось количество посетителей из поисковых систем, каталогов, по ссылкам. С другой стороны не была также учтена стоимость времени на разработку, размещение и поддержание Интернет-сайта, — а это довольно внушительная цифра.

При анализе эффективности собственного проекта необходимо учитывать практически все (в том числе цену за электричество и амортизацию компьютерной техники).

Попытайтесь ответить на вопрос: Сколько стоит ваше время? Это очень важно, — Интернет-сайт требует именно ваших временных затрат.

Глава 5.
Стратегия эффективной работы с партнерскими программами

Начинать, разумеется, необходимо с выбора партнерских программ. Вот несколько моментов, на которые нужно обратить внимание.

- **Концентрация**: выбирайте небольшое количество партнерских программ. Две три — для начала хватит, не стоит распылятся на десяток-другой партнерских программ одновременно.
- **Соответствие тематик**: выбранные партнерские программы должны подходить к тематике вашего сайта. Если ваш сайт о поэзии, то не стоит ожидать от него отдачи от партнерской программы магазина бытовой техники. А вот участие в партнерской программе книжного магазина может дать неплохой эффект.
- **Солидность**: не забывайте, что отправляя посетителей через партнерскую ссылку на сайт рекламодателя, вы лично рекомендуете этот сайт. Поэтому желательно наличие на этом сайте приемлемого содержания, не очень страшного дизайна, а также домена и платного хостинга.
- **Мнения старожилов**: если партнерская программа существует не первый день, то значит у нее уже были участники. Поищите по сети высказывания людей о данной партнерской программе — это пригодиться.
- **Четкость правил**: на сайте рекламодателя должно быть подробно описано: за что и сколько платят, каким образом и когда можно получить заработанное, какие способы рекламы запрещены. Обратите внимание на существование нечетких формулировок вроде этой: «Вы получите до 1000$ с покупки привлеченного клиента!». Данное предложение можно «перевести» и так: «с продажи вы получите не более 1000$». Чувствуете разницу? Также неплохо, когда имеется пользовательское соглашение (terms&conditions), в котором все условия прописаны подробно, причем вам это соглашение стоит внимательно прочитать.
- **Поддержка**: задайте любой вопрос администратору этой партнерской программы. Если вам ответят, значит вы можете надеяться на дальнейшую помощь в урегулировании возникающих вопросов.

В общем случае, работа с партнерской программой ведется по следующей схеме:

- **Предстартовая подготовка**: регистрация в партнерской программе, установка html-кода банеров или ссылок на сайте. Не забудьте до конца разобраться, за что и сколько платят.
- **Начальная проверка**: сделайте оплачиваемое действие с целью проверки статистики и точности начисления денег. Выявите задержку между выполнением оплачиваемого действия и зачислением заработанного на баланс. Этот этап является желательным, но не обязательным.
- **Основная проверка**: наберите минимальную сумму, необходимую для выплаты заработанного. Получите денег. Цель данного этапа — получить уверенность в честности данной партнерской программы.
- **Повышение эффекта**: отслеживайте отдачу от партнерской программы и осуществляйте действия по повышению ее прибыльности.

Остановимся на последнем пункте поподробнее. Необходимо отслеживать и стараться улучшить как качественные, так и количественные параметры результата работы с партнерской программой:

Качественные параметры

CTR банеров или ссылок: процентное отношение количества переходов через партнерскую ссылку(банер) к количеству показов этой ссылки(банера).

Отслеживается статистическими механизмами партнерской программы и (или) своими средствами — скриптами-счетчиками или сторонними системами интернет-статистики.

Способы повышения CTR:

- изменение места размещения партнерского банера или ссылки на более лучший (выше к начала страницы, ближе к контенту)
- выбор более качественного содержания банера или ссылки

- более точный подбор тематики партнерской программы к тематике страницы или всего сайта.
- таргетинг показываемых банеров или ссылок.

CP посетителей — коэффициент исполнения действий. Вычисляется делением количества совершенных «требуемых» (оплачиваемых рекламодателем) действий посетителями к количеству заходов на сайт рекламодателя.

Низкий CP говорит о том, что посетители не делают оплачиваемые рекламодателем действия.

Способы повышения CP:

- более точный подбор тематики рекламодателя к тематике страницы или всего вашему сайту.
- наличие описания того, что нужно делать на сайте рекламодателя посетителю. Возможно даже создать отдельную промо-страницу или даже целей под-сайт.
- таргетинг показываемых банеров или ссылок.

CPM — отдача от партнерской программы в $$ на 1000 показов партнерских банеров и ссылок. Показатель эффективности данной партнерской программы на вашем сайте.

Количественные параметры

Показы: Количество просмотров вашими посетителями рекламы (в тысячах) за данный период времени.

Увеличение количества показов достигается:

- при увеличении посещаемости сайта,
- увеличение объема контента сайта.

Клики: Количество нажатий на банер или ссылку рекламодателя за данный период времени.

Увеличение количества кликов достигается при увеличении CTR банеров и ссылок. Тестируйте разные форматы и виды банеров, меняйте текст текстовых ссылок. Меняйте расположение рекламы.

Действия: Количество сделанных посетителями оплачиваемых действий (покупки) за данный период времени. Для увеличения количества действий «готовьте» посетителей к посещению сайта партнерской программы — разместите краткое описание, что он должен там делать, чтобы добиться обещанного результата.

Поступления денег: сумма, зачисленная на баланс за данный период времени.

Это — основная цель участия в партнерской программе.

Обратите внимание, что качественные характеристики, в отличии от количественных, имеют предел наращивания — скажем CTR не может быть более 100%. Нужно добиться хотя бы приемлемых показателей...

Сохраняйте эти характеристики в файлы за каждый выбранный промежуток времени (через каждую неделю, месяц) для возможности их сравнения с целью выявления эффекта от проведения каких-либо мероприятий.

Глава 6. Преимущества использования Интернета в сетевом маркетинге

XXI век — это век высоких технологий. И, хотим мы этого или нет, но Всемирная Паутина занимает все более прочные позиции в нашей жизни. В том числе, и в нашем бизнесе. Если еще два года назад дистрибьюторов, использующих Интернет в сетевом бизнесе, можно было насчитать всего несколько десятков, а сайты на тему сетевого маркетинга сосчитать на пальцах одной руки, то сегодня картина изменилась кардинальным образом. Сегодня сотни и сотни сетевиков имеют свои web-представительства. Один за другим открываются общеиндустриальные сайты. Сетевики начали понимать преимущества, которые дают им высокие технологии. Они начали осознавать, что Интернет — это мощный инструмент ведения бизнеса, потенциал которого еще до конца не осознал никто! Еще недавно Интернет казался многим «страшным зверем». Но сетевики, с их богатым опытом ведения бизнеса через выстраивание близких отношений, могут (и уже делают это!) укротить Интернет и сделать его послушным и надежным другом. Превратить бездушное железо в инструмент одухотворенных отношений! На смену технофобии приходят новые технологичные способы выстраивания долговременных дружеских и партнерских связей.

Сетевой маркетинг — это искусство выстраивание отношений, путем личных контактов.

Сетевой маркетинг в Интернете — это искусство выстраивания отношений с помощью высоких технологий.

Союз сетевого маркетинга и Интернета это неизбежная тенденция, которая невероятным образом преобразит нашу индустрию. Позволит каждому обычному человеку добиваться неординарных результатов! И это не какое-то там «отдаленное будущее». Это уже реальность сегодняшнего дня. Уровень развития русскоязычного Интернета уже сегодня позволяет успешно делать бизнес с помощью Интернета. Уже сегодня есть пионеры, которые первые вступили на terra incognito и могут поделиться своим опытом с теми, кто идет вслед за ними. Уже появились сетевые компании, которые используют в своей деятельности методы и технологии ведения бизнеса Новой Волны. Пройдет еще совсем немного времени и в Рунете произойдет настоящий «Взрыв». Появятся десятки компаний, предлагающих интернет-продукты и использующие в своей деятельности высокотехнологичные инструменты. Тысячи, десятки тысяч сетевиков ринутся осваивать «непаханную целину» Виртуального пространства. Это будет Прорывом. Точкой взрывообразного роста. Переворотом в сетевой индустрии!

Это будет, хоть и очень скоро, но завтра. А сегодня... сегодня предлагаем вам обратить свой пристальный взор в сторону Интернета. Взглянуть на те возможности, которые он может предоставить каждому из вас. Открыть для себя решение тех возможных проблем, с которыми вы сталкиваетесь в своей повседневной бизнес-деятельности. Ниже предпринята попытка перечислить для вас основные причины — почему необходимо использовать Интернет в вашем бизнесе. На самом деле их больше.

Итак, преимущества (причины) использования Интернета в сетевом маркетинге:

1. Гораздо легче использовать web-сайт, чем проводить «очную презентацию» вы не тратите свое время за рутинной работой — ваш сайт делает это...

2. Использование сайта позволяет кандидатам быстрее начать действовать — они смогут увидеть картину целиком и понять, как использовать свой web-сайт для бизнеса. Теперь, они готовы действовать!

3. Процесс обучения дистрибьюторов становится гораздо проще. Через свой сайт вы доступны всем своим дистрибьюторам с помощью «закрытого доступа», электронной почты, чатов или дискуссионных групп. Позвольте Сети делать работу за вас! В Интернете становится легче общаться с членами вашей организации, проживающих в различных часовых поясах — вы экономите деньги на телефонных переговорах, можете позволить себе высыпаться!

4. Благодаря Интернету, вам не нужно будет тратить деньги на пересылку материалов по почте (включая и стоимость самих материалов)!

5. Позвольте Интернету делать большую часть вашей работы. Нет необходимости тратить свое время на ответы кандидатам или объяснения трудных для понимания аспектов вашего бизнеса (например, план вознаграждений). Все, что теперь нужно сделать, это просто направить кандидата на определенную страницу вашего сайта, на которой он сможет получить ответ на свой вопрос в любое время суток!

6. Интернет позволяет вам автоматизировать существенную часть вашей рутинной бизнес-деятельности. У вас останется больше времени на планирование и творчество!

7. Ваш сайт гораздо легче дуплицировать, чем вас самих!

8. Вы можете позволить себе тратить меньше времени на телефонные разговоры и больше времени на привлечение людей к вашему сайту!

9. Интернет работает значительно быстрее, чем телефонные переговоры и бумажная почта!

10. Посредством Интернета гораздо проще шаг за шагом информировать о ваших возможностях и продуктах потенциальных кандидатов и клиентов. Все, что вам нужно, это создать автоматизированную систему поэтапного он-лайн информирования, использовать ее, и она будет превосходно работать на вас 24 часа в сутки, 365 дней в году!

11. Интернет, по сравнению с другими медийными средствами, позволяет легко выходить на различные целевые аудитории. Вы можете управлять рекламным процессом. При этом, ваши затраты на рекламу могут практически равняться нулю!

12. Более эффективно тратить свое время на привлечение людей к просмотру вашей web-презентации, чем пытаться рассказать о бизнесе по телефону.

13. Интернет делает процесс спонсирования на больших расстояниях гораздо легче — временные зоны, тарифы на телефонные переговоры больше не проблема!

14. Вам не нужно будет давать своим кандидатам презентационные материалы, которые они вам потом могут не вернуть...

15. Вы сберегаете свои деньги на печатание материалов — все на вашем сайте, и вам нет необходимости инвестировать кучу денег в рекрутинговые материалы (проспекты, брошюры, аудио и видеокассеты)!

16. Поиск новых кандидатов в Интернете быстрее выведет вас за пределы «теплого» рынка! У вас больше не будет возникать вопроса: «Где взять людей?»

17. С помощью Интернета гораздо проще проводить «сортировку» потенциальных кандидатов. После того, как кандидаты посмотрят ваш сайт, они уже точно будут знать, что представляет собой ваш бизнес, и хотят ли они к нему присоединиться.

18. Теперь вы тратите свое драгоценное время на телефонные переговоры только с заинтересованными кандидатами. Вы вступаете в переговоры только с теми, кто уже изучил все, что необходимо знать для принятия решения, и они готовы начать действовать!

19. Ваш кандидат увидит, как легко делать бизнес с помощью web-сайта — это путь, по которому он нашел вас... Дупликация в действии!

20. Позвольте Интернету работать за вас с сомневающимися кандидатами. Просто «зарядите» на своем сайте e-mail автоответчик, и вы сможете быть в контакте со своими кандидатами на протяжении 5-ти, 10-ти, 30-ти или даже 90 дней!

21. В Интернете вы имеете возможность вступить в контакт с гораздо большим количеством людей, за меньшее количество времени, чем в «реальной» жизни.

22. Одновременно вашу web-презентацию могут смотреть сотни людей в любое время суток!

23. У вашего web-сайта никогда не бывает плохого настроения. Ваши кандидаты всегда будут видеть одну и ту же великолепную презентацию!

24. Вы не услышите «нет», брошенное вам в лицо.

25. Ваш сайт рассказывает о вашем бизнесе 24 часа в день, 7 дней в неделю, 365 дней в году!

Глава 7. Как получить более 20 000 посетителей в день на свой сайт?

Вы ищете бесплатный способ увеличить посещаемость вашего web-сайта? Существует совершенно бесплатная система обмена ссылками. Система очень проста, но при этом является очень эффективным инструментом повышения посещаемости вашего сайта. Коэффициент об-

мена 1:1. Показывая на одной из страниц (специально выделенной) своего сайта 9 текстовых ссылок, вы спустя некоторое время обнаружите, что ссылка на ваш сайт стоит, как минимум, на 50 000 страниц или сайтов сети!

Инструкции:

1. Используйте ваш браузер (пункт меню «**Сохранить как...**»), чтобы скопировать эту страницу (одновременно скопируется и код страницы).

Сохраните ее в вашей системе, как promote.html (или promote.htm).

2. Удалите последнюю ссылку вместе с описанием, переместите другие вниз и разместите вашу ссылку (ссылку на ваш сайт) вместе с описанием сайта в первое положение(позицию).

3. Удостоверитесь, что путь к вашей ссылке, не относительный, а абсолютный (т.е., начинается с «**http://ваш сайт**». Другие люди будут копировать вашу страницу и соответственно ссылка должна быть правильной, когда по ней щелкнут где-нибудь в другом месте). Постарайтесь дать краткое и вместе с тем точное описание вашего ресурса (ориентировочно до 200 символов, 3-4 строки).

4. Создайте на страницу promote.html хорошо видимую ссылку (текстовую или графическую — вам решать) с главной страницы вашего ресурса или со страницы с наибольшей проходимостью. Можно создать несколько ссылок с разных мест своего сайта. Вы сами, в первую очередь, должны быть заинтересованы, чтобы как можно больше посетителей попало на эту страницу. Удостоверьтесь, что вашу связь действительно видно. Для гарантии, используйте, что-то вроде этого:

```
ЩЕЛКНИТЕ ЗДЕСЬ, чтобы узнать, как получить 22 000 посетителей на
ваш сайт. Бесплатно!
```

5. Поисковики не индексируют зеркала, поэтому копии этой страницы не будут проиндексированы, так как их содержание почти полностью зеркально. Поэтому все ссылки с этой страницы разместите где-нибудь на страницах своего сайта. Описания ссылок поменяйте на очень краткие, по вашему усмотрению, если участник дал несколько ссылок, выберите первую. Этот простой прием значительно усилит эффект нашего с вами предприятия, и, что важно — повысит индекс цитирования участников в Интернете.

Вы должны теперь иметь:

1. Хорошо видимую ссылку на вашей главной (или другой) странице, указывающей на promote.html в вашей системе. Ваша текстовая ссылка с описанием должна стоять на первом месте среди других ссылок.

2. Текстовые ссылки должны стоять на вашей странице promote.html. И позвольте им оставаться там. Аналогично, ваша ссылка будет показана на тысячах других страниц и сайтах.

3. Вся информация располагается в таблице. Не меняйте дизайн таблицы и расположение информации! Категорически запрещено самовольно изменять прилагаемую инструкцию, искажать смысл информации и т.д. В крайнем случае, допустимо изменять цветовую гамму информации, если это не согласуется с вашим дизайном. Например, у вас — белый цвет текста на черном фоне. Однако при изменении цветовых параметров, пожалуйста обратите внимание: удобочитаемость информации не должна пострадать.

Как это работает. Много web-мастеров посетят ваш сайт и присоединится к программе Nika Virtual PROMOTE, взяв себе вашу страницу, вместе с вашей ссылкой и перемещая ее у себя на вторую позицию.

Web-мастера, которые посещают их сайты и страницы — будут делать то же самое. К тому времени, когда ваша ссылка переместится на 5-ю позицию — вы будете иметь порядка 50 000 ссылок на свой ресурс с других сайтов! При хорошем раскладе все это возможно спустя несколько месяцев! Если хотите, то дальше посчитайте сами.

Давайте продемонстрируем это: например, вашу страницу promote.html скопировали к себе и соответственно разместили ссылку на вас 15 человек web-мастеров (на самом деле, эта цифра будет значительно больше, но давайте для примера возьмем это число). Давайте также предположим, что каждого из этих web-мастеров, также посетили по 15 человек и присоединились к программе Nika Virtual PROMOTE и т.д — тот же цикл.

Результат при размещении вашей ссылки на 5 (пятой) позиции: 15x15x15x15= 50 625. Ссылка на ваш сайт будет присутствовать на 50 625 сайтах или страницах! Если же ваша ссылка доберется до 9 позиции, то результат будет просто астрономическим!

И эти ссылки(связи) постоянны! Они ведут потенциальных посетителей на ваш сайт всегда!

Глава 8. Банерная реклама

Любой web-мастер знает, что создание качественного и интересного сайта — это только часть успешно работающего и приносящего доход сайта. В начале его жизни необходимо обеспечить приток новых пользователей каким-либо образом. Далее, если сайт интересен, посетители сами будут приходить на него, ибо если есть человек — он будет куда-то ходить, надо лишь показать ему, куда можно заходить, и убедить, что здесь интересно.

Для того, чтобы привести новых посетителей вполне логично разместить ссылки на ваш сайт с других, более посещаемых ресурсов. Один из простейших и дешевых способов — это регистрация сайта в поисковых машинах, который может приводить на сайт немного посетителей. Однако ресурсов одного типа довольно много — поисковая машина выдает сразу все сайты одного типа — естественно что вероятность того, что пользователь зайдет именно на тот сайт, который надо — невелика. Системы рейтингов приводят на сайт пользователей пропорционально посещаемости самого ресурса, т.е. для того, чтобы люди шли на сайт — он сперва должен быть посещаемым, а чтобы быть посещаемым — он должен как-то рекламироваться иным путем.

Банерообменные системы

Российский Интернет достаточно молод, и денежные отношения в нем еще только строятся. Всего пару лет начали появляться интернет-платежные системы (WebMoney, PayCash), до этого существовала лишь система оплаты через кредитные карты. Потому до сих пор он сохраняет многие принципы натурального хозяйства. Появившиеся банерообменные системы изначально так же строились на этих принципах — на обмене банерных показов друг у друга (несколько сайтов показывали банеры других участников системы, таким образом происходил обмен пользователями, как правило с общей выгодой), т.е. для «раскрутки» сайта первоначально необходимо было иметь уже некоторую посещаемость, которую брали из других ресурсов либо за счет других своих же сайтов, по личному знакомству, либо путем прочих договоров. Однако сами банерообменные системы получают часть показов в свое усмотрение (к примеру, показывая каждый сайт по 10 тыс. банеров у себя — его банеры показываются на сайтах участников к примеру по 9 тыс. показов — т.е. с каждого сайта система получает по 1 тыс. показов). Возникает вопрос: куда же девает сама банерообменная система эти показы, ведь при большом количестве участников это очень большие суммы?

Конечно же пытается продать, хотя некоторая часть уходит на рекламу самой системы. Таким образом новые сайты могут купить показы, и привести пользователей на свой сайт. Но как сайт, уже имеющий аудиторию, может получать прибыль от своей посещаемости, ведь банерообменные системы не предназначены для этого изначально! Т.е. сайт должен собственноручно искать клиентов — в основном владельцев новых сайтов, и предлагать им рекламироваться на своей площади. Конечно это не удобно.

Банерные биржи

Следующим эволюционным этапом в рекламном бизнесе послужило появление банерных бирж — сайтов, где пользователи продают накопленные на своих сайтах показы. Большинство существующих бирж работают на основе доски объявлений, что приводит к возможности мошенничества со стороны «лжепродавцов» (ибо современные интернет-платежные системы работают на принципах самого Интернета — анонимности, и получил деньги заранее, мошеннику ничего не стоит, как исчезнуть). Отдельно стоит биржа BanStock — которая следит за честностью как покупателей, так и продавцов, проводит сделки и обеспечивает именно рыночные цены. Как правило — биржи либо не рыночны, а сама «биржа» устанавливает свою цену на показы, либо же это доска объявлений, где нет никаких гарантий при купле/продаже. Однако основное достоинство банерных бирж является приближенность цены показов к рыночной. При средней цене в $2 за 1000 показов у банерных систем, биржевая цена куда ниже — в среднем $0.3 за 1000 показов. Это обусловлено довольно низкой эффективностью рекламы, а также тем, что крупные клиенты покупают у самих систем, тем самым на «вторичном» рынке остаются средние и мелкие покупатели. Многие банерообменные системы, в связи с появлением вторичного рынка достаточно крупных масштабов пытаются уже сами закупать показы, дабы не потерять возможность дополнительного заработка — ведь рынок есть в любом случае — вопрос лишь в том, кто получает проценты со сделок — СБО (система банерообмена) или биржа. К тому же СБО может за счет своей известности сильно развести цены на покупку и продажу показов.

Где же все таки лучше покупать показы?

Это зависит только от владельца ресурса: при покупке через СБО вы получаете гарантию того, что будут показываться ваши показы в строго установленных вами объемах, возможен дополнительный таргетинг (направление показов на нужные сайты), прочий сервис. Покупая же у участников сети и на биржах — вы сможете сэкономить достаточно большое количество денег на покупке.

Часть 9. Заработок при помощи своего сайта

Глава 1. Бесплатное место под ваш сайт

Естественно, бесплатность здесь — понятие относительное. Все это делается из соображений рекламы. И поэтому при выборе сервера стоит обращать внимание на то, какую рекламу от вас потребуют за бесплатный хостинг. Лучше, конечно, вообще никакой, или маленький логотип где-нибудь в уголочке. Но чаще всего на вашей странице размещается банер, причем на самом видном месте в верхней части вашей страницы. И ничего тут не поделаешь! Но настоящая беда — это всплывающие окна (pop-up). Этот механизм крайне раздражает посетителей и способен испортить все впечатление от вашего сайта. Здесь можно либо избегать публикации на таких хостах, либо пробовать бороться с этим. Иногда удается избавиться от всплывающих окон внесением в тело HTML-документа тэгов **<NOSCRIPT></NOSCRIPT>**.

Сервер может предоставлять или нет возможность использования CGI-скриптов на своем сайте. Возможность использования CGI безусловный плюс для web-дизайнера. Например, можно разместить на странице собственный графический счетчик или гостевую книгу, сделать свой форум или организовать опрос посетителей.

Не все серверы предоставляют одинаковый объем пространства для вашего сайта. Ниже в таблице указано, сколько места под свои нужды вы можете использовать на каждом из хостов. Вы можете переписывать файлы вашего сайта на хост либо какой-то FTP-программой (CuteFTP, BulletProofFTP и др.) или непосредственно из браузера, если это позволяет хост. Последнее не очень удобно и занимает гораздо больше времени.

Ниже перечислены несколько наиболее популярных серверов, предоставляющих бесплатный хостинг:

Narod.ru

Неограниченный объем сайта, сервер работает быстро и достаточно устойчиво. Есть ограничение на размер файла — 5 МБ. Адрес вида http://name.narod.ru. Можно делать сайты при помощи готовых шаблонов — их более сотни на данный момент, а также предоставляется гостевая книга, форум, чат, счетчик посещений. Дается почтовый ящик вида name@narod.ru. Закачка файлов по FTP или при помощи браузера. Реклама — небольшое всплывающее окошко.

Hut.Ru

Неограниченный объем сайта, CGI, MySQL, PHP, SSI. Загрузка по FTP. Реклама — просят разместить кнопочку на первой странице. Адрес вида http://name.hut.ru. На данный момент регистрация прекращена.

Newmail.ru

16 МБ, плюс столько же для трех почтовых ящиков. Адрес http://name.newmail.ru, http://name.nm.ru или http://name.hotmail.ru, или все сразу. Закачка файлов по FTP. При неактивности в течение 50 дней (нет обновлений сайта и не просматривается почта) сайт удаляется вместе с почтовым ящиком. Рекламы нет.

By.ru

Неограниченный объем сайта, загрузка по FTP или WWW. Ограничение на размер файла — 1 МБ. Есть SSI. Дают гостевую книгу, чат. Адрес вида http://name.by.ru. Реклама: всплывающее окно при первом посещении сайта. К сожалению, сервер периодически виснет.

