**Министерство образования и науки Российской Федерации**

Старооскольский технологический институт им. А.А. УГАРОВА

(филиал) федерального государственного автономного образовательного учреждения

высшего образования

«Национальный исследовательский технологический университет «МИСиС»

**ОСКОЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ**

УТВЕРЖДено

НМС опк

пРОТОКОЛ №1

ОТ «01» сентября 2016г.

**ПМ.03. ВЫПОЛНЕНИЕ РАБОТ ПО ПРОФЕССИИ «ОПЕРАТОР ЭЛЕКТРОННО-ВЫЧИСЛИТЕЛЬНЫХ И ВЫЧИСЛИТЕЛЬНЫХ МАШИН»**

**МДК.03.01. ОБРАБОТКА ИНФОРМАЦИИ С ПОМОЩЬЮ ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА**

***Методические указания для студентов очной формы обучения для выполнения практических заданий и внеаудиторной самостоятельной работы***

Часть 4. Технологии создания сайта

Специальность 09.02.04 Информационные системы (по отраслям)

**Старый Оскол 2016**

|  |  |
| --- | --- |
| *Рассмотрены на заседании П(Ц)К 09.02.04*  *Протокол № 1*  *от «01» сентября 2016г.*  *Председатель*  *Назарова О.И.* | *Методические указания составлены в соответствии с рабочей программой по ПМ 03. Выполнение работ по профессии «Оператор электронно-вычислительных и вычислительных машин»*  *Специальности*  *09.02.04 Информационные системы (по отраслям)*  *Зам .директора по М Р*  *к.п.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.М. Степанова* |

***Составитель: Артюхина Д.Д.***

***Рецензенты:***

*внутренний: Коренькова Т.Н.* - преподаватель ОПК СТИ НИТУ МИСиС

*внешний:* Анпилов А.Э. – инженер ООО «КМАЭМ»

**Содержание**

[РАЗДЕЛ 4. ТЕХНОЛОГИИ СОЗДАНИЯ САЙТА 5](#_Toc432537184)

[Практическая работа №23 5](#_Toc432537185)

[Практическая работа №24 12](#_Toc432537186)

[Практическая работа №25 18](#_Toc432537187)

[Практическая работа №26 25](#_Toc432537188)

[Практическая работа №27 40](#_Toc432537189)

[Практическая работа №28 49](#_Toc432537190)

[Практическая работа №29 56](#_Toc432537191)

[Практическа работа №30 69](#_Toc432537192)

[ЗАДАНИЯ ДЛЯ ВНЕАУДИТОРНОЙ САМОСТОЯТЕЛЬНОЙ РАБОТЫ 98](#_Toc432537193)

[Раздел 4. Технологии создания сайта 98](#_Toc432537197)

[Часть 1. Принципы реализации Интернет-проекта 98](#_Toc432537198)

[Часть 2. Проектирование Web-сайта средствами языка HTML на примере сайта информационного агентства. 99](#_Toc432537199)

[Часть 3. Создание Web-сайтов средствами MS Office 113](#_Toc432537200)

[Часть 4. Создание Web-сайтов средствами MS FrontPage 118](#_Toc432537201)

[СПИСОК ЛИТЕРАТУРЫ 132](#_Toc432537202)

**Введение**

Современную жизнь трудно представить без мультимедиа. Однако даже, несмотря на то, что мультимедийными технологиями ежедневно пользуются десятки тысяч людей, далеко не каждый из них имеет представление о том, что означает данное понятие.

В наши дни переоценить значение мультимедиа практически невозможно. Это связано с тем, что мультимедийные технологии с каждым днем все более активно приходят в наши дома. Однако необходимо добавить и то, что благодаря мультимедиа мы получаем огромную пользу. В качестве примера можно сказать о том, что современные мультимедийные технологии нашли свое широкое применение в обучающей сфере. Благодаря их использованию усвоение информации улучшилось в значительной степени.

Термин «мультимедиа» с английского можно перевести как «многие Среды» (от multi - много и media - среда).

В настоящее время мультимедиа-технологии являются бурно развивающейся областью информационных технологий. В этом направлении активно работает значительное число крупных и мелких фирм, технических университетов и студий (в частности IBM, Apple, Motorola, Philips, Sony, Intel и др.). Области использования чрезвычайно многообразны: интерактивные обучающие и информационные системы, САПР и др.

Благодаря развитию мультимедийных технологий появилась возможность объединять многокомпонентную среду (текст, звук, графику, видео, фото) в однородное цифровое представление и надежно и долго сохранять большие объемы информации. Информация гарантировано хранится не менее десяти лет. При этом переработка информации превращается из рутинных операций в творческие.

Основными характерными особенностями этих технологий являются:

* объединение многокомпонентной информационной среды (текста, звука, графики, фото, видео) в однородном цифровом представлении;
* обеспечение надежного (отсутствие искажений при копировании) и долговечного хранения (гарантийный срок хранения - десятки лет) больших объемов информации;
* простота переработки информации (от рутинных до творческих операций).

На сегодняшний день мы не можем не задумываться над тем, что ожидает наших учащихся. Известно, что будущее потребует от них огромного запаса знаний в области современных технологий. Сегодня уже 60% предложений о работе требуют минимальных компьютерных знаний, и этот процент будет возрастать. Но подготовка молодёжи к будущему заключается не только в плане «готовности работать». Учащиеся должны освоить новые жизненно необходимые навыки в связи с тем, что современные информационные технологии всё глубже проникают в нашу жизнь. Информационный депозитарий глобальной компьютерной сети Интернет настолько велик, что умение извлечь из такого большого объёма информации нужный кластер выходит на первый план.

Таким образом, использование мультимедиа технологий в учебном процессе не только целесообразно, но и позволяет достичь цели, которую ставит перед педагогами «Концепция модернизации Российского образования» - подготовка разносторонней развитой личности.

## *РАЗДЕЛ 4. ТЕХНОЛОГИИ СОЗДАНИЯ САЙТА*

## Практическая работа №23

Основные средства языка html

Цель: изучить основные средства языка html

Студент должен:

иметь практический опыт:

осуществления навигации по ресурсам, поиска, ввода и передачи данных с помощью технологий и сервисов Интернета (ПО2);

создания и обработки объектов мультимедиа (ПО3);

уметь:

создавать и редактировать объекты мультимедиа (У2);

знать:

назначение, разновидности и функциональные возможности программ для создания объектов мультимедиа (З2).

Введение в HTML

Основные понятия

Для создания Web-страницы можно воспользоваться специальными программами редактирования документов Всемирной паутины. Другой способ подготовки Web-страниц заключается в «ручном» создании кода документов на языке HTML – HyperText Markup Language – Язык разметки гипертекста. Данный язык представляет собой довольно простой набор команд, описывающий структуру документа. Язык HTML позволяет выделить в документе отдельные элементы – заголовки, абзацы, таблицы и т.д. Файлы с текстом кода на языке HTML имеют расширение .html или .htm.

HTML является описательным языком разметки документов, в нем используются указатели разметки (теги). Теговая модель описывает документ как совокупность контейнеров, каждый из которых начинается и заканчивается тегами, то есть документ НТМL представляет собой не что иное, как обычный файл, с добавленными в него управляющими НТМL-кодами (тегами). В нем разрешено использовать только три управляющих символа: горизонтальную табуляцию, перевод каретки и перевод строки. Это облегчает взаимодействие с различными операционными системами.

Теги НТМL-документов в большинстве своем просты и понятны, ибо они образованы с помощью общеупотребительных слов английского языка, понятных сокращений и обозначений. НТМL-тег состоит из имени, за которым может следовать необязательный список атрибутов тега. Текст тега заключается в угловые скобки ("<" и ">"). Простейший вариант тега — имя, заключенное в угловые скобки, например <HEAD> или <I>. Для ряда тегов характерно наличие атрибутов (т.е. параметров тега), которые могут иметь конкретные значения, устанавливаемые автором для изменения функции тега.

Например, при описании таблицы открывающий тег с атрибутами может выглядеть так:

<TABLE width=570 align=center cellpadding=10 cellspacing=2 border=16>

Эта запись означает следующее: таблица шириной 570 пикселов, выровнена по центру, поле между рамкой и содержимым ячеек 10 пикселов, поле рамки 2 пиксела, ширина бордюра 16 пикселов.

Атрибуты тега следуют за именем и отделяются друг от друга пробелами. Порядок записи атрибутов в теге значения не имеет. Атрибут тега состоит из имени, знака равенства и значения – language=“JavaScript”. В тегах могут использоваться только символы латинского алфавита, а в значениях атрибутов – любые символы. Если значение атрибута — одно слово или число, то его можно просто указать после знака равенства, не выделяя дополнительно. Все остальные значения необходимо заключать в кавычки, особенно если они содержат несколько разделенных пробелами слов. Длина значения атрибута ограничена 1024 символами. Теги принято писать заглавными буквами, а атрибуты и их значения – прописными (например: <BODY text=black>), это не учитывается программой , но облегчает прочтение кода.

Чаще всего элементы разметки HTML или HTML-контейнеры состоят из начального и конечного компонентов, между которыми размещаются текст и другие элементы документа. Имя конечного тега идентично имени начального, но перед именем конечного тега ставится косая черта (/) (например, для тега стиля шрифта — курсив <I> закрывающая пара представляет собой </I>, для тега заголовка <ТIТLЕ> закрывающей парой будет </ТIТLЕ>). Атрибуты прописываются в начальных тегах. Конечные теги никогда не содержат атрибутов. Теги определяют область действия правил интерпретации текстовых документов.

Некоторые элементы разметки не имеют конечного компонента, поскольку являются автономными элементами. Например, тег изображения <IMG>, который служит для вставки в документ графического изображения, конечного компонента не требует. К автономным элементам разметки также относятся разрыв строки (<BR>), горизонтальная линейка (<HR>) и теги, содержащие такую информацию о документе, которая не влияет на его отображаемое содержимое, например тег <META> В некоторых случаях конечные теги в документе можно опускать. Большинство браузеров устроено так, что при обработке текста документа начальный тег воспринимается как конечный тег предыдущего. Самый распространенный тег такого типа — тег абзаца <Р>. Поскольку он используется в документе очень часто, его обычно ставят только в начале каждого абзаца. Когда один абзац заканчивается, следующий тег <Р> сигнализирует браузеру о том, что нужно завершить данный абзац и начать следующий. Большинство авторов тегом конца абзаца не пользуются.

Для краткости и образности мы будем в ряде случаев вместо словосочетания "элемент разметки" применять термин "контейнер".

Общая схема построения контейнера в формате HTML может быть записана в следующем виде:

"контейнер"= <"имя тега" "список атрибутов"> содержание контейнера </"имя тега">

Следует отметить, что в литературе кроме термина "контейнер" еще используется и термин "элемент". Следует быть внимательным, чтобы не путать контейнер (например, BODY) и тег (BODY), используемый при формировании контейнера.

Кроме тегов, элементами HTML являются CER (Character Entity Reference), они предназначены для представления специальных символов в документе HTML, которые могут быть неверно обработаны браузером. Предположим, создается документ HTML, речь в котором идет об элементах данного языка. Если указать имя тега <BODY> просто в документе, браузер может воспринять его как непосредственно старт-тег. Для вывода таких символов и используется CER.

Например, чтобы представить символ "<" в документе HTML, нужно заменить его на &lt;, а символ ">" — на &gt;. То есть, если указать в тексте HTML строку &lt;BODY&gt;, она будет выглядеть на экране как текст <BODY>.

Может возникнуть вопрос: как быть с символами "</>", "&" и со специальными символами, типа знака ударения? Можно выводить их, используя соответствующие CER, например для "&" это будет &amp;, и т. д.

CER легко обнаружить, если посмотреть на структуру любого документа HTML, поскольку каждый из них начинается с амперсанта "&". В отличие от наименований тегов HTML, наименования CER чувствительны к регистру символов. Также наименования CER могут задаваться не в виде имени, а с помощью трехзначных кодов символов в виде &#nnn;. Далее в таблице приведены наиболее часто используемые CER и соответствующие им числовые коды.

|  |  |  |  |
| --- | --- | --- | --- |
| Числовой код | Именная замена | Символ | Описание |
| &#034; | &quot; | " | Кавычка |
| &#038; | &amp; | & | Амперсант |
| &#060; | &lt; | < | Меньше |
| &#062; | &gt; | > | Больше |
| &#160; | &nbsp; |  | Неразрывный пробел |
| &#161; | &iexcl; | ¡ | Перевернутый восклицательный знак |
| &#162; | &cent; | ¢ | Цент |
| &#163; | &pound; | £ | Фунт |
| &#164; | &curren; | ¤ | Валюта |
| &#165; | &yen; | ¥ | Йена |
| &#168; | &uml; | ¨ | Умляут |
| &#169; | &copy; | © | Копирайт |
| &#171; | &laquo; | « | Левая угловая кавычка |
| &#174; | &reg; | ® | Зарегистрированная торговая марка |
| &#177; | &plusmn; | ± | Плюс или минус |
| &#187; | &raquo; | » | Правая угловая кавычка |

Гипертекст породил много специальных терминов:

Элемент – конструкция языка HTML. Это контейнер, содержащий данные и позволяющий отформатировать их определенным образом. Любая Web-страница представляет собой набор элементов. Одна из основных идей гипертекста – возможность вложения элементов.

Тег – начальный или конечным маркеры элемента. Теги определяют границы действия элементов и отделяют элементы друг от друга. В тексте Web-страницы теги заключаются в угловые скобки, а конечный тег всегда снабжается косой чертой.

Атрибут – параметр или свойство элемента. Это, по сути, переменная, которая имеет стандартное имя и которой может присваиваться определенный набор значений: стандартных или произвольных. Предполагается, что символьные значения атрибутов заключаются в прямые кавычки, но некоторые браузеры позволяют не использовать кавычки. Это объясняется тем, что тип атрибута всегда известен заранее. Атрибуты располагаются внутри начального тега и отделяются друг от друга пробелами.

Гиперссылка – фрагмент текста, который является указателем на другой файл или объект. Гиперссылки необходимы для того, чтобы обеспечить возможность перехода от данного документа к другому.

Структура Web-страницы

Структура HTML-документа позволяет задействовать вложенные друг в друга контейнеры. Собственно, сам документ — это один большой контейнер, который начинается с тега <HTML> и заканчивается тегом </HTML>. Он указывает браузеру, что данный текст представляет собой HTML-документ и, содержит в себе теги, которые браузер должен выявить, распознать и правильно интерпретировать.

Типичная Интернет-страница состоит из двух частей: головная часть[[1]](#footnote-1)\* (HEAD) и тела (BODY). Эту базовую структуру в простейшем виде можно представить следующим образом:

|  |  |
| --- | --- |
| <HTML> | Начало HTML-документа |
| <HEAD> | Начало головной части |
| <TITLE> | Начало строки названия страницы |
| … | Строка названия страницы |
| </TITLE> | Конец строки названия страницы |
| </HEAD> | Конец головной части |
| <BODY> | Начало тела документа |
| … |  |
| </BODY> | Конец тела документа |
| </HTML> | Конец HTML-документа |

<HTML> </HTML>

Элемент является самым внешним, так как между его начальным и конечным тегами должна находится вся страница. Этот элемент можно рассматривать как формальность.

<HEAD> </HEAD>

Область заголовка Wеb-страницы. Служит для формирования общей структуры документа. Должен включать элемент TITLE и допускает вложение элемента META.

<TITLE></TITLE>

Элемент разметки TITLE служит для именования документа в World Wide Web. Более прозаическое его назначение — именование окна браузера, в котором просматривается документ. Наличие конечного тега обязательно.

Синтаксис контейнера TITLE в общем виде выглядит следующим образом:

<TITLE>название документа</TITLE>

Заголовок не является обязательным контейнером документа. Его можно опустить. Роботы многих поисковых систем используют содержание элемента TITLE для создания поискового образа документа. Слова из TITLE попадают в индекс поисковой системы. Из этих соображений элемент TITLE всегда рекомендуется использовать на страницах Web-узла.

Нужно позаботиться о том, чтобы это строка, не будучи слишком длинной, достаточно точно отражала назначение документа. Если тег <TITLE></TITLE> отсутствует, в заголовке браузера выводится реальный адрес и имя просматриваемого html-файла.

<META></META>

Содержит служебную информацию, которая не отражается при просмотре. Внутри нет текста, поэтому он не имеет конечного тега. Этот тег специально рассчитан на программу поискового сервера, индексирующую web-страницы. Секция заголовка может содержать несколько элементов <META>, каждый из которых отвечает за определенный набор параметров.

Может содержать:

срок годности документа;

адрес электронной почты;

имя автора страницы;

набор ключевых слов для поиска;

краткое описание содержания Web-страницы;

Описание типа и характеристик Web-страницы;

Указание приложения, в котором была создана Web-страница;

URL

Наличие этого тега значительно увеличивает шансы попасть в первую десятку адресов, найденных поисковым сервером.

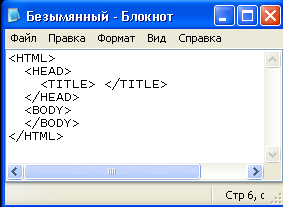
<BODY></BODY>

Это собственно тело документа. Это та произвольная часть документа, которую разрабатывает автор страницы и которая отображается браузером. Конечный тег этого элемента располагается в конце html-кода. В этом элементе могут использоваться все элементы, предназначенные для дизайна web-страницы. Внутри начального тега <BODY> можно расположить ряд атрибутов, обеспечивающих установки для всей страницы целиком, такие как, цвет фона, фоновую картинку, цвет текста и гиперссылок и т.д.

Создание Web-страницы

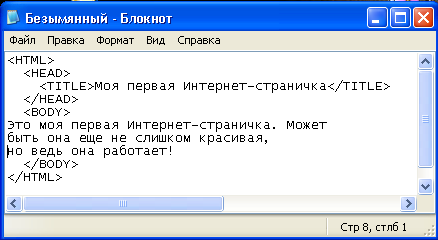
Так как все html-документы имеют одинаковую структуру, рекомендуется создать общий шаблон, в котором будут меняться только название (содержимое тега <TITLE>…</TITLE>) и содержательная часть документа (содержимое контейнера <BODY>…</BODY>.

Создавать html-код лучше в простом текстовом редакторе, например, в программе Блокнот. Для создания документа необходимо запустить программу Блокнот и ввести общий для всех страниц код, который определяет структуру html-документа:



Этот документ можно сохранить под именем «шаблон» (Файл/ Сохранить как/ шаблон.txt) и использовать в дальнейшем в качестве заготовки для создания других документов.

Теперь, чтобы создать web-страницу достаточно открыть файл шаблон.txt, прописать название документа и его содержательную часть:

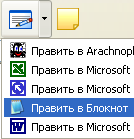


далее сохранить документ в свою папку под другим именем и дать ему расширение .htm (предпочтительно) или .html (Файл/Сохранить как в строке имя файла ввести: имя\_документа.htm и нажимает на кнопку Сохранить):



Теперь, чтобы просмотреть страницу, нам достаточно открыть полученный файл начало.htm. Однако, в браузере мы можем только просматривать страничку, а чтобы вносить изменения можем только html-коде. Есть несколько способов открыть html-код страницы:

Выдрать в меню команду Вид/Просмотр HTML-кода.

Выбрать на Панели инструментов клавишу Править в Блокнот

Кликнуть по экрану правок кнопкой мыши и выбрать в контекстном меню команду Просмотр HTML-кода.

Далее нужно внести изменения в код документа и сохранить в Блокноте. Закрыть Блокнот и в окне браузера нажать на кнопку Обновить на панели инструментов. После этого внесенные изменения отобразятся на экране.

Списки

Список (list)отличается от обычного текста прежде всего тем, что пользователю не надо думать о нумерации его пунктов: эту задачу программа берет на себя. Если список дополняется новыми пунктами или укорачивается, нумерация корректируется автоматически. В случае ненумерованных списков программа ставит перед каждым пунктом маркеры: кружки, прямоугольники, ромбы или другие изображения. В результате список принимает удобочитаемый, «фирменный» вид. Теги для создания списков можно условно разделить на две группы: одни определяют общий вид списка (и позволяют указывать атрибуты), а другие задают его внутреннюю структуру. В списках можно использовать стандартные атрибуты.

В НТМL имеются следующие виды списков: ненумерованный список (неупорядоченный) (Unordered Lists <UL>), нумерованный список (упорядоченный) (Ordered Lists <OL>) и список определений (<DL>).

Ненумерованный список

Самым простым является маркированный (ненумерованный) список.

Ненумерованный список предназначен для создания текста типа:

первый элемент списка;

второй элемент списка;

третий элемент списка.

Записывается данный список в виде последовательности:

<UL>

<LI>Первый элемент списка</LI>

<LI>Второй элемент списка</LI>

<LI>Последний элемент списка</LI>

</UL>

Теги <UL> и </UL> являются своеобразным обрамлением списка. Он позволяет отделить один список от другого. Тег <LI> (List Item) обозначает каждый из пунктов списка. Конечный тег </LI> не является обязательным.

Для тега <UL> предусмотрен атрибут type, определяющий вид маркера, который браузер помещает перед каждым элементом списка. Этот атрибут может отсутствовать или принимать одно из трех значений:

|  |  |  |
| --- | --- | --- |
| Начальный тег |  | Вид метки на экране |
| <UL> |  | Обычный диск (зависит от браузера) |
| <UL type=circle> | ○ | Окружность |
| <UL type=disc> | ● | Диск |
| <UL type=square> | ▫ | Квадрат |

Пример:

|  |  |
| --- | --- |
|  |  |

Нумерованный список

В нумерованных списках каждый элемент снабжен номером, вид и начальное значение которого настраивается специальными атрибутами.

Нумерованный список задается при помощи команды <OL>:

<OL>

<LI>Первый элемент списка</LI>

<LI>Второй элемент списка</LI>

<LI>Последний элемент списка</LI>

</OL>

Конечный тег </LI> не является обязательным.

Вид номера определяется значением атрибута type в теге <OL>:

|  |  |
| --- | --- |
| Начальный тег | Вид номера на экране |
| <OL> | Нумерация выполняется арабскими цифрами (1, 2, 3 и т.д.) |
| <OL type=1> | Нумерация выполняется арабскими цифрами (1, 2, 3 и т.д.) |
| <OL type=А> | Нумерация выполняется прописными буквами (А, В, С и т.д.) |
| <OL type=а> | Нумерация выполняется строчными буквами (a, b, c и т.д.) |
| <OL type=I> | Нумерация выполняется большими римскими цифрами (I, II, III) |
| <OL type=i> | Нумерация выполняется малыми римскими цифрами (i, ii, iii) |

Естественную нумерацию элементов списка (с единицы) можно изменить при помощи атрибута start=n. Число n задает начальное значение нумерации. Если для нумерации используются буквы или римские цифры, то стартовое значение равно n-му элементу в системе нумерации (например, 4 означает D или IV).

Пример:

|  |  |
| --- | --- |
|  |  |

Для элемента <LI> может быть использован атрибут value, который определяет номер для текущего пункта списка:

value=номер

Соответственно, следующий пункт списка будет иметь очередной номер и т.д. При помощи атрибута value, если использовать его для первого в списке элемента <LI>, можно добиться такого же эффекта, что и при помощи атрибута start, или нарушить последовательность номеров, если переопределить другие элементы списка.

Списки определений

Теги списка (Definition List: <DL>, <DT>, <DD>) используют для создания списка терминов и их определений. Схема использования тега следующая.

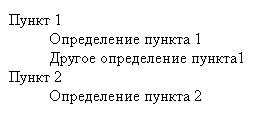
Задается тегом <DL> и имеет следующую структуру:

|  |
| --- |
| <DL>  <DT>Пункт 1  <DD>Определение пункта 1  <DD>Другое определение пункта1  <DT>Пункт 2  <DD>Определение пункта 2  </DL> |

Определяемый термин записывается на одной строке, а его определение — на следующей, с небольшим отступом вправо. Тег <DL> позволяет создавать отдельные абзацы с отступом без нумерации или маркеров. Отступ делается от левого края. Если на странице несколько тегов <DL>, то текст постепенно сдвигается все больше вправо. В конце определения поместите закрывающий тег </DL>. Помните, что тег <DL> сдвигает только левую границу абзаца.

Термин «определение» носит условный характер. Абзацы, размещенные в списке, могут быть определениями, дополнениями, разъяснениями пунктов. По сути, пункт представляет собой заголовок, а определение – произвольный текст под заголовком.

На экране браузера список определений будет выглядеть так:



empty

Вложенные списки

Еще один способ создания сложных списков – использование принципа вложения. Каждый элемент, определяющий пункт списка, может содержать еще один список. Вложенный список располагается с небольшим отступом вправо относительно списка верхнего уровня. Каждый из списков может быть создан и отлажен отдельно, а затем все составляющие могут быть объединены в один большой список.