WALLst.ru

Неограниченный объем сайта, CGI, PHP, готовые скрипты. Загрузка по FTP. Адрес вида http://name.wallst.ru, http://name.dax.ru, http://name.aiq.ru, http://name.dtn.ru, http://name.vov.ru, http://name.tora.ru, http://name.pips.ru, http://name.metastock.ru и http://name.supercharts.ru.

Chat.ru

10 МБ, загрузка по FTP или WWW. Предлагают готовую гостевую книгу. Адрес вида http://name.chat.ru. Рекламы нет.

Boom.ru

50 МБ, загрузка по FTP или WWW. Предлагаются шаблоны для быстрого создания сайта. Адрес вида http://name.boom.ru.

FortuneCity

100 МБ. Возможность использования CGI. Допустимый размер файлов — 3 МБ. Сервер быстрый. Закачка по FTP и WWW. Адрес вида http://members.fortunecity.com/login_name либо длинный http://www.fortunecity.com/слово1/слово2/число. На каждой странице вверху вставляется банер.

VirtualAve

20 МБ, CGI, поддержка расширений FrontPage. Закачка файлов по FTP. Реклама — банер или всплывающее окно.

Freedom 2 Surf

20 МБ, CGI, PHP. Поддержка расширений FrontPage. Закачка по FTP. Адрес вида http://name.f2s.com. Рекламы нет.

Tripod

50 МБ, дают набор своих CGI-скриптов. Закачка файлов по FTP и через браузер. Адрес вида http://name.tripod.com и http://members.tripod.com/name. Реклама: всплывающее окно или банер вверху страницы (на выбор).

Если вы хотите бесплатно зарегистрировать вашу web-страницу в поисковых машинах Интернет, просто обратитесь к одному из ниже перечисленных серверов.

1000 Christian Links In 72 Categories ADD-URL

http://www.newcreations.net/sermoncentral/addurl.html

PM FFA Link Page

http://11pm.com/links/links.htm

Register's Cool Site Award

http://www.123register.com/award.html

Mockingbird Lane FFA Link Page

http://1313mockingbirdlane.com/links/links.htm

Have Fun FFA Link Page

http://www.2havefun.com/links/links.htm

FFA Link Page

http://www.4u2.de/

Free Banner Exchange
http://1-2-free.com/

Banners
http://www.123banners.com/

AmericaMall
http://www.1second.com/1america.htm

StopBiz
http://www.1stopbiz.com/

FFA Link Page
http://212.net/

BuySell
http://www.2buysell.com/

Cool Web Coolest Site
http://www.2coolweb.com/

Hand Internet Market
http://www.2him.com/

Exchange
http://www.x-x.com/

Нередко приходится переезжать с одного сервера на другой. Причины могут быть разные — слишком уж много начали вешать навязчивой рекламы на ваших страницах, или сервер часто глючит и до него порой просто не достучаться. В любом случае, чтобы не терять своих посетителей, а тем более постоянных, которых надо любить, лелеять и вообще беречь как зеницу ока, рекомендуется на старом сервере оставить следующий файл с именем index.htm или index.html:

```
<HTML>
<HEAD>
<TITLE>We've moved!!!</TITLE>
<META HTTP-EQUIV="Refresh" CONTENT="0; URL=http://новый_адрес">
</HEAD>
<BODY></BODY>
</HTML>
```

Этот файл позволит без задержки перенаправлять сервера на ваш новый адрес. Первое значение атрибута **CONTENT** (число 0 в данном

примере) определяет время задержки в секундах перед переходом на адрес, указанный во втором значении атрибута.

Чаще поступают следующим образом: задержку делают секунд 10-20, а на странице пишут, мол, мы переехали на адрес такой-то, и ставят ссылку. Это делается для старых браузеров, которые не поддерживают автоматический редирект (перенаправление). Хотя такими браузерами, наверное, давно никто не пользуется.

Глава 2.
Как заработать на своем сайте

Приступая к работе над своим сайтом, вы вряд ли задумывались о том, что на нем можно еще и заработать. Скорее всего, вы просто хотели о чем-то рассказать человечеству (хотя, возможно, именно ради заработка вы и создавали свою страницу). Сколько можно заработать? Все зависит от количества посетителей и аудитории вашего сайта.

Прежде всего вам необходимо сделать ваш web-сервер популярным (сделать свой сайт популярным — весьма нелегкая задача; для начала вы должны зарегистрировать его в поисковых системах Интернет). Любой популярный (новостной, поисковый, развлекательный) ресурс Интернет способен приносить прибыль через рекламу. Вы просто продаете часть своего web-сервера за реальные деньги, которые вам будут платить рекламодатели.

Второй способ заработать на Интернет — создать качественную сетевую службу (поисковая система, web-хостинг, изготовление банеров и web-серверов). Услуги этой службы оплачиваются тоже реальными деньгами.

Третий способ — собственный виртуальный магазин. В виртуальном магазине можно продавать практически любую продукцию. Главное — это построить такой магазин. Это весьма-весьма нелегкая задача. На ее решение может уйти от 7.000 до 200.000 долларов.

Реально вы можете «получать» и 1-2 доллара в месяц, и несколько сотен долларов в неделю. Все зависит только от вас! Итак... Кто и за что нам платит? В Сети есть немало фирм (как правило — иностранных), готовых заплатить вам за рекламу их услуг и товаров, за посредничество, и даже за чтение их рекламной информации.

- Посредничество при продаже товара — на своем сайте вы рекламируете и предлагаете купить товар. Если кто-то покупает его через вас, то вы получаете процент с

продажи. Очень большие доходы. Иногда достаточно продать одну дорогую вещь, чтобы заработать несколько десятков, а иногда и сотен долларов. Но необходимо иметь англоязычный сайт и работать с зарубежной аудиторией (нашим-то ничего не загонишь!).

- Реклама фирм, их товаров и услуг — наиболее доступный нам способ заработка. Вы размещаете на своем сайте банеры и/или ссылки на рекламодателя. Оплата может быть за показы (в настоящее время встречается очень редко): при загрузке страницы загружается рекламный банер компании, за это вам начисляется некоторая сумма (обычно меньше цента). Хорошо подходит для сайтов с высоким трафиком. Оплата за клики: посетитель кликает по банеру/ссылке и попадает на сайт рекламодателя, на вашем же счете автоматически прибавляется несколько центов. Это не приносит высоких доходов, даже на часто посещаемом сайте. (По статистике по банеру кликает 2-4 посетителя из 100, за клик обычно платят 3-5 центов. Посчитайте, сколько должно быть посетителей, чтобы на счете появилось, скажем, 10 долларов.)
- Заполнение форм — вам платят за регистрацию новых пользователей, web-мастеров и т.д. Вполне подходит для русскоязычных сайтов, т.к. покупать посетителю сайта ничего не надо. Однако, необходимость заполнения форм, особенно англоязычных, отпугивает многих посетителей.

Следует сказать, что нередко спонсоры предлагают смешанные программы (например, за клик — 5 центов, за регистрацию посетителя — 1 доллар).

Итак, сайт у вас, надо полагать, уже есть — можно начинать зарабатывать! Желательно еще хотя бы чуть-чуть знать английский язык. А вообще, чем лучше вы им владеете, тем легче вам будет работать. Поэтому не ленитесь — учите!

Далее рассмотрим конкретных спонсоров (не много, но зато проверенные)... Точнее говоря, это посреднические компании между спонсорами (рекламодателями) и вами.

LinkShare-Get Your Share!

Предлагает очень широкий выбор спонсоров (на данный момент — более 550). После несложной регистрации вы можете выбирать себе наиболее подходящих спонсоров по тематике и видам оплаты. Хотя преобладают проценты с продаж, присутствуют также и клики по банерам, и

заполнение форм (довольно-таки выгодное). Практически у каждого спонсора имеется возможность выбора: представлять их на своем сайте банером, ссылкой, кнопкой и т.п. Есть даже email-ссылки — незаменимая вещь для почтовых рассылок. За каждого зарегистрированного вами web-мастера платят 3 доллара. Периодически рекламодатели проводят розыгрыши призов среди рекламирующих их web-мастеров. Побеждают в них и наши соотечественники. Минимальная сумма для высылки чека определяется каждым спонсором отдельно. Самый большой недостаток — чек высылается отдельно по каждому спонсору. Т.е. вы рекламируете, например, четырех спонсоров. Значит вы и получите потом 4 отдельных чека. Т.е. ваш заработок не суммируется. Обычно наши web-мастера выбирают одну программу: рекламируют саму Linkshare (все-таки $3.00 за каждого web-мастера — достаточно щедрое предложение), иногда еще какого-нибудь наиболее привлекательного спонсора.

cj.com (Commission Junction)

Пожалуй, наиболее привлекательная посредническая контора. Очень простая и быстрая регистрация. Предлагает солидный список различных программ — более 1000 на данный момент. Все программы в основном совмещенные: например, проценты от продаж + клик по банеру. Кстати, за каждого web-мастера, вступившего в Commission Junction по вашей ссылке, вы получите 2 доллара, плюс премия в размере 5 процентов от заработанной им суммы (пожизненно! Кстати, эти проценты не вычитаются из заработка web-мастера, а выплачиваются самой конторой, т.е. web-мастер ничего не теряет). Чеки от 25 долларов высылают ежемесячно, сумму вы можете увеличить при желании. Интересно, что выплаты могут производиться в разной валюте — 27 наименований. В отличие от предыдущей компании все деньги, заработанные у разных спонсоров, суммируются. Вам высылается один общий чек.

При помощи формы справа, предложенной компанией Commission Junction, вы можете поискать подходящих спонсоров для своего сайта. Причем не регистрируясь в самой Commission Junction! В выпадающем списке выбираете категорию спонсора (например, Gift & Flowers — «подарки и цветы») или оставьте all categories — «все категории». Выбираете тип оплаты: за клики (click-through), за продажи (pay-per-sale), за регистрацию (pay-per-lead), за показы (pay-per-view). Ниже можно ввести ключевое слово — если ищете конкретного спонсора. Если «ассортимент» спонсоров понравится, то здесь же можно и зарегистрироваться.

Safe-Audit

Один из старейших спонсоров. Предлагает множество программ. Есть оплата показов банеров, оплата кликов, проценты с продаж и др. Чеки от 20 долларов высылают ежемесячно. Регистрация может длиться

до 10 дней — сперва вашу страничку просмотрят, потом вы получите письмо по E-mail. Если сайт русскоязычный, то сразу много банеров не дадут. Спонсор надежный и платит исправно!

И в заключение хотелось бы сказать, что не стоит пытаться обманывать спонсоров. Не надо кликать по банерам на своем сайте, регистрироваться у самого себя и т.п. Все накрутки достаточно быстро обнаруживаются, с недобросовестным web-мастером отношения немедленно разрываются, а весь его заработок аннулируется.

ClickTrade

Крупная посредническая система. Предлагает размещать как банеры, так и текстовые ссылки. Как это все работает: вы регистрируете у них сайт и начинаете шарить по их базе данных, где содержится около 2000 разных предложений. За клик могут давать 1-99 центов ну и «affiliate programs» имеется немерено конечно.

Когда вы найдете что-нибудь понравившееся, вы посылаете запрос (путем нажатия специальной кнопочки) и ждете.

Когда рекламодатель (не ClickTrade, а именно рекламодатель! ClickTrade здесь только посредник) получает вашу заявку он радостно бежит смотреть ваш сайт, если сайт ему нравится, он говорит сколько будет платить за клик. Поэтому не обольщайтесь, увидев в базе данных предложение с 99 центов за клик — это только прикидка.

Ну и дальше все протекает как обычно — вы размещаете банер или ссылку и живете долго и счастливо. Случаются нюансы — у рекламодателя могут закончиться деньги на аккаунте в ClickTrade, вам в этом случае перестают платить. Рекламодатель может совсем закрыть программу. Так что нужно постоянно наблюдать за событиями. Вообще все система несколько запутана и таинственна, но разобраться можно.

Ad-Up

Оплата в зависимости от посещаемости вашего сайта, берет комиссию:

70% до 1000 в день

65% до 10000

60% до 30000

55% до 60000

50% до 100000

45% до 200000

40% до 300000

35% до 500000

30% свыше 500000 показов в день.

FlyCast

Оплата зависит от рекламодателя.

CyberFirst

Оплата: с рекламодателя они просят $10 — $35 CPM. Комиссию берут 25%.

Спонсор платит за клики — за каждого пользователя, перешедшего с вашего сайта на сайт рекламодателя. В этом случае эффективность банера, его прибыльность для вас будет в сильной мере зависеть от того, насколько этот банер будет привлекателен для посетителей вашего сайта и насколько банер будет «бросаться в глаза» (здесь впрочем тоже не следует кидаться в крайности). Поэтому лучше, когда вы можете сами выбирать, какой банер будет красоваться на вашем сайте. Некоторые из нижеприведенных программ представляют собой так называемые Multi-Advertiser Networks — то есть банерные сетки, крутящие разные банеры на вашем сайте и вы не имеете контроля или имеете слабый контроль за тематикой показываемых банеров. Это плохо. Такими конторами следует пользоваться с осторожностью.

AdClix

Система тут такая — вы регистрируетесь, они смотрят ваш сайт и предлагают вам на выбор несколько банеров соответствующей тематики. Вы выбираете понравившийся и ставите на сайт. Если у вас появляется что-нибудь другой тематики, вы можете попросить их предоставить дополнительные банеры.

Adult спонсоры

Перво-наперво забудем о том, что существуют в Интернет русские безденежные юзеры. Будем думать что в Сети немерено буржуев с кредитками. Забыли? OK. Поехали дальше.

Как устроен adult сервер? Приходит пользователь, смотрит на sample pictures, читает завлекательные описания, и потом, если ему все это понравилось, регистрируется (signup), платит деньги, получает пароль и начинает пользоваться содержимым сервера. Чтобы привлечь побольше посетителей на сервер организаторы сервера объявляют sponsorship program и привлекает web-мастеров для рекламы путем развешивания банеров, рекламирующих этот adult сайт, на страничках web-мас-

теров. Главный критерий для спонсора — коммерческая выгода, т.е. ему не нужно чтобы его банер просто висел у вас на сайте. Ему нужно, чтобы с этого банера к нему шли пользователи с деньгами, готовые сайнапится и платить ему деньги. Если вы рассчитываете получить со спонсора в долгосрочном периоде больше, чем принесете ему прибыли, то губу лучше сразу закатать.

Например. Юзер в среднем пользуется adult сервером 3-4 месяца и платит за эти месяцы в среднем сколько-то баксов. Какой-то частью этой суммы спонсор готов с вами поделиться. Он может сделать это разными путями. Разные программы — кликовые, сайнаповые, partnership — это и есть разные пути. А суть одна, просто варианты разные.

Вариант первый: фиксированная оплата за сайнап. За каждого засайнапившегося юзера спонсор вам платит 20-35 баксов. Здесь все понятно. Вы привели деньги спонсору и получили свой кусок.

Вариант второй: фиксированная оплата за клик (flat rate). Казалось бы, самая выгодная программа — вы посылаете ему клики и не думаете о сайнапах. Так ведь нет же. Если ваш трафик окажется сильно некачественным (сайнапов мало будет), вас перекинут на оплату по сайнапам. Хорошо еще если выплатят уже заработанное. Все правильно — спонсор не хочет платить вам больше, чем ему это выгодно. Про conversion ratio, после которого перекидывают на сайнапы, говорят в terms & conditions, хотя и не всегда. Обычно это 1:500 — 1:1500.

Вариант третий: плавающая оплата за клик в зависимости от качества вашего трафика. Например, так:

1:10 — $3.5

1:50 — $0.70

1:100 — $0.35

1:200 — $0.17

1:500 — $0.07

1:1000 — $0.035

ниже 1:1000 — $25 за signup

Wow! — говорит web-мастер — три с половиной бакса за клик это круто! Но на самом деле если посчитать, то окажется, что это просто видоизмененный первый вариант. Если у вас конвертится 1:100 и вы получаете за клик 35 центов, то сколько вы получаете за того сотого, который засайнапился? Те же 35 баксов. А если трафик сильно некачественный, то и всего 25 баксов. Здесь выгода скорее психологическая — когда сумма на аккаунте прибавляется медленно, по центам, но верно, душе спокойнее, чем ждать этого чертова сайнапа.

Вариант четвертый: «Partnership». Вы получаете, скажем, 50% от всего того, что юзер заплатит на сайте спонсора. Наиболее честный вариант для обоих сторон.

Вариант пятый: спонсор обещает вам, что будет платить вам за клик 10-20 центов независимо от качества вашего трафика. Врет. Не будет он столько платить за плохой трафик. Либо статистику будет занижать, либо вообще не заплатит. Максимум, что может себе позволить платить спонсор без оглядки на трафик 1-2 цента за клик.

Intergal

Невообразимое количество разных программ. В основном правда floating rate (цена клика зависит от качества трафика). Есть программа с консолькой, где вам платят фактически за показы этой консольки по пол цента за показ.

Gamma Entertainment

Всего две программы — одна с flat rate 4 цента за уникальный клик (при некачественном трафике переводят на partnership с выплатой заработанного), другая — partnership 50/50. Платит вовремя.

TrafficCash

Две программы: одна чисто сайнаповая 35 баксов за сайнап, вторая такая: если за период (неделя) не было сайнапов, то вам платят 2 цента клик (клик по моему second page raw), а если были, 16 центов за клик. Говорят платит.

CyberErotica

Тоже много разных программ на выбор. Есть программа с консолькой по центу за показ + 10 баксов за сайнап с этой консольки. Однако пипл рассказывал, что если с этой консольки сайнапов не будет, то аккаунт замораживают и ничего не выплачивают.

Русскоязычные сайты

Здесь собраны спонсоры, для которых язык вашего сайта не имеет особого значения. Смотрите и выбирайте.

http://www.bannerpool.com/

Оплата за клик: от 6 до 14 центов

Минимальная сумма высылаемого чека: 50 долларов

Минимальное количество уникальных посетителей в месяц: не ограничивается

Язык сайта: любой

Содержание сайта: не оговаривается

Примечание: Первые 5 кликов не оплачиваются — дают Вам возможность проверить код.

http://www.valueclick.com/

Оплата за клик: от 6 до 12 центов

Минимальная сумма высылаемого чека: 30 долларов

Минимальное количество уникальных посетителей в месяц: не ограничивается

Язык сайта: любой

Содержание сайта: проверяют

Примечание: Имеют интеллектуальную систему банерокрутилки (подбор банера с максимальным CTR).

http://www.websponsors.com/

Оплата за клик: от 1 до 60 центов

Минимальная сумма высылаемого чека: 50 долларов

Минимальное количество уникальных посетителей в месяц: 500

Язык сайта: любой

Содержание сайта: проверяют

Примечание: Размещают банеры, кнопки, текстовые ссылки, короче все подряд.

http://www.adclub.com

Оплата за клик: 10 центов

Минимальная сумма высылаемого чека: 20 долларов

Минимальное количество уникальных посетителей в месяц: 5.000

Язык сайта: любой

Содержание сайта: кроме warez'a

Примечание: Платят только за CTR до 4% (То есть при 100 показов и 6 нажатиях вам засчитают только 4). Разрешают размещение на странице не более 2 своих банеров, один HTML-код для всех страниц. Имеют спецсистему выбора банеров — робот на основе опыта выбирает какие банеры лучше крутить у вас на сайте, то есть анализирует CTR у всех банеров в базе и крутит именно лучшие.

http://www.pennyweb.com/

Оплата за клик: от 2 до 9 центов

Минимальная сумма высылаемого чека: 25 долларов

Минимальное количество уникальных посетителей в месяц: не ограничивается

Язык сайта: любой

Содержание сайта: не оговаривается

Примечание: банеры других брокеров на той же странице не допускаются.

http://www.clicktrade.com/

Оплата за клик: от 0 до 100 центов

Оплата процента с продаж: от 10 до 75%

Минимальная сумма высылаемого чека: 50 долларов

Минимальное количество уникальных посетителей в месяц: не ограничивается

Язык сайта: любой

http://www.safe-audit.com/

Оплата за клик: от 15 до 20 центов

Оплата за 1000 показов: от 4 до 16 долларов

Минимальная сумма высылаемого чека: 10 долларов

Минимальное количество уникальных посетителей в месяц: не ограничивается

Язык сайта: любой

Содержание сайта: проверяют

http://www.bottomdollar.com/

Оплата за клик: от 2 до 12,5 центов

Оплата за 1000 показов: 4 доллара

Минимальная сумма высылаемого чека: 20 долларов

Минимальное количество уникальных посетителей в месяц: не ограничивается

Язык сайта: любой

Содержание сайта: не проверяют

Примечание: Если вы набрали за месяц меньше 20$, то все аннулируется.

http://www.datais.com/advertising/

Оплата за клик: 10 центов

Минимальная сумма высылаемого чека: 50 долларов

Минимальное количество уникальных посетителей в месяц: не ограничивается

Язык сайта: любой

Содержание сайта: не оговаривается

Примечание: банерокрутилка обычная без всяких там интеллектуальных наворотов, крутит себе их по кругу и все.

http://www.dmnmedia.com/

Оплата процента с дохода: 70%

Минимальная сумма высылаемого чека: 100 долларов

Минимальное количество уникальных посетителей в месяц: 100.000

Язык сайта: любой

Содержание сайта: только о музыке

Примечание: предлагают максимальный во всем WWW процент с доходов — 70%. А ведь именно они всю работу выполняют...

http://www.cyberthrill.com/

Оплата за клик: 20 центов

Минимальная сумма высылаемого чека: 10 долларов

Минимальное количество уникальных посетителей в месяц: не ограничивается

Язык сайта: любой

Содержание сайта: не проверяют

http://www.eads.com/

Оплата за клик: 5 центов

Минимальная сумма высылаемого чека: 10 долларов

Минимальное количество уникальных посетителей в месяц: не ограничивается

Язык сайта: любой

Содержание сайта: проверяют

Примечание: банеры выбираете себе сами, то есть они статические, но зато можно их подбирать под контент сайта.

http://www.paradise-web.com/

Оплата за клик: 0.2 цента

Минимальная сумма высылаемого чека: 50 долларов

Минимальное количество уникальных посетителей в месяц: не ограничивается

Язык сайта: любой

Содержание сайта: не проверяют

Примечание: Нестандартные банеры — 400x40.

Глава 3. Банер и оплата его размещения

Банер — картинка рекламного содержания, ведущая на рекламируемый сайт. Обычные размеры для банера 468x60, но часто встречаются и другие размеры.

Спонсор — организация, которая платит вам за то, что вы размещаете ее рекламу на вашем сайте. В простейшем случае это сам рекламодатель, т.е. сайт, заинтересованный в собственной раскрутке. Чаще же встречаются брокеры — организации, стоящие между рекламодателями и web-мастерами, владельцами сайтов и страничек, желающими рекламу у себя разместить. С рекламодателей брокер берет некоторую сумму за рекламу, и выплачивает ее web-мастерам, забирая себе определенную комиссию.

Показ, impression, adview — одна демонстрация банера посетителю. Факт показа определяется загрузкой картинки, и, в общем то, посетитель вполне мог и не видеть банера, если тот запрятан например далеко внизу.

CPM — оплата за 1000 показов банера на вашем сайте. Обычный CPM — $1-$5.

Клик, переход, click, click-through — факт нажатия банера пользователем.

Чаще встречаются именно программы, где спонсор платит вам за клик (CPC programs). Цена клика варьируется от 1 до 20 центов в зависимости от щедроты спонсора.

Клики могут считаться спонсором по-разному:

- Уникальными (Unique) считаются клики посетителей с разными IP адресами.
- Простым (Raw) считается любой клик.

Raw клики тоже бывают разными. Иногда спонсоры делают такую фишку: банер ведет на входную страницу сайта (первую), с которой можно попасть на сам сайт (вторая страница) и клик засчитывается, если пользователь, придя на входную страницу, не ломанулся назад, а пошел на вторую страницу. В этом случае говорят о second page raw — считается клик на вторую страницу. Это делается спонсором чтобы избежать накруток.

Как именно спонсор считает клики обычно обговорено в условиях программы (terms and conditions). В любом случае полезно знать это заранее.

CTR — процент посетителей, нажавших на банер. Обычно 1-5%, в среднем около 2%. Чем CTR выше, тем выше эффективность банера. Естественно, если банер кликовый вы заинтересованы в его максимальной эффективности. Она зависит от кучи разных факторов:

- если тема банера совпадает с интересами аудитории вашего сайта, кликать будут чаще;
- расположение банера на странице сильно влияет на CTR. Очевидно что висящий на самом верху банер даст больший CTR, чем спрятанный где-то внизу;
- размер банера в Кб. Если банер грузится долго, пользователь вряд ли будет сидеть и ждать, он пойдет дальше по своим делам.

Для кликового банера тоже можно посчитать CPM: банер показан 1000 раз, CTR например получается 3%, стало быть кликнуло 30 человек, и если за клик платят 10 центов получаем CPM = $3.