Пример:

|  |  |
| --- | --- |
|  |  |

В примере нумерованные списки размещены внутри пунктов ненумерованного списка.

**Контрольные вопросы:**

1. Структура HTML-документа и элементы разметки заголовка документа
2. Основные контейнеры заголовка HTML-документа
3. Контейнеры тела документа HTML-документа
4. Использование графики в HTML
5. Таблицы в HTML
6. HTML-формы
7. Фреймы в HTML
8. Базовые типы данных в HTML
9. Общая структура документа HTML

## Практическая работа №24

Использование гиперссылок в html-документе.

Цель: нучиться использовать гиперссылки в html-документе.

Студент должен:

иметь практический опыт:

осуществления навигации по ресурсам, поиска, ввода и передачи данных с помощью технологий и сервисов Интернета (ПО2);

создания и обработки объектов мультимедиа (ПО3);

уметь:

создавать и редактировать объекты мультимедиа (У2);

знать:

назначение, разновидности и функциональные возможности программ для создания объектов мультимедиа (З2).

Гипертекст

Ссылка (гиперссылка) – это возможность «связать» любой находящийся на данной странице фрагмент текста (слово, фразу или целый абзац) либо рисунок с другой Интернет-страницей или файлом данных (например, графическим файлом GIF или JPEG), так что при щелчке мыши на таком тексте или рисунке пользователь автоматически переходит на указанную к ссылке страницу.

Гипертекстовые ссылки среди других элементов текста выделяются цветом и подчеркиванием;

курсор мыши на ссылке меняет свою форму и превращается в указующий перст;

для перехода по ссылке необходимо щелкнуть по ней мышью;

для возврата необходимо использовать навигационную кнопку браузера Назад (Back).

Если ссылки образуют длинную цепочку, то кнопки Назад (Back) и Вперед (Forward) можно использовать для движения по пройденному ссылочному пути в обе стороны. Они работают как традиционные операции «откатка» (отменить) и «накатка» (вернуть) в большинстве прикладных программ.

Ссылки могут уводить не только в разные документы, но и загружать на экран фрагменты одного и того же файла. В последнем случае навигационные средства браузера работают так же.

Переход внутри одного документа

Для задания гипертекстового перехода внутри документа используют две команды <A>. Первая команда с атрибутом href является источником перехода, вторая с атрибутом name – приемником.

|  |  |
| --- | --- |
| Общий вид программы | Комментарий |
| <A href=#метка>текст</A> | Задание перехода по метке |
| … | На экран выводится ссылка: текст |
| … |  |
| … |  |
| <A name=метка></A> | Метка. Сюда браузер приходит по ссылке |
| … | На экране ничего не отображается |

**…**

**<A href**=#метка**>**текст**</A>**

**…**

**<A name**=метка**></A>**

**<P>**

В лесу родилась елочка

переход

браузера

Для организации перехода внутри документа нужно:

Выбрать имя для метки:

имя должно быть уникальным в HTML-программе. Это означает, что других меток с таким именем в файле быть не должно (когда одинаковых меток несколько, браузер выполняет переход на самую первую метку);

имя может заключаться в кавычки;

имя может состоять из любых символов. В том случае, когда имя содержит пробелы или ключевые символы HTML, кавычки обязательны.

Запрограммировать переход по метке.

В месте перехода нужно написать команду <A href=#метка>текст</A>.

Здесь:

# - ключевой символ;

метка – выбранное имя для метки;

текст – запись, которая будет выглядеть на экране браузера как ссылка.

3. Установить тег с меткой в нужном месте HTML-программы.

Перед фрагментом HTML-программы, на которой должен выполниться переход, записать команду-метку <A name=метка></A>.

При этом:

имя метки должно быть то же самое, что и в команде, задающей переход;

парный тег </A> может быть опущен;

на экране тег-метка <A name=метка> никаких изображений не строит.

В результате, при щелчке мыши по тексту, расположенному внутри контейнера <A href=имя\_метки></A>, браузер ищет в программе метку, имя которой указанно в атрибуте href, и показывает на экране то место документа, в котором эта метка расположена.

Пример:

|  |  |
| --- | --- |
| <HTML>  <HEAD>  <TITLE>Переход по меткам внутри одного документа</TITLE>  <BODY leftmargin=300>  <P>Переход в <A href=#finish> конец</A> документа.  <HR>  <H1>Заголовок 1</H1>  <H2>Заголовок 2</H2>  <H3>Заголовок 3</H3>  <H4>Заголовок 4</H4>  <H5>Заголовок 5</H5>  <H6>Заголовок 6</H6>  <P> Существует шесть уровней заголовков, которые обозначаются H1...H6. Заголовок уровня 1 самый крупный, а уровень 6 обеспечивает самый маленький заголовок. Шаблон дает представление об относительных размерах букв в заголовках. Для заголовков можно использовать атрибут, задающий выравнивание влево, по центру или вправо. <HR>  <A name=finish>  </BODY>  <HTML> |  |

Переход к другому документу (файлу)

Для того чтобы браузер загрузил в свое окно новый HTML-документ, нужно записать в программе ссылку при помощи команды <A> с атрибутом href=имя\_файла.

|  |  |
| --- | --- |
| Общий вид программы | Комментарий |
| … | Выполнить файл имя\_файла. На экран выводится ссылка: текст. При щелчке мыши на ссылке браузер строит на экране документ по программе заданной в файле имя\_файла. |
| <A href=имя\_файла>текст</A> |
| … |

*doc1.htm*

**…**

**<A href**=doc2.htm**>**текст**</A>**

**…**

*doc2. htm*

переход

браузера

Пример:

|  |  |
| --- | --- |
| Doc1.htm  <HTML>  <HEAD>  <TITLE>Файл с вопросом</TITLE>  </HEAD>  <BODY>  <H1 align=center>Вопрос</H1>  <HR>  <P> Назовите атрибут команды <A>, который используется для задания гиперссылки.  <P> <A href=./doc2.htm>Ответ</A>  </BODY>  </HTML> | Doc2.htm  <HTML>  <HEAD>  <TITLE>Файл с ответом</TITLE>  </HEAD>  <BODY>  <H1 align=center>Ответ</H1>  <HR>  <P> Для задания гиперссылки используется атрибут href команды &ltA&gt.  <P> <A href=./doc1.htm>Вернуться к вопросу</A>  </BODY>  </HTML> |
|  |  |

Переход к метке другого документа

Можно объединить эти методы и передать управление из одного документа к метке внутри другого.

|  |  |
| --- | --- |
| Общий вид программы | Комментарий |
| Первый HTML-файл  …  <A href=имя\_файла#метка>текст</A>  … | Приступить к показу фрагмента с меткой метка в файле имя\_файла. На экран выводится ссылка: текст. |
| Второй HTML-файл  …  <A name=метка>  … | При щелчке пользователя на ссылке браузер строит на экране документ по программе, заданной в файле имя\_файла, начиная с фрагмента с меткой метка. |

*doc1.htm*

**…**

**<A href**=doc2.htm#3**>**текст**</A>**

**…**

*doc2. htm*

**…**

**<A name**=3>

**<P>**

переход

браузера

Пример:

|  |  |
| --- | --- |
| Doc3.htm  <HTML>  <HEAD>  <TITLE>Содержание</TITLE>  </HEAD>  <BODY leftmargin=300>  <H1 align=center> Афоризмы </H1>  <HR width-=200>  <UL>  <LI> <A href=./doc4.htm#1> Станислав Ежи Лец</A>  <LI> <A href=./doc4.htm#2> Сальвадор Дали</A>  <LI> <A href=./doc4.htm#3> Эрнест Хемингуэй</A>  </BODY>  </HTML> | Doc4.htm  <HTML>  <HEAD>  <TITLE>Афоризмы</TITLE>  </HEAD>  <BODY leftmargin=200>  <A name=1>  <H1 align=center> Станислав Ежи Лец </H1>  <P align=center> Будь альтруистом – уважай эгоизм других.  <HR>  <P align=center> Могло быть хуже. Твой враг мог быть твоим другом.  <HR>  <P align=center>Оптимизм и пессимизм расходятся только в точной дате конца света.  <A name=2  <H1 align=center> Сальвадор Дали </H1>  <P align=center>Если ты способен на жестокость, значит, способен работать.  <HR>  <P align=center> Когда гении перемрут, я останусь в гордом одиночестве.  <HR>  <P align=center> Я не тщеславен, напротив, скромен. И если говорю, что я лучше других, то только потому, что они совсем плохи.  <HR>  <P align=center> Разница между мной и сумасшедшим в том, что я не сумасшедший.  <A name=3>  <H1 align=center> Эрнест Хемингуэй</H1>  <P align=center> Не судите о человеке по его друзьям. Помните, что друзья у Иуды были безукоризненны.  <HR>  <P align=center>Писатель, если он настоящий писатель, каждый день должен прикасаться к вечности или ощущать, что она проходит мимо него.  </BODY>  </HTML> |
|  |  |

Имена файлов и ссылки на них

Имена файлов с HTML-программами имеют расширение html или. Без необходимости обозначение html лучше не использовать, так как не все операционные системы понимают 4 символа в расширении имени файла.

Имена для файлов всегда стараются придумывать осмысленные. Когда файлов много, порядок в именах имеет особое значение.

Если, например, ваш много файловый гипертекст имеет иерархическую структуру, старайтесь в обозначениях использовать иерархию. Так, основной файл с оглавлением можно назвать 00.htm (или index.htm). Файл с первой главой – 01.htm. Когда главы имеют файлы-параграфы их имена получаются из имени главы добавлением двухзначного номера параграфа (пример: файл 0205.htm описывает построение пятого параграфа второй главы).

Браузер ищет файл в том же каталоге, что и файл, содержащий ссылку. Когда необходимо сослаться на файл из другого каталога, можно:

указать в ссылке полный путь к файлу, начиная с имени дисковода;

указать в ссылке путь по отношению к файлу со ссылкой.

Второй способ гораздо лучше, так как позволяет переносить ваш гипертекст в другие каталоги и на другие компьютеры без дополнительных настроек.

Пример:

С

BOOK

00.htm

01

02

02.htm

0201.htm

0202.htm

03

04

Сослаться из файла 00.htm на файл 02.htm:

C:/BOOK/02/02.htm (аблосютная адресация).

./02/02/htm (относительная адресация).

Первый способ приводит к непереносимости вашего продукта. Стоит перенести каталог, и ссылка перестает работать.

Вторая запись предлагает браузеру ориентироваться на текущий каталог – об этом говорит указание «./».

Переход на другие ресурсы

При помощи команды <A> можно программировать переходы на любые ресурсы, доступные в сети.

Гиперсвязь в HTML-документе представляет собой URL-адрес: сначала записывают условное обозначение протокола, по которому доступен файл в Интернете, потом, через символы «://» - адрес сервера, на котором расположен файл, затем через символ «/» - место расположения файла на сервере:

название протокола://адрес сервера/расположение файла

Отправка сообщений по электронной почте

Протокол mailto служит для передачи электронных писем. С его помощью можно отправить сообщение по электронной почте прямо из браузера, не запуская специально для этого почтовую программу. Если браузер не умеет сам работать с почтой, он загрузит почтовую программу.

За название протокола mailto идет простое двоеточие, а не двоеточие с двумя косыми чертами как обычно:

<A href=mailto:адрес>Письмо</A>

**Контрольные вопросы:**

1. Структура HTML-документа и элементы разметки заголовка документа
2. Основные контейнеры заголовка HTML-документа
3. Контейнеры тела документа HTML-документа
4. Использование графики в HTML
5. Таблицы в HTML
6. HTML-формы
7. Фреймы в HTML
8. Базовые типы данных в HTML
9. Общая структура документа HTML

## Практическая работа №25

Размещение графики на веб-странице

Цель: научиться размещать графику на веб-странице

Студент должен:

иметь практический опыт:

осуществления навигации по ресурсам, поиска, ввода и передачи данных с помощью технологий и сервисов Интернета (ПО2);

создания и обработки объектов мультимедиа (ПО3);

уметь:

создавать и редактировать объекты мультимедиа (У2);

знать:

назначение, разновидности и функциональные возможности программ для создания объектов мультимедиа (З2).

Графика

В Интернете наиболее популярны два графических формата: GIF и JPEG.

Обычно используют:

JPEG – для фотографий и очень сложных по цветовой гамме рисунков с плавными цветовыми переходами (в расширении записывается как JPG).

GIF – для логотипов, надписей, заголовков, рисунков, имеющих четкие цветовые граница.

Формат GIF

Картинки в этом формате поддерживают 256 цветов. В большинстве случаев этого вполне достаточно для получения качественных иллюстраций, но до фотографического качества, конечно, далеко.

GIF-формат имеет три дополнительные возможности.

Мультипликация. Используя анимационный GIF-редактор, легко собрать простой «мультик»: нарисовать отдельные кадры, настроить время задержки в последовательности показа, «зациклить кино» или показывать кадры только один раз. Когда анимация готова, она записывается на диск как обычный GIF-файл. Но в этом файле не одна, а целая коллекция GIF-картинок и разная настроечная информация.

В HTML нет различия между заданием вывода на экран простого GIF или анимированного. О том, что картинка «живая» - браузер узнает в момент загрузки GIF-файла и поступает соответственно. Обычные картинки он просто размещает на экране и забывает про них. Анимированные ему приходится опекать постоянно, меняя кадры на экране в соответствии с программой, заложенной в самом GIF-файле.

Прозрачная графика. GIF-формат позволяет один или несколько цветов в картинке объявить прозрачными. Это помогает избавиться от строго прямоугольных иллюстраций и вписывать рисунок в документ более привлекательным образом:

|  |  |
| --- | --- |
| Обычный GIF | Прозрачный GIF |

Прозрачный цвет задается в редакторе при записи GIF-файла на диск.

Чересстрочная развертка. Применяется для больших GIF. Иллюстрация разделяется на четыре части, размером с оригинальную картинку. Первая часть содержит 1, 5, 9-ю и т.д. строки исходного изображения. Вторая – 2, 6, 10-ю и т.д. Третья – 3, 7, 11-ю и т.д. Четвертая – 4, 8, 12-ю и т.д.

Браузер строит на экране сначала первую часть картинки, потом которую, затем третью и четвертую. Получается эффект постепенного проявления изображения. На медленных линиях Интернета этот метод позволяет пользователю еще до загрузки всех частей картинки получить представление об изображении - ведь части строятся браузером по мере их поступления в компьютер. Если картинка загружается не то сети, то различия построения обычного GIF и чересстрочного практически незаметны.

Формат JPEG

Этот формат был разработан специально для передачи фотографий. Он поддерживает миллионы цветов и позволяет получать изображения очень высокого качества.

Конечно, высокое качество отражается на размерах файла. Но формат JPEG имеет одну особенность: в JPG-редакторе при записи на диск можно указать ту степень качества, которая нужна и, тем самым, балансируя на планке «качество-размер файла», достичь нужного компромисса. В самом деле, если из миллиона цветов отсканированной фотографии оставить только 10 000, заменив остальные цвета на близкие, - на глаз такую подмену обнаружить будет практически невозможно, а размер файла существенно сократиться.

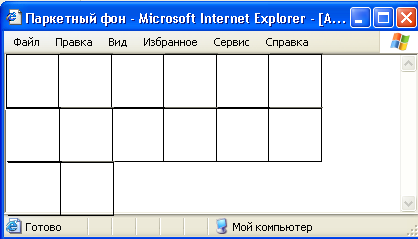
JPEG-формат поддерживает режим, похожий на чересстрочную развертку GIF. Но в отличие от GIF, деление на части происходит не по строкам, а по алгоритму выделения из картинки полноформатных частей. При этом каждая следующая часть по качеству выше предыдущей. JPEG с таким свойством называется прогрессивным.

Обычно графические файлы не смешиваются с HTML-текстами, а помещаются в отдельную папку pic – подкаталог каталога с программами (HTML-файлами).

Узнать формат и размеры графического файла можно кликнув по нему правой кнопкой мыши и выбрав в контекстном меню команду Свойства.

Фоновая картинка

Вместо одноцветной заливки можно использовать картинку. Эта картинка будет, повторяясь укладываться на экранное поле наподобие паркетных плиточек.



Картинка, которую браузер использует как плитку для укладывания паркета.

Чтобы паркет смотрелся красиво, необходима визуальная стыковка «плит» между собой. Это надо учитывать при создании фоновой иллюстрации. Очень важно, чтобы фон оставался фоном и не отвлекал внимание пользователя от главных элементов документа, не мешал читать текст.

Фоновая картинка задается в команде <BODY> атрибутом background:

<BODY background=файл\_с\_картинкой>.

Необходимо также всегда записывать атрибут bgcolor, даже если задан паркетный фон, так как браузер может не поддерживать графики или пользователь отключить показ графики. Кроме того, сначала по сети приходят тексты, и браузер выводит их на экран. Так как фоновая картинка еще не подошла, он эти картинки положит на фон, нарисованный по bgcolor, а если его нет, то – на непредсказуемый фон «по умолчанию». Затем приходит фоновая картинка, и браузер перерисовывает экран. Пользователь получает «удар» по глазам, если цвет фона «по умолчанию» сильно отличается от фоновой картинки.

Можно использовать фоновую картинку высота которой составляет всего один пиксел, а длина – 1600 пикселов (пример: fon3.jpg) или, наоборот, длина – 1 пиксел, а высота 1200 пикселов (пример: fon.gif).

|  |  |
| --- | --- |
|  |  |
|  |  |

При прокрутке длинной страницы фоновая фактура перемещается вместе с текстом и прочими элементами ее содержания. Чтобы зафиксировать фон так, чтобы при прокручивании страницы ее содержание «скользило» по нему используют атрибут bgproperties со значением fixed:

<BODY bgproperties= fixed>

Вставка иллюстраций

Картинку можно помещать в документ почти так же, как текстовый символ. Для вставки этого необычного значка нужно воспользоваться командой <IMG>. Команда не имеет парного закрывающего тега, но имеет много атрибутов.

Атрибут src=имя\_файла

Самым главным атрибутом команды является атрибут src, при помощи которого можно задать имя файла с картинкой. Команда вывода графики будет иметь вид <IMG src=./pic/img.gif>.

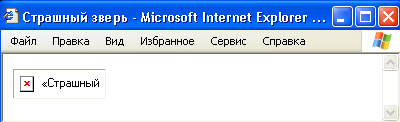
Атрибут alt=“текст\_надписи”

Если браузер не находит картинки в указанном месте на диске, он вместо нее рисует на экране маленький прямоугольник и вписывает в него надпись, которая задана атрибутом alt.

Например, выполняя команду:

<IMG src=./pic/monstr.jpg alt=«Страшный зверь»>

браузер, не найдя картинку выведет на экран такой рисунок:



Атрибут alt рекомендуется использовать всегда. Даже если картинка найдется браузером, задание надписи не будет лишним: стоит пользователю остановить курсор на картинке, как надпись появится в маленьком окошке и сообщит дополнительную информацию.

Атрибуты width=n и height=m

Эти атрибуты задают ширину (width) и высоту (height) (в пикселях) прямоугольника, в который выводится картинка. Если атрибуты не заданы, картинка рисуется в естественных размерах. Когда размеры прямоугольника не совпадают с размерами картинки, браузер производит масштабирование картинки, подгоняя ее под указанные величины. Рекомендуется всегда записывать эти атрибуты и указывать в них естественные размеры картинки. Если размеры указываются неверно – картинка выводится браузером с искажением.

Если указание размеров вовсе проходит – на экране происходит маленькое «землетрясение» в момент прихода картинки из сети. Ведь не зная размеров изображения, браузер оставляет под него на экране стандартный прямоугольник, продолжая строить остальные элементы. Когда картинка загружена, браузеру приходится полностью перерисовывать экран.

Атрибут border=n

Атрибут позволяет задать рамку вокруг иллюстрации толщиной в n пикселов. При отсутствии атрибута или при значении n=0 рамка не рисуется. Однако, чтобы не вводить пользователей в заблуждение в изображениях, которые представляют собой гиперссылки не стоит задавать border=0, поскольку рисунки, применяемые в качестве гиперссылок, обычно выделяются цветной рамкой.

Атрибуты hspace=n vspace=n

Атрибуты hspace и vspace определяют свободное пространство слева и справа (hspace), над и под (vspace) картинкой.

Атрибут align

Определяет положение изображения относительно окружающего его текста.

|  |  |
| --- | --- |
|  | align=top - выравнивает верх изображения по верхнему краю самого высокого элемента в строке окружающего текста. |
|  | align=middle - выравнивает центр изображения по базовой линии строки окружающего текста. |
|  | align=bottom выравнивает нижний край изображения по базовой линии строки окружающего текста. |
|  | align=left определяет огибаемое текстом изображение. Изображение располагается вдоль левой границы документа, а последующие строки текста огибают его справа. |
|  | align=right определяет огибаемое текстом изображение. Изображение располагается вдоль правой границы документа, а последующие строки текста огибают его слева. |

Кроме основных значений атрибута align="ключевое слово" существует еще ряд аргументов, которые расширяют возможности взаимного размещения графики и текста.

align=textop выравнивает верх изображения по верхнему краю самого высокого текстового символа в строке окружающего текста. Действие этого аргумента в большинстве случаев, но не всегда, подобно действию аргумента align=top.

align=absmiddle выравнивает центр изображения по центру строки окружающего текста. Действие этого аргумента в большинстве случаев, но не всегда, подобно действию аргумента align=middle.

align=baseline выравнивает нижний край изображения по базисной линии строки окружающего текста, то есть производит такое же действие, как и align=bottom.

align=absbottom выравнивает нижний край изображения по нижнему краю строки окружающего текста.

Для прекращения обтекания текста слева и справа от картинки используются теги <BR clean=left> и <BR clean=right>

Картинка как ссылка

Очень легко заставить картинку работать как гиперссылка. достаточно вложить команду <IMG> внутрь команды <A>:

<A href=переход…><IMG src=файл…></A>

Курсор при попадании на картинку-ссылку меняет свою форму. Щелчок мыши по картинке заставляет браузер выполнять переход. Картинка-ссылка не подчеркивается, как текстовая ссылка, и не выделяется цветом, а обрамляется в рамочку.

Карты

Элемент <IMG> позволяет использовать изображения, отдельные части которых связаны со ссылками и позволяют выполнять переходы. Такие изображения называются картами (map).

В случае, когда карта обрабатывается в теге <IMG> используется атрибут usemap со значением-ссылкой на описание «чувствительных» областей картинки.

Пример:

<IMG src=./pic/map1.jpg width=385 heigth=399 usemap=#map1 alt=“Мы и теги” border=0>

Значения атрибута usemap записывается аналогично ссылке на текущую страницу, т.е представляет собой конструкцию:

#идентификатор

Атрибут usemap определяет имя карты:

usemap=#Имя

Это имя ставится в соответствии со значением соответствующих атрибутов элементов AREA и MAP, которые определяют конфигурацию карты.

Задание атрибутов usemap придает элементу <IMG> свойства, характерные для элемента <A>, то есть возможность осуществления переходов. Кроме того, мы сталкиваемся со случаем, когда необходимо обязательное совместное использование сразу трех элементов: <AREA>, <IMG> и <MAP>.

Описание «чувствительных» областей размещается в любом месте внутри <BODY>.

В открывающем теге <MAP> указывается атрибут name со значением, равным идентификатору ссылки, размещенной в соответствующем теге <IMG>. Блок <MAP> содержит теги <AREA> с описанием «чувствительных» областей. Для каждой области карты должен быть свой элемент <AREA>. Он не имеет конечного тега.

Атрибуты тега <AREA>

Атрибут shape

Этот атрибут задает вид «чувствительной» области. Об может принимать значения:

|  |
| --- |
| shape=rect – прямоугольник  shape=circle – окружность  shape=poly – многоугольник  shape=default – оставшаяся область |

Атрибут coords

Этот атрибут задает координаты области в пикселях. За начало отсчета принимается верхний левый угол картинки. Координаты задаются через запятые.

Для круга необходимо задать координаты центра и радиус (r), выраженные в пикселях. Координаты центра отсчитываются от левого края (х) и верхнего края (у).

у

х

r

Шаблон для задания круговой области:

shape=circle coords=x, y, r.

Для определения области произвольной конфигурации задают координаты (х, у) каждого из углов многоугольника в порядке их построения. Последняя точка автоматически замыкается на первую:

х1, у1

х2, у2

х3, у3

х4, у4

х5, у5

Шаблон для области произвольной конфигурации:

shape=poly coords=x1,y1, x2,y2, x3,y3, x4,y4, x5,y5

При определении прямоугольной области задаются координаты верхнего левого и правого нижнего углов прямоугольника:

shape=rect coords=x1,y1, x2,y2

Для shape=default атрибут coords не используется.

Атрибут href

Этот атрибут задает ссылку для гипертекстового перехода.

Атрибут nohref

Этот атрибут не имеет значений и указывает область, которая не является гиперссылкой. Атрибут полезен, если необходимо в «чувствительной» области вырезать «нечувствительный» кусок. Его используют также и в сочетании с атрибутом shape=default, закрывая (на всякий случай) «чувствительность» оставшихся после разметки участков картинки.

Атрибут alt

Атрибут задает альтернативный текст для области. Работает в Netscape Navigator, но не работает в Internet Explorer.

Атрибут title

Задает заголовок элементу. Работает аналогично альтернативному тексту в Internet Explorer, но не работает в Netscape Navigator.

Для задания альтернативного текста используйте одновременно alt и title. Это обеспечит показ альтернативного текста в обоих браузерах.

Описание областей карты и соответствующий элемент IMG могут размещаться в разных частях страницы. Переходы, выполняемые с помощью карты, могут происходить как внутри страницы, так и к удаленному ресурсу.

Пример страницы с картами. Карта, имеющая имя karta1 содержит область в виде круга и область произвольной формы. Карта karta2 содержит область прямоугольной формы:

|  |  |
| --- | --- |
| <HTML>  <HEAD>  <TITLE>Карты</TITLE>  </HEAD>  <BODY>  <A name=verh></A>  <H2>Карта 1</h2>  <MAP name=karta1>  <AREA alt=Круг shape=circle coords=119,114,83 href=http://htmldis.narod.ru>  <AREA alt=Многоугольник shape=poly coords=74,242,180,250,249,239,242,278,31,276 href=./doc1.htm>  </MAP>  <IMG src=./pic/map1.gif width=300 height=300 usemap=#karta1 alt=Карта 1>  </BODY>  </HTML> |  |

С изображениями карт удобно работать в стандартном для Windows редакторе Paint. В строке состояния редактора указываются координаты курсора относительно верхнего левого угла рисунка. Значения координат соответствуют требуемым для атрибута coords величинам и идут в том же порядке (х, у).

Одним из способов применения карты является создание меню страницы. Это обычно горизонтальная полоса, состоящая из цветных прямоугольников с текстом команд. Щелчок на каждом из прямоугольников переводит пользователя на другую станицу или в другую точку текущей страницы. Преимущество графического меню заключается в независимости от используемой кодировки символов: буквы всегда будут видны. Можно представить меню в виде одного рисунка-карты и разбить его на несколько многоугольных областей.

Другой популярной областью использования карт является домашняя страница. В этом случае необходимые команды и названия рубрик могут располагаться в произвольном порядке: так, как надо дизайнеру.

Пример:

***ПОЧТА***

***ТОВАРЫ***

***АДРЕС***

***НАША ФИРМА***

**Контрольные вопросы:**

1. Структура HTML-документа и элементы разметки заголовка документа
2. Основные контейнеры заголовка HTML-документа
3. Контейнеры тела документа HTML-документа
4. Использование графики в HTML
5. Таблицы в HTML
6. HTML-формы
7. Фреймы в HTML
8. Базовые типы данных в HTML
9. Общая структура документа HTML

## Практическая работа №26

Создание таблиц в html-документе.

Цель: научиться создавать таблицы в html-документе

Студент должен:

иметь практический опыт:

осуществления навигации по ресурсам, поиска, ввода и передачи данных с помощью технологий и сервисов Интернета (ПО2);

создания и обработки объектов мультимедиа (ПО3);

уметь:

создавать и редактировать объекты мультимедиа (У2);

знать:

назначение, разновидности и функциональные возможности программ для создания объектов мультимедиа (З2).

Таблица

Таблицы являются очень удобным средством форматирования данных на Web-странице. Основное удобство заключается в том, что браузер берет на себя заботу о прорисовке рамки таблицы. Размер рамки может быть автоматически согласован с размером окна просмотра в браузере и, разумеется, с размером находящейся в ячейках таблицы строк текста и рисунков. Кроме того, таблицы позволяют решать чисто дизайнерские задачи: выравнивать части страницы друг относительно друга, размещать рядом рисунки и текст, управлять цветным оформлением и т.д. При создании таблиц используется принцип вложения: внутри основного элемента таблицы (<TABLE>) создается ряд элементов, определяющих строки (<TR>), а внутри этих элементов размещаются элементы для описания каждой ячейки в строке (<TD>, <TН>):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| <TABLE>   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | <TR> | <TD> | </TD> | <TD> | </TD> | </TR> | | <TR> | <TD> | </TD> | <TD> | </TD> | </TR> | | <TR> | <TD> | </TD> | <TD> | </TD> | </TR> |   </TABLE> |

Чтобы разобраться в структуре существующей таблицы или создать новую, необходимо помнить, что последовательность элементов описывает таблицу сверху вниз и справа налево. Например, после элемента <TABLE> указан элемент <TR>, это означает, что начитается описание новой строки таблицы. Все, что расположено за этим элементом, будет размещено в одной строке (справа налево). Это может быть последовательность элементов <TD> (ячеек), другая таблица и т.д. После того, как встретиться новый элемент <TR>, начинается описание следующей строки, и т.д. до конца таблицы (тега </TABLE>):

|  |  |
| --- | --- |
| <HTML>  <HEAD>  <TITLE>Таблица </TITLE>  </HEAD>  <BODY>  <TABLE border=1>  <TR>  <TD>(1,1)</TD>  <TD>(1,2)</TD>  </TR>  <TR>  <TD>(2,1)</TD>  <TD>(2,2)</TD>  </TR>  <TR>  <TD>(3,1)</TD>  <TD>(3,2)</TD>  </TR>  </TABLE>  </BODY>  </HTML> |  |

<TABLE>…</TABLE>

Внешний элемент таблицы Он позволяет задавать общие свойства таблицы и отделяет структуру таблицы от остальной части Web-страницы. Большинство его атрибутов могут использоваться и в других элементах таблицы.

Атрибуты <TABLE>…</TABLE>

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение | Описание |
| align | Left, right, center | Выравнивание по горизонтали |
| width | Число или процент | Ширина таблицы |
| cellpadding | число | Расстояние между содержимым ячейки и рамкой |
| cellspacing | число | Расстояние между ячейками таблицы |
| bgcolor | цвет | Цвет фона таблицы |
| background | цвет | Фоновая картинка |
| border | число | Толщина линий рамки |
| bordercolor | цвет | Цвет линий рамки |
| bordercolordark | цвет | Цвет рамки (снизу и справа) |
| bordercolorlight | цвет | Цвет рамки (сверху и слева) |
| frame | void, above, below, hsides,  vsides, lhs, rhs, border, box | Вид рамки таблицы |
| rules | none, rows, cols, all | Вид сетки внутри таблицы |

Атрибут align

Этот атрибут позволяет определять положение таблицы по отношению к соседним элементам документа, т.е. имеет для таблиц тот же смысл, что и для картинок.

|  |
| --- |
| align=left – выравнивание слева  align=center – выравнивание по центру  align=right – выравнивание справа |

Атрибут width

Задает ширину таблицы. Если атрибут не задан, браузер рисует таблицу минимальных размеров вокруг данных, которые она содержит. Запись width=число заставляет браузер рисовать таблицу шириной в указанное число пикселей. Если реально для таблицы требуется больше места, значение width игнорируется. Если заданный размер уводит таблицу за правую границу окна, браузер добавляет к окну горизонтальную линейку протяжки.

Старайтесь формировать окно документа таким образом, чтобы горизонтальная протяжка не появлялась. Во-первых, линейка протяжки уменьшает полезный размер окна, во-вторых, ухудшает восприятие документа и раздражает пользователя.

Можно также задавать ширину таблицы в процентном отношении к ширине окна браузера: процент (например: width=90%).

Атрибуты cellpadding и cellspacing

Первый атрибут задает расстояние между рамкой таблицы и содержимым ячейки, т.е. размер пустого пространства, окружающего данные в ячейках:

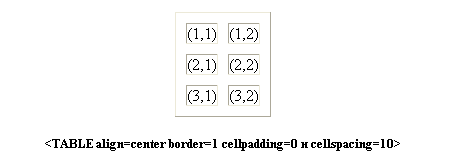
|  |
| --- |
| cellpadding=число пикселей  cellpadding=процент |

Задание этого атрибута делает ячейки больше. Между рамкой и данными всегда сохраняется определенное расстояние. В некоторых случаях это позволяет улучшить восприятие таблицы, сделать текст в ячейках легко читаемым.

Атрибут cellspacing задает расстояние между рамками соседних ячеек (ширину фронтальной грани рамки). Если значение этого атрибута равно нулю, рамка получается тонкой, заостренной.

Когда атрибуты не заданы, браузер устанавливает их по режиму умолчания, обычно: cellpadding=1 и cellspacing=2.

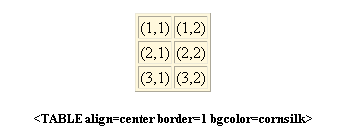
Пример:



Атрибуты bgcolor и background

Если атрибуты не заданы, элементы выводятся прямо на фон документа и таблица выглядит прозрачной. Атрибут bgcolor задает цвет фона таблицы, а атрибут background – паркетную укладку картинкой.

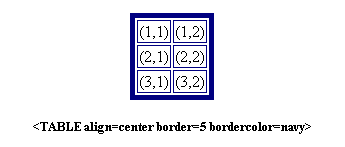
Пример:



Атрибуты border и bordercolor

Первый атрибут толщину рамки вокруг всей таблицы, второй – ее цвет. Если border=0, то рамка, а заодно и прямоугольники клеток, не рисуются.

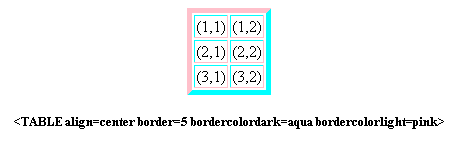
Пример:



Атрибуты bordercolordark и bordercolorlight

Браузер Netscape Navigator эти атрибуты не поддерживает. В Microsoft Internet Explorer они позволяют при необходимости нарисовать на экране объемную рамку. Первый задает цвет ее нижней и правой стороны, второй – левой и верхней. Если необходимо нарисовать рамку плоской (одноцветной), то эти атрибуты не задаются.

Пример:



Атрибут frame

Используется только для элемента <TABLE> и позволяет задать вид рамки таблицы:

frame=параметр

Существуют следующие стандартные параметры:

|  |  |  |  |
| --- | --- | --- | --- |
| frame=void – рамка отсутствует;  frame=above – верхняя сторона рамки;  frame=below – нижняя сторона рамки;  frame=hsides – части рамки сверху и снизу;  frame=vsides – части рамки слева и справа;  frame=lhs – левая часть рамки;  frame=rhs – правая часть рамки;  frame=border – рамка показана полностью;  frame=box – значение повторяет frame=border. | | | |
|  |  |  | |  |
|  |  | |  |  |

Атрибут rules

Определяет вид сетки таблицы внутри, то есть между ячейками:

|  |
| --- |
| rules=none – сетка отсутствует;  rules=rows – горизонтальные линии между строками;  rules=cols – вертикальные линии между колонками;  rules=all – обычная сетка. |

Пример:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Атрибуты <TR>…</TR>

Используя атрибуты команды <TR>, можно установить те или иные свойства одновременно для всех ячеек строки.

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение | Описание |
| align | left, center, right | Выравнивание по горизонтали |
| valigh | top (верхний край), middle (центр), bottom (нижний),  baseline ( по самой верхней текстовой строку) | Выравнивание по вертикали |
| bgcolor | цвет | Цвет фона |
| bordercolor | цвет | Цвет линий рамки |
| bordercolordark | цвет | Цвет рамки (снизу и справа) |
| bordercolorlight | цвет | Цвет рамки (сверху и слева) |

Атрибут align

Горизонтальное выравнивание. Если атрибут не задан, содержимое ячеек выравнивается по левому краю границы ячейки.

Атрибут valign

Вертикальное выравнивание содержимого ячеек. Принимает следующие значение:

|  |
| --- |
| valign=top – выравнивает по верхнему краю ячейки;  valign=bottom – выравнивает по нижнему краю ячейки (не всегда работает);  valign=center – выравнивает по центру;  valign=baseline – выравнивает по первой строке. |

Атрибут bgcolor

Если атрибут не задан, ячейки выводятся на фон, установленный для таблицы, или на фон документа, если фон таблицы не задан. Позволяет задать собственный фоновый цвет для всех ячеек в строке.

Атрибуты <TD> (<TH>)

Каждая табличная строка состоит из ячеек, которые последовательно описываются командами<TD> (обычная ячейка) и или <TH> (ячейка заголовок).

Содержимое обычной ячейки по умолчанию выравнивается по левому краю, ячейки-заголовка – по центру. Кроме того, текст в ячейке-заголовке выделяется полужирным шрифтом.

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение | Описание |
| align | Left, right, blееdleft (прижимает содержимое ячейки вплотную к левому краю) | Выравнивание по горизонтали |
| valign | Top, middle, bottom, baseline | Выравнивание по вертикали |
| width | Число или процент | Ширина ячейки |
| bgcolor | цвет | Цвет фона |
| background | цвет | Фоновая картинка |
| border | число | Толщина линий рамки |
| bordercolor | цвет | Цвет линий рамки |
| bordercolordark | цвет | Темный цвет рамки (снизу и справа) |
| bordercolorlight | цвет | Светлый цвет рамки (сверху и слева) |
| nowrap |  | Выключение автоматического разрыва строки |
| colspan | Определяет сколько табличных столбцов  (n) займет в пределах строки | Ширина большой ячейки в столбцах |
| rowspan | Определяет сколько строк (n) займет в пределах столбца | Высота большой ячейки в строках |

Атрибуты для тега <TD> (<TH>), совпадающие с атрибутами тега <TR>, имеют тот же смысл, но относятся не к строке, а к отдельной ячейке.

Атрибуты width и background имеют тот же смысл, что и для тега <TABLE>, то относятся к отдельной ячейке. Наличие у тега <TD> (<TH>) атрибута background означает, что каждая ячейка может иметь свою собственную фоновую картинку (даже если для всей таблицы в целом фоновая картинка уже указана).

Атрибут nowrap

Обычно абзац форматируется браузером в клетке таблицы так же, как и на полном окне, т.е. когда очередное слово не помещается в строке, оно вместе с остатком абзаца переносится на следующую. И так происходит до тех пор, пока весь текст не будет уложен браузером в оконные строки. Полученный абзац виден на экране по ширине целиком и не требует горизонтальной прокрутки, даже если он очень длинный.

Атрибут nowrap (у него нет присваиваемых значений) запрещает браузеру выполнять описанное выше автоматическое форматирование. На длинных абзацах и небольших размерах окна прием вынуждает браузер показывать горизонтальную линейку прокрутки.

Горизонтальная прокрутка не способствует улучшению комфорта для пользователя, поэтому применение атрибута nowrap ограничивается специальными (и довольно редкими) случаями.

Атрибут colspan

Атрибут colspan=n определяет, сколько табличных столбцов (n) займет ячейка в проделах строки:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| <TABLE>   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | <TR> | <TD colspan=4> </TD> | | | | </TR> | | <TR> | <TD> | </TD> | <TD> | </TD> | </TR> | | <TR> | <TD> | </TD> | <TD> | </TD> | </TR> |   </TABLE> |

Атрибут rowspan

Атрибут rowspan=n определяет, сколько табличных строк (n) займет ячейка в пределах столбца:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| <TABLE>   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | <TR> | <TD rowspan=2> </TD> | </TD> | <TD> | </TD> | </TR> | | <TR> | </TD> | <TD> | </TD> | </TR> | | <TR> | <TD> | </TD> | <TD> | </TD> | </TR> |   </TABLE> |

Применение пустых ячеек

Если ячейка не содержит данных, она не будет иметь границ. Если требуется, чтобы у ячейки были границы, но не было содержимого, необходимо поместить в нее что-то, что не будет видно при просмотре. Можно воспользоваться пустой строкой <ВR> или пробелом (&nbsp;). Можно даже задать пустые столбцы, определив их ширину в пикселах или относительных единицах и не введя в полученные ячейки никаких данных. Это средство может оказаться полезным при размещении на странице текста и графики.

Использование заголовков таблицы — тег <САРТIОN>

Тег <CAPTION> позволяет создавать заголовки таблицы. По умолчанию заголовки центрируются и размещаются либо над (<САРТION аlign=top>), либо под таблицей (<САРТION align=bottom>). Заголовок может состоять из любого текста и изображений. Текст будет разбит на строки, соответствующие ширине таблицы. Иногда тег <САРТION> используется для подписи под рисунком. Для этого достаточно описать таблицу без границ.

Пример таблицы:

|  |  |
| --- | --- |
| <HTML>  <HEAD>  <TITLE>Таблица</TITLE>  </HEAD>  <BODY>  <TABLE border="4" cellspacing=3  background=./pic/fon01.gif>  <CAPTION>Таблица с объединенными  ячейками и фоновым рисунком</CAPTION>  <TR>  <TH rowspan=2>&nbsp;</TH>  <TH colspan="2">Заголовок 1</TH>  </TR>  <TR>  <TH>Заголовок 1.1</TH>  <TH>Заголовок 1.2</TH>  </TR>  <TR>  <TH>Заголовок 2</TH>  <TD>Ячейка 1</TH>  <TD>Ячейка 2</TD>  </TR>  <TR>  <TH>Заголовок 3</TH>  <TD>Ячейка 3</TD>  <TD>Ячейка 4</TD>  </TR>  </TABLE>  </BODY>  </HTML> |  |

Таблицы, как дизайнерский инструмент

В обычных книгах текст не начинается с самого края – страничные поля выделяют информационные блоки и, следовательно, помогают глазам более комфортно воспринимать материал.

В гипертекстовом документе можно создать поля, если размещать информацию внутри таблицы, состоящей всего из одной клетки с невидимыми границами. При изменении значения атрибута cellpadding, меняется ширина полей.

Используя таблицы, можно выводить на экран информацию в несколько колонок.

Используя таблицы, можно выровнять информационные блоки на экранеи разместить их в соответствии с дизайн-проектом документа.

Бегущая строка

Этот видеоэффект обычно используется на Web-страницах для тех же целей, что и в реальной жизни: например, с его помощью можно вывести при дефиците места какую-то рекламную или новостную информацию. Однако, злоупотреблять этим не стоит – даже одна такая строка не на своем месте может изрядно подпортить впечатление от внешнего вида станицы, а от нескольких, да еще расположенных рядом друг с другом, начинает рябить в глазах.

Несмотря на то, что название «бегущая строка» подразумевает однострочный текст, этот тег позволяет «привести в движение» практически любой объект Web\_страницы, будь то одна или несколько строк и даже абзацев текста, иллюстрация, таблица и т.д., либо их произвольная комбинация. Достаточно заключить желаемый фрагмент между тегами <MARQUEE>….</MARQUEE>. Наибольшего эффекта можно добиться, если удачно подобрать атрибуты.

Атрибуты <MARQUEE>….</MARQUEE>

|  |  |  |
| --- | --- | --- |
| Атрибут | Значение | Описание |
| bgcolor |  | Цвет фона. |
| height | пиксели или проценты | Высота полосы. |
| width | пиксели или проценты | Ширина полосы. |
| align | top, middle, bottom | Вертикальное выравнивание текста. |
| direction | left, right | Направление движения строки. |
| behavior | scroll, slide, alternate. | Управление поведением строки. |
| hspace | пиксели | Сдвиг полосы по вертикали в право. |
| vspace | пиксели | Пустое пространство сверху и снизу полосы. |
| loop | число | Ограничение количества проходов по экрану. |
| scrollamount | число | Скорость движения полосы как число пикселей, которые проходит строка за каждый шаг. |
| scrolldelay | число | Скорость движения полосы за временной интервал. |

Атрибут bgcolor

Задает цвет полосы, на которой находится бегущая строка. Если цвет задан, то элемент будет выглядеть как цветная полоса.

Атрибуты height и width

Задают ширину и высоту строки. Задаются в пикселях или процентах. Если высота полосы задается в пикселях, то можно рекомендовать задавать ее в диапазоне 30…50. Высоту можно задавать и в процентах. Процент определяет долю от высоты видимой части гипертекста внутри окна браузера. Эта величина, разумеется, не является постоянной и зависит от размера окна. Если заданная высота недостаточна для данного содержимого, она будет автоматически увеличена до требуемой.

Атрибут align

Задает вертикальное выравнивание текста. Имеет значения:

|  |
| --- |
| align=top – выравнивание по верхнему краю;  align=middle – выравнивание по центру;  align=bottom – выравнивание по нижнему краю |

Однако этот атрибут поддерживают не все браузеры.

Атрибут loop

Этот атрибут задает количество повторений «прокручивания» заданного содержимого. В качестве его значения указывается целое число (1 и более) или значение infinite (непрерывное повторение).

Атрибут scrollamount

Определяет скорость «прокручивания». Указывает число пикселей, на которое строка смещается при каждом шаге прокрутки. Чем меньше значение атрибута, тем плавнее, без рывков, движется содержимое строки. Если число равно 1, то строка будет еле ползти по сравнению с режимом по умолчанию. Если число больше 10, то строка будет двигаться очень быстро.

Атрибут scrolldelay

Второй атрибут скорости. Задает задержку (в миллисекундах) между шагами. Чтобы ускорить движение строки без снижения плавности, необходим уменьшить значение атрибута (задержки).

Атрибут behavior

Атрибут управляет поведением строки и имеет следующие значения:

behavior=scroll – строка появляется из-за одного края, проплывает по экрану и дойдя до края экрана уходит из поля зрения. Если согласно значению loop действие должно повториться, то строка снова появляется с противоположной стороны, а по завершению всех циклов повторения строка остается пустой.

behavior=slide – строка появляется из-за края окна, достигает противоположного края и останавливается.

behavior=alternate – строка движется вправо или влево, «отражаясь от края окна и меняя направление движения до тех пор, пока не будет исчерпано заданное число повторений, тогда содержимое строки остается на экране неподвижным.

Атрибут direction

Указывает направление движения строки при behavior=scroll и behavior=slide, а также первоначальное направление строки при behavior=alternate. Значения атрибута:

|  |
| --- |
| direction=right – движение слева направо;  direction=left – движение справа налево. |

Атрибуты hspase и vspase

Бегущую строку можно, как и иллюстрации, разместить между блоками текста в отдельном абзаце или вставить непосредственно в строку текста, в этом случае полезными оказываются атрибуты hspase и vspase, которые задают отступы (в пикселях). Атрибут hspase задает отступ слева и справа, атрибут vspase - отступы сверху и снизу.

Пример бегущей строки в приложенном файле.

Задание для самостоятельной работы

Формы

Методами, доступными нам при изучении данного курса мы можем только создавать формы, но не можем организовать их отправку.

Форма, или формуляр, представляет собой отдельную Интернет-страницу или часть страницы, на которой размещены разнообразные типовые для Windows элементы интерактивного диалога: поля ввода текста, флажки и радиокнопки с поясняющими строками текста, раскрывающиеся списки и одна или несколько кнопок, обычно служащих для очистки формы (приведения в исходный вид в случае неправильного заполнения) и отправки введенных данных сформированного информационного запроса) на сервер.

Форма (формуляр) в HTML-документе определяется в виде блока <FORM>…</FORM>, внутри которого располагаются теги, задающие те или иные формы, а также играющие роль подсказок текстовые строки и обращения к графическим файлам. Внутри формы могут использоваться обычные HTML-теги, например, табличные.

В составе тега <FORM> записываются атрибуты, определяющие способ и характеристики обработки заключающейся в форме информации, в частности, способ пересылки ее на сервер и кодирования данных в процессе этой пересылки.

Команда <FORM> имеет следующие атрибуты:

|  |  |
| --- | --- |
| Атрибут | Описание |
| action | Задает адрес для отправки формы по сети. Вы можете послать форму на сервер, где она будет обработана программой (CGI-скриптом[[2]](#footnote-2)), а можете указать в качестве значения атрибута адрес электронной почты, и форма будет послана браузером по e-mail (пример: FORM action=mailto: адрес электронной почты). |
| enctype | Задает кодирование передаваемых по сети данных (пример: enctype= “text=pain” задаст передачу обычного текста). |
| method | Задает протокол для пересылки данных на сервер. По умолчанию используется протокол get, но лучше использовать протокол post как более универсальный. |

Каждому элементу формы соответствует тег <INPUT>, который позволяет создавать различные части формы. Этот элемент не имеет конечного тега, так как все параметры задаются при помощи атрибутов. Атрибут тега type указывает тип данного элемента.