Так что CPM кликового банера зависит в сильной степени от творческого подхода. Причем не обязательно чем выше платят за клик, тем больше денег вы получите. Красивый, подобранный по тематике 5-центовый банер может принести вам больше, чем распугивающее посетителей вашего сайта 20-центовое нечто.

Еще более многочисленная группа программ — CPA programs (affiliate programs, «per action»). Здесь рекламируются коммерческие сайты — там, где можно что-нибудь купить, заказать, подписаться, заплатив им за это деньги. Так вот в программах «per action» вы становитесь как бы посредником, и если посетитель, пришедший с вашего сайта что-нибудь приобретет/подпишется/ зарегистрируется короче заплатит за что-нибудь, то вы получите либо процент от сделки (такие программы называются еще «partnership program») либо заранее оговоренную сумму.

По вполне понятным причинам вешать такие банеры на сайты с российской аудиторией имеет мало смысла, однако попадаются «per action» программы, которые не требуют от посетителя что-либо заплатить, а предоставляют ему какой-нибудь free stuff или бесплатно же регистрируют его где-нибудь. В этом случае к таким программам стоит присмотреться поближе.

Conversion ratio — показатель эффективности для банера «per action». Это соотношение числа купивших (зарегистрировавшихся) к числу пришедших с вашего сайта. Если conversion ratio очень низкий, то говорят о неэффективном трафике с вашего сайта.

Как спонсор платит деньги?

Один раз месяц или раз в две недели (это оговорено в условиях) спонсор смотрит сколько баксов у вас набежало на аккаунте. Если сумма на аккаунте превышает некоторую минимальную сумму (которая тоже оговорена, обычно это $10 — $100) вам высылается чек. Именной банковский чек (бывают еще дорожные чеки — это совсем не то).

Некоторые спонсоры делают еще и Wire Transfer — прямой перевод денег на банковский счет. Это, конечно, быстрее, чем чеки, но минимальная сумма для такого перевода $500 -$1000.

Что делать с чеком?

Ломимся в банк, который работает с именными чеками и отдаем чек на «инкассо». При этом у вас его забирают, отправляют почтой в тот банк, откуда он выписан, тот банк переводит деньги на счет вашего банка, а вам банк выплачивает баксы. Оплата за инкассо и скорость проведения данной операции сильно разнятся от банка к банку. В Сбербанке

например берут 4% от суммы чека (но не меньше $3) и делают это 1.5-2 месяца. Другие банки могут делать это около 2-х недель.

Глава 4. Влияние местоположения банера на его эффективность

Гипотеза 1. Если банер расположен около полосы прокрутки, то кликабельность повышается потому что указатель мышки там рядом, не надо напрягаться тащить мышку черт знает куда.

Результаты исследования. На банер 125x125, расположенный в нижней части экрана (экрана! — не страницы!) у полосы прокрутки кликали в среднем в 3 раза чаще, чем на банер 468x60, расположенный в верхней части страницы.

Гипотеза 2. Если банер расположен не на самом верху страницы, а примерно на 1/3 от верха, то кликабельность повышается.

Результаты исследования. Кликабельность действительно повысилась в среднем на 77%.

Гипотеза 3. Если вдобавок к верхнему банеру добавить его же, но внизу страницы, кликать будут больше.

Результаты исследования. Заметного повышения кликабельности не обнаружено.

Глава 5. Банерокрутилка на JavaScript

Эффективность банерной рекламы можно увеличить, если показывать одному и тому же пользователю не один, а несколько банеров — повышается вероятность, что он заинтересуется каким-то из них. Это можно делать варварским способом — размещая несколько банеров на странице, а можно и сделать так, чтобы единственный висящий на странице банер через некоторое время перегружался, демонстрируя новую картинку.

Проще всего, конечно, это сделать, используя CGI скрипты, но не все могут их размещать у себя. К примеру, большинство халявных хостингов не предоставляет такой возможности. Выход — использовать JavaScript. Приведенный ниже скрипт перегружает банер через указанное количество миллисекунд.

Противопоказания к применению: многие спонсоры предоставляют вам кусок HTML кода, который менять категорически запрещают. В этом случае этот скрипт использовать не получиться, так как вам придется разобрать код на адрес ссылки и адрес картинки. Если все же очень надо, напишите спонсору жалостливое письмо с описанием ситуации — может и разрешат.

Все выделенное синим цветом заменяете на свое. Количество строчек с loc[..] и imsrc[..] должно соответствовать количеству банеров maxid.

```
<html>
<head>
<title> </title>
<script language="JavaScript">
var id=1;
var maxid=КОЛИЧЕСТВО банерОВ;
loc=new Array(maxid);
loc[1]="http://АДРЕС ССЫЛКИ ДЛЯ банерА 1";
loc[2]="http://АДРЕС ССЫЛКИ ДЛЯ банерА 2";
loc[3]="http://АДРЕС ССЫЛКИ ДЛЯ банерА 3";
loc[4]="http://АДРЕС ССЫЛКИ ДЛЯ банерА 4";
imsrc=new Array(maxid);
imsrc[1]="http://АДРЕС банерА 1";
imsrc[2]="http://АДРЕС банерА 2";
imsrc[3]="http://АДРЕС банерА 3";
imsrc[4]="http://АДРЕС банерА 4";
function got(){
window.location.href=loc[id];
}
function nextAd(){
if(++id>maxid)id=1;
document.ad.src=imsrc[id];
window.setTimeout('nextAd();',ЗАДЕРЖКА в миллисекундах);
}
</script>
<body onLoad="window.setTimeout('nextAd();',ЗАДЕРЖКА ПЕРЕД ПЕРВЫМ
ИЗМЕНЕНИЕМ банерА в миллисекундах);">
<!-- Здесь идет ваша страница -->
<!-- А следующую строчку следует вставить на то место, где у вас
должен быть банер -->
<a href="javascript:got();"> <img name="ad" alt="" src="АДРЕС
ПЕРВОГО банерА" ></a>
<!-- Здесь идет ваша страница -->
```

```
</body>
</html>
```

Глава 6. Бесплатное размещение web-страницы на сервере www.geocities.com

Вы можете бесплатно разместить свою Web-страничку и бесплатно получить адрес e-mail на сервере GeoCities. Эта компания полагает, что каждый человек имеет право разместить в киберпространстве собственную домашнюю страничку. Программа поддержки бесплатных Home Page создана для представления индивидуальных проектов в World Wide Web. Эта программа поддерживается спонсорами, которые имеют постоянные коммерческие Web-страницы.

Вам достаточно выполнить все указания Main Homesteading Page. Если ваш адрес электронной почты действительно существует, то на него придет письмо с информацией относительно использования вашей Web-странички.

GeoCities разработала уникальный Home Page Editor, с помощью которого вы можете создать персональную Web-страничку и специальная секция Neighborhoods, где выбирается тема страницы.

Выбираем тему

Вы можете выбрать тему для своей странички из двадцати девяти нижепубликуемых рубрик. Вам достаточно щелкнуть на названии понравившейся вам рубрики, получить дополнительную информацию, заполнить форму, получить письмо и, наконец, приступить к публикации собственного гипертекстового документа.

Area51 или Area51/Vault

Научная фантастика.

Athens или Athens/Acropolis

Все, что может быть связано с образованием. Преподаватели, философы, и все, кто интересуется литературой, будут чувствовать себя здесь как дома.

BourbonStreet

GeoCities не может не включить в свои рубрики тему, связанную с великолепным легким джазом.

Broadway

Театры и шоу.

CapeCanaveral

Наука и технологии. Инженерия, математика, авиация и т.д.

CapitolHill

Политика, государство и национальные интересы. Участники этой рубрики весьма смело высказываются обо всем этом.

CollegePark

Почувствуйте вкус университетской жизни!

Colosseum или Colosseum/Field

Если вы любите спорт, то здесь вы можете разместить соответствующую страничку. Олимпийские игры, профессиональный спорт и т.д.

Enchanted Forest

Место для детей. Игры, рассказы, учебные ссылки и домашние странички, созданные детьми.

Heartland или Heartland/Plains

Отцы, дети и проблемы в семье.

Hollywood или Hollywood/Hills

Кинематограф, телевидение и живое видео!

HotSprings

Центр вашего здоровья и профессиональной пригодности. Правильная пища, медицинские проблемы, лечебные курорты.

MotorCity

С волной пестрого флага, MotorCity приветствует автомобильных фанатов.

NapaValley

Откупорьте бутылку вина, возьмите прекрасную сигару и присоединяйтесь к NapaValley, чтобы стать истинным гурманом и ценителем вкусной пищи.

Paris или Paris/LeftBank

Сообщество романтиков. Великолепный Париж. Поэзия, искусство, прекрасное вино и континентальный образ жизни.

RainForest

Сохранение окружающей среды.

RodeoDrive

Магазин для весьма обеспеченных господ.

SiliconValley, SiliconValley/Park или SiliconValley/Heights or SiliconValley/Pines

Компьютеры и программное обеспечение. Самая популярная рубрика в GeoCities.

SoHo или SoHo/Lofts

Писатели и еще раз писатели.

SouthBeach или SouthBeach Marina

Местечко для праздношатающихся гуляк.

SunsetStrip или SunsetStrip/Alley, или SunsetStrip/Towers

Рок-н-ролл, блюзы, панк-клубы.

TheTropics

Каждый когда-нибудь мечтал оказаться в Раю. Рубрика для всех тех, кто в каникулы любит путешествовать.

TelevisionCity

Все о телевидении.

TimesSquare или TimesSquare/Arcade

Игры и приключения.

Tokyo или Tokyo/Towers

Для всех тех, кто интересуется Дальним Востоком.

Vienna

Классическая музыка, оперетта и балет.

WallStreet

Здесь GeoCitizens делает бизнес. Инвестиции, финансы, бизнес и коммерция.

WestHollywood

Геи, лесбиянки и прочие сексуальные меньшинства.

Yosemite

Любители свежего воздуха. Путешествия, альпинизм, лыжный спорт, рэфтинг, кемпинг и все то, что можно делать на свежем воздухе.

Три шага — и ваш мегабайт в киберпространстве!

Вам нужно сделать только три шага для того, чтобы ваша Home Page стала полновластным членом киберпространства. Ниже публикуется информация относительно этого.

Просмотрите каталог **Neighborhood Directory** и выберите тему для своей странички. Будьте уверены в том, что выбранная тема соответствует вашей страничке.

Найдите незанятый адрес в соответствующей секции **Neighborhood**. Для этого существует два способа. Выбрав из секции **Neighborhood** опцию **main page**, вы можете вести поиск через пронумерованные блоки адресов уже существующих страниц до тех пор, пока не найдете свободное место. Вы можете также просто щелкнуть на слове **Join** и следовать предложенным инструкциям. Выбрав **Yes** на следующей странице, вы сможете автоматически найти свободное место посредством поля данных **Vacancy Locator**.

Щелкните на **Apply for this address** для резервирования места под свою страничку.

После этого вы автоматически переместитесь на страничку с регистрационной формой. Здесь вы должны ввести свои персональные данные. Особо хотим подчеркнуть – правильно укажите ваш адрес электронной почты. Именно по этому адресу будет выслан пароль, без которого невозможен доступ к вашей страничке.

Подтверждение регистрации

Если вы правильно заполнили регистрационную форму, то вы получите так называемое подтверждение регистрации. Вас поприветствуют, сообщат выбранную рубрику и укажут электронный адрес, по которому будет выслана дополнительная информация. Если вам известен пароль, то вы можете изменить свой адрес электронной почты посредством **Personal Profile Editor**.

Поддерживаемые типы файлов

Вы можете просмотреть специальный список, в котором указаны все поддерживаемые типы файлов. Если вам необходима поддержка какого-либо другого типа, вы должны обратиться с соответствующим запросом на сервер Geocites. Отошлите соответствующее письмо по адре-

су: comments@geocities.com. В целях безопасности не поддерживаются скрипты CGI и EXE-файлы. Это, конечно, не относится к счетчикам, формам и картам картинок, разработанными Geocites.

Относительно файла index.html

Вы можете модифицировать свой документ, находясь у себя дома или делать все то же самое, используя Home Page. Несмотря на то, что в вашем каталоге может находиться несколько гипертекстовых страниц, файл вашей первой странички должен быть назван как index.html.

Именно этот файл будет загружать документ, когда кто-нибудь обратится по вашему адресу. Вы должны удалить старый файл index.html из вашего каталога. Это можно сделать посредством **Disk Usage Utility**, то есть специальной странички, которая позволяет манипулировать файлами вашего каталога. Все остальные гипертекстовые страницы могут иметь любые названия. Помните, что вы можете создать новую страницу непосредственно на сервере Geocites с помощью **Home Page Editor**.

Если вы изменили свою страничку через **Home Page Editor**, то вам достаточно нажать на кнопку **Create/View**. Теперь ваша страничка модифицируется и вы сможете увидеть произведенные изменения в режиме предварительного просмотра.

Соглашения относительно имен при пересылке гипертекстовых файлов

В пределах вашего документа HTML, вы должны использовать определенное соглашение, касающееся имен файлов, размещенных на сервере Geocites. Конечно, вы можете через встроенные ссылки обращаться к другим гипертекстовым документам WWW. Но файлы, переданные на сервер через FTP, должны удовлетворять следующим требованиям:

```
<IMG SRC = “image.gif”>
```

или

```
<A HREF = “util.zip”>
```

Вы можете обратиться к любому типу файла, который поддерживается сервером Geocites. Например, вы можете обратиться к страничке page2.html, находящейся внутри index.html, используя ссылку

```
<HREF = “hobbies.html”>
Это моя вторая страница </A>
```

и обратиться к картинке, находящейся в вашем текущем каталоге, используя ссылку

```
<IMG SRC = “photo.jpg”>
```


Использование протокола передачи

Вы можете пересылать ваши файлы на сервер ftp.geocities.com. В поле **Login** укажите **Member Name**, а в поле **Password** ваш пароль (эти две вещи должны быть вами получены от Geocites по электронной почте). Соединившись с ftp.geocities.com, вы автоматически окажетесь в вашем текущем каталоге. Этот каталог предназначен для файлов вашей Web-странички.

Вы не можете создавать подкаталоги на сервере Geocities. Передавайте все ваши файлы только в свой текущий каталог. Пересылка всех не текстовых файлов (GIF, JPG, ZIP) осуществляется в бинарном режиме или через команду **raw**. Файлы HTML и другие текстовые файлы должны передаваться в режиме ASCII.

Вы можете иметь в своем текущем каталоге столько файлов, сколько хотите, но пространство на диске ограничено для использования только одним мегабайтом.

Ваш собственный универсальный локатор ресурса

Каждый зарегистрированный пользователь получает собственный уникальный URL. Этот локатор вы можете сообщать всем поисковым службам Интернет. Ваш URL будет всегда начинаться с **http://www.geocities.com** и заканчиваться выбранной рубрикой и полученным номером вашего адреса. Например, если ваш адрес есть 2090 и находится в Athens Neighborhood, то ваш URL будет выглядеть так:

```
http://www.geocities.com/Athens/2090/
```

Предварительный просмотр, обновление и удаление

Ваши файлы будут автоматически перемещены посредством FTP в ваш текущий каталог. Вы можете просмотреть свою страничку, выбрав в рубрике **Neighborhood** соответствующий каталог и свой адрес.

Кроме этого, вы можете обновлять ваш гипертекстовый документ посредством обыкновенной пересылки файлов через FTP в ваш текущий каталог. Вы можете так же удалить все переданные файлы, используя утилиту **Personal Profile Deletion Utility**. Вам не нужно удалять старые файлы перед тем, как передавать новые. Когда вы будете передавать файлы, то в вашем каталоге файл с прежним именем просто автоматически перезапишется.

Удостоверьтесь в том, что имя нового файла точно такое же как текущий файл, особенно при посылке вашего файла index.html.

При использовании **Basic Home Page Editor** вы можете получить сообщение: *"Page not created with Home Page Editor"*, означающее, что ваша страничка не создана в HTML-редакторе Geocities. Вы пересылаете через FTP файл index.html и затем можете модифицировать основанный на этом файле гипертекстовый документ с помощью основного редактора Geocities или **Advanced HTML Editor**. Вы можете, конечно, не производить никаких изменений, а просто передать файл index.html через FTP или **EZ File Upload Utility**.

Скорее всего, это сообщение появилось из-за того, что вы добавили тэги HTML в строку Title основного или расширенного редактора, а система Geocities не смогла понять этого. Итак, если вы попали в такую ситуацию, то у вас имеется два решения:

- Не изменяя текущего аккаунта, удалите ваш профайл и измените свой адрес посредством **Profile Deletion Utility**
- Сделайте все будущие изменения вашего гипертекстового документа index.html через FTP или **Advanced HTML Editor**

Ссылки на другие страницы

Вы можете добавить в свой гипертекстовый документ ссылки на другие страницы, находящиеся в вашем текущем каталоге. Для этого вы должны пользоваться файлами с непосредственными ссылками. Например, если ваш файл index.html имеет ссылку ко второй странице в пределах вашего каталога, то такая ссылка должна быть организована следующим образом:

```
<a href="secondpage.html">
```

Это означает, что вам вообще не нужно указывать начало ссылки, то есть http:/www.geocities.com/.

Аналогичный метод используется и в том случае, если вы хотите связать вашу страницу со страницей другого пользователя.

Например, если вы хотите связаться с 1000 Colosseum, то вы просто пишите:

```
<a href="/Colosseum/1000/">
```

Скрипты

Вы не можете передавать скрипты на сервер Geocities. Это связано с безопасностью. Теоретически, запущенная на сервере скрипт-программа может открыть доступ ко всем областям сервера. Поэтому

Geocities предлагает только свои разработки. Вы можете использовать формы трех скриптов:

- imagemap
- counter
- mailto

Все остальные формы должны передаваться только на электронный адрес сервера. Они будут рассматриваться, изучаться, в общем, тестироваться.

Серверы, совершенно бесплатно предлагающие web-пространство для вашей web-страницы

http://www.geocities.com

http://www.xoom.com

http://www.hypermart.net

http://members.tripod.com

http://www.angelfire.com

http://www.crosswinds.net/index.html

http://www.fortunecity.com

http://www.fsn.net

http://a1bbs.dzone.co.kr

http://www.easyspace.com

http://www.cynetcity.com

http://www.cybercities.com

http://www.royaltystudios.com

http://www.internet-club.com

http://www.student.toplinks.com/freehome.htm

http://rampage.ml.org/freeinfo.html

http://www.cybercity.hko.net/

http://bip.concept.se/user

http://www.nether.net/

http://www.metrocity.net

http://mkn.co.uk/

http://www.nettaxi.com

http://home.onestop.net/

http://www.yi.com/

http://www.howdyneighbor.com/

http://www.schlund.de/

http://www.cqws.com/rates.html

http://jungle.fapema.br/

http://free.prohosting.com/

http://www.home.ch

http://www.earthonline.net/

http://www.freenation.com/

Часть 10. Уроки мастерства

Глава 1. Выбираем и настраиваем домашний Web-сервер

Часто возникает ситуация, когда необходимо проверить полный вид страницы. Однако чаще всего это невозможно при работе дома — технологии SSI и CGI, например, точно требуют сервера. Но как это сделать? И можно ли? Ответ: можно. Нужно установить на ваш домашний компьютер (пусть даже не подключенный к Интернету) специальную программу — Web-сервер.

К примеру, Apache — полноценный web-севеp. Вот далеко неполный список функций, им выполняемых:

- полноценный web сервер;
- proxy сервер;
- поддержка CGI;
- поддержка SSI;
- поддерживает виртуальные серверы.

Сколько же стоит такое удовольствие? А нисколько. Дело в том, что Apache — сервер, создававшийся усилиями массы программистов по всему миру. Apache — дитя Интернет, а следовательно — распространяется бесплатно.

Итак, что же надо сделать, чтобы Apache заработал на вашем компьютере? А нужно следующее. На сайте Apache надо найти и загрузить Apache для Win32 (то есть Win 95/98/NT). Загрузив дистрибутив сервера, запустите исполняемый файл. Во время установки следует указать директорию c:\usr\local\apache для установки вместо той, что стоит там по умолчанию. Дело в том, что такое расположение сервера соответствует расположению его или подобного на реальном сервере web, а чем ближе в плане конфигурации мы будем к серверу, тем лучше и эффективнее сможем работать.

Итак, после того, как инсталляция завершена, надо произвести конфигурацию сервера. Конфигурация Apache производится посредством изменения эдаких *.ini файлов, они имеют расширение *.conf.

Основной конфигурационный файл называется httpd.conf и отвечает за основную startup конфигурацию сервера. Файлы конфигурации лежат в /usr/local/apache/conf/, но после инсталляции вы их там не найдете. Однако они никуда не пропали — они все лежат в папке /usr/local/apache/.tmp/. Следующие файлы требуется скопировать в директорию /usr/local/apache/conf/:

- httpd.conf — основная конфигурация сервера;
- access.conf — конфигурация доступа к серверу;
- mime.types — типы расширений MIME;
- srm.conf — довесок к конфигурации.

Итак, первым (и единственным) файлом, который необходимо изменить будет httpd.conf. Откройте его в NotePad. Каждый раздел и параметр здесь очень умно комментирован, но не зная точно, что это значит, разобраться сложно.

Конфигурируя сервер, мы установим достаточно слабую защищенность — мы его дома будем использовать. «Слабую» в том плане, что, например, CGI у нас будут разрешены к исполнению везде.

Далее описаны инструкции, что они значат и чему должны быть равны.

- Server root — «корень» сервера. Должен быть равен c:/urs/local/apache, т.к. именно туда мы установили сервер;
- Server admin — адрес e-mail администратора сервера;
- Document root — путь к файлам http — документам;

После фразы **# This should be changed to whatever you set DocumentRoot to...** должно быть написано **<directory "тут_путь_к_файлам_http">**.

Следующий параметр (options) должен содержать слова Indexes, Includes и ExecCGI, что даст возможность серверу:

- исполнять CGI скрипты;
- показывать содержание директорий;
- выполнять инструкции SSI;

Перед инструкцией вставить две строчки:

```
AddHandler Server-Parsed .htm .html .shtml
AddHandler Cgi-Script .cgi
```

Эти инструкции определяют т.н. обработчики — указания серверу «***обрабатывать файлы *.xxx как...***»

Параметр **UserDir** стоит установить равным document root. Он отвечает за размещение директорий пользователей, в поисках которых при запросе www.address.com/~username пойдет сервер. То есть, искать он будет в userdir/username.

В **DirectoryIndex** можно добавить index.htm (многие делают не .html а .htm).

В **ServerName** пишется любое имя сервера.

На самом деле, все. Теперь можно запускать сервер. Еще посоветую сделать вот что: создайте ярлыки (например, на рабочий стол) на c:\usr\local\apache\apache.exe и второй на c:\usr\local\apache\apache.exe -k -shutdown, и обзовите их «Apache» и «Shut Apache Down». Таким образом, эти ярлыки будут запускать сервер и выключать его. Также можно сделать ярлык c:\usr\local\apache\apache.exe -k restart для перезапуска сервера после изменения файлов конфигурации.

Сервер Apache доступен буквально из любого браузера под Windows под адресом http://127.0.0.1/ или http://localhost/ после чего пишется адрес документа, лежащего под **Document Root**.

Глава 2.
Выводим иллюстрации в отдельном окне

Иногда необходимо организовать вывод графических изображений в отдельном окне, но вам хотелось бы, чтобы новое окно браузера не содержало кнопок навигации, адресной строки, полос прокрутки и чтобы его заданный первоначально физический размер был фиксированным. Как этого добиться? Достаточно просто: следует использовать несложный сценарий JavaScript, запрещающий изменение пользователем размеров окна и удаляющий из него все ненужные вам компоненты.

Скрипт, написанный на языке Java, интегрируется в web-страницу при помощи тэга **<SCRIPT>** с атрибутом **LANGUAGE**, который помещается в начало кода html-документа, либо между тэгами **<HEAD>** и **</HEAD>**, или же сразу после директивы **<BODY>**. Сам текст сценария не отображается в окне браузера при загрузке документа, он исполняет-

ся подобно подпрограмме в случае определенного действия пользователя, например, открытия или закрытия окна. Встроенные в web-страницу сценарии JavaScript интерпретируются браузером вместе с кодом разметки гипертекста и вызываются на исполнение из тела html-документа специальными директивами.