Контейнер <TEXTAREA>…</TEXTAREA> задает многострочное текстовое окно.

Контейнер <SELECT>…</SELECT> используется для создания раскрывающегося списка.

Радиокнопки <INPUT type=radio>

Радиокнопки (селекторные кнопки, переключатель) предназначены для выбора одного варианта из нескольких альтернативных. Подобно кнопочному переключателю диапазона волн в радиоприемнике выбор одной кнопки автоматически снимает отметку с другой.

В группе радиокнопок активной может быть только одна. Отметка на новой кнопке автоматически снимает отметку на старой.

Радиоктопка задается командой:

<INPUT type=radio>

Дополнительные атрибуты:

Атрибут name

Значением является идентификатор (имя). Кнопки, входящие в одну взаимосвязанную группу, должны иметь одинаковое имя. Именно по этому критерию (одинаковое имя) браузер распознает переключатель и организует снятие отметки с кнопки, когда выбрана другая в этой же самой группе.

Атрибут checked

Логический атрибут, для него не указывается значение. Присутствие атрибута означает, что отметка на данной кнопке установлена по умолчанию.

Атрибут value

Значение этого атрибута будет служить указанием на выбор пользователя, когда форма отправляется по электронной почте.

Пример:

|  |  |
| --- | --- |
| <FORM metod=post action=mailto:html@list.ru>  Что будете ловить?<BR>  <INPUT type=radio name=fish value=“Карась”> Карась<BR>  <INPUT type=radio name=fish value=“Щука” checked> Щука<BR>  <INPUT type=radio name=fish value=“Сом”> Сом<BR>  </FORM> |  |

Что придет по e-mail

Обычно браузер не сам отсылает форму, а поручает работу почтовой программе, установленной по умолчанию в операционной системе компьютера (например: Outlook).

В указанном выше примере отметка стоит на позиции «Щука». При нажатии на кнопку, по указанному в теге <FORM> адресу уйдет письмо с темой «Форма отправлена из Microsoft Internet Explorer» (для браузера IE) следующего содержания:

fish=Щука

Таким образом, браузер составляет письмо из пар «имя=значение», которые выбираются по порядку из полей формы.

В качестве обратного адреса браузер указывает адрес пользователя компьютера, с которого отправляется форма.

Флажки <INPUT type=checkbox>

Флажки, или кнопки с независимой фиксацией, служат для многокомпонентного выбора.

Флажки не зависят один от другого. Их можно устанавливать и сбрасывать в любой комбинации.

Атрибуты:

name - имя флажка (при объединении нескольких флажков в группу всем им дается одно и то же имя);

value – значение флажка (записанная в кавычках произвольная комбинация букв и цифр, передаваемая на сервер, если данный флажок будет помечен пользователем);

checked - что отметка на данном флажке по умолчанию.

Пример:

|  |  |
| --- | --- |
| <FORM metod=post action=mailto:html@list.ru>  <INPUT type=checkbox name=html> HTML-конструирование<BR>  <INPUT type=checkbox name=js checked> JavaScript-конструирование<BR>  <INPUT type=checkbox name=dtml> DTML-конструирование<BR>  <INPUT type=checkbox name=al checked> Алгоритмика  </FORM> |  |

Значение атрибута name с добавкой «=on» передается в сеть, если флажок отмечен. Например, если выбраны книги «Алгоритмика» и «JavaScript-конструирование», в сеть будет передана запись:

js=on

al=on

Атрибут checked задает начальную отметку флажка.

Поле ввода имени пересылаемого файла <INPUT type=file>

Предназначено для пересылки на сервер вместе с заполненной формой указанного пользователем файла. Имя файла и путь к нему вводятся в однострочном текстовом поле или, после щелчка мышью по расположенной справа кнопке Обзор, требуемый файл выбирается в стандартном окне открытия файла, а его полное имя автоматически вводится в поле ввода.

Атрибуты:

name – имя данного диалогового элемента;

size – ширина текстового поля;

maxlenght – максимально допустимое количество символов имени файла (вместе с путем).

Пример:

|  |  |
| --- | --- |
| <FORM>  <INPUT type=file name=MyFile size=20  maxlenght=40>  </FORM> |  |

Поле ввода текста <INPUT type=text>

Представляет собой прямоугольное поле, предназначенное для ввода одной строки текста.

В эту строку вводится текстовая информация.

Атрибуты:

name – уникальное имя поля;

size – ширина поля (в символах);

maxlenght – максимально допустимое количество вводимых символов;

value – задает начальную запись в строке ввода (записывается в кавычках).

Пример:

|  |  |
| --- | --- |
| <FORM>  <INPUT type=text name=MyFam size=20  maxlenght=40 value=“Введите фамилию”>  </FORM> |  |

Значение атрибута name с текущим содержанием строки ввода передается в сеть:

MyFam=Введите фамилию

Атрибут size задает ширину строки ввода в символах (рекомендуется кодировать с запасом).

Поле ввода пароля <INPUT type=password>

Прямоугольное текстовое поле, отличающееся от поля для ввода текста только тем, что вводимый текст отображается в нем как соответствующее количество «звездочек» или «точек». Имеет те же атрибуты, что и при type=text.

Пример:

|  |  |
| --- | --- |
| <FORM>  <INPUT type=password name=MyPass size=20  maxlenght=40 value=“Пароль”>  </FORM> |  |

Многострочное поле ввода <TEXTAREA>…</TEXTAREA>

Прямоугольное текстовое поле с заданными шириной и высотой. При необходимости автоматически генерируется вертикальная линейка прокрутки содержимого.

Этот элемент удобен для ввода в текст нескольких строчек. Начальное содержимое поля кодируется так же, как и содержимое блока <PRE>, т.е. текст выводится в том виде, в каком он представлен в HTML-файле без автоматического форматирования.

Атрибуты:

name – имя области;

cols – ширина области (в символах);

rows – высота области (в строках);

wrap – управление автоматическим переносом строк внутри окна и при передаче текста на сервер.

Атрибут может принимать следующие значения:

wrap=off – без переноса;

wrap=physical – автоматический перенос с вставкой в соответствующие места текста символов перехода на новую строку;

wrap=virtual – автоматический перенос строк без добавления символов перехода на новую строку (т.е. на сервер текст передается как одна строка).

Пример:

|  |  |
| --- | --- |
| <FORM>  <TEXTAREA name=MyFam rows=7 cols=20 wrap=virtual> Для прокручивания содержимого в этом поле ввода используйте вертикальную линейку прокрутки или клавиши PgUp, PgDn либо клавиши управления курсором.  </TEXTAREA>  </FORM> |  |

Раскрывающиеся списки <SELECT>…</SELECT>

Предназначается как для предоставления пользователю возможности выбора одной или нескольких опций, так и для использования в качестве своеобразного меню. В HTML кодируется при помощи контейнера <SELECT>…</SELECT>.

Атрибуты <SELECT>…</SELECT>

name – имя списка;

size – количество одновременно отображаемых пунктов;

multiple – разрешение помечать (при одновременно нажатых клавишах Ctrl и Shift) сразу несколько пунктов меню.

Если атрибут multiple и/или значение атрибута size больше 1, то список выводится развернутым (тогда значение size определяет количество одновременно видимых на экране пунктов списка, для просмотра остальных используется линейка прокрутки). В противном случае (при отсутствии параметра multiple и при size =1) выводится список, раскрывающийся щелчком мыши на расположенной справа кнопке со стрелкой вниз.

Значение отдельных пунктов списка определяется при помощи тегов <OPTION>, каждый из которых определяет один пункт.

Атрибуты <OPTION>

value – записанная в кавычках произвольная комбинация букв и цифр, передаваемая на сервер, если данный пункт выбран; при отсутствии данного параметра на сервер передается название выбранного пункта списка;

selected – определяет данный пункт как помеченный по умолчанию (для раскрывающегося списка, т.е. при size =1 и отсутствии атрибута multiple, данный атрибут может быть присвоен только одному пункту; при отсутствии атрибута selected изначально выбранным по умолчанию является первый пункт списка) В IE 5.0 этот параметр обязателен, иначе поле списка выводится пустым.

Название пункта записывается после тега <OPTION>.

Пример:

|  |  |
| --- | --- |
| <FORM>  <TABLE bgcolor=silver>  <TR>  <TD>  <P>Укажите автора:  <P>Здесь лапы у елей дрожат на весу,<BR>  Здесь птицы щебечут тревожно.<BR>  Живешь в заколдованном диком лесу,<BR>  Откуда уйти невозможно.  </TD>  </TR>  <TR>  <TD>  <SELECT name=autor>  <OPTION>Выберите автора</OPTION>  <OPTION>А.Пушкин</OPTION>  <OPTION>М.Лермонтов</OPTION>  <OPTION>В.Высоцкий</OPTION>  <OPTION>Б.Окуджава</OPTION>  </SELECT>  </TD>  </TR>  </TABLE>  </FORM> |  |

Для раскрывающегося списка самый первый пункт (демонстрируемый по умолчанию) обычно является «фиктивным и содержит текст краткой «подсказки» для пользователя, например: «Выберите автора».

Командные кнопки

Кнопка для отправки формы в сеть <INPUT type=submit>

Атрибуты тега <FORM> определяют, куда и как будет послана форма. Атрибут submit служит для создания кнопки, щелчок по которой вызывает отправку информации формы на сервер.

Атрибуты

name – имя кнопки. Должен присутствовать во всех элементах INPUT, кроме кнопок подтверждения и сброса. Значение этого атрибута определяет имя поля формы, то есть блока данных, введенных в это поле. Программа сервера по этому имени может выделить необходимые данные.

value - надпись на кнопке. При отсутствии данного параметра выводится стандартная надпись Подача запроса или Submit.

Пример:

|  |  |
| --- | --- |
| <FORM>  <INPUT type=submit name=znach value=Пуск>  </FORM> |  |

Кнопка очистки формы <INPUT type=reset>

Щелчок мышью по этой кнопке очищает форму (т.е. приводит ее к изначальному виду, в том числе восстанавливая изначальные значения в текстовых полях, изначальную полетку флажком и радиокнопок и выбор пунктов списков.

Атрибуты:

name – имя кнопки. Необязательный параметр, нигде не используется, разве что при работе со скриптами;

value - надпись на кнопке. При отсутствии данного параметра выводится стандартная надпись Сброс или Reset.

Пример:

|  |  |
| --- | --- |
| <FORM>  <INPUT type=reset value=Сброс>  </FORM> |  |

Обычные кнопки <INPUT type=button>

Браузер самостоятельно не обрабатывает нажатие на такую кнопку. Разработчик должен предусмотреть обработку при помощи программы (скрипта), написанной, например, на JavaScript.

Для этого в теге <INPUT> используются атрибут onclick, который определяет программу, которая обрабатывает нажатие кнопки.

Атрибуты:

name – имя кнопки;

value - надпись на кнопке.

Пример:

|  |  |
| --- | --- |
| <FORM>  <INPUT type=button value=“Нажми меня” onclick= alert(“Привет!”)>  </FORM> |  |

В примере использована функция alert, которая выводит на всплывающую панель значение своего аргумента.

Графические командные кнопки <INPUT type=image>

По назначению аналогичны пользовательским кнопкам стандартного вида, но здесь в качестве кнопки используется произвольное изображение, хранящиеся в отдельном графическом файле. Фактически, это рисунок-гиперссылка, но для графической кнопки на сервер, кроме ее имени, дополнительно передается информация о координатах курсора мыши в момент нажатия.

Атрибуты:

name – имя кнопки;

align – выравнивание кнопки относительно окружающего текста (bottom, left, middle, right, top). Необязательный атрибут;

border – толщина рамки вокруг кнопки. Необязательный атрибут;

src – имя (и, если требуется, путь или адрес URL);

value – текстовая строка.

Пример:

|  |  |
| --- | --- |
| <FORM>  <INPUT type=image name=button value="Графическая кнопка" src=./pic/image.jpg>  </FORM> |  |

Интерфейсные элементы блока <FORM>

|  |  |  |
| --- | --- | --- |
| Элемент | Тег | Описание |
| Радиокнопки | <INPUT type=radio> | Выбор одного варианта из нескольких альтернативных. |
| Флажки | <INPUT type=checkbox> | Установка меток. Любое число меток. |
| Строка ввода | <INPUT type=text> | Ввод в границах одной строки. |
| Поле с паролем | <INPUT type=password> | Введенная информация отображается звездочками |
| Поле ввода | <TEXTAREA>  …  </TEXTAREA> | Ввод в многострочное поле. |
| Меню | <SELECT>  <OPTION></OPTION>  <OPTION></OPTION>  <OPTION></OPTION>  </SELECT> | Выбор вариантов из нескольких альтернативных. |
| Обычные кнопки | <INPUT type=button value=Вид> | Нажатие на такую кнопку обрабатывается при помощи скриптовой программы. |
| Восстановление формы | <INPUT type=reset  value=Сброс> | Приведение элементов формы в начальное состояние (то, которое было до воздействий пользователя) |
| Посылка форма в сеть | <INPUT type=submit value=Пуск> | После нажатия на эту кнопку браузер отправляет форму по указанному в атрибуте action тега <FORM> адресу. |
| Кнопка с рисунком | <INPUT type=image> | Для указания графического файла используется атрибут src. |
| Присоединение файла к форме | <INPUT type=file> | Пользователю предлагается записать имя файла в поле ввода. Браузер автоматически создает рядом с полем ввода кнопку Обзор, которая позволяет запустить стандартный диалог выбора файлов. |
| Скрытый элемент | <INPUT type=hidden> | Такие элементы используются для того, чтобы включить в набор данных формы некую фиксированную информации. Это определение имени переменной и ее значения. |

Пример страницы с формами в приложенном файле.

**Контрольные вопросы:**

1. Таблицы в HTML
2. HTML-формы
3. Фреймы в HTML
4. Базовые типы данных в HTML
5. Общая структура документа HTML

## Практическая работа №27

Создание фреймов в html-документе.

Цель: научиться создавать фреймы в html-дщкументе

Студент должен:

иметь практический опыт:

осуществления навигации по ресурсам, поиска, ввода и передачи данных с помощью технологий и сервисов Интернета (ПО2);

создания и обработки объектов мультимедиа (ПО3);

уметь:

создавать и редактировать объекты мультимедиа (У2);

знать:

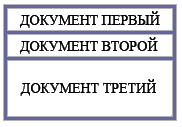
назначение, разновидности и функциональные возможности программ для создания объектов мультимедиа (З2).

Фреймы.

Итак, фреймы.. для чего они нужны и какую пользу в себе несут? Об этом я и постараюсь рассказать в этой главе, ну и естественно о том, как их внедрять и работать с ними..

Часто при создании сайта возникает необходимость открывать в одном окне браузера одновременно несколько HTML документов.. так вот фреймы как раз таки и созданы для того чтобы определить рабочие области для каждого документа. Кроме того, фреймы - это хороший инструмент, с помощью которого можно заниматься версткой страницы, они служат достойной "альтернативой" табличному способу верстки страницы.. в кавычки взял слово "альтернатива" потому, что это совершенно иной способ построения сайта со своими достоинствами и недостатками и его достаточно трудно сопоставлять с доселе привычным построением сайта.. но обо всем по порядку..

Ну что, начнем? Допустим нам необходимо в одном окне браузера открыть сразу три HTML документа и расположить их к примеру вот таким способом:



Что нам для этого нужно? Для начала естественно нам необходимо создать три отдельных html документа которые мы собственно и будем открывать в одном окне.. Пусть первый документ будет нести в себе графическое изображение и будет выполнять роль логотипа назовём его logotype.html , вторым документом будет некое содержание.. назовём menu.html , ну и третьим документ с кучей текста. text.html. Конечно Вы можете придумать свои названия, как впрочем и содержание документов, но пока лучше всё же копируйте мои.. так будет удобнее и мне и Вам.

Итак, есть три файла logotype.html, menu.html и text.html.. которые нам необходимо разместить под одной крышей, а крыши у нас как раз таки ещё пока и нет..

Пишем "крышу", главный документ к которому будем подключать наши файлы. Раз он будет главным, присваиваем ему имя index.html.

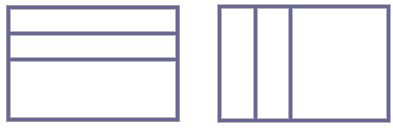
<html>  
<head>  
<title>фреймы</title>  
</head>  
<body>  
</body>  
</html>

Вот привычная для нас структура документа которую мы пережевали ещё в самом начале нашего обучения.. Фреймы ломают сложившиеся стереотипы! Структура фрейм документа имеет вот такой вид:

<html>  
<head>  
<title>фреймы </title>  
</head>  
**<frameset>**  
**</frameset>**  
</html>

Отсутствие тега **<body>** компенсирует новый тег **<frameset>** - установить фрейм или набор фреймов.. с этим товарищем мы собственно и будем работать дальше.

Тег **<frameset>** имеет атрибуты **rows** и **cols** - эти атрибуты указывают браузеру на то, как следует размещать фреймы в окне браузера

**rows** - горизонтально                **cols** - вертикально  


В нашем случае нам необходимо горизонтальное размещение фреймов.. значит пишем так:

<html>  
<head>  
<title>фреймы </title>  
</head>  
<frameset **rows="15%,15%,70%"**>  
</frameset>  
</html>

Проценты после знака равенства - это не что иное, как размеры наших окошек- фреймов в одном большом окне браузера, у нас предполагается размещение трёх окон - следовательно и значений через запятую тоже три. Вспомните как мы задавали размеры ячеек для таблицы, здесь тот же принцип.. Как и в случае с ячейками таблицы, размеры фреймов можно указывать в процентах от общей площади и в пикселях.

Вот несколько примеров написания:   
**<frameset rows="15%,15%,70%">** - такая запись говорит о том, что будет три горизонтальных окошка где последнее займет 70% площади окна браузера, а первые два по 15%.   
**<frameset cols ="200,240,200">** - здесь три окна расположены вертикально ширина которых указана в пикселях.   
**<frameset rows="100,\*,180">** - такая запись значит, что первое и третье окно будут иметь ширину 100 и 180 пикселей соответственно, а второй фрейм займет всю оставшеюся площадь.

С размещением документов покончили, теперь собственно осталось их подключить и насладится первым результатом..

Тег **<frame>** и его атрибут **src** укажет браузеру путь к html документу который следует открыть в отведенном для него месте. У нас имеется три отдельных документа logotype.html, menu.html и text.html теперь к каждому нужно прописать путь, что мы собственно и делаем. (предполагаю, что у Вас все четыре файла находятся в одной папке и пути к ним имеют самую простую запись по типу: **<frame src="logotype.html">**)

Пример:

Файл index.html   
<html>  
<head>  
<title>фреймы</title>  
</head>  
<frameset rows="15%,15%,70%">  
**<frame src="logotype.html">**  
**<frame src="menu.html">**  
**<frame src="text.html">**  
</frameset>  
</html>

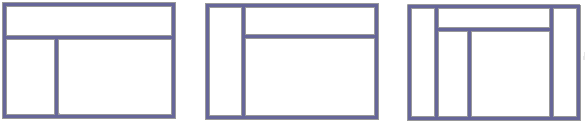
Файл logotype.html   
<html>  
<head>  
<title>фреймы</title>  
</head>  
<body>  
<img src="logotype.jpg" width="800" height="120" alt="Кулинария">  
</body>  
</html>

Файл menu.html   
<html>  
<head>  
<title>фреймы</title>  
</head>  
<body bgcolor="#e8e8e8" background="fon1.jpg">  
<font face="Monotype Corsiva" size="6"color="#ff0000">Меню:</font><br>  
<font face="Comic Sans MS" size="3">  
<a href="text.html">Суп из шампиньонов</a><br>  
<a href="text1.html">Бобы в горшочке по-итальянски</a><br>  
<a href="text2.html">Австралийский летний салат</a><br>  
... ... ...  
</font>  
</body>  
</html>

Файл text.html   
<html>  
<head>  
<title>фреймы</title>  
</head>  
<body bgcolor="#f5f5f5">  
<center><font face="Comic Sans MS" size="3">Суп из шампиньонов</font></center>  
<br>  
куча текста..  
</body>  
</html>

В данном примере я выложил все четыре HTML документа где index.html является основным, а остальные три, это подключаемые страницы, каждая из которых в принципе может работать автономно и содержать в себе всё что угодно картинки, таблицы, тексты, ссылки.. сделал я это для того что бы дать Вам понять принцип работы с фреймами. В дальнейших примерах я буду выкладывать лишь головной файл с фреймами, (а то уж больно много места на странице они все вместе занимают), а Вы знайте что у меня "где то там" есть все остальные файлы содержащие в себе графику, тексты, может быть ещё что ни будь.. ну и не отставайте за мной потихоньку редактируйте свои собственные странички к будущему тренировочному сайту.. Не знаю про что Вы будите писать свой, а я тут пообещался помочь одной девушке в написании сайта посвящённого кулинарии..)) так что решил так сказать сразу на двух зайцев замахнуться.. на девушку зайку и сайт..)) как-то так..))

В выше изложенном примере мы расположили все окна горизонтально, поменяв атрибут **rows** на **cols** можно расположить их в вертикальном порядке. А как быть, если требуется разместить наши окна



Выход есть. Разберём на примерах.

Начнём с первого случая.. что мы видим? А видим мы две строки, где вторая поделена на два столбца.

A теперь по порядку:

**<frameset rows="20%,80%">** *- делим окно браузера на две строки*  
**<frame src="logotype.html">***- в первую строку загружаем наш логотип*  
**<frameset cols="30%,70%">***- а вторую строку делим на два столбца*  
**<frame src="menu.html">** *- в первом будет меню*  
**<frame src="text.html">***- а во втором куча текста*  
**</frameset>** *- закрываем тег деления на столбцы*  
**</frameset>** *- закрываем тег деления на строки*

Вообще то говорить столбцы и строки неверно, так как фреймы никакого отношения к таблицам не имеют кроме как визуального подобия, правильно говорить горизонтальные и вертикальные фреймы.. ну это я так изрекаюсь что б Вам понятнее было..

Ну да ладно, смотрим пример:

<html>  
<head>  
<title>фреймы</title>  
</head>  
<frameset rows="15%,85%">  
<frame src="logotype.html">  
<frameset cols="20%,80%">  
<frame src="menu.html">  
<frame src="text.html">  
</frameset>  
</frameset>  
</html>

Во втором случае у нас имеется два столбца в котором второй поделен на две строки, значит будем писать так:

**<frameset cols="30%,70%">** *- делим окно на два столбца*  
**<frame src="menu.html">** *- в первом будет содержание*  
**<frameset rows="20%,80%">** *- а второй делим на две строки*  
**<frame src="logotype.html">***- логотип*  
**<frame src="text.html">** *- и основной текст*  
**</frameset>** *- закрываем деление на строки*  
**</frameset>***- закрываем деление на столбцы*

Пример:

<html>  
<head>  
<title>фреймы</title>  
</head>  
<frameset cols="20%,80%">  
<frame src="menu.html">  
<frameset rows="15%,85%">  
<frame src="logotype.html">  
<frame src="text.html">  
</frameset>  
</frameset>  
</html>

Третий случай немного посложней, но бояться его не стоит.. тем более что именно такое построение лично я выбрал для сайта про кулинарное искусство, чуть ниже объясню почему. Что мы имеем? три столбца причем во второй столбец по сути заключён наш первый случай..

**<frameset cols="\*,800,\*">** *- делим окно на три столбика (обратите внимание второй столбец занимает ровно 800 пикселей а два крайних точного размера не имеют и поделят пополам оставшееся место)*  
**<frame src="dekor.html">** *-в первый столбец загружаем html документ который будет выполнять сугубо декоративную функцию*  
  
**<frameset rows="15%,85%">** -  
**<frame src="logotype.html">** -  
**<frameset cols="20%,80%">** - *во второй столбец вставляем наш "первый случай"*  
**<frame src="menu.html">** -  
**<frame src="text.html">** -  
**</frameset>** -  
**</frameset>** -  
  
**<frame src="dekor.html">***- в третий столбец загружаем всё тот же файл с декорациями*  
**</frameset>***- захлопываем*

Посмотрите пример, а потом я объясню почему избрал путь с пятью окнами

Пример:

<html>  
<head>  
<title>фреймы</title>  
</head>  
<frameset cols="\*,800,\*">  
<frame src="dekor.html">  
<frameset rows="120,\*">  
<frame src="logotype.html">  
<frameset cols="200,600">  
<frame src="menu.html">  
<frame src="text.html">  
</frameset>  
</frameset>  
<frame src="dekor.html">  
</frameset>  
</html>

Так почему пять окон? Помнится мне, я уже писал про то, что у разных пользователей сети Интернет на мониторах выставлены разные разрешения и соответственно на разных экранах наш сайт будет выглядеть по разному.. и если нет конкретных размеров все наши рисунки, тексты, таблицы что называется "поплывут" у тех пользователей, разрешения мониторов которых отлично от Вашего. Когда мы верстали сайт с помощью таблицы, вопрос с размерами страницы решался при помощи присуждения этой таблице конкретной ширины и высоты, к сожалению, с фреймами так поступить нельзя.. даже если указывать ширину фреймов не в процентах, а в пикселях, всё равно последний столбец растянется на оставшеюся ширину окна браузера и страница, что называется, потеряет "товарный вид". Так как же быть? Приходится идти на маленькие хитрости.. Задав центральному столбцу (в котором у нас собственно и размещена вся страница) размер 800 пикселей мы раз и навсегда определи его ширину, а безразмерные первый и третий столбцы помимо декора выполняют роль своего рода "пружин" на которых подвешен центральный столбец.. Так, у людей с маленьким разрешением монитора эти окна/поля будут узкими, а людей с большим разрешением широкими, таким образом центральный столбец никак не пострадает и теперь мы можем с полной уверенностью размещать в нём какие либо объекты, с точной привязкой по месту, не переживая за их дальнейшую судьбу. Сравните первый пример, где три окна и третий, где их уже пять, неправда ли так лучше?