Итак, для того чтобы организовать вывод графического изображения в отдельном окне фиксированного размера без кнопок навигации и полос прокрутки, в тело html-документа необходимо поместить следующий код:

```
<BODY>
<SCRIPT LANGUAGE="JavaScript">
function picture( ){
window.open("URL. определяющий адрес расположения картинки",
"newwindow", config=''width-ширина картинки, height-высота
картинки, toolbar=0, location=0, directories=0, status=1,
menubar=0, scrollbars=0, resizable=0");
}
</SCRIPT>
Тело html-документа
</BODY>
```

Записываемый в коде сценария JavaScript адрес картинки лучше всего приводить в абсолютном виде, с указанием протокола передачи данных и имени целевого графического файла, например, http://www.myserver.ru/images/picture.jpg. Атрибут config определяет конфигурацию создаваемого окна. Значения параметров **width** и **height**, устанавливающих ширину и высоту картинки в пикселах, следует увеличить на 10-15 точек, поскольку в новом окне изображение будет иметь отступы сверху и снизу, а значит, определенная его часть может оказаться скрытой за границей окна. Остальные параметры атрибута config принимают одно из двух возможных значений: 0 — определяемый параметром элемент не отображается на экране и 1 — определяемый параметром объект отображается вместе с другим содержимым. Так, параметр **toolbar** выводит на экран (или не выводит) панель инструментов с функциональными кнопками, параметр **location** — адресную строку, **directories** — список директорий, **status** — строку состояния в нижней части окна, **menubar** — системную панель, содержащую меню **Файл**, **Правка**, **Вид** и т.д., параметр **scrollbars** — полосы прокрутки, и, наконец, параметр **resizable** разрешает или запрещает произвольное изменение размеров окна пользователем.

Определитель **picture( )**, записанный правее директивы **function**, является уникальным именем данной функции JavaScript, которое будет использовано в дальнейшем для вызова этой функции на исполнение. Если вы планируете выводить на экран подобным образом несколько

разных графических файлов, для каждого из них придется написать отдельную функцию, отличающуюся от предложенной выше лишь именем, адресом расположения целевого файла и значениями размеров картинки.

Для того чтобы активизировать созданный сценарий JavaScript, в теле html-документа необходимо разместить соответствующую гиперссылку, включающую ряд необходимых параметров. Для подготовки такой ссылки можно использовать графическую миниатюру рисунка, отображаемого браузером в отдельном динамически появляющемся окне. Код гиперссылки в этом случае будет выглядеть следующим образом:

```
<A HREF="javascript:picture()"><IMG SRC="URL миниатюры картинки"
WDTH="ширина миниатюры" HEIGHT="высота миниатюры"
BORDER="0"ALT="Альтернативный текст"></A>
```

где **picture()** — уникальное имя функции, вызываемой при активизации гиперссылки.

Для контроля всех перечисленных параметров используются возможности JavaScript, а конкретно метод open объекта window. Вот его все возможные параметры:

```
window.open('http://www.mysite.ru','namewin','top, left, menubar,
toolbar, location, directories, status, scrollbars, resizable,
width, height')
```

Здесь 3 группы параметров.

Рассмотрим эти параметры:

- http://www.mysite.ru — это адрес сайта,который открывается в новом окне браузера,
- **namewin** — это имя, которое будет присвоено открываемому окну.
- **top** — отступ открываемого окна от верхней части экрана. Значение задается в пикселах, например, top=100
- **left** — отступ открываемого окна от левой части экрана. Значение задается в пикселах, например, left=150
- **menubar** — определяет показывать строку меню браузера в открываемом окне или нет. Значениями являются yes или no, вы также можете использовать в виде значений соответственно 1 и 0.

- **toolbar** — определяет показывать в открываемом окне панель Обычные кнопки (назад, вперед) или нет. Например, toolbar=no.
- **location** — определяет показывать панель Адресная строка или нет. Например, location=0.
- **directories** — определяет показывать Ссылки, или нет. Например, directories=yes.
- **status** — определяет показывать строку состояния или нет. Например, status=1.
- **scrollbars** — определяет можно ли при необходимости, когда содержимое страницы не помещается на экране показывать панели прокрутки или нет. Например, scrollbars=yes.
- **resizable** — определяет возможность пользователя изменять размеры открываемого окна. Если задано значение resizable=0, то в открываемом окне недоступной становится кнопка «**Развернуть**» (при доступных «**Свернуть**» и «**Закрыть**»), а в Netscape 6 вообще не отображаются никакие кнопки, кроме доступной «**Закрыть**».
- **width** — ширина открываемого окна. Значение задается в пикселах, например, width=640
- **height** — высота открываемого окна. Значение задается в пикселах, например, height=480

Для использования метода open объекта window его сопоставляют с обработчиком события, рассмотрим это на примерах:

```
<form>
<input
onclick="window.open('http://www.mysite.ru','Window','top=20,
left=30, menubar=0, toolbar=0, location=0, directories=0,
status=0, scrollbars=1, resizable=0, width=800, height=600')"
type=button value="Жми тут ">
</form>
```

Мы создали форму с единственной кнопкой. Если пользователь нажмет на нее, то будет открыто новое окно браузера без каких-либо панелей. Рассмотрим создание текстовых ссылок, использовав вызов заранее созданной функции JavaScript, которая выполняет открытие нового окна:


```
<head>
<SCRIPT LANGUAGE="JavaScript">
<!--
function new_window()
{
window.open('http://www.mysite.ru','Brouserwindow','top=25,
left=40, menubar=0, toolbar=0, location=0, directories=0, sta-
tus=0, scrollbars=0, resizable=0, width=600, height=400');
}
// -->
</SCRIPT>
</head>
<a href="javascript: new_window()">Mysite.ru</a>
```

Непосредственно при нажатии на ссылку выполняется соответствующая функция JavaScript.

Глава 3. Добавляем страницу в Избранное

Наверняка вы встречали на некоторых сайтах Интернета текстовые или графические гиперссылки, при нажатии на которые данная страничка автоматически добавляется в список ресурсов, расположенный в меню Избранное браузера Microsoft Internet Explorer. Самым простым методом реализации такой функции является использование соответствующего сценария JavaScript.

Нарисуйте небольшое изображение, которое впоследствии сыграет роль графического представления вашей ссылки, после чего сохраните его на диске под именем favorites.gif. Можно переходить к редактированию содержимого web-страницы, адрес которой вы хотите добавить в список меню **Избранное**. В общем виде код данной функции будет выглядеть следующим образом:

```
<A HREF="javascript:window. external. AddFavorite (URL вашей
web-странички', 'Краткое описание вашей странички')"><IMG SRC="URL
расположения файла favorites.gif" ALT="Альтернативный текст"
BORDER="0" WIDTH="Ширина изображения" HEIGHT="Высота
изображения"></A>
```

А вот пример применения подобного сценария JavaScript:

```
<A HREF="javascript:window. external. AddFavorite
('http://www.xxx31337.ru'. 'Юморная страничка BAtvisa')"><IMG
SRC="../images/favorites.gif" ALT="Добавить сайт в 'Избранное'"
BORDER="0" WIDTH="15" HEIGHT="10"></A>
```

Этот текст размещается в том месте листинга html-документа, в котором вы хотели бы представить гиперссылку. Если вместо графической гиперссылки вы планируете использовать обычную, текстовую, код функции заметно упрощается:

```
<A HREF="javascript:window. external. AddFavorite
('URL вашей web-странички'. 'Краткое описание вашей странички')">
Добавить сайт в 'Избранное'</A>
```

Можно реализовать данную возможность иным, менее компактным и более трудоемким способом, а именно — с использованием специальной функции JavaScript, помещенной в заголовок html-документа. Для этого наберите в любой строке html-кода между тэгами **<HEAD>** и **</HEAD>** следующий текст:

```
<SCRIPT LANGUAGE="JavaScript">
var url="URL вашего сайта"
var title="Краткое описание вашего сайта "function bookmark(){
if (document.all)
window. external. AddFavorite(url.title)
}
</SCRIPT>
```

В предложенной выше функции, имеющей уникальное на данной странице имя bookmark, определены две переменные, url и title. Первая из них указывает на адрес, по которому размещается ваша страничка в сети Интернет, вторая содержит ее краткое описание. При вызове функции bookmark на исполнение браузер автоматически заносит значения обеих переменных в папку Избранное. Для того чтобы выполнить данный сценарий, в код html-документа необходимо добавить следующий текст гиперссылки:

```
<A HREF="javascript:bookmark()">Добавить сайт в 'Избранное'</A>
```

Вместо текста «**Добавить сайт в Избранное**» может быть использовано любое графическое изображение, специфицированное тэгом **<IMG>**.

Глава 4. «Откат назад» с помощью JavaScript

Все очень просто, нам надо применить вот такую форму:

```
<center><FORM><input style="font-size: 9pt"
onclick="history.back(-1)" type=button value="&lt;
Назад"></FORM></center>
```


history.back(-1) в скобках можем менять значение, на сколько страниц делать откат, ну конечно как правило это значение менять не требуется.

Это самый удобный способ, тут не надо указывать куда надо откатывать, а просто используется **Hstory back**. Тут в принципе все понятно, так что просто скопируйте и установите на странице.

Глава 5. Индикатор состояния ICQ

Если вы пользуетесь популярным Интернет-пейджером программой ICQ, вашу домашнюю страничку наверняка украсит небольшой графический индикатор, показывающий посетителям, находитесь ли вы в настоящий момент в режиме online или отключены от Интернета. Установить такой индикатор можно, включив в соответствующую строку листинга своей web-страницы отрывок html-кода, который будет иметь следующий вид:

```
<IMG SRC="http://online.mirabilis.com/scripts/online.dll?icq=
ВАШ_НОМЕР_ICQ&amp: img=КОД_ИЗОБРАЖЕНИЯ" WIDTH="Ширина картинки"
WIDTH="Высота картинки">
```

Вместо слов **ВАШ_НОМЕР_ICQ** в данный код следует вставить ваш ICQ UIN, строка «код изображения» определяет внешний вид индикатора, а параметры ширины и высоты картинки могут варьироваться в зависимости от выбранного вами графического представления индикатора. В качестве наглядного примера использования индикатора состояния ICQ в составе web-страницы, если принять, что номер ICQ — 31337, можно привести следующий отрывок кода:

```
<IMG
SRC="http://online.mirabilis.com/scripts/online.dll?icq=31337&arnp
:img=3"
WIDTH="249" HEIGHT="17">
```

Глава 6. Как поменять цвет скролл-бара

Для задания оформления скролл-бара необходимо прописать таблицу стилей:

```
<style type=text/css>
<!--
```

```
body { scrollbar-face-color:#000000;
scrollbar-highlight-color: #888888;
scrollbar-shadow-color: #666666;
scrollbar-3dlight-color: #666666;
scrollbar-arrow-color: #FFFFFF;
scrollbar-track-color: #333333;
scrollbar-darkshadow-color: #666666; }
//-->
</style>
```

scrollbar-face-color — задает основной цвет скролл-бара, а также цвет двух завершающих кнопочек со стрелочками. Если вы даже не укажите свойство **scrollbar-track-color**, то цвет трека определиться автоматически, как более яркий нежели цвет **scrollbar-face-color**.

scrollbar-highlight-color — устанавливает яркий цвет подсветки, создающий эффект объемности, то есть цвет, для освещенной части скролл-бара и кнопочек. Этим цветом отображается левый верхний угол самого скролл-бара и двух кнопочек (когда они не нажаты).

scrollbar-shadow-color — устанавливает темный цвет подсветки, создающий эффект объемности, то есть цвет, для теневой части скролл-бара и кнопочек (цвет тени). Этим цветом отображается правый нижний угол самого скролл-бара и двух кнопочек (когда они не нажаты).

scrollbar-3dlight-color — определяет цвет падающего цвета для создания трехмерности скролл-бара. На практике это выражается в цвете тоненькой кромки, находящейся в левых верхних углах, ближе к краю нежели **scrollbar-highlight-color**.

scrollbar-arrow-color — определяет цвет двух маленьких стрелочек, находящихся на кнопочках крайних позиций скролл-бара.

scrollbar-track-color — задает цвет трека, то есть дорожки прокрутки, по которой собственно и перемещается сам скролл-бар.

scrollbar-darkshadow-color — определяет цвет откинутого цвета для создания трехмерности скролл-бара. На практике это выражается в цвете тоненькой кромки, находящейся в правых нижних углах, ближе к краю нежели **scrollbar-shadow-color**.

Значение этих свойств (цвет) определяется в виде именованных цветов или кодов цветов (для этого ставится символ «#», а за ним без пробела шесть шестнадцатеричных чисел, определяющих цвет в кодировке RGB). Да и будьте осторожны, вы можете испортить весь стиль своего сайта...

Глава 7. Как «обмануть» фреймы

Очень часто бывает, когда посетители добавляя фреймовую страницу в Избранное, при последующем заходе по ссылке они видят лишь ту страницу, где текст, без меню, в таких случаях надо либо дублировать меню либо добавить скрипт, приведенный ниже (лучше расположить его в тэгах head).

Да, надо код ставить на все страницы вашего сайта, или будет выскакивать только та страница, где расположен этот код.

```
<script LANGUAGE="JavaScript">
<!--if (top.frames.length!=0) top.location=self.document.loca-
tion;//--></script>
```

Глава 8. Свойства тэга mailto

Если нам надо разместить на странице ссылку для отправки электронной почты мы обычно пишем вот такое:

```
<a href=mailto:w-w@list.ru>Письмо мастеру</a>
```

Еще мы можем заменить надпись любой картинкой:

```
<a href="mailto:w-w@list.ru"><img src="folder.gif" width="16"
height="16" border="0" alt="w-w@list.ru">
```

Еще возможна отправка с автоматическим заполнением Темы письма и Заголовка в вашем почтовом клиенте, вот пример:

```
<a href="mailto:w-w@list.ru?subject=Пишу с твоего сайта&Body=
Уважаемый мастер!"> Письмо мастеру</a>
```

Глава 9. Пример практического создания сайта

Представьте себе, что вы решили написать в Инете о себе любимом, ну и чуть-чуть о своей подруге. Далеко не лучшая идея — вряд ли кому-то это интересно, разве ближайшим друзьям и родственникам. Но для примера сойдет.

Начнем, пожалуй, с того, что застолбим себе местечко на каком-нибудь сервере, появится стимул побыстрее его заполнить (предполагается, что почтовый ящик у вас уже есть). Возьмем, к примеру, Fortune

City — дают 100 МБ. Думаю, достаточно. Для регистрации жмем вот сюда, выбираем тематику сайта (не мучайтесь, любую выбирайте!), имя пользователя и пароль. Далее заполняем предлагаемые формы. Если вы не очень уверены в своем английском, то предварительно кликните ссылку Registration Help! в верхней части страницы. Там найдете переводы всех форм, которые предстоит заполнять при регистрации на FortuneCity.

...Ну вот. Место у вас есть, и появился адрес. Что-то вроде http://www.fortunecity.com/business/fax/339.

Принимаемся за создание странички. Прикидываем, как все будет выглядеть: заглавная страничка будет о себе, про девушку — отдельная страничка. Еще у вас есть парочка отсканированных фотографий, по штучке на каждую страничку пристроим обязательно. Т.е. наш сайт будет состоять из следующих файлов:

- заглавная страничка — index.html (обычно она всегда так называется);
- страничка про девушку — назовем ее girl.html;
- две фотографии — пусть это будут myphoto.gif и herphoto.gif.

Все файлы расположим в одной папке (их всего 4, поэтому и мудрить не будем).

Теперь для создания самих страничек можете воспользоваться Front Page или Word. А можно открыть что-нибудь типа Блокнота или WordPad и сделать все вручную. Мне лично второй вариант больше нравится. Итак, набираем следующее (назначение тэгов — это то, что находится между символами «<» и «>»):

```
<HTML>
<HEAD>
<TITLE>Обо мне и моей любимой</TITLE>
</HEAD>
<BODY BGCOLOR="#A0F0F0">
<H2 ALIGN=CENTER>Тимон Тимонович</H2>
<IMG SRC="myphoto.gif" WIDTH="122" HIGHT="183" ALIGN=LEFT ALT="Моя
фотка">
<P>Это я! Родился, рос, потом учился, и все время рос. Вот такое
у меня бурное детство...</P>
<P>Чем я занимаюсь сейчас? Да все тем же, чем и раньше - ем и пи-
саю.<BR>
<I>Вот такой вот я есть!</I></P>
```



```
<HR>
<P ALIGN=CENTER>
Про мою девушку вы можете узнать <A HREF="girl.html">здесь</A>.
</P>
</BODY>
</HTML>
```

Этот файл назовите index.html и сохраните на диске. Для этого выберите в меню **«Файл»** опцию **«Сохранить как...»**. Выберите тип файла **«Текстовый документ»**, а имя — «index.html». Теперь наберем следующий файл:

```
<HTML>
<HEAD>
<TITLE>Обо мне и моей любимой</TITLE>
</HEAD>
<BODY BGCOLOR="#A0F0F0">
<H2 ALIGN=CENTER>Тима</H2>
<IMG SRC="herphoto.gif" WIDTH="128" HIGHT="185" ALIGN=RIGHT
ALT="Фото моей девушки">
<P>Знаете, ее биография не очень-то отличается от моей...</P>
<P>Лучше про ее маму расскажу:
<UL>
<LI>
<I>На днях теща 50 копеек проглотила. Подавилась. И знаешь, ме-
лочь, а приятно!</I>
<BR><BR>
<LI>
<I>Вчера хоронили тещу. Порвали три баяна...</I>
</UL></P>
<HR>
<P ALIGN=CENTER>
Чтобы вернуться ко мне, <A HREF="index.html">жмите здесь</A>.
</P>
</BODY>
</HTML>
```

Сохраните этот файл как girl.html. Сюда же скопируйте myphoto.gif и herphoto.gif. Можете скопировать любые два графических файла с расширением gif из имеющихся в наличии. Главное не забудьте их переименовать в myphoto.gif и herphoto.gif. Теперь откройте папку, куда вы все сложили, дважды кликните по index.html и увидите результат своих трудов.

Осталось только перебросить все это хозяйство на сервер (upload). Можно использовать только браузер (обновление по WWW), а можно

FTP-клиент (по FTP). Второй вариант гораздо быстрее и удобнее. Рассмотрим оба варианта.

Upload по WWW

Заходим на www.fortunecity.com, находим вверху ссылку «Build» («Создать») и жмем на нее, затем выбираем «Update with advanced homebuilder» и в появившемся окне вводим свой логин и пароль. Попадаете на страничку «build» — строительство своего сайта. Там пока пусто. В верхней части страницы — ваш адрес (чтоб не забыли!). Ниже — список имеющихся на вашем сайте файлов (собственно говоря, пока там ничего не имеется) и их размер: html files (HTML-файлы), graphic files (графические файлы), other files (другие файлы) и directories (папки). Кнопки: edit (редактировать), rename (переименовать), view (просмотреть), delete (удалить), create directory (создать папку), create new html file (создать новый HTML файл). Так как нам пока редактировать и удалять нечего, смотрим далее. Внизу страницы под заголовком «**Uploading files**» находим 10 окошек с кнопками «**Обзор**». Жмем на одну из кнопок, в открывшемся окне выбираем свой файл «**index.html**», нажимаем открыть. То же повторяем и для «girl.html», «myphoto.gif» и «herphoto.html». Еще чуть ниже нажимаем на «upload files». Готово. Теперь набираем в окне браузера адрес своей странички и смотрим, что получилось.

Upload по FTP

Рассмотрим на примере программы CuteFTP. Сперва настраиваем ее для работы со своим сайтом: в FTP Site Manager (открывается при каждом запуске программы) нажимаем **Add Site**. Придумываем Site Label, можно любое имя (ну, например, site01). **Host Address** — вводим ftp.fortunecity.com, **User ID** — ваш логин для входа на сайт, **Password** — пароль, остальное — по умолчанию. Нажимаем «**Connect**». Если все сделали правильно и сервер FortuneCity не глючит (что иногда случается с ним), то менее, чем через минуту мы можем редактировать свой сайт. В левом окне выбираем файлы, которые следует загрузить на сервер, нажимаем upload (стрелка вверх) или в меню **Commands** выбираем **upload** — выбранные файлы выгружаются на сервер. Все. Закрывайте программу и заходите на свой сайт.

Часть 11. Тонкости и секреты

Глава 1. Фреймы

Для начала давайте решим, для чего используются фреймы. Они появились еще во второй версии Netscape Navigator и предназначались для облегчения навигации при создании страничек (так тогда казалось).

Техника использования фреймов заключается в том, что все окно браузера делится на несколько областей, в каждую из которых можно загрузить независимую страничку. Кроме этого, был введен механизм, позволяющий управлять любой страничкой из любого окна. Например, можно в одном окне организовать меню сайта, а в другом показывать его содержимое. Причем, щелчок по ссылке в окне меню открывал страничку совсем в другом окне. Такое построение сайта встречается чаще всего, но ничто не мешает нам сделать не два фрейма, а, например, 3, 4, 5... и т.д. Но пока не будем торопиться с обсуждением механизма работы фреймов, а поговорим об их недостатках и уместности применения.

Основные недостатки сайта, построенного с применением фреймов, следующие:

- Странички сайта не индексируются обычными поисковыми системами, исключая первую страницу. Это происходит из-за того, что страница описания фреймов не содержит в себе ссылок вида **...** и поисковые роботы, естественно, не могут попасть на внутренние странички. Приходится прикладывать дополнительные усилия, чтобы все же осуществить это. В частности, вводить вышеупомянутые ссылки именно для роботов.
- В случае попадания не на первую страничку сайта не существует «официального» способа перейти на первую страничку сайта — приходится вручную редактировать путь в адресной строке браузера.

- В ввиду того, что фреймовая структура сайта придает ему достаточно узнаваемый вид, то большинство подобных страничек выглядят достаточно однообразно.
- Невозможно поставить закладку на внутреннюю страничку сайта. А это уже серьезно! Представьте, что вы наткнулись на очень интересную статью и желаете, например, поместить ссылку в свою коллекцию или послать другу ее адрес. Так вот, ни то, ни другое вы сделать не сможете — фреймы скрывают истинный адрес странички. Ради справедливости, нужно сказать, что этот адрес все же можно узнать, открывая ссылку в новом окне.
- Проблемы отображения странички в разных версиях браузеров. От этого уже никуда не уйти и это предложение можно автоматически добавлять практически к любым расширениям HTML.

В каких случаях уместно применять фреймы?

Честно говоря, есть очень мало задач, которые нельзя решить без использования фреймов. Первоначально введенные для того, чтобы облегчить создание механизма навигации по сайту, на данный момент они, по-моему, ясно показали, что не справляются с этой задачей. Для этой цели гораздо лучше подходит технология SSI.

Тем не менее, в каких случаях оправдано их применение:

- в случае, если стоит задача быстро создать сайт и все странички уже написаны, а ни времени, ни желания их переделывать нет
- если нужно, чтобы часть странички (чаще всего логотип или меню) всегда находились перед глазами
- для дизайнерских изысков, наконец...

Но есть один удачный способ применения фреймов — создание системы помощи для сложных сайтов. В этом случае очень удобно открывать новое окно, где уже и используется фреймовая структура. Это очень похоже на встроенную систему помощи Windows.

Механизм работы фреймов

Если все же решено применять фреймы, то давайте разберемся с тем, как правильно писать HTML-код и работать с ними.

Любая страничка, содержащая фреймы, начинается с написания специальной странички-контейнера, которая сама не показывается, но

содержит в себе указания для организации фреймовой структуры и ссылок на участвующие файлы. Вот, как она выглядит:

index.htm — страничка контейнер.

```
<html>
<head>
<title>Frame page</title>
</head>

<frameset cols="160,*">
<frame src="left.htm" name="menu" scrolling=no noresize>
<frame src="right.htm" name="content" scrolling=no noresize>
</frameset>

<noframes>
<p>Ваш браузер не поддерживает фреймы, пожалуйста, обновите
его.</p>
</noframes>

</html>
```

И сразу напишем код для страничек, входящих в фреймовую структуру.

left.htm — страничка, содержащая меню.

```
<html>
<head>
<title>Menu page</title>
</head>

<body>

<a href="topic_1.htm" target="content">topic #1</a><br>
<a href="topic_2.htm" target="content">topic #2</a><br>
<a href="topic_3.htm" target="content">topic #3</a><br>
<a href="topic_4.htm" target="content">topic #4</a><br>

</body>
</html>
```

right.htm — страничка, в которой будут отображаться основные статьи сайта.

```
<html>
<head>
<title>Content page</title>
```

```
</head>

<body>

<p>Эта страничка будет грузиться по умолчанию, и
на ней обычно размещают приветственный текст.</p>

</body>
</html>
```

Как можно заметить, страничка-контейнер отличается от обычной лишь тем, что вместо тэга **<body>** мы используем тэг **<frameset>**, в котором непосредственно и определяются фреймы. Но сам по себе он описывает лишь количество и размеры фреймов и не описывает странички, входящие во фрейм. Это можно сделать при помощи тэга **<frame>**. Давайте-ка подробнее разберем наш пример:

```
<frameset cols="160,*">
<frame src="left.htm" name="menu" scrolling=no noresize>
<frame src="right.htm" name="content" scrolling=no noresize>
</frameset>
```

В параметре тэга **<frameset>** мы определяем, что страничка будет разделена по вертикали и состоять из двух колонок (если нам нужно разделить окно по горизонтали, то тогда вместо параметра cols, нужно применять параметр rows).