На этом с размещением и размерами фреймов закончим.. и так здесь долго задержались.. идём дальше.

Приводим фреймы в опрятный вид.

В последнем нашем примере первое, что бросается в глаза, это целая куча полос прокруток, которые стоят где надо и не надо.. Давайте от них избавимся, ну можно и оставить кое где.. Делается это при помощи атрибута **scrolling** - тега **<frame>**, он может иметь одно из трёх значений:

**no**- никогда не показывать полосу прокрутки,

**yes**- всегда показывать,

**auto**- показывать в том случае если она необходима.

Пример:

<html>  
<head>  
<title>фреймы</title>  
</head>  
<frameset cols="\*,800,\*">  
<frame src="dekor.html" **scrolling="no"**>  
<frameset rows="120,\*">  
<frame src="logotype.html" **scrolling="no"**>  
<frameset cols="200,600">  
<frame src="menu.html">  
<frame src="text.html">  
</frameset>  
</frameset>  
<frame src="dekor.html" **scrolling="no"**>  
</frameset>  
</html>

Поля фреймов или иначе расстояние от границ фрейма до текста или картинки, как в нашем случае, задаются в пикселях при помощи атрибутов **marginwidth** и **marginheight** тега **<frame>**

Пример:

<html>  
<head>  
<title>фреймы</title>  
</head>  
<frameset cols="\*,800,\*">  
<frame src="dekor.html" scrolling="no">  
<frameset rows="120,\*">  
<frame src="logotype.html" scrolling="no" **marginwidth="0" marginheight="0"**>  
<frameset cols="200,600">  
<frame src="menu.html">  
<frame src="text.html" **marginwidth="10" marginheight="10"**>  
</frameset>  
</frameset>  
<frame src="dekor.html" scrolling="no">  
</frameset>  
</html>

Поговорим немного о рамках вокруг наших фреймов.

Если Вы обратили внимание, то в последнем примере наведя курсор на границу фрейма он, курсор то бишь, приобретает другой вид (захватить и переместить) и теперь эту границу можно перетащить в какую либо сторону удерживая левой кнопкой мыши. Иногда такая "мобильность" границ фреймов играет на руку веб-мастеру, но чаще всё же мешает.. Для того, чтобы запретить пользователю играться с размерами окон для тега **<frame>** придумали атрибут **noresize**

Пример:

<html>  
<head>  
<title>фреймы</title>  
</head>  
<frameset cols="\*,800,\*">  
<frame src="dekor.html" scrolling="no" **noresize**>  
<frameset rows="120,\*">  
<frame src="logotype.html" scrolling="no" marginwidth="0" marginheight="0" **noresize**>  
<frameset cols="200,600">  
<frame src="menu.html" **noresize**>  
<frame src="text.html" marginwidth="10" marginheight="10" **noresize**>  
</frameset>  
</frameset>  
<frame src="dekor.html" scrolling="no" **noresize**>  
</frameset>  
</html>

А вот давно нам знакомый атрибут **border** задаёт ширину в пикселях этих самых рамок между фрёймами.. пишется он внутри тега **<frameset>**. Как и раньше значение **border="0"** вовсе избавит нас от рамок.

Пример:

<html>  
<head>  
<title>фреймы</title>  
</head>  
<frameset cols="\*,800,\*" **border="0"**>  
<frame src="dekor.html" scrolling="no" noresize>  
<frameset rows="120,\*" **border="0"**>  
<frame src="logotype.html" scrolling="no" marginwidth="0" marginheight="0" noresize>  
<frameset cols="200,600" **border="0"**>  
<frame src="menu.html" noresize>  
<frame src="text.html" marginwidth="10" marginheight="10" noresize>  
</frameset>  
</frameset>  
<frame src="dekor.html" scrolling="no" noresize>  
</frameset>  
</html>

Фреймы и ссылки.

Настало время оживить наш сайт ссылками, да вот беда, знакомая нам обыкновенная ссылка **<a href="text1.html">Бобы в горшочке по-итальянски</a>** откроет данный документ в том же фрейме где она и находится, в нашем случае непосредственно в фрейме с содержанием, а само содержание в момент перехода по этой ссылки уйдёт в небытие.. можете взглянуть на   [**корявый пример..**](http://www.webremeslo.ru/html/primer/primer42.html)    кликнете в нём на любую ссылку.. Что бы впредь такого не происходило, необходимо указывать браузеру в каком фрейме следует открывать необходимый нам документ, ну если конечно Вам и в правду не нужно открывать его в этом же фрейме.

Помнится мы уже знакомились, в главе посвящённой ссылкам, с атрибутами **name**- имя и **target**- цель, применяются они также и при работе с фреймами механизм немного изменён а так почти тоже самое. первым делом фрейму в котором мы хотели бы открывать какие либо документы необходимо присвоить индивидуальное имя.

пишется так:

<frame src="text.html" **name="osnovnoe"**> имя можно придумать любое.. главное его не забыть..

А теперь в документе с ссылками (в нашем случае это файл menu.html)необходимо указать цель то есть собственно определить в каком окне стоит открывать тот или иной документ

Пишется так:

<a href="text1.html" **target="osnovnoe"**>Бобы в горшочке по-итальянски</a>

Ну я думаю Вам уже не стоит объяснять, что прежде чем ссылаться на какие либо документы, их необходимо создать.. в моём случае файлы (рецепты) имеют имена text.html, text1.html, text2.html…

Посмотрите пример:

Файл index.html  
<html>  
<head>  
<title>фреймы</title>  
</head>  
<frameset cols="\*,800,\*" border="0">  
<frame src="dekor.html" scrolling="no" noresize>  
<frameset rows="120,\*" border="0">  
<frame src="logotype.html" scrolling="no" marginwidth="0" marginheight="0" noresize>  
<frameset cols="200,600" border="0">  
<frame src="menu.html" noresize>  
<frame src="text.html" **name="osnovnoe"** marginwidth="10" marginheight="10" noresize>  
</frameset>  
</frameset>  
<frame src="dekor.html" scrolling="no" noresize>  
</frameset>  
</html>

Файл menu.html   
<html>  
<head>  
<title>фреймы</title>  
</head>  
<body bgcolor="#e8e8e8" background="fon1.jpg" link="#b40000" alink="#900000" vlink="#900000">  
<font face="Monotype Corsiva" size="6"color="#ff0000">Меню:</font><hr>  
<font face="Comic Sans MS" size="3">  
<a href="text.html" **target="osnovnoe"**>Суп из шампиньонов</a><hr>  
<a href="text1.html" **target="osnovnoe"**>Бобы в горшочке по-итальянски</a><hr>  
<a href="text2.html" **target="osnovnoe"**>Австралийский летний салат</a><hr>  
... ... ...  
</font>  
</body>  
</html>

Так же как и раньше документ можно открыть в отдельном окне. Напомню, пишется так:

<a href="text1.html" **target="\_blank"**>Бобы в горшочке по-итальянски</a>

Или же присвоив атрибуту **target** значение **\_top** открыть его в этом же окне браузера, но на весь экран.. "обнулив" при этом всё что бы там не находилось.. пишется так:

<a href="text1.html" **target="\_top"**>Бобы в горшочке по-итальянски</a>

Такой вот сайтик получился.. конечно над ним ещё работать и работать.. к тому же по моёй задумке он будет иметь несколько иную структуру в плане навигации по сайту, в нем будет ёщё целая куча страниц, красивое меню, но что касается фреймовой структуры, думаю она останется прежней..

Плавающий фрейм

Иногда в страницу содержащую в себе не фреймовую структуру необходимо в отдельном окне вставить другой HTML документ или даже ряд таковых документов. Для выполнения этой цели существует тег **<iframe>** - так называемый плавающий фрейм.

Данный тег имеет ряд атрибутов:

**src** - обязательный атрибут, указывающий путь к открываемой странице  
**width** - ширина плавающего фрейма в пикселях или процентах  
**height** - высота плавающего фрейма  
  
**scrolling** - показ полосы прокрутки

**no** - никогда не показывать полосу прокрутки,

**yes** - всегда показывать,

**auto** - показывать в том случае если она необходима.

**align** - выравнивание пваюшего фрейма

**left** - слева

**right** - справа

**top** - выше

**bottom** - ниже

**frameborder** - наличие рамки вокруг плавающего фрейма

1 - включить рамку

0- выключить рамку

Всё вместе пишется так:

**<iframe src="primer.html" width="300" height="250" align ="left" scrolling="auto" frameborder="1" ></iframe>**

Пример документа с плавающим фреймом:

<html>  
<head>  
<title>Плавающий фрейм</title>  
</head>  
<body>  
<center><h2>Плавающий фрейм</h2></center>  
В эту страницу введён так называемый "плавающий фрейм".  
В отдельном окне он открывает для показа другой документ html.  
<br>  
**<iframe src="primer.html" width="250" height="250" align ="left" scrolling="auto" frameborder="1"></iframe>**  
<br>  
… … …  
<br>  
</body>  
</html>

Noframes

Некоторые браузеры не поддерживают фреймовую структуру документа или неправильно её интерпретируют, кроме того зачастую пользователи в настройках своих браузеров умышленно отключают поддержку фреймовой структуры html документа. И хотя процент таких браузеров и пользователей невелик, но тем не менее они есть.

Теперь представьте что Вы построили свой сайт с помощью фреймовой структуры и вот некоторые посетители, может быть сами не подозревая в чем проблема, пытаются открыть Ваш сайт, а их браузер выдает ошибку! Что они подумают о Вашем сайте? я думаю, что то типа: "Фу.. ерунда какая то.. больше сроду сюда не зайду!".

Для того что бы дать понять пользователю, что его браузер/настройки браузера не поддерживают фреймы существует тег**<noframes>**.

Тег **<noframes>** выводит текст, заключенный в него в том случае, если браузер пользователя не поддерживает фреймы или они принудительно выключены в его настройках. Если же фреймы поддерживаются браузером пользователя, то данный тег попросту игнорируется.

Пример:

<html>  
<head>  
<title>фреймы</title>  
</head>  
<frameset cols="\*,800,\*" border="0">  
**<noframes>Извините, но Ваш браузер не поддерживает фреймы..</noframes>**  
<frame src="dekor.html" scrolling="no" noresize>  
<frameset rows="120,\*" border="0">  
<frame src="logotype.html" scrolling="no" marginwidth="0" marginheight="0" noresize>  
<frameset cols="200,600" border="0">  
<frame src="menu.html" noresize>  
<frame src="text.html" marginwidth="10" marginheight="10" noresize>  
</frameset>  
</frameset>  
<frame src="dekor.html" scrolling="no" noresize>  
</frameset>  
</html>

Результат примера будет заметен, если Ваш браузер и в правду не поддерживает фреймы (тут я надолго задумался.. :) если это так, то зачем вообще читать эту главу?) или же Вы в качестве эксперимента выключили поддержку фреймов в своём браузере.

Тег **<noframes>** должен быть расположен внутри тега **<frameset>**

С плавающим фреймом всё ещё более просто, достаточно написать нужный текст между **<iframe> </iframe>** и данная надпись будет выводится на экран, в том случае если браузер не поддерживает фреймы.

Вот так:

<iframe src="primer.html" width="300" height="250" align ="left" scrolling="auto" frameborder="1">**Извините, но Ваш браузер не поддерживает фреймы..**</iframe>

**Контрольные вопросы**

1. Структура HTML-документа и элементы разметки заголовка документа
2. Основные контейнеры заголовка HTML-документа
3. Контейнеры тела документа HTML-документа
4. Использование графики в HTML
5. Таблицы в HTML
6. HTML-формы
7. Фреймы в HTML
8. Базовые типы данных в HTML
9. Общая структура документа HTML

## Практическая работа №28

Применение каскадных таблиц стилей (CSS)

Цель: научиться применять каскадные таблицы стилей (CSS)

Студент должен:

иметь практический опыт:

осуществления навигации по ресурсам, поиска, ввода и передачи данных с помощью технологий и сервисов Интернета (ПО2);

создания и обработки объектов мультимедиа (ПО3);

уметь:

создавать и редактировать объекты мультимедиа (У2);

знать:

назначение, разновидности и функциональные возможности программ для создания объектов мультимедиа (З2).

*Дизайн* Web-узлов — это точное *размещение* компонентов *HTML*-страниц относительно друг друга в рабочей области окна браузера.

Недостатки такого определения Web-дизайна очевидны. В нем не учтены ни цвет, ни форма, ни другие свойства компонентов *HTML*-страниц. Главное в этом определении — показать ограниченность возможностей *HTML*-разметки. *Позиционирование* компонентов на странице является одним из самых слабых мест в *HTML*.

К компонентам страницы относятся: блоки текста, *графика* и встроенные приложения. Размер и границы каждого из этих компонентов в рамках *HTML*-разметки задаются с разной степенью точности. Размер графики и приложений можно задать с точностью до пикселя. Размеры текстовых блоков в *HTML* задать нельзя: они вычисляются браузером исходя из *относительного размера* шрифта по умолчанию.

*Автор* страницы не может заранее определить настройки браузера пользователя, что существенно ограничивает число вариантов представления информации на странице.

Нельзя сказать, что разработчики браузеров не пытались изменить данную ситуацию. В ранних версиях браузеров *CERN* для платформы *NEXT* и в браузерах WWWC *автор* страницы имел возможность переопределять настройки браузера по умолчанию через*HTML*-разметку. Но этот подход не получил продолжения в коммерческих продуктах.

Другой способ управления настройками браузера — *программирование* на JavaScript. Бурное развитие этого языка позволяет говорить о возможности полного контроля над процессом отображения *HTML*-страниц. Недостаток JavaScript — отказ от декларативного характера разметки и относительно большой объем кода для *переопределения* свойств *элементов разметки*.

*Спецификация CSS* (Cascading *Style* Sheets) позволяет остаться в рамках декларативного характера разметки страницы и полностью контролировать форму представления *элементов HTML-разметки*.

*Каскадные таблицы стилей* призваны разрешить противоречие между точностью определения размеров картинок и приложений, с одной стороны, и точностью определения размеров блоков текста и его начертания — с другой.

Таблицы стилей также позволяют определить цвет и начертание текстового фрагмента, изменять эти параметры внутри текстового блока, выполнять выравнивание текстового блока относительно других блоков и компонентов страницы.

Наличие подобных возможностей позволяет говорить о *CSS* как о средстве разделения логической структуры документа и формы его представления. *Логическая структура* документа определяется *элементами HTML-разметки*, в то время как форма представления каждого из этих элементов задается *CSS*-описателем элемента.

*CSS* позволяет полностью переопределить форму представления *элемента разметки* по умолчанию. Например, <I>...</I>определяет *отображение* текста курсивом:

<I>Отобразим текст курсивом</I>

А теперь переопределим стиль отображения для *элемента разметки* I ([открыть](http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/101/objects/1/files/example1.htm)):

<I STYLE="text-decoration:underline;

font-style:normal;">

Отобразим текст курсивом

</I>

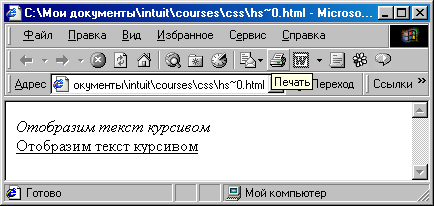


Рис. 1.1.

Этот пример показывает, что привычный стиль отображения элементов может быть полностью изменен при помощи *CSS*. В данной технологии *HTML*-разметка носит чисто декларативный характер.

Практическое *значение* *CSS* для Web-инжиниринга (совокупности технологий разработки и сопровождения Web-узлов) заключается в том, что процесс создания узла можно формализовать и представить в виде последовательности действий:

необходимо определиться с номенклатурой страниц, т.е. все страницы проектируемого Web-узла разбить на типы, например домашняя страница, навигационные страницы, информационные страницы, коммуникационные страницы и т.п. У каждого узла этот перечень может быть свой;

для каждого типа страниц требуется разработать определенную логическую структуру (стандартный набор компонентов страницы);

следует создать навигационную карту узла и форму ее реализации на страницах;

для каждого стандартного компонента страницы нужно разработать стиль его отображения (CSS-описатель);

теперь остается только рисовать картинки, создавать анимацию, писать программы, вручную вводить текст и графику или генерировать содержимое страниц автоматически при обращении к ним.

Объяснив таким образом роль и назначение *CSS* среди многообразия Web-технологий, мы переходим непосредственно к обсуждению применения *каскадных таблиц стилей*.

Способы применения CSS

Под способами применения CSS мы в данном разделе понимаем форму декларирования стиля на HTML-странице и форму связывания описания стиля отображения элемента разметки с самим элементом. Речь идет о том, где и в какой форме автор страницы (или дизайнер) описывает стиль, и как и в какой форме на него ссылается.

Итак, различают четыре способа применения стилей:

переопределение стиля в элементе разметки ;

размещение описания стиля в заголовке документа в элементе STYLE ;

размещение ссылки на внешнее описание через элемент LINK ;

импорт описания стиля в документ.

Переопределение стиля

<H1 STYLE="font-weight:normal;

font-style:italic;

font-size:10pt;">

Заголовок первого уровня

</H1>

Атрибут style можно применить внутри любого элемента разметки. Например, мы можем через style определить ширину и выравнивание элемента hr (горизонтальное отчеркивание):

<HR STYLE="width:100px;">

Очевидно, что не все параметры стиля можно установить для конкретного элемента разметки. О типах элементов и соответствующих параметрах стилей мы поговорим в разделе "Понятия блочного и строкового элементов".

Здесь же нужно отметить следующее: стили разработаны в первую очередь для управления отображением текста. Не следует увлекаться стилями при управлении отображением нетекстовых элементов HTML-разметки.

Элемент STYLE

Применение элемента STYLE — это основной способ внедрения каскадных таблиц стилей в ткань HTML-документа. Помимо управления отображением элементов разметки, элемент STYLE позволяет описывать стилевые свойства элементов, которые можно изменять при программировании на JavaScript.

Элемент STYLE дает возможность определить стиль отображения для:

стандартных элементов HTML-разметки ;

произвольных классов ( селектор   CLASS );

НTML-объектов ( селектор   ID ).

К сожалению, работа с селекторами в браузерах различных производителей может преподносить различного рода сюрпризы, для избежания которых следует придерживаться стандартов W3C.

Понятие селектора, применение селекторов и формальный синтаксис CSS мы обсудим в разделах "Синтаксис" и " Наследование ипереопределение ".

Стандартные элементы разметки описываются в элементе STYLE следующим образом:

<HEAD>

<STYLE>

p { color:darkred;text-align:justify;

font-size:8pt; }</STYLE>

</HEAD>

<BODY>

...

<P>

Этот параграф мы используем как пример

применения описания стиля для стандартного

элемента HTML-разметки.

</P>

...

</BODY>

Теперь все параграфы документа будут отображаться стилем из элемента STYLE, если только стиль не будет каким-либо способом переопределен. В STYLE можно определить стиль любого элемента разметки.

Ссылка на внешнее описание

Ссылка на описание стиля, расположенного за пределами документа, осуществляется при помощи элемента LINK, который размещают в элементе HEAD . Внешнее описание может представлять собой файл, содержащий описание стилей. Описание стилей в этом файле будет по синтаксису в точности совпадать с содержанием элемента STYLE.

Ниже приведен пример ссылки на внешнее описание стилей:

<LINK TYPE="text/css" REL="stylesheet"

HREF="http://intuit.ru/my\_css.css">

Здесь важны значения атрибутов REL и TYPE. Атрибут REL должен иметь значение stylesheet. Type может принимать значения text/css или text/javascript. Второй тип описания стилей введен Netscape. Его мы в данном учебном курсе не обсуждаем.

Атрибут HREF задает универсальный локатор ресурса (URL) для внешнего файла описания стилей. Это может быть ссылка на файл с любым именем, а не только на файл с расширением \*.css.

Импорт описания стилей

Импорт описателей стилей в некотором смысле составляет конкуренцию представленному выше указанию на внешний описатель стиля.

Импортировать стиль можно либо внутрь элемента STYLE, либо внутрь внешнего файла, который представляет собой описатель стиля. Оператор импорта стиля должен предшествовать всем прочим описателям стилей:

<STYLE>

@import url(http://intuit.ru/style.css)

A { color:cyan;text-decoration:underline; }

</STYLE>

Импортируемый стиль можно переопределить либо через описатель элемента в STYLE, либо через атрибут элемента STYLE.

Синтаксис

Формально стиль отображения *элементов разметки* задается ссылкой в *элементе разметки* на *селектор* стиля. *Синтаксис* описания стилей в общем виде представляется следующим образом:

selector[, selector[, ...]]

{ attribute:value;

[attribute:value;...] }

или

selector selector [selector ...]

{ attribute:value;

[attribute:value;...] }

В первом варианте перечислены *селекторы*, для которых действует данное описание стиля. Второй вариант задает иерархию вложенности *селекторов*, для совокупности которых определен стиль. Напомним, что речь в данном случае идет об описаниях стилей в нотации text/css. Описания стилей размещаются либо внутри элемента STYLE, либо во *внешнем файле*.

В качестве *селектора* можно использовать имя *элемента разметки*, *имя класса* и *идентификатор объекта* на *HTML*-странице.

*Атрибут* (attribute) определяет свойство отображаемого элемента, например левый *отступ* параграфа (*margin-left*), а*значение* (value) — *значение* этого атрибута, например, 10 *типографских пунктов* (10 *pt*).

Селектор — имя элемента разметки

Когда автор Web-узла хочет определить общий стиль всех страниц, он просто прописывает стили для всех *элементов HTML-разметки*, которые будут использоваться на страницах. Это дает возможность скомпоновать страницы из логических элементов, а стиль отображения элементов описать во *внешнем файле*.

Такой способ создания сайта позволяет автору изменять внешний вид всех страниц путем внесения изменений в файл описания стилей, а не в файлы HTML-страниц.

*Внешний файл* при этом может выглядеть следующим образом:

I, EM {color:#003366;font-style:normal;}

A I {font-style:normal;font-weight:bold;

text-decoration:line-through;}

В первой строке этого описания перечислены селекторы-элементы, которые будут отображаться одинаково:

<I>Это курсив</I> и это тоже <EM>курсив</EM>

Последняя строка определяет стиль отображения вложенного в гипертекстовую ссылку курсива:

<A NAME=empty><I>intuit</I></A>

В данном случае *переопределение* состоит в том, что текст отображается внутри гипертекстовой ссылки перечеркнутым, причем жирным шрифтом.

Селектор — имя класса

***Имя класса*** не является каким-либо стандартным именем *элемента HTML-разметки*. Оно определяет описание класса *элементов разметки*, которые будут отображаться одинаково. Для того, чтобы отнести *элемент разметки* к тому или иному классу, нужно воспользоваться его атрибутом CLASS ([открыть](http://www.intuit.ru/EDI/12_08_14_3/1407852794-22209/tutorial/101/objects/1/files/example2.htm)):

<STYLE>

.test {color:white;background-color:black;}

</STYLE>

...

<P CLASS="test">

Этот параграф мы отобразим белым цветом по

черному фону

</P>

...

<P>

Эту <A CLASS="test">гипертекстовую ссылку</A>

мы отобразим белым цветом по черному фону.

</P>

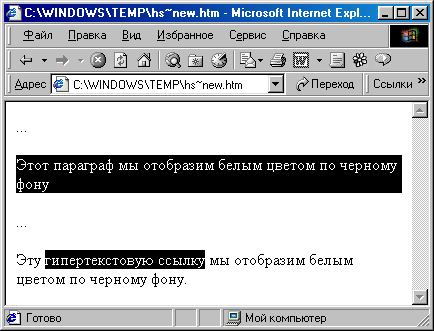


Рис. 1.2.