В принципе, колонок может быть и больше, чем две. Их количество определяется значением параметра cols (rows), которое представляет собой цифры и знаки, разделенные запятыми. Каждое значение определяет ширину колонки. Ширина, как, впрочем, и высота, может задаваться в процентах, пикселах и при помощи знака звездочки, обозначающей — «все оставшееся пространство». Вот примеры определения фреймов:

- **<frameset cols="200,*">** — 2 колонки, одна из которых имеет фиксированную ширину в 200 пикселей.
- **<frameset rows="25%, 50%, 25%">** — 3 строчки, высоты которых определены в процентах от высоты окна браузера.
- **<frameset rows="*, 2* ,*">** — то же самое, что и предыдущая строчка, но записанная при помощи звездочек. Цифра перед звездочкой указывает количество повторов. Ширина одной звездочки определяется как среднее арифметическое между всеми звездочками в строке с учетом коэффициентов перед ними.

Тэг **<frameset>** является контейнером, и в качестве его элементов должны быть написаны строчки для каждой колонки (строки) при помощи тэга **<frame>**. Вот пример описания:

```
<frame src="left.htm" name="menu" scrolling=no noresize>
```

src — URL странички, которая будет помещена во фрейм.

name — имя странички, по которому к ней можно будет обращаться.

scrolling, noresize, ... — параметры, управляющие поведением фрейма — возможностью изменять пользователем ширину фрейма и управлять появлением полосы прокрутки в случае, если содержимое странички не помещается на экране.

Обратите внимание на параметр **name** — он является очень важным и позволяет нам манипулировать с фреймом. Как же это осуществляется? Внимательно рассмотрим файл left.htm, в котором содержится наше меню. В самом файле нет ничего необычного, а вот в написании ссылок на странички есть. Если присмотреться, то можно заметить, что добавился еще один параметр **target="content"** — вот именно он и отвечает за то, в каком окне будет загружаться файл по ссылке:

```
<a href="topic_1.htm" target="content">topic #1</a>
```

В качестве значения параметра указывается то самое имя, которое мы присвоили нашему фрейму. Таким образом, мы можем открыть любую ссылку из любого окна и в любом окне. Достаточно лишь знать его имя.

Кроме имен, определяемых непосредственно нами, есть часть уже определенных, и наиболее важные из них следующие:

- **_blank** — открывает ссылку в новом окне
- **_top** — открывает ссылку на все окно, если она находилась во фрейме. Другими словами, она разрушает структуру фрейма и загружает файл во все окно.
- **_parent** — открывает ссылку в родительском окне.

До сих пор в качестве элементов контейнера **<frameset>** мы использовали простые тэги **<frame>**, но мы также можем использовать и контейнер **<frameset>**. В этом случае мы получаем т.н. вложенные фреймы. Вот пример:

```
<frameset rows="40,*">
<frame src="up.htm" name="logo" scrolling=no noresize>
<frameset cols="160,*">
<frame src="left.htm" name="menu" scrolling=no noresize>
```

```
<frame src="right.htm" name="content" scrolling=no noresize>
</frameset>
</frameset>
```

В данном случае определены три фрейма — один горизонтальный вверху, который мы можем использовать, например, как логотип. И два вертикальных, которые разделили второй горизонтальный фрейм.

В общем случае можно использовать сколько угодно фреймов (в пределах разумного, конечно).

Небольшое замечание. В принципе, в качестве источника фрейма может быть указана не HTML-страничка, а рисунок в формате GIF или JPEG, но применение такого механизма считается дурным тоном.

Теперь о подводных камешках. Самый большой из них следующий — Netscape Navigator не точно соблюдает ширину фрейма в пикселах и может принимать лишь некие кратные шаги. Поэтому в случае, если возникла такая проблема, нужно просто подкорректировать ширину фрейма под стандарты Netscape.

Кстати, рамку фрейма можно убрать, указав параметр:

```
frameborder=0
```

И еще одно! Для браузеров, которые не понимают фреймов, есть специальный тэг **<noframes>**. Как раз он и определяет, что увидит пользователь с браузером, который не поддерживает фреймов. Обычно в этом месте либо размещают надпись, либо указывают список на основные раздела сайта, чтобы человек все же смог просмотреть странички.

Глава 2. Ускоряем загрузку графики

Многие начинающие web-дизайнеры, разрабатывая свой первый самостоятельный проект, сталкиваются иногда с весьма трудной задачей, связанной с отображением графики в документах HTML. Проблема заключается в том, что согласно изученным вами ранее «постулатам» web-дизайна каждый html-документ должен быть максимально компактным, тогда как большое количество иллюстраций, появляющихся, например, при создании виртуального фотоальбома, значительно перегружает файл, пропорционально увеличивая время его считывания с сервера.

Решить эту проблему, оптимизировав объем итогового html-документа, можно двумя методами. Первый метод заключается в следующем. Создавая фотоальбом, подготовьте две копии каждого изображения: одну — нормального размера и качества, такую, которую вы и хотели бы поместить на свою web-страничку, вторую — в уменьшенном масштабе и с низким разрешением.

Разместите на web-странице уменьшенные копии картинок, назначив для каждой из них свойства гиперссылки с атрибутом **TARGET="_blank"**, которая перенаправит пользователя к большому графическому файлу. В документ вставьте пояснение для посетителей о том, что щелчок кнопкой мыши на любом рисунке приведет к его загрузке в отдельном окне браузера в увеличенном масштабе. Тогда пользователю, не заинтересованному в разглядывании ваших фотографий, не придется подолгу дожидаться окончания считывания странички с удаленного узла. Если же он все-таки захочет полюбоваться на некоторые из опубликованных вами иллюстраций, ему не составит труда выбрать наиболее интересные из них. Такой подход к представлению графики в Интернете носит название «предварительного просмотра», или, по-английски, «preview».

Если мы назовем файл, содержащий уменьшенную копию картинки, picture1.jpg, а файл нормального размера — picture2.jpg, то html-код гиперссылки будет выглядеть следующим образом:

```
<A HREF="picture2.jpg" TARGET="_blank"><IMG SRC="../picture1.JPG"
WIDTH="100" HEIGHT="98" ALT="Уменьшенная картинка" BORDER="0"></A>
```

Второй метод ускорения загрузки графических файлов с сервера несколько отличается от предыдущего. Если, предположим, ваш сайт содержит фото галерею, доступ к которой открывается с пятой — шестой странички по счету, все содержащиеся в ней графические файлы можно загрузить в клиентский компьютер заранее, на предыдущих страницах, выводя их на экран размером 1x1 пиксел. Код, осуществляющий подобный вывод графики, выглядит так: **<IMGSRC="URL графического файла" WIDTH="1" HEIGHT="1">**. Изображение при этом станет невидимым для посетителя сайта, но сохранится в кэше его машины, а при последующем открытии данной картинки она будет загружаться уже с жесткого диска локального компьютера. Таким образом, у пользователя создается впечатление, что изображение выводится на экран очень быстро. Однако за все надо платить: время загрузки предыдущих страниц при этом пропорционально увеличивается. Дабы посетитель вашего сайта не скучал, в процессе выгрузки графики его можно развлечь чем-нибудь другим: например, предложить почитать новости или подборку свежих анекдотов.

Глава 3.
Создаем систему быстрой навигации

Если объем разрабатываемого вами сайта достаточно велик, к тому же он имеет сложную логическую структуру, а количество составляющих его документов и разделов превышает возможности человеческой памяти или фантазии, пользователю не всегда представляется удобным использовать для путешествия по такому ресурсу стандартные средства навигации, выполненные в виде кнопок или текстовых ссылок.

Для того чтобы облегчить посетителям задачу поиска среди обилия опубликованных вами документов того, который им нужен, опытные web-дизайнеры дублируют традиционные кнопки, направляющие пользователя к основным разделам ресурса, системой быстрой навигации, представляющей собой выпадающее меню.

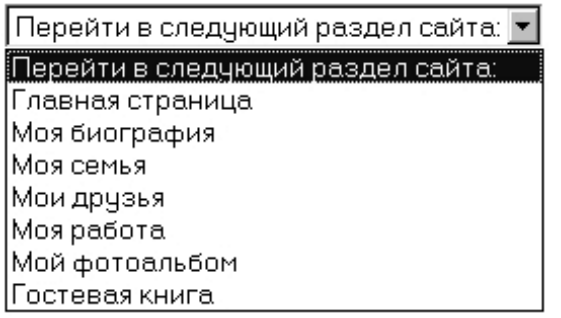
Выбор одного из пунктов этого меню переводит посетителя на соответствующую web-страницу. Такое меню можно создать с применением стандартных средств JavaScript.

Общий вид кода формы меню, выполняющего функции системы быстрой навигации, таков:

```
<FORM>
<SELECT NAME="form1" SIZE="1"
onchange="top. location.
href=this.options[this.selectedindex].value;
this.selectedindex=0">
<OPTION SELECTED>Перейти в следующий раздел сайта:</OPTION>
<OPTION VALUE="URL первой странички">Страница 1</OPTION>
<OPTION VALUE="URL второй странички">Страница 2</OPTION>
<OPTION VALUE="URL N-ой странички">Страница N</OPTION>
</SELECT>
</FORM>
```

В качестве значения атрибута **value** тэга **OPTION** следует указать полный либо сокращенный адрес URL, по которому находится страничка, определяемая каждым пунктом меню.

Между тэгами **<OPTION>** и **</OPTION>** записывается название данной странички. Внешний вид отображенной в окне браузера формы быстрой навигации по сайту показан ниже.



Если вы хотите, чтобы документ, открываемый при выборе какого-либо пункта навигационного меню, загружался в новом окне, вторую строку предложенного выше кода необходимо заменить следующей:

```
<SELECT NAME="form1" SIZE="1"
onchange="window.open(this.options[this.selectedlndex].value):
this.selectedindex=0">
```

Глава 4.
Защитим страницу паролем

В ряде случаев возникает необходимость запретить доступ к какому-либо разделу вашего сайта неавторизованным пользователям, то есть лицам, не знающим соответствующий пароль. Традиционный алгоритм установки пароля на web-страницу подразумевает использование подключенного к обычной форме CGI-скрипта, управляющего процедурой передачи данных в момент нажатия пользователем кнопки Отправить и осуществляющего процесс авторизации.

Примечание: Авторизацией называется процедура проверки введенного пользователем в какую-либо интерактивную форму пароля или сочетания логического идентификационного имени и пароля на подлинность. Очевидно, что применение такого метода на практике неизбежно связано с необходимостью размещения сценариев CGI в специальной серверной директории и запуска программ на удаленном узле, что требует как минимум наличия у вас прав администратора. Альтернативным путем, не вызывающим необходимости вступать в переговоры с владельцами сервера, предоставляющего вам web-хостинг, для получения у них разрешения на использование технологии CGI, является простая защита вашего сайта от несанкционированного доступа посредством применения JavaScript.

Общий механизм такой защиты действует следующим образом. В одной из директорий вашего сайта с условным названием private создается независимый раздел, содержимое которого и будет составлять вашу защищенную зону. В другой папке с условным названием access содержится пустой html-документ, имя которого совпадает с придуманным вами паролем, а заголовок **<HEAD>** содержит META-определитель, осуществляющий автоматическую переадресацию(редирект) браузера на стартовую страницу защищенной зоны. Постараюсь пояснить изложенную здесь методику чуть подробнее. Если, например, стартовый документ закрытого паролем раздела сохранен в файле с именем getting.html, асам пароль представляет собой следующую последовательность цифр и символов: xQ17kDFr4ty, то файл, хранящийся в папке access, должен будет называться xQ17kDFr4ty.html и содержать в заголовке директиву **<META HTTP-EQUIV="Refresh". CONTENT="10: URL=. ./private/getting.html">**. Теперь вам осталось только разместить на web-странице, с которой осуществляется доступ в закрытый раздел, содержащую JavaScript форму с предложением ввести в ее текстовое поле пароль.

После нажатия на кнопку **Отправить** JavaScript автоматически добавит вконец заданной пользователем символьной последовательности расширение .html и переадресует браузер к web-странице с получившимся названием, которая, в свою очередь, переадресует его к стартовому документу защищенной зоны. Очевидно, что такой алгоритм позволяет размещать в папке access неограниченное количество страниц с именами, соответствующими различным паролям, благодаря чему вы можете менять число и состав посетителей, имеющих доступ к закрытому разделу.

Следует отметить, что разработка такой элементарной защиты имеет смысл лишь в том случае, когда в настройках web-сервера, предоставляющего вам хостинг, установлена функция отображения файла not_found.html с сообщением об ошибке, если пользователь пытается загрузить страницу с неправильным именем или отсутствующий на сервере документ. Если данная функция отключена, сервер продемонстрирует в рабочем окне браузера список всех файлов, имеющихся в данном разделе его диска, причем каждый из них пользователь сможет открыть по собственному желанию.

Администрация многих узлов, предоставляющих бесплатный web-хостинг, понимает, что содержимое пользовательских директорий может являться информацией сугубо конфиденциальной, и потому запрещает посетителям доступ к списку файлов, составляющих сайты своих клиентов.

Очевидно также, что существует элементарный способ обойти такую несложную защиту: достаточно узнать у имеющих доступ к закрытой зоне вашего ресурса пользователей имя ее стартового файла и название директории, в которой он размещается. Выход из этой ситуации может быть только один: почаще меняйте пароли и название стартового документа защищенного раздела. А теперь самое главное — код интерактивной формы, которая позволит пользователям осуществить переход в защищенную зону вашего ресурса:

```
<FORM NAME="access1">
Пожалуйста, введите пароль:
<INPUT NAME="passwd" TYPE="password">
<INPUT TYPE="button" VALUE="Отправить"
onclick=(window. location. href=document. accessl. passwd value
+" html")>
</FORM>
```

Глава 5. Устанавливаем счетчик

Счетчик посещений представляет собой интерактивный графический элемент дизайна web-страницы, управляемый соответствующим сценарием CGI с данного либо с удаленного сервера. Изображение счетчика состоит из набора отдельных графических файлов, каждый из которых содержит символ одной цифры. CGI-скрипт фиксирует открытие документа в клиентском браузере и прибавляет к последнему значению счетчика единицу, соответствующим образом изменяя составляющую его картинку. Таким образом, владелец ресурса может определить количество посещений, или «хитов» (от англ., hit — нажатие), имевших место в течение какого-либо интервала времени.

Внешнее оформление счетчиков может быть совершенно различным и зависит, в первую очередь, от фантазии создававшего их дизайнера.

Сами же счетчики можно условно разделить на две обширные категории по их функциональному назначению. Первая категория включает в себя счетчики, изначально предназначенные исключительно для подсчета посещений.

Поскольку включение в состав web-страницы такого счетчика требует от разработчика ресурса администраторского доступа к серверной директории, в которой размещаются CGI-скрипты, можно пойти несколько иным путем: зарегистрироваться на одном из многочислен-

ных сайтов, предлагающих посетителям установку бесплатных счетчиков, и пользоваться сценарием CGI, запускаемым с этого узла. После регистрации вы получите отрывок html-кода, который следует просто вставить в листинг вашей web-страницы через буфер обмена. В качестве своеобразной оплаты за это сервер, сдавший вам в аренду один из своих счетчиков, включит в его изображение небольшой рекламный банер либо станет еженедельно отсылать в ваш почтовый ящик корреспонденцию с подробной статистикой посещений вашей web-странички за истекшие семь дней и, разумеется, неизбежной рекламой.

Счетчики предоставляются пользователю популярными поисковыми машинами, и помимо своей основной функции они осуществляют подсчет рейтинга странички в соответствующем разделе тематического каталога данной поисковой системы. Рейтинг определяется количеством уникальных посетителей вашего сайта в сутки: CGI-скрипт, установленный на поисковом сервере, фиксирует IP-адрес посетителя, открывающего данную страницу в своем браузере, и текущее состояние счетчика изменяется на единицу только в том случае, если этот IP-адрес не содержится в базе посещений вашего сайта, обновляющейся каждые двадцать четыре часа. Повторные загрузки документа в одну и ту же клиентскую машину игнорируются. Сделано это с целью пресечь попытки владельцев сайтов искусственно «накрутить» показания счетчика, поскольку от этих показаний зависит положение вашей странички в каталоге ресурсов поискового сервера: чем больше посещений — тем выше ссылка на ваш сайт находится в списке прочих ресурсов Интернета и, соответственно, тем больше новых пользователей каталога станут вашими читателями.

Для того чтобы получить такой счетчик и установить его на своей страничке, вам необходимо зарегистрироваться в соответствующем разделе выбранной вами поисковой системы, ввести в форму регистрации адрес, по которому размещается ваш сайт (если ресурс еще не создан или не опубликован в Интернете, поисковая система откажется от регистрации), указать внешний вид счетчика, в случае если возможно несколько вариантов, и, наконец, поместить через буфер обмена предложенный вам поисковой машиной код в соответствующую строку листинга html-документа. Счетчик запущен. Давайте кратко рассмотрим особенности и возможности нескольких наиболее популярных счетчиков:

- Rambler (http://www.rambler.ru). Счетчик, предоставляемый этой поисковой системой, считается на сегодня наиболее популярным и авторитетным в русскоязычной части Интернета. Пользователю предлагаются три основных статистических параметра: количество уникальных хитов, количество уникальных посетителей и количество повторных перезагрузок

страницы. На самом сервере www.rambler.ru можно при желании получить более подробную статистику, например, среднее количество посетителей по часам и по дням недели. Возможен выбор из множества вариантов внешнего оформления счетчика, оптимизированных для разного цветового решения страницы. Можно выбрать счетчик, не демонстрирующий посетителю количество хитов, а просто указывающий на то, что данная страница входит в состав пользователей системы рейтинга Rambler.

- Aport (http://www.aport.ru). Данный счетчик во многом аналогичен счетчику поисковой системы Rambler и обсчитывает тот же ассортимент статистических параметров. Подробную статистику посещений за истекшие сутки можно также получить на сервере поисковой машины.

- SpylLOG (http://www.spylog.ru). Счетчик Spy LOG является наиболее полным из всех счетчиков, предоставляемых другими аналогичными системами. Он отражает количество посетителей, навестивших ресурс с момента установки счетчика на странице, за истекшие 24 часа, а также количество пользователей, присутствующих на сайте в настоящий момент времени. Имеется широкий выбор из множества вариантов оформления счетчика, причем все они достаточно удачны с тонки зрения дизайна. На сервере http://www.spylog.ru доступна подробная статистика посещений использующей счетчик странички, причем данные обсчитываются более чем по 600 параметрам. Вам продемонстрируют распределение ваших пользователей по регионам, часам и дням недели, предоставят данные о том, каким программным обеспечением они пользуются для выхода в Интернет, покажут наиболее популярные маршруты по ресурсу, наиболее популярные разделы сайта и предоставят множество других полезных данных.

- List.RU (http://www.list.ru). Этот счетчик является весьма популярным в российском Интернете. На его изображении индицируется информация о хитах и уникальных посещениях, те же данные можно получить на сервере статистики в сводке за последние 30 суток. Помимо прочего, List.RU помесячно обсчитывает информацию о локации ваших читателей по странам мира

и формирует отчет о других web-страницах, на которых размещена ссылка на ваш ресурс.

- Count.RU (http://www.count.ru). Статистическая служба Count.RU появилась на свет относительно недавно, но уже завоевала признание многочисленных владельцев интернетовских информационных ресурсов. Помимо традиционной статистики по количеству хитов и уникальных посетителей за сутки и последние 30 дней система Count.RU анализирует информацию о распределении пользователей вашего сайта по странам и регионам России, предлагает отчет о ресурсах, публикующих ссылки на вашу страницу. Так же как и с помощью счетчика List.RU, на сервере Count.RU вы можете узнать данные о типе браузера, установленного на компьютерах ваших посетителей, версию используемой ими операционной системы, значения экранного разрешения и цветовой палитры их мониторов.

Глава 6. Десять советов web-дизайнеру

Создавая новый проект, который впоследствии, согласно вашим замыслам, будет привлекать сотни и тысячи посетителей в день, помните десять простых правил, которые помогут вам быстрее добиться поставленной цели.

1. Не перегружайте дизайн декоративными элементами. Излишнее количество графики «утяжеляет» страницу, увеличивая время ее загрузки в браузер, усложняет процесс обновления ресурса, делает практически невозможным внесение изменений в компоненты оформления страниц и, наконец, просто затрудняет восприятие информации.

2. Сделайте вашей настольной книгой орфографический словарь русского языка. Обилие ошибок в тексте сайта несказанно раздражает пользователей.

3. Обращайте пристальное внимание на мелкие детали оформления страницы: кнопки навигации, разделительные линии, маркеры списков. Незначительная на первый взгляд неаккуратность в мелочах моментально испортит общее впечатление о ресурсе.

4. Не публикуйте иллюстрации и тексты, взятые с чужих web-страниц, если только на них специально не указано, что данные элементы

можно свободно копировать и использовать. То же самое касается и концептуальных дизайнерских решений. Слухи распространяются по Интернету очень быстро, и если неожиданно выяснится, что вы незаконно воспользовались результатами чьих-либо трудов, это испортит вашу репутацию на всю жизнь.

5. Не злоупотребляйте клипартами и шаблонами. Безусловно, возможность воспользоваться готовыми решениями на начальном этапе является тяжелым искушением. Однако где гарантия, что этого не сделает сотня — другая ваших коллег? Если вы используете клипарт или готовый шаблон, внесите в его дизайн или компоновку хотя бы незначительные изменения.

6. Не воруйте сами у себя. Если несколько ваших проектов будут похожи как две капли воды, отличаясь только цветовыми решениями и формой кнопок, у пользователей могут возникнуть нехорошие мысли, касающиеся степени развитости вашей фантазии. Узнаваемый авторский стиль — это одно, слепое копирование дизайнерских решений — совсем другое. Клонирование до добра не доводит.

7. Если посетители вашего сайта присылают вам письма с предложениями, пожеланиями и комментариями, отвечайте на всю приходящую корреспонденцию незамедлительно. Игнорирование читательских писем — кратчайший путь к скатыванию популярности вашего ресурса до нулевого значения. Если занятость не позволяет вам ежедневно отвечать на приходящую почту, заготовьте заранее с десяток шаблонов, содержащих ответы на возможные послания. Эти ответы должны как минимум включать слова благодарности за проявленное беспокойство, обещание рассмотреть все пожелания посетителя и выражение надежды на то, что читатель снова заглянет на вашу страничку. Если вы не имеете возможности даже бегло проглядывать почту, заведите себе «секретаря».

8. Реагируйте на советы посетителей оперативно. Если хотя бы два человека сообщат вам о том, что щелчок мышью на одной из опубликованных вами гиперссылок приводит к появлению сообщения об ошибке, разберитесь с проблемой, не откладывая ее в долгий ящик. Иначе вы рискуете больше никогда не увидеть этих пользователей, а также их знакомых и знакомых друзей их знакомых на своей страничке.

9. Публикуйте только проверенную информацию. Если, например, вы напишете на своем сайте, что Билл Гейтс — женщина, обосновав данное предположение сотней различных аргументов, вы, безусловно, без труда добьетесь нескольких сотен посещений вашей странички в день. Но при этом вы, во-первых, подвергаете себя опасности получить по голове лично от Билла Гейтса, а во-вторых, ваша репутация рас-

пространителя непроверенной информации вряд ли поможет вам в бизнесе, поиске друзей или деловых партнеров.

10. Старайтесь избегать использования в текстах технического и нетехнического сленга, просторечий и ненормативной лексики. Кто-нибудь обязательно не поймет, что именно вы имеете в виду, остальные, скорее всего, поймут вас неправильно. Помните: как вы будете относиться к своим посетителям, в точности так же и они будут относиться к вам.

Глава 7. Каскадные таблицы стилей — CSS

Что такое CSS и для чего это нужно? CSS — Cascading Style Sheets, что означает дословно «каскадные таблицы стилей». Смысл — описывается стиль всего документа или сайта в целом, либо отдельных его элементов. Т.е. вы можете один раз определить, что, например, на всех страницах вашего сайта все заголовки будут красного цвета, абзацы выровнены по центру, а ссылки останутся неподчеркнутыми (то-то посетители помучаются, пока найдут их!). И к тому же вы сможете изменить стиль вашей страницы или даже всего сайта, подправив в тексте лишь пару строк!

Чтобы сразу объяснить суть таблиц стилей, рассмотрим пример. Допустим, у вас есть такая страница:

```
<HTML>
<HEAD>
<TITLE>Изучим таблицы стилей!</TITLE>
</HEAD>
<BODY>
<H1>Большой заголовок</H1>
<H2>Заголовок поменьше</H2>
<P>А это уже абзац - самый обычный абзац.</P>
</BODY>
</HTML>
```

А теперь где-нибудь в заголовке страницы (т.е. в секции **HEAD**) разместим тэг **<STYLE>**:

```
<STYLE TYPE="text/css">
<!--
H1 {font-size: 40px; background: red; text-align: right;
font-family: helvetica, arial, sans-serif}
H2 {font-size: 20px; font-style: italic; font-family: helvetica,
arial, sans-serif}
P {background: silver; text-align: center; font-family: courier,
fixed, monospace}
-->
</STYLE>
```


Можете набрать этот пример в каком-либо текстовом редакторе и сохранить затем с расширением htm или html (хорошая практика!), а можете сразу посмотреть этот пример в действии.

Теперь разберем этот пример подробнее.

Атрибут **TYPE="text/css"** и информация в комментарии (между **"<!--" и "-->"**) как раз и определяют стиль нашей страницы. Метки комментария здесь используются для совместимости со старыми версиями браузеров — они просто проигнорируют незнакомый тэг **<STYLE>** и содержимое комментария, и ваша страница будет показана как самая обычная HTML-страница. Строка **H1** указывает браузеру, что всякий текст, находящийся между метками **<H1>** и **</H1>**, должен отображаться шрифтом helvetica, arial или sans-serif высотой в 40 пикселов на красном фоне и быть выровнен вправо. Строка **H2** определяет, что шрифт в заголовках **<H2>** должен быть наклонным и высотой в 20 пикселов из семейства helvetica. И, наконец, строка **P** определяет поведение всех абзацев на странице: на сером фоне, выровнены по центру, шрифт — courier, fixed или monospace.

Вот и все. Как говорится, просто и со вкусом. Таким образом вы можете управлять практически всем, что появляется на вашей странице. Т.е. практически любому тэгу можно придать новые свойства. А теперь подробнее.

Font-family — шрифт, используемый для отображения данного элемента. В идеале, конечно, было бы неплохо, если бы у всех ваших посетителей были одинаковые наборы шрифтов. На практике же такого не бывает. Поэтому указывайте список шрифтов. Браузер просматривает этот список пока не встретит имеющийся в наличии шрифт. Например, компьютер с ОС Windows как правило имеет шрифт Arial, в компьютерах Macintosh наиболее похож на него шрифт Helvetica. В конце списка желательно иметь один из следующих: serif, sans-serif, monospace, fantasy или cursive. Пример: **"P {font-family: arial, helvetica, sans-serif}"**

Font-size — размер шрифта. Может указываться в точках (pt), пикселах (px).

Font-style: italic — курсив (наклонный шрифт).

Font-weight: bold — жирный шрифт. Значение может быть также числовым, только различные браузеры реагируют на это по-разному.

Text-transform — преобразование текста: допустимые значения: **uppercase** (все буквы будут заглавные), **lowercase** (все буквы будут строчные), **capitalize** (каждое слово будет начинаться с заглавной буквы) и none (т.е. никаких действий).

Text-decoration — выделение текста, допустимые значения: **underline** (подчеркнутый), **line-through** (перечеркнутый), **blink** (мигающий) и **none** (ничего). Например, чтобы ссылки в тексте не выделялись подчеркиванием, можно включить в заголовок документа следующий текст:

```
<STYLE TYPE="text/css">
<!--
a:link {text-decoration: none}
a:visited {text-decoration: none}
a:active {text-decoration: none}
-->
</STYLE>
```

Color — изменение цвета текста, линий, рамок. Скажем, в предыдущем примере мы сделали так, что ссылки на странице не подчеркиваются. Чтобы они не выделялись еще и цветом, сделаем одинаковым цвет ссылок и текста (для краткости здесь и далее метки **STYLE TYPE**...будут опускаться):

```
body {color: black}
a:link {color: black; text-decoration: none}
a:visited {color: black; text-decoration: none}
```

Т.е. цвет текста на странице — черный (1-я строчка), ссылки (обычно голубого цвета) — неподчеркнутые черного цвета (2-я строчка), посещенные ссылки (обычно фиолетовые) — также черного цвета и неподчеркнутые (3-я строчка).

Background-color — определяет цвет фона для какого-либо элемента. Например:

```
strong {background-color: yellow}
```

Текст между метками **<STRONG>** и **</STRONG>** будет показан на желтом фоне.

Background-image — позволяет сделать фон в виде рисунка. Пример:

```
blockquote {background-image: url(../pictures/million.jpg)}
```

В результате цитаты (т.е. текст между метками **<BLOCKQUOTE>** и **</BLOCKQUOTE>**) будет размещен на фоне рисунка «million.jpg».

Text-align — выравнивание текста. Применяется только в абзацах, заголовках и списках. А также для выравнивания картинок на страничке (несмотря на слово «text»). Например:

```
IMG {text-align: center}
```

Text-indent — позволяет делать отступ в первой строке абзаца (красная строка — как мы привыкли видеть в книгах). Ширина отступа измеряется в пикселах (px) или точках (pt). Может быть также величиной отрицательной, тогда первая строка будет выступать влево от остального текста. Пример (он же и применен к этому абзацу):

```
p {text-indent: 10pt}
```

Margin — устанавливает отступ текста. Пример использования:

```
body {margin-left: 10pt; margin-right: 15pt}
```

В результате текст на странице будет отступать слева на 10 точек от края страницы, справа — на 15 точек. Также возможны варианты: margin-top (отступ сверху) и margin-bottom (отступ снизу).

Рамки

Каждый элемент может заключен в рамку. Рамка может иметь следующие свойства:

- **border-width** — ширина рамки. Значение числовое (в пикселах или точках) или одно из зарезервированных слов — **thin** (тонкая), **medium** (средняя), **thick** (толстая);
- **border-color** — цвет рамки. Числовое или текстовое значение цвета. Также может принимать значение **transparent** (прозрачная);
- **border-style** — стиль рамки. Может иметь следующие значения:
  - **dashed** — пунктирная в виде черточек
  - **dotted** — пунктирная в виде точек
  - **double —** двойная линия
  - **inset —** с эффектом вдавленности
  - **outset** — с эффектом выпуклости
  - **ridge** — выпуклая рамка
  - **groove** — врезанная рамка
  - **solid** — непрерывная линия

Можно определить каждую сторону рамки отдельно. Для ширины рамки это выглядит так: **border-top-width** (ширина верхней стороны), **border-right-width** (правая сторона), **border-bottom-width** (нижняя сторона) и **border-left-width** (левая сторона). Для определения цвета и стиля достаточно просто перечислить свойства в таком порядке: верх-право-низ-лево. Пример:

```
blockquote {border-top-width: 3px; border-right-width: 5px;
border-bottom-width: 3px; border-left-width: 5px; border-color:
red green navy green; border-style: double double solid double }
```

Здесь определена рамка со следующими границами: верхняя — красная двойная в 3 пиксела шириной, правая и левая — зеленые двойные в 5 пикселов шириной, нижняя — синяя сплошная шириной в 3 пиксела.

А что, если захочется изменить стиль только одного абзаца? Или каждому абзацу определить свой стиль? Можно! В таком случае надо использовать атрибут **STYLE** с необходимыми свойствами. Например:

```
<P STYLE="text-indent: 15pt; color: red; background-color:
white">
```

или

```
<H3 STYLE="background-color: silver; border-width: 3px;
border-style: groove;
border-color: #F0F0F0; text-align: center">
```

Теперь представим такую ситуацию: у вас на странице 25 абзацев, из них 15 вы хотите выдержать в каком-то определенном стиле. Чтобы не писать атрибут **STYLE** каждый раз, можно поступить следующим образом: определить в заголовке документа в тэге **STYLE** необходимый стиль и дать ему название, например:

```
<style type="text/css">
<!--
.krasota {text-indent: 20px; background-color: aqua; color: red;}
-->
</style>
```

Точку перед названием надо ставить обязательно! В примере стиль назван «krasota», вы же, естественно, можете обозвать его как угодно, хоть «vasja» или «MySuperStyle». Теперь в тексте достаточно указать название своего стиля (уже без точки), и он будет применен:

```
<P CLASS="krasota">
```

И напоследок. Можно определить стиль для всего сайта! Для этого создается текстовый документ, где перечисляются все стили, используемые на страницах сайта (заголовки и т.п. не требуются) и сохраняется

с расширением .css (например, «stili.css»). Потом достаточно с каждой страницы сайта, где используются эти стили, сделать ссылку на этот файл. Ссылка размещается в заголовке страницы (в секции **HEAD**) и имеет такой вид:

```
<LINK REL=stylesheet HREF="stili.css" TYPE="text/css">
```

Теперь достаточно сделать изменения в одном файле — в «stili.css» (или как вы там его назовете) — чтобы изменился стиль всего сайта, пусть даже он состоит из 200 или 300 страниц!

Если на какой-то странице сайта определены свои стили, которые перекрывают стили, описанные в .css файле, то на ней будут применены ее стили (т.е. определенные в заголовке этой страницы). В свою очередь, если на странице в заголовке определены какие-то стили, и существует стиль, указанный в тэге (например, **<P STYLE= ".....">**), то к данному тэгу будет применяться второй стиль.

Вопросы и ответы

Что такое HTML и какие есть редакторы для HTML?

Для установления соединения с удаленным сервером используется сетевой адрес компьютера, который называется универсальным указателем ресурса — URL (Uniform Resource Locator). В ответ сервер посылает документы в формате HTML.

HTML — HyperText Markup Language — язык разметки гипертекста. Документы на языке HTML, также называемые web-документами, позволяют пользователю, указав на выделенное слово или фразу, получить доступ к файлу или перейти на другой HTML-документ, который связан с указанным участком текста гиперссылкой. Такие гипертекстовые связи между файлами и документами, расположенными на серверах по всему миру, позволяют системе работать так, как будто она представляет собой огромную паутину информации.

Существует два типа редакторов — так называемые WYSIWYG (What You See Is What You Get) и редакторы, работающие напрямую с HTML. Если вы чайник в HTML, то вам помогут следующие редакторы:

- FrontPage
- FrontPad
- Netscape Editor (Composer)
- Hot Metal — мощный, но тормозной
- Word 97
- HomePage Publisher for OS/2 (http://ourworld.compuserve.com/homepages/clerin/)
- Macromedia DreamWeaver

Не рекомендуем ими пользоваться если вы серьезно занялись HTML-дизайном, лучше купите книгу по HTML и пользуйтесь со следующим типом редакторов:

- Notepad (c:\windows\notepad.exe)
- Текстовый редактор в DOS Navigator/NC/etc
- HTML Pad
- Hot Dog

- HTMLed32 (http://www.ist.ca/)
- HomeSite (http://www.allaire.com)

А какой редактор лучший?

HomeSite! Некоторым больше нравятся более простые, такие как notepad, htmled.

Как заставить HS4 не заменять русские буквы на непонятные символы?

```
Settings/Tag Help/Automatically convert special characters
= off
```

Как сделать так, что бы картинки и/или текст с моей странички никто не мог украсть (скопировать)?

Никак.

Как зарегистрироваться на Geocities?

При регистрации вы получаете:

- 6Mb свободного пространства
- Бесплатный e-mail адрес
- Удобный FileManager
- Доступ по FTP

Для начала вам надо занять свободный виртуальный «домик» в Geocities.

- Войдите на http://www.geocities.com/neighborhoods и выберите нужное вам направление (например, если на вашей страничке будет информация о «железе», софте, то выбирайте SiliconValley).
- Далее в полученном окне нажмите: **join this neighborhood**
- Теперь вы должны выбрать свой будущий адрес. Найдите свободный «домик» (vacant) (Там будет написано сколько и где их свободно). **Совет**: Выбирайте то место, где больше свободных «домов», т.к. во время оформления регистрационной карточки у вас могут перехватить адрес другие желающие). Появится улица с домиками. Выбирайте свободный (с надписью vacant). Действовать надо очень быстро, т.к. «домик» могут перехватить. Теперь вам надо заполнить регистрационную карточку. Все, что не помечено красной звездочкой (*) можно не заполнять.

- В строке **membername** укажите ваше имя в **Geocities** (так же будет выглядеть новый e-mail адрес (<membername> @geocities.com). После заполнения карточки нажмите кнопку **Submit**. Теперь вы получили «домик», и вам надо «заселить» его в течение двух недель, иначе его освободят для других желающих. Пароль вам пришлют по почте.

Как отослать страничку на сервер Geocities?

Чтобы отослать вашу страничку на Geocities, проделайте следующее:

- Зайдите на http://www.geocities.com/members/tools/file_manager.html
- Для оправки файлов вам понадобится Internet Explorer или Netscape Navigator.
- Введите ваш **membername** и пароль и нажмите кнопку **Submit**.
- В следующем окне в **EZ File Upload** нажмите **Browse...** и выберите те файлы, которые надо послать. Нажмите **Upload Files**.
- После перекачки файлов откроется окно с осведомлением.
- Все! Для проверки откройте вашу страницу на Geocities.

Как установить счетчик посещений?

Для www.geocities.com

- Вставьте в вашу страничку тэг
  `<IMG SRC=/cgi-bin/counter/member>`
  где **member** — ваше имя пользователя.
- Затем идите на http://www.geocities.com/cgi-bin/counter/member.password где вместо **member** — ваше membername, а вместо password — пароль на Geocities.

Это самый простой счетчик без статистики.

Также вы можете взять другие счетчики на:

- SuperStats (http://www.superstats.com)
- NedStat (http://www.nedstat.nl)
- FreeStats (http://www3.freestats.com)
- Peresvet (http://www.peresvet.net)

- Lanka (http://lankaonline.com/public/counter)
- PagaCout (http://www.pagecount.com)
- Dux (http://counter.dux.ru)
- Dino (http://counter.bloke.com)
- SiteFlow (http://www.siteflow.com)
- ICount (http://www.icount.com)
- Aaddzz (http://www.aaddzz.com/pages/counters)
- SiteStat (http://www.site-stats.com)
- Ranker (http://www.ranker.ru)
  - Заходим на www.ranker.ru/add.asp
  - Читаем правила, соглашаемся
  - Заполняем регистрационную форму (Ranker FAQ находится по адресу: www.ranker.ru/faq.asp — рекомендуем прочитать перед заполнением формы)
  - Кликаем на кнопке **Submit**.
  - Вам покажут код, который вы должны вставить в страничку. Также его пришлют вам на e-mail
- Rambler (http://www.rambler.ru/top100)
  - Заходим на www.rambler.ru/top100
  - Кликаем **Add Site** (там на картинке нарисованы кнопки)
  - Прочитайте правила и нажмите кнопку внизу (не нажимайте дважды)
  - Заполните регистрационную форму и нажмите кнопку **Зарегистрироваться**.
  - Теперь вы получите HTML код для вашей странички по e-mail
  - Вставьте этот код в страничку.
- The Counter (http://www.thecounter.com)
  - У них вместо картинки надо вставить скрипт в страничку... за то они присылают еженедельно отчеты о каждом дне.

Как создать прозрачную графику?

Для создания прозрачной графики вам понадобится графический редактор, например Adobe Photoshop.

- Откроем файл
- Закрасим всю область, которую надо сделать прозрачной в один цвет
- **Image ⇨ Mode ⇨ Indexed Color**
- Сохраним его в формате GIF89a: **File Export ⇨ GIF89a Export...**
- Пипеткой выделим тот цвет на рисунке который надо сделать прозрачным.
- Кликаем **OK**

Как создать «чересстрочную» графику?

Для создания чересстрочной графики вам понадобится графический редактор, например Adobe Photoshop. Если вы хотите, чтобы ваши GIF'ы загружались постепенно, т.е. сначала появлялось расплывчатое изображение, потом четче и т.д., (кажется, что картинки загружаются быстрее), то проделайте следующее:

- Откроем файл **File ⇨ Open**
- Сохраним его в формате **GIF89a: File Export ⇨ GIF89a Export...** Не забудьте проверить, стоит ли галочка на **Interlace**
- Кликаем **OK** и сохраняем файл.

Как создать анимации?

Картинки для анимаций можно делать в Photoshop, LViewPro.

Сами анимации. Запустите **мастер** в Gif Construction Set: **File ⇨ Animation Wizard...** и кликните **Next** и еще раз **Next**. Если вы хотите, чтобы ваша анимация после выполнения остановилась, то выбирайте **Animate once and stop**, а если хотите, чтобы она постоянно работала — **Loop Indefinitely** и кликайте **Next**.

Далее выберите качество изображения (фотореалистичное, рисованное) **Next**. Теперь надо выбрать промежуток времени между сменой кадров **Next**.

- Теперь выбираем GIF'ы для анимации **Next**

- Кликаем **Done**
- Сохраните **File ⇨ Save As...**
- Для проверки нажмите **View** или запустите анимацию в браузере.

Можно ли сделать домен?

Да!

- Заходим на http://members.ml.org и нажимаем **SIGNUP**
- Теперь надо заполнить регистрационную карточку.
- Пароль и логин вам пришлют по почте.
- Теперь возвращаемся на начальную страницу и нажимаем **LOGIN**
- Вводим логин и пароль
- Теперь в новом окне нажимаем **Login** (внизу)
- В новом окне нажимаем **/HOME**, далее **New Host** и задаем параметры нового домена. Вы можете выбрать **ваше_имя.home.ml.org** или **ваше_имя.base.org**.
- В графе URL укажите ваш прежний адрес.
- Поставьте точку на **I Agree with...** и нажмите **Submit**
- Все! Через несколько минут проверьте работает ли домен.

Как сделать, чтобы при нажатии на ссылку, загружалась бы другая страничка?

```
<A HREF="http://www.server.com">Здесь ваш текст</A>
```

Как сделать, чтобы при нажатии на ссылку, появлялся бы бланк отправления сообщения?

```
<A HREF="mailto:yourname@server.com">Здесь ваш текст</A>
```

Можно добавить автоматическое подставление строки Subject:

```
<A HREF="mailto:yourname@server.com?Subject=ваш_subj">Здесь ваш текст</A>
```

Как сделать, чтобы при нажатии на картинку, делалось одно из выше перечисленного?

```
<A HREF="http://www.server.com"><IMG SRC="URL_вашей_картинки"></A>
```

Как сделать, чтобы при нажатии на ссылку, она открывалась в другом окне?

```
<A HREF="http://www.server.com" TARGET="_blank">Здесь ваш
текст</A>
```

или

```
<A HREF="http://www.server.com" TARGET="_new">Здесь ваш текст</A>
```

Как сделать, чтобы при наведении мышки на ссылку, выдавался мой текст, а не "http://www.server.com/Firma/Name/1234/...."?

```
<A HREF="http://www.server.com" onMouseOver="window.status='мышка
на ссылке';return true" onMouseOut="window.status='мышка за преде-
ламиссылки'; return true">Здесь сама ссылка</A>
```

Как подавить подчеркивание ссылок?

В **<head></head>** вставьте кусок:

```
<style>
a.noneline text-decoration: none;
</style>
```

В **<a href..>** впишите:

```
<a href=url class="noneline">
```

Как сделать, чтобы при нажатии на ссылку, в одном фрейме, она открывалась в другом?

```
<A HREF="file.htm" TARGET="название_фрейма">
```

Как сделать, чтобы фреймы открывались в том же окне, но поверх других фреймов?

```
<A HREF="file.htm" TARGET="_top">
```

Как сделать, чтобы при нажатии на ссылку, она открывалась в новом окне?

```
<A HREF="file.htm" TARGET="_blank">
```

или

```
<A HREF="file.htm" TARGET="_new">
```

Как сделать, чтобы при нажатии на ссылку, она открывалась в том же фрейме?

```
<A HREF="file.htm" TARGET="_self">
```

Как сделать, чтобы при нажатии на ссылку менялось содержимое двух фреймов?

а)

```
<A HREF="file.htm" onclick="top.frames[2].location='newframe2';
onclick="top.frames[3].location='newframe3';">text</A>
```


Где цифра в frames[x] — номер изменяемых фреймов.

б)

```
<A HREF="newframes" TARGET="_top">text</A>
```

Где newframes содержит тот же фреймсет, но с измененными фреймами.

Как сделать, чтобы нельзя было изменять размер фрейма/растягивать?

Добавить в описание фрейма noresize

Пример:

```
<frame name="test" frameborder="yes" scrolling="auto" noresize>
```

Как определить, что юзер открыл страницу не в фрейме, а ее надо смотреть в фрейме?

а) Пользователю предлагают перейти на главную страницу.

```
<SCRIPT LANGUAGE="JavaScript">
<!--
if (self.parent.frames.length == 0)
document.writeln("Best view with<A HREF="index.html">main
frame</A>");
// -->
</SCRIPT>
```

б) Сразу грузится главный фрейм.

```
<SCRIPT LANGUAGE="JavaScript">
<!--
if (self.parent.frames.length == 0)
self.parent.location="index.htm";
// -->
</SCRIPT>
```

Как убрать 1 пиксел между фреймами в Netscape?

```
marginweight=0 marginheight=0 в <body> теге зафрейменных страниц.
```

Именно в <body>!

Что такое таблицы?

Таблица начинается и заканчивается тэгами **<TABLE></TABLE>**. Не забывайте закрывать ее, а то некоторые браузеры не покажут ее содержимое. Опции:

- **<TR></TR>** — Table Rows. Строка таблицы. Если в таблице содержатся два набора тэгов **<TR>** и **</TR>**, в ней будут две строки. Весь текст, графику, другие таблицы и

атрибуты, которые вы хотите поместить одну строку, должны быть помещены между тэгами **<TR>** и **</TR>**.

- **<TD></TD>** — Внутри таблицы размещаются ячейки с данными. Каждая ячейка, содержащая текст или графику должна быть заключена в тэги **<TD></TD>**.
- **<TH></TH>** — Table Header. Заголовок таблицы. По умолчанию текст между тэгами <**TH**> и </**TH**> записывается жирным шрифтом и располагается по середине ячейки. Центрирование можно отменить и выровнять текст по левому или правому краю — **<ALIGN=LEFT>** и **<ALIGN=RIGHT>**.
- **NOWRAP** — Чтобы текст, не помещающийся в одну строку, не переходил на следующую используйте данный атрибут (для **<TH></TH>** и **<TR></TR>**).
- **COLSPAN=xx** — Соединение столбцов. Если вы хотите, сделать какую-нибудь ячейку шире, чем верхняя или нижняя, используйте данный атрибут. (Укажите на сколько ячеек надо «растянуть» данную ячейку — **COLSPAN**)
- **ROWSPAN=** — Аналогичен **COLSPAN**, только он задает число строк, на которые растягивается ячейка.
- **WIDTH=** — Применяется в двух случаях. Можно поместить его в тэг **<TABLE>** для задания ширины всей таблицы, а можно в тэгах **<TR></TR>** или **<TH></TH>** для задания ширины ячейки или группы ячеек.
- **CELLPADDING=** — Определяет ширину пустого пространства между содержимым ячейки и ее границами, т.е. задает поля внутри ячейки.
- **ALIGN=** — Выравнивание текста или графики в ячейке по горизонтали (**LEFT**, **RIGHT**, **CENTER**, **JUSTTIFY**).
- **VALIGN=** — Выравнивание текста или графики в ячейке по вертикали (**TOP**, **BOTTOM**, **CENTER**, **BASELINE**).
- **<CAPTION></CAPTION>** — Заголовок таблицы. Центрируется по умолчанию
- **BORDER=** — Рамка таблицы.
- **CELLSPACING=** — Атрибут определяет ширину промежутков между ячейками (по умолчанию — 2pix)

- **BGCOLOR=""** — Фоновый цвет таблицы.
- **BORDERCOLOR= (IE) —** Цвет линий, обрамляющих таблицу или ячейку.
- **BORDERCOLORDARK= BORDERCOLORLIGHT= —** Атрибуты для создания трехмерных таблиц.

Пример:

```
<TABLE BORDER=1 CELLSPACING=0 CELLPADDING=0 HEIGHT="1" BORDERCOL-
OR="#33CCFF" bordercolordark="#33CCFF" bordercolorlight="#33CCFF">
<CAPTION>TABLE</CAPTION>
<TR BGCOLOR="#000099">
<TD WIDTH="400">
<BR>
</TD>
</TR>
<TR BGCOLOR="white">
<TD WIDTH="400">
<BR>
</TD>
</TR>
</TABLE>
```

Как используется тэг <META>?

<META> используется в рамках тэга **<HEAD>**, чтобы указать какую-либо полезную информацию, не определяемую другими HTML документами. В частности некоторые поисковики индексируют страницу этим тэгам.

Формат:

```
<META HTTP-EQUIV="имя" CONTENT="содержимое"> или
<META NAME="имя" CONTENT="содержимое">
```

Атрибуты и их аргументы:

- **HTTP-EQUIV="Content-Type"**

Указывает тип документа. Можно вдобавок указать кодировку документа. Но с этим будьте аккуратны.

```
<META HTTP-EQUIV="Content-type" CONTENT="text/html; charset=WIN-
1251">
```

- **HTTP-EQUIV="Refresh"**

Время, через которое браузер автоматически обновит документ. Можно сделать автоматическую загрузку другой странички.

```
<META HTTP-EQUIV="Refresh" Content="5;
URL=http://www.server.com/">
```

Через 5 секунд начнется загрузка указанного URL. Если не хотите ждать, пишите

```
Content="0; URL=...">
```

- **NAME="KeyWords"**

Поисковые машины смотрят наличие этого параметра, за которым идет слово **CONTENT="<words>"**, где **<words>** — слова, на которые ваша страничка ориентирована. Например, ваша страничка посвящена фильму «Титаник»:

```
<META NAME="KeyWords" CONTENT="Titanik, Dion">
```

На ключевое слово **Titanik**, поисковик выдаст ссылку на вашу страничку.