Таким образом в любом *элементе разметки* можно сослаться на описание класса отображения. При этом совершенно необязательно, чтобы *элементы разметки* были однотипными. В примере к одному классу отнесены и параграф, и гипертекстовая ссылка в другом параграфе.

Лидирующую точку в *имени класса* можно опустить. Она задается из соображений сохранения единства описания. Например, можно определить классы отображения однотипных *элементов разметки*:

a.menu { color:red;background-color:white;

text-decoration:none; }

a.paragraph { color:navy;

text-decoration:underline; }

В данном примере класс гипертекстовых ссылок menu имеет одно описание стиля, а класс гипертекстовых ссылок paragraph — совершенно другое. При этом каждый из этих классов нельзя применить к другим *элементам разметки*, например, параграфу или списку. Если имя *элемента разметки* не задано, это означает, что класс можно отнести к любому *элементу разметки* — корневой класс описания стилей. Это очень похоже на обозначение имени корневого домена в системе доменных имен. Собственно ничего удивительного здесь нет, т.к. система классов объектов на HTML-странице представляет собой дерево. *Элементы разметки* — это узлы дерева.

Селектор — идентификатор объекта

***Объектная модель документа*** (Document Object Model) описывает документ как дерево объектов. Объектами являются: сам документ, его разделы (элемент DIV ), картинки, параграфы, приложения и т.п. Каждому из объектов можно дать имя и обращаться к нему по имени. Данная возможность используется при программировании страниц на стороне клиента.

Применение *идентификатора объекта* оправдано еще и в случае модификации атрибута описания стиля для данного объекта в его CSS-описании. Вместо двух описаний классов, которые отличаются только одним из параметров, можно создать одно описание класса и описание *идентификатора объекта*. Описание стиля для объекта задается строкой, в которой *селектор* представляет собой имя этого объекта с лидирующим символом "#":

a.mainlink { color:darkred;

text-decoration:underline;

font-style:italic; }

#blue { color:#003366 }

...

<A CLASS=mainlink>основная гипертекстовая

ссылка</A>

<A CLASS=mainlink ID=blue>модифицированная

гипертекстовая ссылка</A>

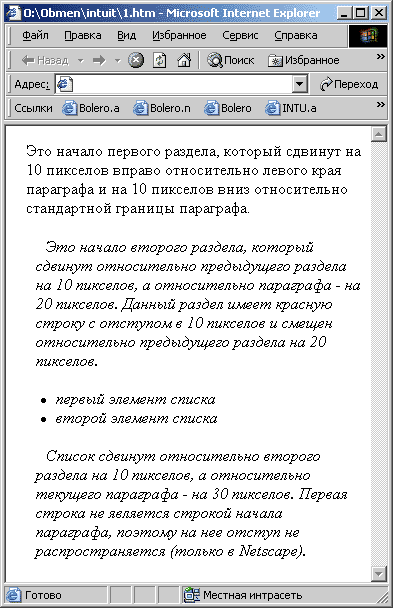
Следует отметить, что интерпретация *идентификаторов объектов* в Internet Explorer и Netscape Navigator различна. Существует еще атрибут name у *элемента разметки*. При идентификации объекта Netscape Navigator обычно имеет дело именно с этим атрибутом, а Internet Explorer — с атрибутом ID.

Различия в интерпретации ID в браузерах при декларативном использовании CSS не очень страшны. Другое дело, если автор решится программировать стили, т.е. изменять значения атрибутов описателей стилей. В этом случае разница *объектных моделей документов* в Netscape Navigator и Internet Explorer проявится в полной мере. Фактически, придется для каждого из браузеров разрабатывать совершенно разные страницы.

Наследование и переопределение

При обсуждении технических спецификаций часто бывает полезно вникнуть в смысл названия. В названии принято точно определять суть и назначение стандарта или спецификации. Описание стилей отображения *элементов HTML-разметки* носит название *"Каскадные таблицы стилей"*. Со словом "стилей" все более-менее понятно. Под словом "таблицы" следует понимать набор свойств *элемента разметки*, который можно представить в виде строки в таблице свойств, т.е. *элементы разметки* — строки, а свойства — столбцы. А вот *слово* "каскадные" требует пояснения.

Во-первых, существует *иерархия* *элементов разметки* (*дерево* объектов на странице). Во-вторых, свойства этих объектов могут наследоваться. Таким образом в дереве объектов образуется *ветвь*, которая ведет к листу дерева — *элементу разметки*, например, элементу списка или параграфу. Его свойства определяются *элементами разметки*, в которые вложен элемент, и описателями стиля для данного элемента:



Предыдущий текст закодирован в терминах разделов и списка следующим образом:

<DIV STYLE="margin-left:10px;margin-top:10px;">

Это начало первого раздела, который сдвинут

на 10 пикселов вправо относительно левого

края параграфа и на 10 пикселов вниз

относительно стандартной границы параграфа.

<DIV STYLE="margin-left:10px;margin-top:20px;

text-indent:10px;font-style:italic;">

Это начало второго раздела, который

сдвинут относительно предыдущего раздела

на 10 пикселов, а относительно параграфа —

на 20 пикселов. Данный раздел имеет красную

строку с отступом в 10 пикселов и смещен

относительно предыдущего раздела на 20

пикселов.

<UL STYLE="margin-left:10px;">

<LI>первый элемент списка

<LI>второй элемент списка

</UL>

Список сдвинут относительно второго раздела

на 10 пикселов, а относительно текущего

параграфа — на 30 пикселов. Первая строка не

является строкой начала параграфа, поэтому на

нее отступ не распространяется (только в

Netscape).

</DIV>

</DIV>

Таким образом отступы отсчитываются относительно элемента, в который вложен текущий элемент. Все параметры, которые не были переопределены в текущем элементе, наследуются из старшего по иерархии элемента. Последнее хорошо продемонстрировано в применении стилей отображения списка, который вложен в раздел и поэтому отображается курсивом.

Когда объяснение некоторого феномена *HTML*-разметки растягивается на несколько параграфов, имеет смысл воспользоваться приведенной ниже графической схемой построения страницы.

При использовании стилей действуют следующие правила ***старшинства стилей***:

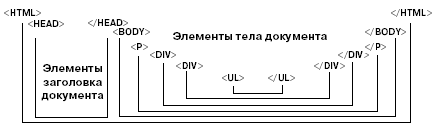


Рис. 1.3.

сначала применяются стили браузера по умолчанию;

стили браузера по умолчанию переопределяются прилинкованными стилями (элемент LINK заголовка документа);

прилинкованные стили переопределяются описаниями стилей в элементе STYLE ;

стили элемента STYLE переопределяются атрибутом STYLE в любом из *элементов разметки*.

Не все атрибуты стиля могут наследоваться. Например, "набивка" (*отступ* содержания элемента от его границ) элемента BODY не наследуется вложенными в него элементами и определяется по умолчанию или прописывается для каждого элемента отдельно. Алгоритмы *наследования* в *Internet* Explorer и в Netscape Navigator разные, поэтому для единства отображения элементов следует прописывать стиль по максимуму атрибутов.

**Контрольные вопросы:**

1. Структура HTML-документа и элементы разметки заголовка документа
2. Основные контейнеры заголовка HTML-документа
3. Контейнеры тела документа HTML-документа
4. Использование графики в HTML
5. Таблицы в HTML
6. HTML-формы
7. Фреймы в HTML
8. Базовые типы данных в HTML
9. Общая структура документа HTML

## Практическая работа №29

Способы обмена данными в рамках протокола HTTP с применением HTML-форм и CGI-скриптов.

Цель: рассмотреть способы обмена данными в рамках протокола HTTP с применением HTML-форм и CGI-скриптов.

Студент должен:

иметь практический опыт:

осуществления навигации по ресурсам, поиска, ввода и передачи данных с помощью технологий и сервисов Интернета (ПО2);

создания и обработки объектов мультимедиа (ПО3);

уметь:

создавать и редактировать объекты мультимедиа (У2);

знать:

назначение, разновидности и функциональные возможности программ для создания объектов мультимедиа (З2).

Главным достижением технологии World Wide Web по праву считают унификацию интерфейса пользователя при работе с информационными ресурсами Internet. Универсальный мультипротокольный браузер, будь то Netscape Navigator или Internet Explorer, позволяет путем выбора гипертекстовой ссылки получить доступ к FTP-архиву, архиву Gopher, новостям из конференции Usenet или отправить письмо по электронной почте. До эпохи Web для каждого из этих ресурсов пришлось бы запускать отдельную программу.

Однако, кроме текстов, которые можно читать, или картинок, которые можно просматривать, существует множество ресурсов, требующих ввода информации в процессе работы с ними. К таким ресурсам, в частности, относятся информационно-поисковые системы, где пользователь должен вводить список ключевых слов, или реляционные (да и любые другие) базы данных, гдепользователь может формулировать запросы к отношениям. Более того, для любой страницы, которая требует аутентификации пользователя, необходимо вводить идентификатор и пароль.

На сегодня уже сложился определенный стиль графического интерфейса приложения. Существует достаточно большое число прикладных пакетов, которые позволяют "прилаживать" такой интерфейс к программе. Однако на такое "прилаживание" или прямоепрограммирование уходит до 80% трудозатрат программистов. При этом в большинстве случаев все сводится к разбору введенных параметров с последующей выдачей результатов в виде отформатированного текста.

Форматирование страниц в Web-технологии достигается за счет HTML-разметки. Остается только создать инструмент ввода данных через рабочее окно браузера или через HTML-документ. В 1991 году эта проблема была решена специалистами NCSA. Они разработали и реализовали две взаимосвязанные спецификации: HTML-формы и Common Gateway Interface.

Формы произвели настоящую революцию в HTML-разметке: авторы документов получили возможность создавать сложные шаблоны ввода информации в рамках HTML-страницы, пользователи — эти шаблоны заполнять. При этом авторы форм опирались на свойства HTTP-протокола и универсальный локатор ресурсов URL с учетом того, что при HTTP-обмене можно использовать различные методы доступа к ресурсам. Это позволило сделать механизм интерпретации форм расширяемым и легко приспосабливаемым к дальнейшему развитию Web-технологии. Таким образом, кроме HTTP, можно было использовать и другие протоколы, которые поддерживали универсальный браузер, например mailto.

Common Gateway Interface — это спецификация обмена данными между прикладной программой, выполняемой по запросу пользователя, и HTTP-сервером, который данную программу запускает. До появления CGI новые функции нужно было внедрять непосредственно в сервер. CGI позволила разрабатывать программы независимо от сервера, а механизм передачи им управления и данных был унаследован от программирования в среде командной строки. Последнее резко сократило трудозатраты на разработку приложений, так как не надо было программировать интерфейс пользователя: его функции выполняли формы.

Слушатели данного учебного курса научатся создавать документы с формами, программировать на стороне сервера с использованием CGI и обрабатывать данные, передаваемые браузером серверу. В рамках курса будут подробно рассмотрены различные способы такой обработки, а также основные приемы построения интерактивных страниц Web-узла.

Введение

Обмен данными в Web-технологии подразделяется в соответствии с типами методов доступа протокола HTTP и видами запросов в спецификации CGI.

Основных методов доступа два: GET и POST. Помимо них часто используются HEAD и PUT.

Виды запросов CGI разделяют на два основных MIME-типа: application/x-www-form-urlencoded и multipart/form-data. Второй тип запроса специально создан для передачи больших внешних файлов.

Эту классификацию можно представить в виде таблицы:

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | | Клиент --> Сервер | Клиент <-- Сервер |
| GET | По умолчанию | Только HTTP-заголовок | HTTP-заголовок и страница, как тело HTTP-сообщения |
| isindex | Только HTTP-заголовок (список ключевых слов включен в URL. Слова разделены символом "+". Кодирования кириллицы не производится) | HTTP-заголовок и страница, как тело HTTP-сообщения |
| form-urlencoded | Только HTTP-заголовок (данные из формы включены в URL страницы. Производится кодирование специальных символов и кириллицы) и тело HTTP-сообщения | HTTP-заголовок и страница, как тело HTTP-сообщения |
| POST | form-urlencoded | Только HTTP-заголовок (данные из формы включены в URL страницы. Производится кодирование специальных символов и кириллицы) и тело HTTP-сообщения | HTTP-заголовок и страница, как тело HTTP-сообщения |
| form-data | HTTP-заголовок и составное тело HTTP-сообщения. Первая часть тела — данные из формы, для которых производится кодирование, вторая часть тела — присоединенный файл как он есть | HTTP-заголовок и страница, как тело HTTP-сообщения |
| PUT | | HTTP-заголовок и документ, как тело HTTP-сообщения | HTTP-заголовок. В качестве тела можно передать комментарий к коду возврата |
| HEAD | | HTTP-заголовок | HTTP-заголовок |

При реализации нестандартных методов доступа, например, DELETE, могут быть несколько иные комбинации содержания откликов и ответов.

Мы рассмотрим все эти типы обменов.

HyperText Transfer Protocol

Все данные в рамках *Web*-технологии передаются по протоколу HTTР. *Исключение* составляет обмен с использованием программирования на *Java* или обмен из Plugin-приложений. Учитывая реальный объем трафика, который передается в рамках *Web*-обмена по *HTTP*, мы будем рассматривать только этот протокол. При этом мы остановимся на таких вопросах, как:

общая структура сообщений;

методы доступа;

оптимизация обменов.

Общая структура сообщений

***HTTP*** — это протокол прикладного уровня. Он ориентирован на модель обмена "клиент-сервер". Клиент и сервер обмениваются фрагментами данных, которые называются HTTP-сообщениями. Сообщения, отправляемые клиентом серверу, называют запросами, а сообщения, отправляемые сервером клиенту — откликами. Сообщение может состоять из двух частей: заголовка и тела. Тело от заголовка отделяется пустой строкой.

Заголовок содержит служебную информацию, необходимую для обработки тела сообщения или управления обменом. Заголовок состоит из директив заголовка, которые обычно записываются каждая на новой строке.

Тело сообщения не является обязательным, в отличие от заголовка сообщения. Оно может содержать текст, графику, аудио- или видеоинформацию.

Ниже приведен HTTP-запрос:

GET / HTTP/1.0

Accept: image/jpeg

пустая строка

И отклик:

HTTP/1.0 200 OK

Date: Fri, 24 Jul 1998 21:30:51 GMT

Server: Apache/1.2.5

Content-type: text/html

Content-length: 21345

пустая строка

<HTML>

...

</HTML>

Текст "пустая строка" — это просто обозначение наличия пустой строки, которая отделяет заголовок HTTP-сообщения от его тела.

Сервер, принимая запрос от клиента, часть информации заголовка HTTP-запроса преобразует в переменные окружения, которые доступны для анализа *CGI-скриптом*. Если запрос имеет тело, то оно становится доступным скрипту через поток стандартного ввода.

Методы доступа

Самой главной директивой HTTP-запроса является метод доступа. Он указывается первым словом в первой строке запроса. В нашем примере это GET. Различают четыре основных метода доступа:

GET;

HEAD;

POST;

PUT.

Кроме этих четырех методов существует еще около пяти дополнительных методов доступа, но они используются редко.

Метод GET

Метод GET применяется клиентом при запросе к серверу по умолчанию. В этом случае клиент сообщает адрес ресурса (URL), который он хочет получить, версию протокола *HTTP*, поддерживаемые им MIME-типы документов, версию и название клиентского программного обеспечения. Все эти параметры указываются в заголовке HTTP-запроса. Тело в запросе не передается.

В ответ сервер сообщает версию HTTP-протокола, код возврата, тип содержания тела сообщения, размер тела сообщения и ряд других необязательных директив HTTP-заголовка. Сам ресурс, обычно HTML-страница, передается в теле отклика.

Метод HEAD

Метод HEAD используется для уменьшения обменов при работе по протоколу *HTTP*. Он аналогичен методу GET за исключением того, что в отклике тело сообщения не передается. Данный метод используется для проверки времени последней модификации ресурса и срока годности кэшированных ресурсов, а также при использовании программ сканирования ресурсов World Wide Web. Одним словом, метод HEAD предназначен для уменьшения объема передаваемой по сети информации в рамках HTTP-обмена.

Метод POST

Метод POST — это альтернатива методу GET. При обмене данными по методу POST в запросе клиента присутствует тело HTTP-сообщения. Это тело может формироваться из данных, которые вводятся в HTML-форме, или из присоединенного внешнего файла. В отклике, как правило, присутствует и заголовок, и тело HTTP-сообщения. Чтобы инициировать обмен по методу POST, в атрибутеMETHOD контейнера FORM следует указать значение " post ".

Метод PUT

Метод PUT используется для публикации HTML-страниц в каталоге HTTP-сервера. При передаче данных от клиента к серверу в сообщении присутствует и заголовок сообщения, в котором указан URL данного ресурса, и тело — содержание размещаемого ресурса.

В отклике тело ресурса обычно не передается, а в заголовке сообщения указывается код возврата, который определяет успешное или неуспешное размещение ресурса.

Оптимизация обменов

Протокол *HTTP* изначально не был ориентирован на постоянное соединение. Это означает, что как только сервер принял запрос от клиента и ответил на него, соединение между клиентом и сервером разрывается. Для нового обмена данными нужно устанавливать новое соединение. Такой подход имеет как достоинства, так и недостатки.

К достоинствам относится возможность одновременного обслуживания большого количества коротких запросов. Даже на популярных серверах число открытых соединений может не превышать сотни при обслуживании порядка миллиона запросов в сутки. При этом один клиент может открыть до 40 соединений одновременно, и с точки зрения сервера все они равноправны. При высокоскоростных линиях связи это позволяет добиться малого времени отклика на запрос клиента для всей страницы (текст, графика и т.п.).

К недостаткам такой схемы обмена относятся: необходимость каждый раз устанавливать соединение и невозможность поддерживать сессию работы с информационным ресурсом. При инициализации соединения по транспортному протоколу TCP и разрыве этого соединения требуется передать довольно большой объем служебной информации. Отсутствие поддержки сессий в *HTTP* затрудняет работу с такими ресурсами как базы данных или ресурсы, требующие аутентификации.

Для оптимизации числа открытых TCP-соединений в HTTP-протоколе версий 1.0 и 1.1 предусмотрен режим keep-alive. В этом режиме соединение инициализируется только один раз, и по нему последовательно можно реализовать несколько HTTP-обменов.

Для обеспечения поддержки сессий к директивам HTTP-заголовка были добавлены "печенюшки" (cookies). Они позволяют сымитировать поддержку соединения при работе по протоколу *HTTP*.

Виды интерфейса пользователя в Web-технологии

Страницы World Wide *Web* по функциональному назначению можно разделить на несколько типов: информационные страницы, навигационные страницы, страницы обмена данными. Во многих случаях эти функции можно объединить в одной странице.

Информационные страницы — это последовательное изложение информации с возможностью гипертекстовых контекстных переходов. *Пользователь* просматривает их последовательно. Гипертекстовые ссылки обычно применяют для создания сносок, примечаний или отсылок к спискам литературы и других ассоциативных материалов. Типичными примерами таких страниц являются подсказки, руководства, описания компаний, исторические справки и т.п.

Навигационные страницы — это совокупность гипертекстовых ссылок, которая позволяет ориентироваться в материалах *Web*-узла. Типичный пример такой страницы — *Home* page (*домашняя страница*). Как правило, на ней нет пространных текстовых описаний и иллюстраций, она состоит из совокупности различных *меню*. Эти *меню* можно реализовать через списки, таблицы ссылок или imagemap.

Страницы обмена данными позволяют передать на *сервер* некоторый объем информации, отличный от стандартного адреса (*URL*) ресурса. При просмотре и навигации *пользователь* просто выбирает гипертекстовые ссылки, по которым загружаются новые страницы. При обмене данными на *сервер* передается не только *адрес* ресурса, но и дополнительная *информация*, которую вводит*пользователь*.

В зависимости от функционального назначения страниц изменяется вид интерфейса ресурса, с которым *пользователь* имеет дело. В первых двух случаях достаточно манипулятором "*мышь*" выбрать гипертекстовую ссылку, как тут же загрузится новая страница. В случае страниц обмена данными следует заполнить поля HTML-форм и отправить данные на *сервер*.

При этом формы обеспечивают практически все необходимые виды полей ввода и *меню*. Единственное, чего не позволяют реализовать HTML-формы, так это вложенные *меню*. Формы можно применять не только при обмене данными. Достаточно развитые*механизмы* обработки форм присутствуют в JavaScript.

Спецификация Common Gateway Interface

Данная спецификация определяет стандартный способ обмена данными между прикладной программой и HTTP-сервером. Спецификация была предложена для сервера *NCSA* и является основным средством расширения возможностей обработки запросов клиентов HTTP-сервером.

В *CGI* имеет смысл выделить следующие основные моменты:

понятие *CGI-скрипта* ;

типы запросов;

механизмы приема данных скриптом;

механизм генерации отклика скриптом.

Основное назначение *CGI* — обработка данных из HTML-форм. В настоящее время область применения *CGI* гораздо шире.

Понятие CGI-скрипта

***CGI-скриптом*** называют программу, написанную на любом языке программирования или командном языке, которая осуществляет обмен данными с HTTP-сервером в соответствии со спецификацией *Common Gateway Interface* .

Наиболее популярными языками для разработки скриптов являются Perl и С.

Типы запросов

Различают два типа запросов к *CGI-скриптам*: по методу GET и по методу POST. В свою очередь, запросы по методу GETподразделяются на запросы по типам кодирования: isindex и form-urlencoded, а запросы по методу POST —multipart/form-data и form-urlencoded.

В запросах по методу GET данные от клиента передаются скрипту в переменной окружения *QUERY\_STRING*. В запросах по методуPOST данные из формы передаются в потоке стандартного ввода скрипта. При передаче через поток стандартного ввода в переменной окружения CONTENT\_LENGTH указывается число передаваемых символов.

Запрос типа ISINDEX — это запрос вида:

http://intuit.ru/somthing-cgi/cgi-script?word1+word2+word3

Главным здесь является список слов после символа " ?". Слова перечисляются через символ " + " и для кириллицы в шестнадцатеричные последовательности не кодируются. Последовательность слов после символа " ?" будет размещена в переменной окружения *QUERY\_STRING*.

Запрос типа form-urlencoded — это запрос вида:

http://intuit.ru/somthing-cgi/cgi-script?field=word1&field2=word2

Данные формы записываются в виде пар " имя\_поля-значение ", которые разделены символом " & ".

Приведенный пример — это обращение к скрипту по методу GET. Все символы после " ?" попадут в переменную окружения*QUERY\_STRING*. При этом если в значениях полей появляется кириллица или специальные символы, то они заменяются шестнадцатиричным кодом символа, который следует за символом " % ".

При обращении к скрипту по методу POST данные после символа " ?" не будут размещаться в *QUERY\_STRING*, а будут направлены в поток стандартного ввода скрипта. В этом случае количество символов в потоке стандартного ввода скрипта будет указано в переменной окружения CONTENT\_LENGTH.

При запросе типа multipart/form-data применяется составное тело HTTP-сообщения, которое представляет собой данные, введенные в форме, и данные присоединенного внешнего файла. Это тело помещается в поток стандартного ввода скрипта. При этом к данным формы применяется кодирование как в form-urlencoded, а данные внешнего файла передаются как есть.

Механизмы приема данных скриптом

Скрипт может принять данные от сервера тремя способами:

через переменные окружения;

через аргументы командной строки;

через поток стандартного ввода.

При описании этих механизмов будем считать, что речь идет об обмене данными с сервером Apache для платформы Unix.

Переменные окружения

При вызове скрипта сервер выполняет системные вызовы fork и exec. При этом он создает среду выполнения скрипта, определяя ее переменные. В спецификации *CGI* определены 22 переменные окружения. При обращении к скрипту разными методами и из различных контекстов реальные значения принимают разные совокупности этих переменных. Например, при обращении по методу POST переменная *QUERY\_STRING* не имеет значения, а по методу GET — имеет. Другой пример — переменная окружения*HTTP\_REFERER*. При переходе по гипертекстовой ссылке она определена, а если перейти по значению поля location или через JavaScript-программу, то *HTTP\_REFERER* определена не будет.

Получить доступ к переменным окружения можно в зависимости от языка программирования следующим образом:

#Perl

$a = $ENV{CONTENT\_LENGTH};

...

// C

a = getenv("CONTENT\_LENGTH");

В случае доступа к скрипту по методу GET данные, которые передаются скрипту, размещаются в переменной окружения*QUERY\_STRING*.