- **NAME="Description"**

В поле **CONTENT="<word>"** — описание вашей странички.

```
<META NAME="Descripton" CONTENT="My Home Page">
```

- **NAME="GENERATOR"**

Обычно вставляется редакторами WYSIWYG (FP'98). Совершенно необязателен. Вот что вставляет Netscape Composer:

```
<META NAME="GENERATOR" CONTENT="Mozilla/4.03 [en] (Win95; I)
[Netscape]">
```

Пример использования:

```
<HEAD>
<META NAME="Description" CONTENT="ЗУБЫ HomePage">
<META NAME="KeyWords" CONTENT="Teeth, Director">
<META NAME="GENERATOR" CONTENT="Microsoft Notepad">
<TITLE>ИЧП "ЗУБЫ"</TITLE>
</HEAD>
```

В чем отличия <b> и , <i> и ?

<b> и <i> — тэги физического выделения, то есть вы принудительно заставляете выделять текст полужирным или наклонным шрифтом. и — тэги логического выделения, то есть каждый браузер может по своему выделить текст внутри этих тэгов.

Как разместить картинку в центре экрана?

Можно сделать ее в таблице:

```
<table width="100%" height="100%">
<tr>
<td align="center" valign="middle">
<img src="kartinka.gif">
</td
</td
</table>
```


Как сделать неподвижный фон(background)?

Это расширение для IE 3.01 и позже. На навигаторе и старых версиях IE не работает.

```
<BODY BACKGROUND="URL_вашей_картинки" BGPROPERTIES="FIXED">
```

Как вставить музыку в html файл?

Просто вставьте данный скрипт в вашу страничку:

```
<SCRIPT LANGUAGE="JavaScript"><!-- Hide the script from old
browsers
function BGSound()
// find out what browser it is
var Cmd = " "
var Brwsr = window.navigator.appName
if (Brwsr == "Microsoft Internet Explorer")
// Use BGSOUND for MSIE
Cmd = "<BGSOUND SRC=" + "music.mid" + ">"
else
// Everyone else gets EMBED
Cmd ="<EMBED SRC='music.mid' HIDDEN=TRUE>"
return Cmd
// do it!
document.write(BGSound())
// end hiding here -->
</SCRIPT>
```

Как сделать так, чтобы при наведении мышки на текст с ссылкой, он (текст) поменял цвет?

Вставьте данный скрипт в вашу страничку:

```
<SCRIPT language="JScript">
<!--
ua=navigator.userAgent;
v=navigator.appVersion.substring(0,1);
if ((ua.lastIndexOf("MSIE")!=-1) && (v!='1') && (v!='2') &&
(v!='3')) document.body.onmouseover=makeCool;
document.body.onmouseout=makeNormal;
function makeCool()
src = event.toElement;
```

```
if (src.tagName == "A")
src.oldcol = src.style.color;
src.style.color = "FF0000";
function makeNormal()
src=event.fromElement;
if (src.tagName == "A")
src.style.color = src.oldcol;
//-->
</SCRIPT>
```

Как сделать неподчеркнутые ссылки?

Вставить в <head> вот такой кусок:

```
<STYLE type="text/css">
<!--
A:link text-decoration: none;
A:visited text-decoration: none;
A:active text-decoration: none;
A:hover text-decoration: none;
-->
</STYLE>
```

Как сделать кнопку «Back»?

```
<a href="javascript:history.back()">Назад</a>
```

Что такое «форма»?

Форма начинается и заканчивается тэгами <FORM>, она может содержать различные поля и кнопки.

Формат:

```
<FORM ACTION="URL" METHOD="GET/POST">
....
</FORM>
```

В поле ACTION — адрес программы, которая будет запущена по нажатию кнопки Sumbit(срабатывание формы). Поле METHOD может принимать одно из двух значений: GET или POST. GET — введенная пользователем информация помещается в конец URL.

Поскольку размер формы может быть большой, то получится большой URL при выполнении формы.

POST — информация от пользователя вливается в общий поток данных

Что такое «разворачивающиеся списки»?

Формат:

```
<SELECT NAME="name">
<OPTION SELECTED>Text_1</OPTION>
<OPTION>Text_2</OPTION>
<OPTION>......</OPTION>
<OPTION>Text_N</OPTION>
</SELECT>
```

NAME — уникальное имя элемента. Каждый элемент должен иметь уникальное имя, чтобы при получении введенной пользователем информации. <OPTION></OPTION> — добавляет новый элемент списка. Внутри тэга вводится имя этого элемента.

Если у элемента определена опция VALUE, программе-обработчику будет передано указанное значение, иначе — сам текст элемента.

Также один любой элемент может быть исходно выделен. Для этого используется опция SELECTED внутри тега <OPTION>.

Что такое «список со множественным выбором»?

Формат такой же как и у обычного разворачивающегося списка, только в тег SELECT добавляются еще две опции:

```
<SELECT NAME="name" SIZE="2" MULTIPLE="MULTIPLE">
<OPTION VALUE="1">Text_1<OPTION>
<OPTION VALUE="2">Text_2<OPTION>
<SELECT>
```

SIZE — определяет сколько элементов должно одновременно отображаться в списке.

MULTIPLE — указывает, что одновременно можно выбрать более одного элемента из списка.

VALUE — дает команду посылать заданный параметр _вместо_ текста между тэгом <OPTION>.

Что такое «флажки»?

Формат:

```
<INPUT TYPE="CHECKBOX" NAME="name" VALUE="value">
```

Что такое «переключатели»?

Отличие переключателей от флажков в том, что вы можете выбрать только один переключатель из группы name, но любое количество флажков.

Формат:

```
<INPUT TYPE="RADIO" NAME="name" VALUE="value">
```

Что такое «текстовые поля»?

Формат:

```
<INPUT TYPE="TEXT" NAME="name" SIZE="XX" MAXLENGTH="XX">
```

MAXLENGTH — указывает максимально возможную длину вводимой строки.

SIZE — видимый размер поля.

Что такое «текстовые области»?

Формат:

```
<TEXTAREA NAME="name" ROWS="XX" COLS="XX">
```

ROWS — количество строк.

COLS — количество символов в строке.

Что такое «парольные поля»?

Формат:

```
<INPUT TYPE="PASSWORD" NAME="name" SIZE="XX" MAXLENGTH="XX">
```

Отличие от простого текстового поля только в том, что вводимые символы на экране не изображаются (заменяются звездочками).

Что такое «скрытые поля»?

Формат:

```
<INPUT TYPE="HIDDEN" NAME="name" VALUE="value">
```

Данное поле нужно для передачи параметров программе-обработчику без предоставления пользователю возможности поменять их.

Для чего нужны кнопки Submit & Reset?

Формат:

```
<INPUT TYPE="SUBMIT" VALUE="Отправить">
<INPUT TYPE="RESET" VALUE="Сброс">
```

По кнопке SUBMIT происходит отправка формы, а по кнопке RESET — сброс в исходное состояние.

Мне нужно, чтобы формы отсылались на мой e-mail. Как это сделать?

а) Если надо отослать одно поле, то подойдет следующая конструкция:

```
<FORM ACTION="mailto:your@email" METHOD="POST">
```

б) Если же полей много (имя, фамилия), то надо использовать халявные сервера со скриптами — например, cgi-free. Недостаток — реклама: перекидывает после **Послать** на страничку с своей рекламой.

```
<FORM action="http://www.cgi-ree.com/mailer.cgi?your@email;
temp.htm" method=POST name="form1">
```

Здесь **your@email** — ваш email (на который вышлется), **temp.htm** — полный адрес к странички благодарности *«что вы заполнили форму»*

Как использовать «карту»?

Для того, чтобы при нажатии на определенную область картинки происходил переход на один адрес, а при нажатии в другой области — на другой, то проще всего воспользоваться следующей конструкцией:

```
<IMG SRC="адрес_картинки" USEMAP="#имя_мэпа">
```

В любом месте html-файла описать сам "мэп"(карту):

```
<MAP NAME="имя_мэпа">
<AREA элементы>
</MAP>
```

Где «элементы»:

1. SHAPE — очертания области. Принимает значения: RECT, CIRCLE, POLY. По умолчанию RECT.

а) RECT — прямоугольник. COORDS="x1, y1, x2, y2"

x и **y** задают координаты верхнего левого и нижнего правого углов.

б) CIRCLE — круг. COORDS="x0, y0, r"

x0 и **y0** — координаты центра. **r** — радиус.

в) POLY — многоугольник. COORDS="x1, y1, x2, y2, x3, y3..." x и y — координаты вершин.

Все координаты указываются в пикселах, за исключением случаев COORD="0,0,50%, 100%" (в процентах).

2. COORDS — координаты области. Форма, определяемая SHAPE.

3. HREF — адрес документа (URL)

4. NOHREF — означает, что этот регион не действует. Используйте его — когда хотите «вырезать» область.

5. ALT — текстовое описание области. Используется для того, чтобы показывать текстовые метки, которые могут высвечиваться в строке статуса, когда мышь или другое устройство управления курсором находится над горячей зоной, или же для создания текстовой строки для нефафической программы просмотра.

Обратите внимание, что при описании «мэпа» пишется просто **имя_мепа**, а при описании картинки пишется **#имя_мэпа**. Пример:

```
<IMG SRC="Stamp.GIF" HEIGHT=168 WIDTH=173 USEMAP="#m1">
<map name="m1">
<area href="file1.htm" shape=rect coords="0,0,50,50">
<area href="file2.htm" shape=circle coords="50,0,100">
</map>
```

Так же можно воспользоваться специальными программами для нарезки мэпов.

Как отформатировать текст по ширине?

```
<p align="justify">
text
</p>
```

Как изменить абзацный отступ?

```
<p style="text-indent: 15pt">... Абзац ...</p>
```

Как можно изменить параметры <hr>?

- **SIZE=xx** — Атрибут указывает высоту линии.
- **WIDTH=xx** — Пикселы или проценты. Длина линии.
- **ALIGN=word** — Выравнивание (CENTER, LEFT, RIGHT).
- **NOSHADE** — Атрибут для создания линий без тени.
- **COLOR=word** — Имя цвета или #**RRGGBB**. Цвет линии. Работает только в IE.

Тэг <**HR**> можно (но не нужно) использовать для создания окружностей:

```
<HR NOSHADE SIZE=100 WIDTH="1%"> (SIZE - диаметр)
```

Примеры:

```
<HR COLOR="red" NOSHADE SIZE=1 WIDTH="90%">
<HR COLOR="#cdef1a" SIZE=2 WIDTH="500" ALIGN=CENTER>
```


Как определить разрешение у юзера и в зависимости от разрешения посылать на разные страницы?

```
<html>
<head>
<script language="JavaScript">
var height=0;
var width=0;
if (self.screen)  // for NN4 and IE4
width = screen.width
height = screen.height
else if (self.java) // for NN3 with enabled Java
var jkit = java.awt.Toolkit.getDefaultToolkit();
var scrsize = jkit.getScreenSize();
width = scrsize.width;
height = scrsize.height;
</script>
</head>
<body>
<script language="JavaScript">
<!--
if (width == 800 && height == 600)
location.href = "800x600.htm"
else if (width == 640 && height == 480)
location.href = "640x480.htm"
else
location.href = "unknown.htm"
//-->
</script>
</body>
</html>
```

Как сделать автоматическую надпись «Последнее обновление»?

Вставьте в страничку следующий текст:

```
Последнее обновление:
<SCRIPT LANGUAGE="JavaScript"><!--
document.writeln(document.lastModified)
// --></SCRIPT>
```

Как сделать так, чтобы при наведении мышки на картинку-ссылку, картинка изменилась?

```
<SCRIPT LANGUAGE="JavaScript">
<!--//
* if (navigator.appName == "Netscape" &&
parseFloat(navigator.appVersion)
```

```
>= 3.0) roll = 'true';
* else if (navigator.appName == "Microsoft Internet Explorer" &&
parseFloat(navigator.appVersion) >= 3.0) roll = 'true';
else roll = 'false';
function over(img,ref) if (roll == 'true')
document.images[img].src = ref;
function out(img,ref) if (roll == 'true')
document.images[img].src = ref;
if (roll == 'true')
a1=new Image;a1.src="over.gif";
a2=new Image;a2.src="out.gif";
aX=new Image;aX.src="image.img"
//-->
</SCRIPT>
```

a1,a2,.. — так подгружаем картинки нажатой и ненажатой кнопок. Не забудьте про это!

Затем прописывайте картинку-ссылку:

```
<A HREF="URL" onmouseover="over('mover','over.gif');"
onmouseout="out('mover','out.gif');"><IMG SRC="out.gif"
NAME="mover"></A>
```

Здесь **mover** — имя картинки, необходимо чтобы java-скрипт знал какую картинку менять. У каждой картинки должно быть свое имя.

Приложения

Арсенал web-строителя

Для изготовления профессиональных страничек вам придется профессионально изучить несколько профессиональных программ.

Программа обработки растровой графики

Одна из важнейших программ, которую вам придется освоить. Она вам поможет при сканировании фотографий и их коррекции. При помощи этой программы вы сможете создавать те невообразимые эффекты, которые вы видите в Интернет, рекламах и т.д. Обязательными являются следующие умения для этого класса программ:

- Сканирование фотографий;
- Коррекция отсканированных и готовых фотографий, в том числе — тоновая и цветовая коррекция;
- Ретушь фотографий;
- Умение кадрировать;
- Понимание различий в форматах графических файлов;
- Грамотное использование фильтров;
- Работа со слоями.

В качестве конкретных программ можно привести следующие:

- Adobe PhotoShop — данная программа является лидером в области графических программ такого рода, но она требует и соответствующих ресурсов от вашего компьютера.
- Paint Shop Pro — по моему мнению, лучшая shareware-программа, которая, к тому же, поддерживает фильтры от Adobe PhotoShop и очень быстро работает с объемными (>20M) фотографиями. Может импортировать и экспортировать изображения в 40-50 разных форматов. Некоторые работы вам придется делать, используя разные редакторы. Идеальных редакторов нет, некоторые лучше делают одно, некоторые — другое.

Программа обработки векторной графики

Еще одна из важных для дизайнера программ. Позволяет создавать с нуля или с использованием клипарта различные логотипы, кнопки, эффектные надписи и т.п. вещи. Принципы векторных редакторов сильно отличаются от растровых, поэтому осваивать их придется отдельно; зато, освоив их, вы заодно сможете изготавливать для себя фирменные визитки, бланки, брошюры (как побочный эффект). Вам потребуются следующие навыки:

- Умение работать с графическими объектами (группировка, наложение, получение нестандартных объектов);
- Умение работать с кривыми и узлами кривых;
- Умение работать с направляющими и сеткой;
- Умение разбираться в цветовых моделях (RGB, CMYK, HSB);
- Работа с текстом — умение разместить его на любой кривой;
- Грамотное использование градиентов и заливок (включая фрактальные);
- Грамотное использование эффектов.

Типичными представителями данного класса являются Corel DRAW и Adobe Illustrator — они обе являются лидерами в своих областях и, соответственно, располагают самыми последними достижениями в области векторной графики. В противовес этим тяжеловесам можно порекомендовать симпатичную и быструю программку Corel Xara!; пусть вас не вводит в заблуждение слово Corel — данная программа разработана фирмой XARA, которая была куплена корпорацией, как один из конкурентов. Данная программа, в отличие от двух предыдущих, очень быстрая и маленькая, но некоторые эффекты и команды, доступные в других программах, в ней сделать или трудно, или вообще невозможно.

Программы просмотра web-страничек

Или браузеры — в настоящее время имеется только два браузера, заслуживающих упоминания — это Microsoft Internet Explorer и Opera. Сейчас приближаются уже седьмые версии данных программ, но, по статистике, наиболее используемыми являются их 5-е версии. Поэтому для контроля внешнего вида ваших страничек вам придется использовать их. Причем на компьютере должны стоять обе версии. Хорошая web-страничка должна одинаково выглядеть в любом из этих браузеров.

Простой текстовый редактор

Понадобится для ручного исправления и добавления HTML-кода, т.к. существующие визуальные редакторы не могут полностью контролировать процесс создания web-странички (а если вы делаете страничку со сложным дизайном, то, возможно, вам придется весь код писать в редакторе. В качестве примера подойдет обычный «Блокнот» из стандартной поставки Windows или один HTML-редакторов, которые имеют встроенные команды на проверку правильности тэгов и структуры документов, например, HoTMetaL.

Визуальные редакторы

Позволяют быстро разрабатывать несложные web-странички и корректировать уже написанные, но с ними нужно быть осторожными, т.к., благодаря именно им, ваша страничка может плохо глядеться в другом браузере. Наиболее известные кандидаты — это Macromedia Dreamweaver. Иногда вам придется вручную исправлять код, сгенерированный данными программами.

Текстовый процессор

С возможностью проверки орфографии — очень полезен для набора текста и исправления ошибок в распознанных текстах. Как пример — обычный Microsoft Word.

Программы распознавания текста — могут вам сэкономить массу времени, избавляя от ручного набора напечатанных текстов. Программ по распознаванию русского языка всего две — это CuneiForm и FineReader. Используйте по возможности последние версии. По качеству распознавания они примерно одинаковы.

Некоторые специальные программы — эти программы позволят вам выполнить некоторые эффекты и справиться с такими задачами, которые другими способами трудновыполнимы или не выполнимы вообще. Ниже перечислены некоторые из этих программ:

- Ulead GIF Animator — программа, позволяющая вам создавать анимированные GIFы. Обеспечивает полный контроль над выходным файлом. Обладает очень мощными средствами оптимизации.
- Фильтры для Adobe PhotoShop — их количество просто огромно, но реально понадобится немного. Они способны существенно повысить работоспособность и двумя-тремя нажатиями создать впечатляющие эффекты. Но только не переборщите с ними — во всем нужна мера.

- Macromedia Flash MX — практически стандарт для использования в web векторных изображений. Обладает собственной средой разработки и позволяет создавать впечатляющую векторную анимацию.
- Программы для создания VRML-миров или 3D-программы, позволяющие экспортировать в данный формат. В качестве примера могу порекомендовать неплохую программу создания VRML-миров Internet3D Space Builder.
- Программы для обработки звука — могут понадобиться, если того требует Ваша страничка. Это программы вообще отдельного класса, но для простой обработки звука, например, подойдет CoolEdit, eJay.

Пожалуй, это более-менее полный список программ для создания web-страничек. Не все из них, конечно, нужно сразу же иметь, чтобы начать писать свои странички. Для начала достаточно какого-нибудь браузера и визуального редактора, а уже по мере накопления опыта вы освоите и другие программы.

Тэги HTML

****

Ссылка

- **target=_blank** — открывает ссылку в новом окне.
- **name="anchor1"** — имя закладки.

<area>

Определяет геометрические области внутри карты и ссылки, связанные с каждой областью.

- **shape="rect" (circle, poly)** — тип области (прямоугольник, круг, многоугольник)
- **coords="x1,y1,..."** — координаты, количество зависит от типа области
- **alt="описание"** — описание, подсказка
- **href="document.html"** — ссылка

****

Жирный текст

<big></big>

Крупный текст

<body></body>

Обязательный тэг

- **<body text="#cc0000">** — задает цвет текста всего документа
- **<body bgcolor="#000000">** — задает цвет фона документа
- **<body background="картинка.jpg">** — задает фоновую картинку
- **<body link="#ff9999">** — задает цвет ссылки
- **<body alink="#ff9999">** — задает цвет активной (нажатой) ссылки
- **<body vlink="#ff9999">** — задает цвет посещенной ссылки
- **<body topmargin="0">** — определяет ширину верхнего и нижнего полей документа для IE
- **<body leftmargin="0">** — определяет ширину левого и правого полей документа для IE
- **<body marginheight="0">** — определяет ширину верхнего и нижнего полей документа для NN
- **<body marginwidth="0">** — определяет ширину левого и правого полей документа для NN

**
**

Перенос строки (принудительный)

- clear="all" (left, right) — завершение обтекания текстом объекта (картинки).

** **

Центрирование текста

****

Употребляется с одним из параметров, приведенных ниже:

- **текст** — задает цвет текста выбранной части документа.

- **текст** — задает размер шрифта текста (от -2 до +4)
- **текст** — задает определенный шрифт в документе (можно прописать какой-нибудь экзотический шрифт)
- **<frame src="menu.html">** — определяет фрэйм и его свойства внутри frameset
- **marginheight="0"** — определяет ширину верхнего и нижнего полей фрэйма.
- **marginwidth="0"** — определяет ширину левого и правого полей фрэйма.
- **scrolling="yes" (no, auto)** — линейка прокрутки будет всегда (никогда, если надо).
- **name="window-1"** — имя фрейма, используется для ссылки на него из других документов (фреймов), с помощью параметра тэга <a> target (target="имя_фрейма").

<frameset></frameset>

Определяет фрэймовую структуру документа. Открывает и закрывает список фрэймов

- **rows="100,200,*"** — определяет количество и размеры горизонтальных фрэймов (рядов). Размеры задаются в процентах или пикселях.
- **cols="10%,20%,70%"** — определяет количество и размеры вертикальных фрэймов (колонок).Размеры задаются в процентах или пикселах.
- **border** — определяет ширину рамок фрэймов в пикселах.

<Hx></Hx>

Заголовок

<head></head>

Обязательный тэг

<hr>

Линия

- **<Hr align="right">** — center или left
- **<Hr width="30%">** — ширина линии в процентах

- **<Hr size="6">** — толщина линии
- **<Hr NoShade>** — отмена объемности
- **<Hr color="cc0000">** — цвет линии, только в IE

<html></html>

Обязательный тэг

****

Выделяет текст курсивом

<img src="ваша_картинка.jpg">

Рисунок

- **Border="5"** — рамка вокруг картинки
- **width="500"** — ширина картинки
- **height="100"** — высота картинки
- **align="left"** — расположение текста по отношению к картинке (right, top, middle, left, bottom)
- **vspace="10"** — расстояние от картинки до текста по вертикали
- **hspace="30"** — расстояние от картинки до текста по горизонтали
- **alt="описание"** — описание картинки
- **usemap="#karta1"** — ссылка на карту
- **<map></map>** — навигационные карты
- **name="karta1"** — имя карты

<marquee></marquee>

Бегущая строка (текста)

- **height="16"** — высота строки
- **width="250"** — ширина строки
- **bgcolor="#99CCFF"** — цвет фона
- **vspace="10"** — расстояние от бегущей строки до текста (картинок, др. объектов) по вертикали

- **hspace="20"** — расстояние от бегущей строки до текста (картинок, др. объектов) по горизонтали
- **loop="2"** — сколько раз прокрутится строка
- **direction="left" (right, up, down)** — направление движения строки — влево (вправо, вверх, вниз)
- **behavior="scroll" (slide, alternate)** — поведение строки — обычная прокрутка (прокрутка с остановкой, от края к краю)
- **scrollamount="1"** — скорость движения строки, может принимать значения от 1 до 10 (1 — самое медленное движение, 10 — самое быстрое)

**
**

Списки

- **type="disk"** — полный кружок (по умолчанию)
- **type="circle"** — полый кружок
- **type="square"** — квадратик

** **

Параграф

- ** текст ** — центрирование текста.
- ** текст ** — выравнивание текста по правому краю документа.
- ** текст ** — выравнивание текста по левому краю документа.
- ** текст ** — выравнивание текста по обоим краям документа.

<pre></pre>

Форматированный текст

<s></s>

Перечеркнутый текст

<small></small>

Малый текст

<stike></strike>

Перечеркнутый текст

****

Нижний индекс

****

Верхний индекс

<title></title>

Название документа

<table></table>

Таблицы

<tr></tr>

Строчка (ряд) таблицы.

<td></td>

Столбец (ячейка) таблицы.

- **bgcolor="#FFCC33"** — цвет фона таблицы (строки, ячейки).
- **background="картинка.gif"** — задает фоновый рисунок для таблицы (строки, ячейки).
- **width="50"** или **width="50%"** — ширина таблицы (строки, ячейки) в пикселях или процентах.
- **height="45"** или **height="45%"** — высота таблицы (строки, ячейки) в пикселях или процентах.
- **align="center" (right, left)** — выравнивает содержимое ячейки относительно ее центра (правого или левого краев).
- **valign="middle" (top, bottom)** — вертикальное выравнивание содержимого строки (ячейки) по середине (наверху или внизу).
- **colspan="2"** — растянуть ячейку на несколько столбцов.
- **rowspan="2"** — растянуть ячейку на несколько рядов.
- **cellspacing="5"** — задает пространство между ячейками.
- **cellpadding="5"** — задает пространство внутри ячейки между ее содержимым и границами.