Аргументы командной строки

Как ни странно звучит, но у *CGI-скрипта* может быть такой элемент операционного окружения как командная строка. Это не означает, что скрипт реально можно вызвать из командной строки через сервер. Тем не менее получить доступ к содержанию командной строки скрипта можно с помощью тех же функций, что и при вызове его из-под интерактивной оболочки:

#Perl

foreach $a (@ARGV)

{

print $a,"\n";

}

// C

void main(argc,argv)

int argc;

char \*argv[];

{

int i;

for(i=0;i<argc;i++)

{

printf("%s\n",argv[i]);

}

}

В обоих примерах показана распечатка аргументов командной строки для программ на Perl и C соответственно.

Аргументы командной строки появляются только в запросах типа ISINDEX.

Поток стандартного ввода

Ввод данных в скрипт через поток стандартного ввода осуществляется только при использовании метода доступа к ресурсу (скрипту) POST. При этом в переменную окружения CONTENT\_LENGTH помещается число символов, которое необходимо считать из потока стандартного ввода скрипта, а в переменную окружения CONTENT\_TYPE помещается тип кодирования данных, которые считываются из потока стандартного ввода.

При посимвольном считывании в C можно применить, например, такой фрагмент кода:

int n;

char \*buf;

n= atoi(getenv("CONTENT\_LENGTH"));

buf = (char \*) malloc(n+1);

memset(buf,'\000',n+1);

for(i=0;i<n;i++)

{

buf[i]=getchar();

}

free(buf);

В данном фрагменте применено динамическое размещение памяти в скрипте, поэтому при выходе из него память следует освободить. Вообще говоря, память будет автоматически освобождена операционной системой после завершения скрипта. Однако, если переносить скрипт на спецификацию FCGI (Fast CGI), что требует минимума переделок, из-за неаккуратной работы с памятью могут возникнуть проблемы.

Механизм генерации отклика скриптом

Существует только один способ вернуть данные серверу и, соответственно, браузеру пользователя — писать в поток стандартного вывода (STDOUT). При этом скрипт должен формировать HTTP-сообщение.

Сначала выводятся директивы HTTP-заголовка. В минимальном варианте это либо

Content-type: text/html,

либо

Location: http://intuit.ru/

В первом случае определяется тип тела HTTP-сообщения, а во втором осуществляется перенаправление запроса.

После заголовка генерируется отклик в виде тела HTTP-сообщения, которое должно быть отделено от заголовка пустой строкой:

#!/bin/sh

echo Content-type: text/plain

echo

echo Hello

В данном случае используется командный интерпретатор sh.

Если скрипт начинает формирование заголовка с директивы версии HTTP-протокола, то сервер не анализирует отклик и передает его как есть. Если в заголовке, сгенерированном скриптом, эта директива отсутствует, то сервер считает, что заголовок неполный, и вставляет в него дополнительные директивы.

Введение в программирование CGI-скриптов и программирование скриптов на bash

При обсуждении обмена данными между клиентом и сервером в Web-технологии логично было бы рассмотреть вопросы разработки прикладного программного обеспечения на стороне сервера. Как правило, это CGI-скрипты.

К наиболее популярным средствам разработки таких скриптов относятся:

shell (командный язык);

Perl;

С.

Тем, кто имеет представление об этих языках, достаточно выполнить контрольные тесты и перейти к целевому программированию скриптов. Для тех же, кто смутно представляет себе процесс программирования, изучение данного раздела учебного курса обязательно. Раздел содержит много простых и полезных программ, которые применяются для отладки более сложных скриптов или являются их составными частями.

Командные языки являются тем первым инструментом программирования, который попадает в руки любого пользователя. В Windowsэто cmd (речь идет об Windows NT или Windows 95), в Unix — различного рода shell. cmd оставим для учебных курсов Microsoft и сосредоточимся на командных языках Unix.

Среди различных командных языков оболочек (shell) выберем тот, который является общим для большинства Unix-платформ — GNUbash (Bourne Again Shell). Прообраз bash — самый первый shell ( sh ), поэтому bash наследует многие его свойства.

Для программирования CGI-скриптов bash удобен тем, что наглядно демонстрирует многие свойства окружения среды Unix, которые используются и в других системах программирования. Кроме того, часто программирование на командном языке применяется для сравнительных описаний программ разработанных на С или Perl.

Структура bash-скрипта

Для того чтобы выполнить bash -скрипт, требуется интерпретатор bash. При этом скрипт запускается HTTP-сервером и, в общем случае, не определяет его операционное окружение (точнее, оно определяется окружением сервера). По этой причине в начале файла скрипта следует указать, что для его исполнения требуется интерпретатор bash:

#!/usr/local/bin/bash

echo Hello BASH

Первая строчка этой записи указывает на то, что содержание файла будет рассматриваться как программа (скрипт) на bash. Кстати,bash эту конструкцию также воспринимает как запуск программы интерпретации содержимого файла, поэтому в процессе выполнения скрипта для отдельных операций можно вызывать другие скрипты со своими интерпретаторами.

В общем случае символ "#" рассматривается как начало комментария, который распространяется до конца строки. При программировании скриптов его чаще всего приходится употреблять для маскирования строк программы во время отладки.

Более bash -скрипт ничем не выделяется. Команды bash обычно вводятся каждая на отдельной строке. Если это по каким-то причинам затруднительно, то команды разделяются символом ";". Исключение составляют конвейеры: в них команды находятся в пределах одной строки и разделены символом "|".

При программировании на bash нужно четко различать команды, встроенные в bash, и команды операционной системы. Например,echo — это команда операционной системы, а let — встроенная команда bash.

Стандартный поток вывода

Собственно, сам командный язык bash не имеет механизма организации вывода данных. Среди встроенных в bash команд нет команды печати. Но зато можно воспользоваться командами Unix. Самой простой из них является команда echo, которая копирует свои аргументы в поток стандартного вывода. При этом объединять разные слова во фразу каким-либо образом не нужно:

bash>echo Perl meets CGI

Perl meets CGI

bash>

В данном случае echo вывела три своих аргумента и символ перевода строки — приглашение (prompt) bash находится на новой строке.

На первый взгляд, такое простое решение для стандартного вывода кажется примитивным. На самом деле, его вполне достаточно для генерации HTML-страниц. Механизмы, которые делают echo в совокупности с bash эффективным средством генерации отчетов в HTML-формате, таковы:

подстановка переменных (substitution);

маскирующие кавычки (quoting);

подстановка результатов выполнения команд.

В совокупности они представляют собой мощный инструмент.

Substitution позволяет формировать строку вывода путем включения в нее значений переменных. Например, если нужно распечатать позиционные параметры скрипта, сделать это можно следующим образом:

echo first\_arg#$1 second\_arg#$2

В данном случае распечатываются первый и второй аргументы командной строки скрипта. Другой пример — распечатка переменной окружения:

echo QUERY\_STRING:$QUERY\_STRING

Quoting используется для маскирования специальных значений некоторых символов. Такие символы называют метасимволами. Например: ">" и "<" — это символы перенаправления потоков ввода-вывода и, следовательно, их надо маскировать при выводе. Для такого маскирования проще всего использовать простые одинарные кавычки:

echo '<H1>QUOTING</H1>'

В данном случае мы напечатаем заголовок первого уровня в HTML-документе. При маскировании следует помнить, что внутри кавычек bash не выполняет интерпретации кода скрипта, поэтому переменные внутрь одинарных кавычек вставлять нельзя:

echo '<H1>'$QUERY\_STRING'</H1>'

В данном случае строка вывода будет состоять из трех частей: тега начала заголовка, значения переменной QUERY\_STRING и тега конца заголовка.

Подстановка результата выполнения команды осуществляется с использованием обратных кавычек (традиционный вариант) или формы $ (command). При этом в строку вывода включается значение, которое возвращает выполненная команда:

echo '<H1>'`date`'</H1>'

или

echo '<H1>'$(date)'</H1>'

Таким образом можно вставлять не только отдельные команды, но и целые последовательности команд. Главное, чтобы эта последовательность что-нибудь возвращала.

Переменные окружения

Переменные окружения (оболочки) создаются в момент старта bash -скрипта. При этом существует два типа переменных — те, которые действуют только в данной оболочке, и те, которые наследуются извне. Для просмотра переменных окружения можно использовать команду set:

bash-2.01$ set

bash=/bin/bash

bash\_versinfo=([0]="2" [1]="01" [2]="0"

[3]="1" [4]="release"

[5]="i386-pc-freebsd2.2.2")

bash\_version='2.01.0(1)-release'

columns=106

dirstack=()

euid=1010

...

Здесь не приводится полный список всех переменных окружения. Показано только, как этот список отображается. Каждаяпеременная передается парой "имя=значение". При этом каждая такая пара записывается с новой строки. Попробуем распечатать все переменные окружения скрипта в виде HTML-таблицы, используя bash:

#!/usr/freeware/bin/bash

echo Content-type: text/html

echo

echo '<HTML><HEAD></HEAD><BODY>'

echo '<H1>переменные окружения</H1>'

echo '<TABLE BORDER=1>'

echo '<TR><TD>Имя</TD><TD>значение</TD></TR>'

IFS='='

set | while read x y

do

echo '<TR><TD>'$x'</TD><TD>'$y'</TD></TR>'

done

echo '</TABLE>'

echo '<HR>'

echo '</BODY></HTML>'

Первой командой echo формируется предложение HTTP-заголовка. Вторая команда echo обеспечивает пропуск строки между заголовком HTTP-сообщения и его телом. Затем начинает формироваться тело HTML-документа. Обратите внимание на прямые одинарные кавычки " ' ". Они применяются для того, чтобы защитить от интерпретации угловые скобки " < " и " > ", которые используются в bash для перенаправления стандартных потоков ввода/вывода.

Далее присваивается значение переменной окружения bash, которая не генерируется сервером HTTP — IFS. Переменная IFSхранит список символов-разделителей слов. По умолчанию это пробел и табуляция. Но нам нужно разделить имя переменной и его значения, которые на самом деле разделены символом "=".

Теперь вызываем команду set. При этом ее стандартный поток вывода перенаправляем при помощи "|" команде read, которая считывает строку из стандартного ввода, при этом присваивая переменным x и y значения последовательно от начала строки выделенных слов. А слова мы разделяем символом "=".

Читаем стандартный ввод в цикле while условие do.... done. В качестве условия все та же команда read — если считываем данные, то "истина", если нет, то — "ложь". При этом внутри цикла выводим строки таблицы "имя — значение".

B конце скрипта приводим документ к стандартному виду HTML-документа.

Обратиться к значению переменной окружения можно, конечно, гораздо проще — по имени:

#!/usr/freeware/bin/bash

echo Content-type: text/html

echo

echo '<HTML><HEAD></HEAD><BODY>'

echo '<H1>QUERY\_STRING</H1>'

echo QUERY\_STRING = $QUERY\_STRING

echo '<HR>'

echo '</BODY></HTML>'

Здесь по команде echo будет просто распечатано значение переменной окружения   QUERY\_STRING.

Аргументы командной строки

Позиционные параметры или аргументы командной строки — это последовательность строковых констант, которые указываются в командной строке после имени скрипта. Любая встроенная в bash команда или команда Unix может запускаться с набором этих параметров. Например, для того, чтобы подсчитать число активных в данный момент процессов httpd, администратор систем выдает такую последовательность команд:

bash>ps -ax | grep httpd | wc -l

Здесь указано три команды, организованные в конвейер. Каждая из них имеет по одному аргументу командной строки:

ps задана с аргументом -ax ;

grep задана с аргументом httpd ;

wc задана с аргументом -l.

Позиционные параметры (аргументы командной строки) задаются встроенными переменными $1 — $n, где n — число аргументов. Аргументы командной строки появляются при запросах типа ISINDEX. Число аргументов командной строки определяется встроенной переменной bash — $#. Если мы вызовем скрипт по ссылке типа:

http://www.intuit.ru/cgi-bin/

argv.cgi?arg1+arg2+arg3,

то переменная $# примет значение 3, а переменные: $1 — arg1, $2 — arg2, $3 — arg3. Кстати, $0 — это имя самого скрипта. Распечатка параметров в виде HTML-таблицы может выглядеть следующим образом:

#!/usr/freeware/bin/bash

echo Content-type: text/html

echo

echo '<HTML><HEAD></HEAD><BODY>'

echo '<H1>Аргументы</H1>'

echo '<TABLE BORDER=1>'

echo '<TR><TH>Номер</TH><TH>Значение</TH></TR>'

let i=0

for x in $@

do

let i=i+1

echo '<TR><TD>arg['$i']</TD><TD>'$x'</TD></TR>'

done

echo '</TABLE>'

echo '</BODY></HTML>'

Последовательность команд echo формирует HTTP-сообщение. Команда let позволяет выполнять арифметические вычисления. Перед циклом for производим инициализацию переменной i. Цикл for "пробегает" по всем аргументам командной строки, которые объединены в переменной $@ и разделяются в ней пробелами. Фактически они представляют собой список слов, покоторому и бежит переменная цикла x. Обратите внимание на отличие данного цикла от стандартного цикла for в С или Perl: в нем не используются арифметические операции, а идет работа со списком.

Внутри цикла при помощи команды let мы увеличиваем индекс аргумента командной строки (значение переменной i ) и распечатываем этот индекс и значение переменной x в виде элементов HTML-таблицы.

Если аргументов мало и их местоположение известно, то к каждому из них можно просто обращаться по встроенному имени, например, первый аргумент — это $1.

Стандартный поток ввода

По большому счету, для чтения данных из стандартного потока ввода в рамках программирования CGI-скриптов bash непригоден. Дело в том, что в нем нет механизма посимвольного считывания данных. Bash -скрипт способен читать только строками и останавливает считывание лишь в случае появления в потоке символа конца файла. Как известно, HTTP-сервер такого символа в стандартный поток ввода скрипта при работе по методу POST не передает. Тем не менее чтение стандартного ввода в рамках программирования CGI-скриптов на bash применяется.

Примером тому может служить генерация гипертекстовых ссылок на файлы текущего каталога:

#!/usr/freeware/bin/bash

echo Content-type: text/html

echo

echo '<HTML><HEAD></HEAD><BODY>'

echo '<Ul>'

ls -a | while read x

do

if test -f $x; then

echo '<LI><A HREF=./'$x'>'$x'</A>';

fi

done

echo '</BODY></HTML>'

В данном случае команда ls доставляет в скрипт имена файлов. Один файл — это отдельная строка. Эти имена обрамляются гипертекстовыми ссылками и вставляются в HTML-страницу. При этом печатаются только обычные файлы, все остальные игнорируются.

Другой пример — фильтрация. При приеме по методу GET запрос размещается в переменной QUERY\_STRING. Но он там находится в форме form-urlencoded. Для его фильтрации вызывается внешняя программа, стандартный вывод которой перенаправляется настандартный ввод одной из команд скрипта:

echo $QUERY\_STRING | tr '+' ' ' | while read x

do

for y in $x

do

echo $y

done

done

Существуют и другие способы применения чтения из стандартного ввода при программировании CGI-скриптов на BASH.

Типы данных и переменные

В *bash* существует только два типа данных: скаляры и одномерные массивы. При этом возможно *вычисление* арифметических выражений, результат выполнения которых становится значением скаляра. *По*-другому эти типы можно интерпретировать как текстовые строки и списки.

Существует два типа переменных: встроенные переменные *bash* и переменные, определяемые пользователем (переменные пользователя). Не перечисляя всех встроенных переменных, назовем наиболее употребительные:

$1-$n — аргументы командной строки скрипта;

$0 — имя скрипта;

$@ — список аргументов командной строки;

$# — число аргументов командной строки;

$IFS — список разделителей;

$PATH — путь поиска команд.

*Переменные окружения*, которые генерируются сервером — это переменные пользователя, импортируемые скриптом при его запуске. *Пользователь* внутри скрипта может установить собственные переменные:

IFS="="

В данном случае мы отменили *значение* *по* умолчанию для списка разделителей и назначили в качестве разделителя знак равенства "=". IFS — это глобальная *переменная*, поэтому она передается от скрипта к скрипту *по* умолчанию. Если требуется назначить собственную переменную и передать ее в другой *скрипт*, который вызывается из текущего скрипта, ее нужно будет экспортировать:

bash>QUERY\_STRING=arg1+arg2+arg3;

export QUERY\_STRING

В данном случае в целях отладки скрипта в командной строке *bash* определена *переменная окружения*   *QUERY\_STRING*. Если запустить *скрипт* без предварительного экспорта, то *значение* этой переменной ( $*QUERY\_STRING* ) будет неопределенным.*Команда* export позволяет передать это *значение* в тестируемый *скрипт*.

Управление потоком вычислений

Изо всех возможностей управления порядком выполнения команд в *bash* -скрипте мы рассмотрим только if, while и for. Пользуясь этими встроенными возможностями *bash*, следует иметь в виду, что *логические выражения*, которые применяются в качестве условий данных команд, строятся вокруг строк, а не чисел. Использовать числовое условие в *bash* крайне затруднительно.

if

Команда if имеет вид:

if list; then list; [elif list; then list;]

...[ else list;] fi

Сначала выполняется список команд, который стоит после if. Если он завершился успешно, то выполняется список команд после первого then. Значение и логика выполнения других частей этой команды очевидна. Команда начинается символами "if" и должна закончиться символами "fi". Часть команды в квадратных скобках — это необязательные конструкции, которые при необходимости можно опустить.

Рассмотрим в качестве примера проверку метода доступа к скрипту. Для *bash* это может быть только GET:

#!/usr/freeware/bin/bash

echo Content-type: text/plain

echo

if test $REQUEST\_METHOD = "POST"; then

echo POST;

elif test $REQUEST\_METHOD = "GET"; then

echo GET;

else echo Unknown method $REQUEST\_METHOD;

fi

В данном случае мы используем сравнение строк (символ "="). Если нужно сравнивать арифметические выражения, то следует использовать другие операции сравнения:

-eq — равенство операндов;

-ne — неравенство операндов;

-lt — первый операнд меньше второго;

-le — первый операнд меньше либо равен второму;

-gt — первый операнд больше второго;

-ge — первый операнд больше либо равен второму.

Команда test чрезвычайно полезна при работе с файловой системой. Например, при проверке наличия файла и прав на чтение можно использовать следующую комбинацию:

if test -r file.txt;

then echo file.txt is readable; fi

Помимо проверки наличия файла и прав можно определять тип файла ( -d — каталог, -f — обычный файл и т.п.).

while

Команда while позволяет выполнять список команд до тех пор, пока справедливо условие использования данного списка, которое задается аргументом while. Чаще всего в наших примерах эта команда применяется при фильтрации входного потока:

ps -axj | grep httpd | while read id pid

do

if test $id = "root"; then kill -1 $pid; fi

done

В данном случае в системе FreeBSD просматривается список активных процессов с именем httpd (HTTP-сервера), отыскивается процесс-родитель и перезапускается.

for

Вид команды for в *bash* отличается от обычного; когда в команде инициализируется переменная цикла, происходит проверка условия для переменной цикла и производится изменение ее значения. В *bash* переменная бежит по списку и выполняет цикл до тех пор, пока список не будет исчерпан:

for var; in list; do list; done

Переменная var принимает значения из списка, указанного за in, до тех пор, пока этот список не кончится. При этом для каждого значения var выполняется список команд, заключенный между "do" и "done". Примером использования for может служить разбор входных строк:

ls -ax | while read x

do

for y in $x

do

echo $y

done

done

Считываемая из стандартного ввода строка разбивается на слова, и каждое слово печатается отдельно на новой строке.

Конвейеры

Одним из наиболее полезных свойств командных языков является организация конвейеров и *перенаправление ввода/вывода*. Под конвейером понимают последовательность программ (команд), соединенных друг с другом через ввод/*вывод*. *Стандартный поток вывода* первой программы подается на стандартный *поток* ввода второй программы, *стандартный поток вывода* второй программы, в свою *очередь*, подается на стандартный *поток* ввода третьей и т.д. Выглядит это примерно следующим образом:

ps -ax | grep httpd | wc -l

В данном случае *вывод* статистики об активных процессах подается на ввод фильтра grep, который выводит только те строки, где встречается *программа* httpd. В свою *очередь*, *выход* grep подается на вход счетчика строк (*атрибут* -l ), который в итоге сообщает, сколько в данный момент в системе крутится *HTTP*-серверов. Символы "|" как раз и обозначают конвейеризацию выполнения программ.

Кроме конвейеров, к *стандартному вводу/выводу* можно применять перенаправление. Так, символ "<" означает чтение из стандартного потока ввода, а символ ">" — *вывод* в *файл*. Этот метод стоит использовать при перенаправлении стандартного потока ошибок в *стандартный поток вывода*:

ls hhjhj > intuit 2>&1

В данном случае задана заведомо ошибочная с точки зрения синтаксиса *команда*. *Стандартный вывод* перенаправляется в *файл*"intuit", а стандартный *поток* ошибок перенаправляется в *стандартный вывод* и, следовательно, тоже попадает в *файл* "intuit".

Файлы и каталоги

Работа с файлами и каталогами в *bash* опирается на механизм перенаправления и команды Unix. Проверку состояния файлов удобно выполнять с помощью команды test. Для чтения данных из файла достаточно направить его содержимое в стандартный *поток*ввода:

Bash> intuit.cgi < intuit.txt

Можно для этих же целей построить конвейер:

Bash> cat intuit.txt | intuit.cgi

В данном случае *команда* cat читает данные из файла и передает их на *поток* *стандартного ввода* скрипта.

Для создания файлов в *bash* можно применять функцию копирования из пустого файла в нормальный *файл*:

bash>cp /dev/null intuit.txt

Для прохода *по* дереву каталогов используют команду cd, а получить *список* файлов с их атрибутами можно при помощи командыls:

#!/usr/freeware/bin/bash

echo Content-type: text/html

echo

echo '<HTML><HEAD></HEAD><BODY>'

echo '<UL>'

ls | while read x

do

if test -f $x; then

echo '<LI><A HREF=./'$x'>'$x'</A>'; fi

done

echo '</UL>'

echo '</BODY></HTML>'

В результате выполнения этого скрипта в HTML-документ будет вставлен фрагмент из отчета команды ls. При этом будут отображаться только регулярные (обыкновенные) файлы. Каталоги и специальные файлы в *список* файлов каталога не попадут.

Удалить *файл* можно при помощи команды rm "имя\_файла", переименовать — mv "имя\_старое" "имя\_новое" и т.п. Все это типовые команды Unix.

**Контрольные вопросы:**

1. Структура HTML-документа и элементы разметки заголовка документа
2. Основные контейнеры заголовка HTML-документа
3. Контейнеры тела документа HTML-документа
4. Использование графики в HTML
5. Таблицы в HTML
6. HTML-формы
7. Фреймы в HTML
8. Базовые типы данных в HTML
9. Общая структура документа HTML

## Практическа работа №30

Создание динамического html-документа

Цель: освоить основные методы и приемы создания динамического html-документа

Студент должен:

иметь практический опыт:

осуществления навигации по ресурсам, поиска, ввода и передачи данных с помощью технологий и сервисов Интернета (ПО2);

создания и обработки объектов мультимедиа (ПО3);

уметь:

создавать и редактировать объекты мультимедиа (У2);

знать:

назначение, разновидности и функциональные возможности программ для создания объектов мультимедиа (З2).

Динамический HTML

Данная глава предназначена для того, чтобы познакомить вас с применением сценариев для создания динамических HTML?документов на примере языка программирования JavaScript.

Основы использования сценариев в HTML-документе

Помещение сценария в документ

Для помещения сценария в документ достаточно использовать HTML?элемент SCRIPT. Этот элемент задается парными тегами <SCRIPT> и </SCRIPT> и имеет следующие атрибуты:

• src – URI файла, в котором записан код внешнего сценария;

• type – задает тип содержимого элемента SCRIPT или файла, определенного атрибутом src, принимает значения вида text/язык\_сценария (например, text/javascript, text/vbscript);

• language – задает язык сценария (например, javascript или vbscript); при использовании атрибута type этот атрибут излишен.

Элемент SCRIPT может появляться как в заголовке, так и в теле документа произвольное количество раз. Рассмотрим пример внедрения в документ простейшего сценария на JavaScript (язык рассмотрим чуть позже) (пример 13.1).

Пример 13.1. Сценарий в HTML-документе

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Frameset//EN»

"http://www.w3.org/TR/html4/frameset.dtd">

<HTML>

<HEAD>

<TITLE>Простейшая страница со сценарием</TITLE>

</HEAD>

<BODY>

<SCRIPT type = "text/javascript">

var date = new Date;

if (date.getHours() < 8 || date.getHours() > 22)

document.write("<P align = right><FONT color = black><I>Не спится?</I></FONT></P>");

else if (date.getHours() < 11)

document.write ("<P align = right><FONT color = blue><I>Доброе утро </I></FONT></P>");

else if (date.getHours() < 17)

document.write ("<P align = right><FONT color = red><I>Добрый день</I></FONT></P>");

else document.write ("<P align = right><FONT color = green><I>Добрый вечер</I></FONT></P>");

</SCRIPT>

<H1>Страница, содержащая сценарий</H1>

<P>Прочий текст страницы...