- **border="3"** — задает толщину рамки таблицы.
- **bordercolor="#000000"** — задает цвет рамки таблицы.

<tt></tt>

Фиксированный (моноширинный) текст

<u></u>

Подчеркнутый текст

**
**

Отступ (табуляция)

META-тэги

Мета тэги — это необязательные атрибуты страницы, размещенные в ее заголовке, которые содержат описание страницы, ключевые слова к ней, некоторую информацию об авторе, управляющие команды для поисковых роботов и прочую служебную информацию, не предназначенную для всех посетителей.

<META NAME="Description" CONTENT="Описание вашей страницы.">

Зачастую именно то, что вы здесь укажете, будет отображаться поисковиками (краткое содержание страницы). Сделайте описание ярким, зазывающим. Длинна описания не должна превышать 200 символов.

<META NAME="Keywords" CONTENT="Ключевые слова.">

Очень важный тэг. Подумайте, какие слова будет набирать пользователь, при поиске информации в поисковике. Вот их то и вписывайте. Не пишите в этом тэге одних и тех же слов — поисковые машины это страсть как не любят. Длина списка до 800 символов.

<META HTTP-EQUIV="Content-Type" content="text/html; charset=windows-1251">

Указание типа документа и его кодировки. В России в основном windows-1251.

Если содержание Вашей страницы часто меняется, то вставьте следующие два тэга:

<META NAME="Document-state" CONTENT="Dynamic">

<META NAME="Revizit-after" CONTENT="10 days">

Робот будет переиндексировать вашу страницу каждые 10 дней. Периодичность можете выбрать сами.

Если страница изменяться не будет, вставьте:

<META NAME="Document-state" CONTENT="Static">

Мета-тэг от известного белорусского web-дизайнера Сергея Осипова:

<META HTTP-EQUIV="Pragma" CONTENT="no-cache">

или:

<META HTTP-EQUIV="no-cache">

или:

<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">

или:

<META HTTP-EQUIV="Expires" CONTENT="Wed, 2 Mar 1996 00:00:05 GMT">

Контроль кэширования для HTTP/1.0. Страницы с этими тэгами не будут кешироваться браузерами. Незаменимы при выдаче результатов работы скриптов.

<METAHTTP-EQUIV="Content-Language" CONTENT="en,ru">

Этот тэг отвечает за язык, использованный на странице. В данном случае это русский и английский.

<META HTTP-EQUIV="Refresh" CONTENT="5; URL=page.html">

Этот тэг через 5 секунд загрузит страницу page.html в текущем окне браузера. Это то, что называют "редиректом".

<META HTTP-EQUIV="Keywords" CONTENT="Ключевые слова.">

Ёще один тэг с ключевыми словами.

<META NAME="Classification" CONTENT="Классификация.">

Тэг отвечает за аудиторию, на которую направлен сайт. Например Web Masters.

<META NAME="Robots" CONTENT="index, follow">

Этот тэг отвечает за управление поисковыми роботами. Вот, какие могут быть значения CONTENT:

- **index** — индексировать эту страницу.

- **follow** — индексировать страницы на которые есть ссылки с этой странице.
- **all** — эквивалентно двум предыдущим через запятую.
- **noindex** — не индексировать страницу, но идти по ссылкам.
- **nofollow** — индексировать, но не идти по ссылкам.
- **none** — эквивалентно двум предыдущим через запятую.

По умолчанию **CONTENT="index, follow"**.

<META NAME="author" content="Автор">

Тэг, в который вы можете вписать своё имя, название студии, или ваш e-mail. Формат произвольный.

<META HTTP-EQUIV="Reply-to" content="spam@sitemaker.ru">

Тэг для указания адреса электронной почты автора.

<META NAME="Copyright" content="Авторские права">

Описание авторских прав на документ. Формат произвольный.

Хостинг

Бесплатный хостинг Американский

http://www.geocities.com

http://www.xoom.com

http://www.hypermart.net

http://members.tripod.com

http://www.angelfire.com

http://www.crosswinds.net/index.html

http://www.fortunecity.com

http://www.fsn.net

http://a1bbs.dzone.co.kr

http://www.easyspace.com

http://www.cynetcity.com

http://www.cybercities.com

http://www.royaltystudios.com

http://www.internet-club.com

http://www.student.toplinks.com/freehome.htm

http://rampage.ml.org/freeinfo.html

http://www.cybercity.hko.net/

http://bip.concept.se/user

http://www.nether.net/

http://www.metrocity.net

http://mkn.co.uk/

http://www.nettaxi.com

http://home.onestop.net/

http://www.yi.com/

http://www.howdyneighbor.com/

http://www.schlund.de/

http://www.cqws.com/rates.html

http://jungle.fapema.br/

http://free.prohosting.com/

http://www.home.ch

http://www.earthonline.net/

http://www.freenation.com/

Бесплатная регистрация в поисковиках

1000 Christian Links In 72 Categories ADD-URL

http://www.newcreations.net/sermoncentral/addurl.html

PM FFA Link Page

http://11pm.com/links/links.htm

Register's Cool Site Award

http://www.123register.com/award.html

Mockingbird Lane FFA Link Page

http://1313mockingbirdlane.com/links/links.htm

Have Fun FFA Link Page
http://www.2havefun.com/links/links.htm

FFA Link Page
http://www.4u2.de/

Free Banner Exchange
http://1-2-free.com/

Banners
http://www.123banners.com/

AmericaMall
http://www.1second.com/1america.htm

StopBiz
http://www.1stopbiz.com/

FFA Link Page
http://212.net/

BuySell
http://www.2buysell.com/

Cool Web Coolest Site
http://www.2coolweb.com/

Hand Internet Market
http://www.2him.com/

Exchange
http://www.x-x.com/

Лучшие русскоязычные хостинг-серверы

H1.ru

Предоставляют очень качественный хостинг, с очень большими возможностями. Неограниченное пространство, нужно только доказать его необходимость. Единственный недостаток — их банер на каждой вашей странице. Но множество достоинств (таких как FTP, CGI, серверные логи, почтовый адрес, поддержка DNS), с лихвой его покрывают.

By.ru

Новый, набирающий силу хостинг. Пока для новых проектов — неограниченное место + нет рекламы (оставляют за собой право на ее размещение). CGI нет, но зато есть поддержка SSI(!) FTP, гостевая книга и т.д. Существует программа партнерского хостинга (полный набор услуг платного хостинга), в которую вступить практически невозможно (требуют 1000 уникальных посетителей в сутки).

Webservis.ru

Тоже относительно новый сервис с огромным количеством услуг. Неограниченное пространство, поддержка CGI, но в скрипты обязательно нужно вставлять их банеры, большое количество разных услуг для облегчения сложной жизни web-мастера. Если вам не нужна поддержка скриптов, то банеры размещать не обязательно. Интересные домены: имя.bos.ru, имя.al.ru и еще несколько доменов, дают почту (например, имя@bos.ru = любое@имя.bos.ru).

Lgg.ru

Возрождение старого интересного проекта, дают 30 Мб, поддержка cgi, mySQL, php4; их банеры устанавливать не обязательно. Возможность создать 3 субдомена сайта (друг.мойсайт.lgg.ru) и дать ему свой отдельный e-mail (5 штук всего). Устойчивость еще неизвестна, но скорость высокая. Если ваш проект будет посещаемым, то дают домен второго уровня.

Hut.ru

Качественный, профессиональный хостинг. Поддержка всего. Нужно устанавливать их банер 486x60 (но можно договорится самим устанавливать). Абсолютно те же условия, что и у H1, даже сами правила как будто переписали у известного хостера!

Narod.ru

Широко известный сервер. Много Мб. FTP, Почта. CGI нет, но много разных примочек (гостевые книги, анкеты, чаты, форумы, счётчики и прочее). Рекламы нет, но оставляют за собой право на её размещение. Подходит для начинающих.

Boom.ru

Мало чем отличается от Narod'a. 50 (может уже больше) Мб, FTP. CGI нет. Примочек не так много, но быстро развивается. Рекламы нет, но оставляют за собой право на её размещение.

Newmail.ru

Аналог 2-х предыдущих серверов. 32 Мб, FTP, почта. CGI нет. Рекламы нет, но оставляют за собой право на её размещение. Если вы житель Москвы, для вас доступ на сервер по обычному телефонному номеру.

Sunday.ru, Tam.ru, Zk.ru, Home.c99.ru

Четыре аналогичных сервиса. Предоставляют 100 Мб места. Набор готовых к установке скриптов. Открыли новый сервер, позволяющий размещать cgi-скрипты. Рекламы нет.

Diaspora.ru

Web-портал «Планета Диаспора» предоставляет место для интересных проектов. Мегабайты роли не играют. FTP, CGI, Почта. Реклама есть, но ограничиваются всего несколькими банерами. Хорошо работает служба поддержки.

Chat.ru

Дают адрес chat.ru/имя или имя.chat.ru. Гостевая и другая мелочь, очень средний сервер, поддержки cgi нет. Дают майл адрес.

Каталог бесплатного хостинга

Сайт	Язык	Размер
AGAVA	Russian	999
AngelCities	English	15
Angelfire	English	30
AOL	English	12
Barry's World	English	20
Bobsville	English	2,5
Boom.Ru	Russian	50
Bootbox	English	10
BurgeNet	English	20
By.ru	Russian	999
CFM-resources	English	30
Chat	Russian	20
Cheatman	English	999

ClassicGaming	English	999
Cool Free Pages	English	50
Cybercities	English	25
Demented	English	20
Digitalrice	English	5
DomainDLX	English	15
Easyspace	English	25
Eccentrix	English	15
EGAO	English	50
Finitesite	English	15
FortuneCity	English	100
FreeHomepages	English	5
FreeServers	English	20
Freewebfile	English	50
GameDummys	English	25
Geocities	English	15
Graffiti	English	20
Grlspace	English	10
HealthBlast	English	15
HostUltra	English	999
HotyellowCity	English	15
Idlecity	English	20
Imbri	English	20
Infhost	English	5
IntelCities	English	3
JoinMe	English	5
KM.RU	Russian	999
Live Domain	English	20

LiveUniverse	English	0,5
MainQuad	English	15
Megspace	English	50
MixHost	English	25
MSN	English	12
Narod.Ru	Russian	999
Nerdcities	English	100
Netcolony	English	35
Netomia	English	60
Netscape	English	11
NewMail.Ru	Russian	32
Onenet	English	10
Ournet	English	2
Phatstart	English	5
PHP50	English	50
Polbox	English	2
Power Serve	English	5
Qwikpages	English	100
Real Hosting	English	20
Rediff	English	10
RedRival.com	English	20
Rootsweb	English	100
Russian Zone	Russian	999
SDF	English	10
Searchalot	English	10
Sitepalace	English	8
Solcities	English	5
Southhosting	English	2

Space Towns	English	50
Spaceports	English	10
SpanCity	English	999
Sphosting.com	English	25
Spree	English	999
Starving-art	English	5
Stas	English	999
StaticFree	English	10
Stormloader	English	20
T35	English	35
TalkCity	English	12
Telefragged	English	999
Terrashare	English	25
TheDeck	English	1
TheGeekHost	English	20
Thirdage	English	9
TimeCities	English	15
Tiscali.cz	English	10
Topcities	English	50
Tripod	English	50
Tripod UK	English	12
U-build-it	English	10
Ugo	English	12
UnoDosTres	English	2
Vavo	English	999
Verica	English	50
VGF.com	English	100
Viaduk	English	5

Virtue	English	5,5
Web1000	English	50
Webpagebiz	English	3
WebSamba	English	30
Webservis.ru	Russian	100
WebSiteForFree	English	7,5
WhoWhere	English	5
Wigloo	English	5
Women	English	5
Worldzone	English	100
Xism	English	20
X-Mail	English	20
Xoing	English	20
Xoom	English	999
Yoogo	English	15
YourOwnSpace	English	5
Yourwebhome	English	50
YouthofAmerica	English	20
Zcities	English	1
Zenation	English	50
ZoomGo	English	20
Zy	English	10

Краткий словарь Интернета

Active Channel

Узел Web, автоматически поставляемый на рабочий стол пользователя.

Active Desktop

Интерфейс, интегрированный с рабочим столом Windows и обозревателем Microsoft Internet Explorer.

ActiveMovie

Технология цифрового видео, позволяющая через Web просматривать файлы AVI, QuickTime или MPEG.

ActiveX

Термин программного интерфейса технологии Microsoft, позволяющей разработчикам создавать интерактивное содержимое для WWW, а также для компонентов программного обеспечения, написанных на различных языках. Основные элементы технологии ActiveX — COM и DCOM.

Address

Уникальный код с информацией, находящейся в файле конкретной сети.

Anchor

То, что собственно, и образовывает гипертекстовую ссылку.

Anonymous

Один из методов получения доступа к той или иной информации. Вы можете только копировать файлы, передавать свои — нет.

ANSI

American National Standards Institute.

Выделенный канал

Постоянное соединение, всегда позволяющее передать поток информации от одного компьютера к другому.

Authorization

Право, которое дается пользователю на тот или иной ресурс компьютерной системы.

BBS

Bulletin Board System. Тип компьютерного сервиса. Юзеры могут читать и публиковать различные сообщения. Передавать или скачивать файлы.

Big mail

Электронная почта огромных размеров. Передается некоторыми сволочами в качестве мести.

Bookmark

Закладка. Файл Gopher или WWW. Информация в этом файле размещена так, что получить доступ к желаемой странице можно без дополнительного серфинга.

Browser

Нечто, похожее на графический интерфейс. Позволяет блуждать по той или иной сети, или искать себе на определенное место приключений.

Bullet

Маркер. Элемент оформления текста. Например, маленький круг, квадрат, звездочка.

Character Set

Символы, которые объединены в определенную группу. Например, ASCII.

Character

Любой символ, введенный с клавиатуры в компьютер.

Chart

Диаграмма. Рисунок, показывающий взаимодействие тех или иных данных.

Check Box

Флажок в виде крестика. Один из многочисленных элементов графической операционной системы. Позволяет включать или отключать то или иное действие.

CompuServe

Коммерческая компьютерная сеть США.

Cookies

Технология, позволяющая сохранять сугубо индивидуальную информацию о пользователе сети.

Cracker

Хакер, который ломает программы + «делает лекарства».

Cybermall

Электронный магазин.

Cyberpunk

Самоутвержденный хакер.

DejaNews

Ищем новости в конференциях Usenet по ключевому слову, автору статьи или по адресу. Доступ через http://www.dejanews.com.

DNS

Domain Name System. Доменная система адресации. Нечто, похожее на базу данных и конвертирующее буквенное сетевое имя в набор цифр. Этот набор цифр является числовым адресом Интернет. Понятен только компьютерам и некоторым фанатам с шизофренической памятью.

Domain Name Server

Имя сервера домена. Каждое подразделение Интернет имеет два домена. Основной DNS обычно располагается на сетевой машине. DNS — сервера используют в своих обращениях к удаленным узлам 32 — битные адреса IP, мнемонически заключенные в четырехразрядную буквенную комбинацию. Любой хост может получить соответствующий DNS у ближайшего информационного сервера DNS по известной системе Domain Name Server через сетевой протокол DNS. Просто хост посылает запрос на известный IP — адрес DNS — сервера свой IP — адрес и имя сервера. Сервер DNS штудирует собственную базу данных, находит IP — адрес и отправляет на хост соответствующий ответ DNS. Схема весьма примитивная. Если же сервер DNS не находит искомую буквенную комбинацию, то он отсылает запрос на так называемый корневой сервер, который, в свою очередь, сверяет информацию с файлом настроек root.cache. Так происходит до тех пор, пока имя хоста не будет найдено в Интернет.

Domain

Подразделение Интернет. У каждого домена есть своя метка. Например, метки .com, .net, .mil, .org означают, что это соответственно домен коммерческой, сетевой, военной и общественной организации.

Download

Закачивание софта с другого компьютера на собственный винчестер.

Ethernet

Один из видов сети с весьма классной пропускной способностью. Довольно часто компьютеры, использующие протоколы TCP/IP подсоединяются к Интернет через Ethernet.

Font

Шрифт или семейство знаков с индивидуальным стилем.

Freeware

Бесплатное программное обеспечение.

FTP

File Transfer Protocol. Метод пересылки файлов на ваш компьютер. В Интернет имеются тысячи мест, поддерживающих этот метод. Иногда единственная возможность заиметь файл — это воспользоваться протоколом FTP. Помните об этом протоколе.

FTP-client

Софт, позволяющий вам приконнектиться к серверу FTP.

FTP-команды

Команды с синтаксисом FTP.

FTP-mail

Крутейшая штучка. Вы можете получать файлы с серверов FTP по электронной почте.

FTP-server

Компьютер, битком набитый всякой всячиной и поддерживающий протокол FTP.

GIF

Graphic Interchange Format. Один из стандартных графических файлов WWW.

HTML

HyperText Markup Language. Язык, лежащий в основе формирования документов World Wide Web.

HTTP

HyperText Transport Protocol. Система, передающая документы HTML по World Wide Web.

Hypermedia

Гиперсреда. Через определенные ссылки связывается текст, графика, звук, видео и картинки.

Hypermedia

Нечто, которое хорошо знают хакеры. Грубо говоря то, что дискретно может отобразить фотографию на выбранном вами web-обозревателе.

Hypertext

Нечто представляющее данные так, что можно легко организовать межстраничные связи различных документов Интернет.

Internet

Мировая с англоязычным уклоном свободная конфедерация компьютерных сетей, объединяющая более 12 тысяч локальных сетей, более одного миллиона компьютеров и около 30 миллионов юзеров. Негласный победитель FIDO.

IP Address

Тридцатидвухбитовый (ну и словечко) адрес протокола Интернет, включающий в себя номер узла и сети.

IP

Самый важный из всех протоколов, на котором основана Интернет. Через этот протокол осуществляется прямое подключение к Интернет.

IRC

Internet Relay Chat. Одна из форм WWW-психоза. IRC чем-то напоминает работу в конференциях Usenet. Но если там вы общаетесь не в реальном времени, то здесь может вестись живой разговор. Разве что, — вас никто не слышит. Вас могут прочитать. Вы набиваете текст на клавиатуре. Ваша информация попадает на общий дисплей. Различные группы видят ваш бред. Если интересно — отвечают.

JPEG

Joint Photographic Experts Group. Графический формат, принятый всеми в качестве стандартного формата при оформлении Web-страниц.

Lamer

Юзер-идиот. Термин, придуманный американскими системными администраторами.

Login

Регистрационное имя пользователя.

Lurk

Пассивные наблюдатели. Например, те, кто только читают новости конференций Usenet или сообщения, проходящие по каналам IRC.

Mail server

Почтовый сервер. Компьютер, обрабатывающий электронную почту.

Mail

Обмен частными текстовыми сообщениями в Интернет или в другой компьютерной сети.

Mailing list

Список рассылки электронных почтовых сообщений, классифицированных по определенным темам. Своеобразная подписка.

Marquee

Бегущая строка в документе HTML.

Media

Почти то же, что Multimedia.

Microsoft FrontPage Express

Редактор web-страниц. Средство для создания и оформления документов HTML в режиме непосредственного отображения.

MIME

Multipurpose Internet Mail Extensions. Протокол передачи звука, графики и других двоичных данных. Применяется при передаче почтовых сообщений. И как только этот MIME не интерпретируют!

Mosaic

Древнейший графический пользовательский интерфейс, позволяющий просматривать World Wide Web. Имеются версии для X-Window, Windows и Macintosh.

MPEG

Moving Pictures Expert Group. Протокол, через который упаковываются видеозаписи.

Multimedia

Весьма туманный термин, не переводящийся на нормальный язык.

Netscape Communicator

Еще один обозреватель Интернет. Графический пользовательский интерфейс, позволяющий просматривать World Wide Web. Имеются версии для X Window, MS Windows и Macintosh. Весьма неплохой.

Newsgroup

Область сообщений в конференциях Usenet.

NNTP

Сетевой протокол, с помощью которого весь мир пользуется прелестями от конференций Usenet.

Offline

Автономный режим работы компьютера.

Online

Интерактивный режим работы сетевого компьютера.

Page

Страница. Любой документ World Wide Web.

Prompt

Поле данных удаленного терминала.

Protocol

Метод, с помощью которого передается информация от хоста к юзеру.

Provider

Поставщик некоторых услуг. В частности, — услуг Интернет.

216 web-безопасных цветов

FFFFFF	CCCCCC
999999	666666
333333	000000
FF0000	FF3333
CC0000	FF6666
CC3333	990000
FF9999	CC6666
993333	660000
FFCCCC	CC9999
996666	663333
330000	FF3300
FF6633	CC3300
FF9966	CC6633
993300	FF6600
FF9933	CC6600
FFCC99	CC9966
996633	663300
FF9900	FFCC66
CC9933	996600
CC9900	FFCC33
FFCC00	FFFF00
FFFF33	CCCC00
FFFF66	CCCC33
999900	FFFF99
CCCC66	999933
666600	FFFFCC
CCCC99	999966

666633	333300
CCFF00	CCFF33
99CC00	CCFF66
99CC33	669900
99FF00	99FF33
66CC00	CCFF99
99CC66	669933
336600	66FF00
99FF66	66CC33
339900	66FF33
33CC00	33FF00
00FF00	33FF33
00CC00	66FF66
33CC33	009900
99FF99	66CC66
339933	006600
CCFFCC	99CC99
669966	336633
003300	00FF33
33FF66	00CC33
66FF99	33CC66
009933	00FF66
33FF99	00CC66
99FFCC	66CC99
339966	006633
00FF99	66FFCC
33CC99	009966
3300CC	9966FF

6633CC	330099
6600FF	9933FF
6600CC	CC99FF
9966CC	663399
330066	9900FF
CC66FF	9933CC
660099	CC33FF
9900CC	CC00FF
FF00FF	FF33FF
CC00CC	FF66FF
CC33CC	990099
FF99FF	CC66CC
993399	660066
FFCCFF	CC99CC
996699	663366
330033	FF00CC
FF33CC	CC0099
FF66CC	CC3399
990066	FF0099
FF3399	CC0066
FF99CC	CC6699
993366	660033
FF0066	FF6699
CC3366	990033
FF3366	CC0033
FF0033	33FFCC
00FFCC	00FFFF
33FFFF	00CCCC

66FFFF	33CCCC
009999	99FFFF
66CCCC	339999
006666	CCFFFF
99CCCC	669999
336666	003333
00CCFF	33CCFF
0099CC	66CCFF
3399CC	006699
0099FF	3399FF
0066CC	99CCFF
6699CC	336699
003366	0066FF
6699FF	3366CC
003399	3366FF
0033CC	0033FF
0000FF	3333FF
0000CC	6666FF
3333CC	000099
9999FF	6666CC
333399	000066
CCCCFF	9999CC
666699	333366
000033	3300FF
00CC99	6633FF

Список использованных материалов

Альтернатива «Блокноту», или в чем писать HTML?
Дмитрий Степанов

Что такое портал?
Андрей Акопянц

Так значит вам нужен именно web-мастер?
Андрей Ерофеев

Сделать сайт в оффлайне, от и до? Реально!
Олег Титов

О структуре оптимизированных страниц
Бурцев Игорь

Как правильно создать pop-up страницы
Бурцев Игорь

Как изменить цвета скролл-бара у Internet Explorer 5.5 и выше
Евгений Жданов

Несколько слов о хорошей раскрутке
Полянко Александр

Как создать вирусный трафик с помощью бесплатных электронных книг
Дмитрий Смакотин

Как правильно раскрутить проект и привлечь нужных посетителей.
А. М. Максимов

Экономика проекта
Сергей Хрипунов

25 преимуществ использования Интернета в сетевом маркетинге
Дмитрий Смакотин

Какой вид дизайна выбрать?
Поликарпов Роман

Критические ошибки
Поликарпов Роман

Права доступа (CHMOD)
Поликарпов Роман

Web-дизайн
Бочкарёв Антон

Что такое web-дизайн?
Бочкарёв Антон

Основы Web-мастерства
В. Холмогоров

HTML-редакторы для профессионалов
Елена Гореткина

Материалы сайта http://www.web-window.ru

Научно-популярное издание

Орлов Леонид Владимирович

Web-сайт без секретов

Главный редактор *Б. К. Леонтьев*
Шеф-редактор *А. Г. Бенеташвили*
Оригинал-макет *И. В. Царик*
Художник *О. К. Алехин*
Художественный редактор *М. Л. Мишин*
Технический редактор *К. В. Шапиро*
Корректоры *Л. С. Зимина, К. В. Толкачева*

Подписано в печать 20.05.2006. Формат 60×90/16.
Гарнитура «Ньютон». Бумага офсетная. Печать офсетная.
Печ. л. 32. Тираж 3000.