</HTML>

Участки кода между <SCRIPT> и </SCRIPT>, начинающиеся с document.write(), должны быть записаны в одну строку, иначе пример работать не будет. Что же делает сценарий в приведенном примере? Да ничего особенного, просто при загрузке страницы печатает в правом верхнем углу приветствие, цвет и текст которого зависят от времени суток.

Стоит сказать несколько слов о еще одном HTML?элементе, имеющем отношение к сценариям, – это NOSCRIPT. Этот HTML?элемент задается парными тегами <NOSCRIPT> и </NOSCRIPT> и полезен, когда автор документа хочет подстраховаться на случай, если его документ будет открыт в браузере, не поддерживающем сценарии вообще или не поддерживающем сценарии на используемом языке программирования. Итак, если сценарий не может быть выполнен, то браузер отобразит содержимое элемента NOSCRIPT (в нем может быть, например, гиперссылка на версию документа, не использующую сценарии).

Скрытие сценария

Рассмотрим прием, который часто применяется для того, чтобы браузер, вообще «не знающий» HTML?элемента SCRIPT, не показал пользователю текст сценария. Это может случиться, если сценарий помещен в тело документа.

Для предотвращения возникновения такой ситуации текст сценария помещают в HTML?комментарий следующим образом:

<SCRIPT type = «text/javascript»>

<!–

//Текст программы, помещенной здесь, пользователь не увидит случайно

–>

</SCRIPT>

Браузеры, поддерживающие элемент SCRIPT, должны проигнорировать HTML?комментарий внутри этого элемента (хотя некоторые версии браузера Netscape Navigator не воспринимали сценарий, заключенный в комментарий HTML).

Другим вариантом скрытия, который должен точно работать, является помещение сценария во внешнем файле и подключение его при помощи атрибута src элемента SCRIPT.

13.2. Исполнение сценария

Теперь рассмотрим, как организовать выполнение написанных и внедренных в документ сценариев. Итак, сценарий может исполняться в двух случаях: при загрузке документа и при возникновении события (а также при вызове его интерпретатором, например, по таймеру, но это рассмотрим особо).

Исполнение при загрузке документа

Интерпретатор браузера выполняет операторы языка JavaScript, записанные вне тела какой?либо функции, только один раз по мере загрузке документа. Причем положение HTML?элемента SCRIPT с текстом сценария определяет момент времени, в который сценарий будет выполняться. Так, сценарий в примере 13.1 выполнялся именно до того, как было загружено остальное содержимое документа, поэтому выведенный им текст и появился раньше основного содержимого документа.

Чтобы при загрузке HTML?документа выполнялась какая?либо функция, в нужном месте сценария должен быть записан вызов функции. Так, пример 13.1 можно переписать следующим образом (пример 13.2 сокращен).

Пример 13.2. Вызов функции при загрузке документа

...

<BODY>

<SCRIPT type = "text/javascript">

function greeting(){

//Те же действия, что и в примере 13.1...

}

//Вызов функции greeting();

</SCRIPT>

<H1>Страница, содержащая сценарий</H1>

<P>Прочий текст страницы...

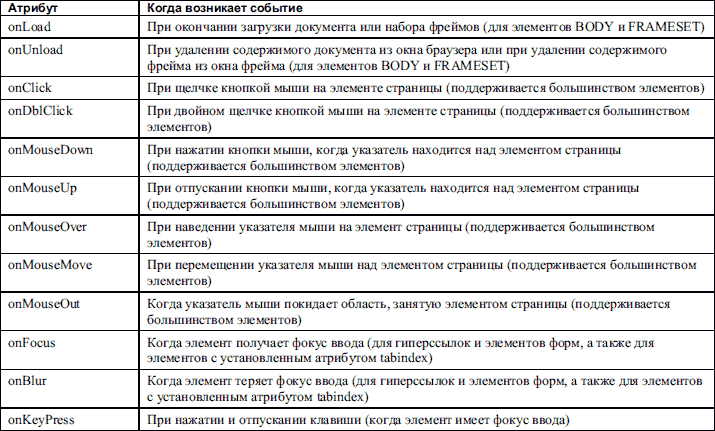
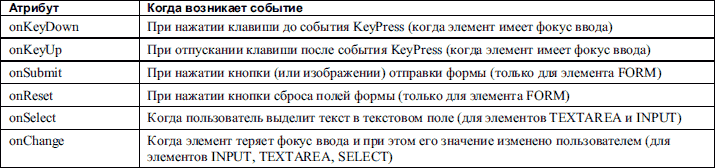
</HTML>

Если бы в коде сценария не было явного вызова функции greeting(), то на странице не появилось бы приветствие.

Реакция на события

Сценарий может быть также запущен браузером при возникновении на странице какого?либо события, с которым сопоставлен сценарий, – обработчика события. Генерацию событий могут вызывать различные действия пользователя: щелчок кнопкой мыши на элементе страницы, наведение указателя мыши на элемент и др.

Для назначения обработчиков события используются атрибуты HTML?элементов, приведенные в табл. 13.1.

Таблица 13.1. Атрибуты для назначения обработчиков событий  


Значениями приведенных в таблице атрибутов могут быть фрагменты кода сценариев, например:

<P onClick = «alert('Не давите на меня!!!')»>Текст абзаца

<P onClick = "

{

//Аккуратно оформленный блок кода, ведь строки HTML-разметки можно

//безнаказанно разрывать

alert('Лучше нажимайте на соседний абзац.');

}">Текст абзаца

Обратите внимание, что поскольку текст обработчика помещается в двойные кавычки, то сами кавычки в тексте сценария использовать не следует. Обычной практикой является создание функций?обработчиков события (обычной функции JavaScript) вместо записи действий по обработке события прямо в теге элемента. В таком случае в атрибут onСобытие записывается код вызова функции?обработчика.

События, возникающие в дочерних элементах, передаются вверх по иерархии родительским элементам. Так, например, если над текстом элемента B в приведенном ниже примере произойдет щелчок кнопкой мыши, то событие получит сначала элемент B, потом элемент P, а затем и элемент BODY:

<BODY onClick = «body\_click()»>

<P onClick = "p\_click()">Обычный текст

<B onClick = "b\_click()">полужирный текст</B>

13.3. Объектная модель документа

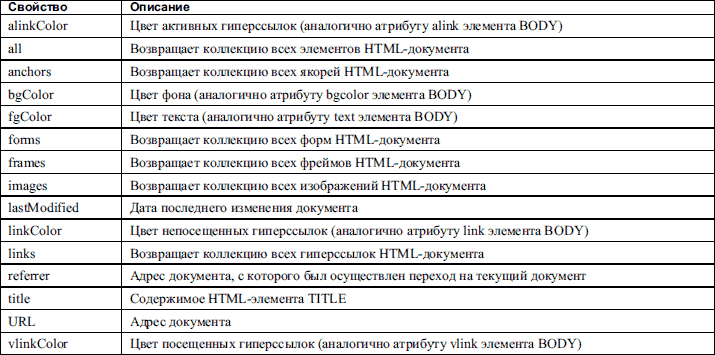
Чтобы можно было успешно применить полученные значения по программированию на JavaScript, нужно рассмотреть еще один специфический момент – это то, как сценарии могут воздействовать на HTML?документ. Для этого необходимо изучить технологию представления HTML?документа в виде совокупности объектов – объектную модель документа (DOM, Document Object Model). DOM включает не только объекты, из которых состоит документ, но и объекты, позволяющие получать различную информацию о браузере, системе (в частности, о видеосистеме компьютера), работать с окнами и многое другое.

Объект document

Для программиста на JavaScript HTML?документ представляется в виде объекта document. Этот глобальный объект существует в единичном экземпляре. О создании объекта document заботится интерпретатор.

Свойства и методы объекта document

Итак, объект document предоставляет ряд свойств и методов, позволяющих осуществлять практически любые манипуляции с HTML?документом. Основные свойства объекта document приведены в табл. 13.2.

Таблица 13.2. Свойства объекта document

Свойства, отвечающие за цветовое оформление, хранят целочисленные значения. Другие свойства, кроме возвращающих коллекции, хранят строки. Особо следует рассмотреть свойства, которые возвращают коллекции: all, anchors, forms, frames, images и links. В этих коллекциях находятся объекты, описывающие соответствующие элементы HTML?документа.

Коллекции – это тоже объекты, во многом похожие на массивы. Однако коллекции предоставляют доступ к своим элементам c помощью метода item(). Причем доступ может осуществляться как по номеру элемента в коллекции, так и по его имени (задается атрибутом id или name элемента). Нумерация элементов в коллекциях начинается с нуля. Кроме того, для коллекций предусмотрено свойство length, хранящее количество элементов в коллекции.

В качестве примера рассмотрим, как получить доступ к элементам следующего HTML?документа (пример 13.3).

Пример 13.3. HTML-документ, к элементам которого нужно получить доступ

<HTML>

<HEAD>

<TITLE>Заголовок документа</TITLE>

</HEAD>

<BODY>

<H1 id = "main\_part">Главный заголовок документа</H1>

<P id = "par1">Текст документа...

</BODY>

</HTML>

Если осуществлять доступ к элементам документа при помощи коллекции all с использованием номеров, то доступ ко всем элементам документа будет выглядеть следующим образом:

var elements = []; //Массив, в который скопируем ссылки на объекты страницы

var i;

for (i=0; i<document.all.length; i++)

elements[i] = document.all(i);

В примере 13.3 в коллекции all содержится шесть элементов, поэтому массив elements после выполнения приведенного фрагмента программы должен содержать шесть элементов. Обратите внимание, как осуществляется доступ к элементам коллекции all: имя метода item() можно опускать.

Если бы доступ к элементам коллекции осуществлялся по имени HTML?элементов, то можно было бы получить объекты, описывающие заголовок и абзац, следующим образом:

var h = document.all(«main\_part»);

var p = document.all("par1");

Возможно также прямое обращение к поименованным элементам документа. При использовании этого способа предыдущий фрагмент программы будет выглядеть следующим образом:

var h = main\_part;

var p = par1;

Кроме достаточно богатого набора свойств, можно пользоваться методом write() объекта document для добавления к HTML?документу любого текста прямо из сценария. Метод write() принимает строку, в которой может содержаться любое HTML?форматирование, например:

document.write('<H1 id = «part2»>Текст, напечатанный сценарием</H1>')

При выполнении этого фрагмента программы не только появляется текст заголовка в окне браузера, но и создается объект с именем part2, к которому можно получить доступ с использованием той же коллекции all.

Использование методов open(URI\_документа) и close() объекта document позволяет открывать новые HTML?документы и закрывать окно браузера с текущим документом, например:

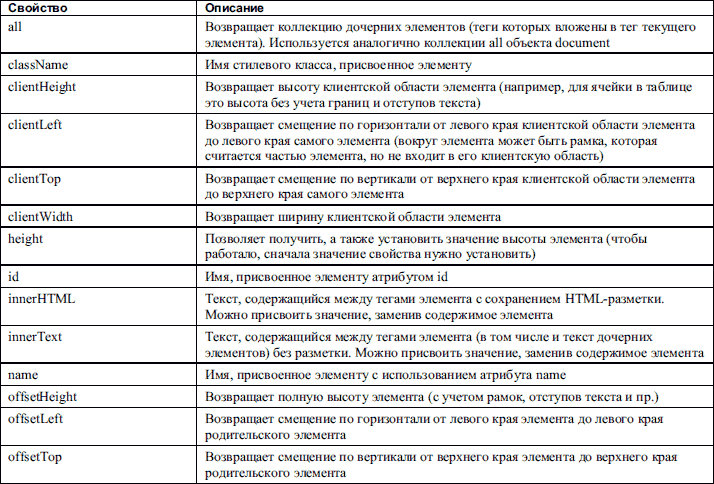
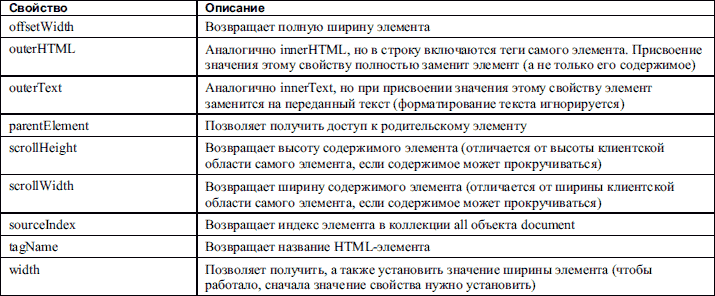
open(«13.1.html») //Открываем пример 13.1 в новом окне

close(); //Пытаемся закрыть текущий документ

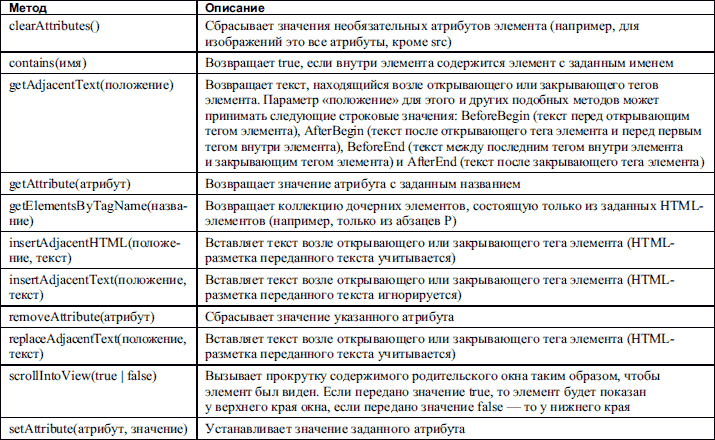
Управление элементами документа

Итак, выше рассмотрено, как можно получить доступ к объектам, описывающим HTML?элементы документа. Теперь же рассмотрим, какие общие действия можно производить с полученными объектами.

Свойства, которые доступны для большинства элементов документа, приведены в табл. 13.3.

Таблица 13.3. Основные свойства элементов документа  


В табл. 13.4 приводятся основные методы, которые можно использовать для манипулирования большинством элементов документа.

Таблица 13.4. Основные методы элементов документа

С использованием приведенных в таблицах свойств и методов можно осуществлять практически любые манипуляции с документом, показываемым в окне браузера. Сейчас дополнительно будут рассмотрены некоторые особенности доступа к элементам таблиц: доступ к строкам и ячейкам.

Объекты, описывающие таблицы, поддерживают внутреннюю коллекцию rows, с помощью которой организовывается доступ к отдельным строкам таблицы. Эти объекты поддерживают также методы insertRow(номер) и deleteRow(номер), которые принимают в качестве параметра номер строки и позволяют вставить или удалить строку таблицы. Если метод insertRow() вызвать без параметра, то строка будет добавлена в конец таблицы. При успешном добавлении строки метод insertRow возвращает ссылку на объект, описывающий созданную строку.

Каждым элементом коллекции rows является объект, одним из свойств которого является коллекция cells, содержащая объекты, управляющие ячейками таблицы. Каждый элемент коллекции cells позволяет оперировать конкретными ячейками таблицы. Добавление ячеек в строку таблицы можно осуществлять при помощи метода insertCell(номер) объектов коллекции rows. Удалять же ячейки можно, используя метод deleteCell(номер) объектов той же коллекции.

Наконец, как завершающий этап знакомства с объектом document создадим несколько страниц, использующих его возможности.

Страница следующего примера будет представлять собой своеобразный каталог изображений. Изображения вместе с текстом, описывающим их, будут помещены в таблице. При этом помещение и удаление данных должно выполняться интерактивно (то есть этим управляет пользователь). Внешний вид страницы примера приведен на рис. 13.1.

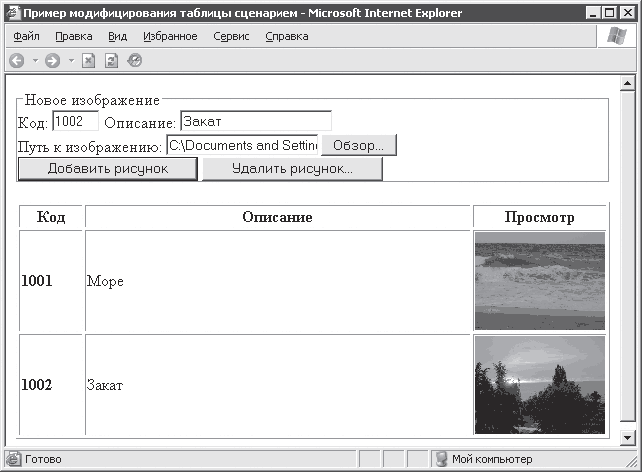


Рис. 13.1. Изменяемая страница (каталог изображений)

При реализации примера код сценария и HTML?код страницы расположены в отдельных файлах. Файл документа может иметь произвольное имя. Он выглядит следующим образом (пример 13.4).

Пример 13.4. Файл HTML-документа

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»>

<HTML>

<HEAD>

<TITLE>Пример модифицирования таблицы сценарием</TITLE>

<SCRIPT type = "text/javascript" src = "script\_13\_4.js"></SCRIPT>

</HEAD>

<BODY>

<FIELDSET>

<LEGEND>Новое изображение</LEGEND>

Код: <INPUT name = "txtCode" maxlength = "4" size = "5">

Описание: <INPUT name = "txtName" maxlength = "500">

<BR>Путь к изображению: <INPUT type = "file" name = "txtFile">

<BR><INPUT type = "button" value = "Добавить рисунок"

onClick = "add\_image(txtCode.value, txtName.value, txtFile.value)">

<INPUT type = "button" value = "Удалить рисунок..."

onClick = "delete\_image()">

</FIELDSET>

<P>

<TABLE width = "100%" border>

<!–Определения столбцов таблицы–>

<COL width = "60">

<COL>

<COL width = "130">

<THEAD>

<TR><TH>Код<TH>Описание<TH>Просмотр

</THEAD>

<TBODY id = "mytable">

<!–Сюда сценарий вставляет записи–>

</TBODY>

</TABLE>

</BODY>

</HTML>

В приведенном примере обратите внимание на элемент TBODY таблицы. Присвоив ему имя, вы можете работать с телом таблицы как с самостоятельной таблицей. Это же справедливо для THEAD и TFOOT. Обратите также внимание на получение значений, введенных в текстовые поля: в большинстве случаев можно пользоваться не методами getAttribute() и setAttribute(), а свойствами, которые имеют такие же названия, как и соответствующие атрибуты.

Теперь очередь файла с кодом сценария. В примере он имеет имя script\_13\_4.js (JS – стандартное расширение для файлов со сценариями на JavaScript) (пример 13.5).

Пример 13.5. Файл script\_13\_4.js

//Функция принимает код, название, путь изображения и добавляет

//запись в таблицу function add\_image(code, imagename, path){

if (valid\_data(code, imagename, path)){

//Формирование строки таблицы (в части TBODY)

var row = mytable.insertRow();

//Ячейка с кодом рисунка row.insertCell().innerHTML = "<B>" + code + "</B>";

//Ячейка с названием row.insertCell().innerHTML = imagename;

//Ячейка с изображением row.insertCell().innerHTML = '<IMG width = "130" src = "'+ path +'">';

}

}

//Функция проверяет правильность введеных данных function valid\_data(code, imagename, path){

//Проверка, введены ли все значения if (code == "" || imagename == "" || path == ""){

alert("Введите значения во все поля");

return false;

}

else{

//Проверим, чтобы код изображения не дублировался var i;

for (i=0; i<mytable.rows.length; i++){

if (mytable.rows(i).cells(0).innerText == code){

alert("Изображение с кодом "+ code +" уже присутствует в таблице");

return false;

}

}

}

return true;

}

//Функция удаления записи из таблицы (код изображения

//вводит пользователь)

function delete\_image(){

var code = prompt("Введите код удаляемого изображения", "");

if (code != null){

//Находим и удаляем запись var i;

for (i=0; i<mytable.rows.length; i++){

if (mytable.rows(i).cells(0).innerText == code){

mytable.deleteRow(i);

return true;

}

}

alert("Изображение с кодом " + code + " не найдено.");

}

}

Как видно, в файле script\_13\_4.js реализованы три функции. Первая функция add\_image() используется для добавления записей в таблицу. Перед добавлением каждой новой записи она проверяет (с помощью функции valid\_data()), чтобы были введены все данные (код, описание и путь изображения), а также, чтобы код нового изображения не дублировался кодом одного из изображений, ранее добавленных в таблицу. Третья функция delete\_image() используется для удаления записи из таблицы.

Из примера 13.5 можно увидеть применение коллекций rows и cells таблицы на практике.

Помещаем свое меню на страницу

Рассмотрим еще один довольно любопытный пример, позволяющий разнообразить оформление страницы. Используя таблицы, CSS и простые сценарии, создадим свое красочное меню. В пункты этого меню можно вставлять маленькие изображения. Пункты будут подсвечиваться при наведении на них указателя мыши. Внешний вид меню представлен на рис. 13.2.

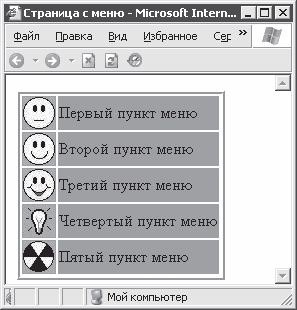


Рис. 13.2. Внешний вид меню

Ниже приводится текст HTML?документа с созданным меню (пример 13.6).

Пример 13.6. Документ с меню

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Frameset//EN»>

<HTML>

<HEAD>

<TITLE>Страница с меню</TITLE>

<STYLE type = "text/css">

.item {background-color: rgb(170, 170, 170)}

.selected {background-color: magenta}

.menu {border-style: ridge}

</STYLE>

<SCRIPT src = "menu.js" type = "text/javascript"></SCRIPT>

</HEAD>

<BODY>

<TABLE id = "menu1" class = "menu">

<!–Первый пункт меню–>

<TR id = "item1" class = "item" onClick = "item1\_click()"

onMouseOver = "item1.className = 'selected'"

onMouseOut = "item1.className = 'item'">

<TD><IMG src = "icons/2.jpg"><TD>Первый пункт меню

<!–Второй пункт меню–>

<TR id = "item2" class = "item" onClick = "item2\_click()"

onMouseOver = "item2.className = 'selected'"

onMouseOut = "item2.className = 'item'">

<TD><IMG src = "icons/2.jpg"><TD>Второй пункт меню

<!–Третий пункт меню–>

<TR id = "item3" class = "item" onClick = "item3\_click()"

onMouseOver = "item3.className = 'selected'"

onMouseOut = "item3.className = 'item'">

<TD><IMG src = "icons/3.jpg"><TD>Третий пункт меню

<!–Четвертый пункт меню–>

<TR id = "item4" class = "item" onClick = "item4\_click()"

onMouseOver = "item4.className = 'selected'"

onMouseOut = "item4.className = 'item'">

<TD><IMG src = "icons/4.jpg"><TD>Четвертый пункт меню

<!–Пятый пункт меню–>

<TR id = "item5" class = "item" onClick = "item5\_click()"

onMouseOver = "item5.className = 'selected'"

onMouseOut = "item5.className = 'item'">

<TD><IMG src = "icons/5.jpg"><TD>Пятый пункт меню

</TABLE>

</BODY>

</HTML>

Из приведенного текста можно увидеть, каким образом используется таблица: пунктами меню являются строки таблицы. Чтобы строки таблицы подсвечивались при наведении указателя мыши, их стилевой класс динамически изменяется при обработке событий onMouseOver, onMouseOut. Изменив определения стилевых классов item, selected, menu, можно легко добиться нужного вида меню.

При выборе каждого из пунктов меню вызывается соответствующая функция?обработчик (см. значения атрибутов onClick для элементов TR). Все функции?обработчики собраны в файле menu.js, текст которого приводится ниже (пример 13.7).

Пример 13.7. Содержимое файла menu.js

/\*

В этом файле содержатся функции-обработчики для каждого пункта меню

\*/

function item1\_click(){

alert("Вы выбрали первый пункт меню");

//Другие действия...

}

function item2\_click(){

alert("Вы выбрали второй пункт меню");

//Другие действия...

}

function item3\_click(){

alert("Вы выбрали третий пункт меню");

//Другие действия...

}

function item4\_click(){

alert("Вы выбрали четвертый пункт меню");

//Другие действия...

}

function item5\_click(){

alert("Вы выбрали пятый пункт меню");

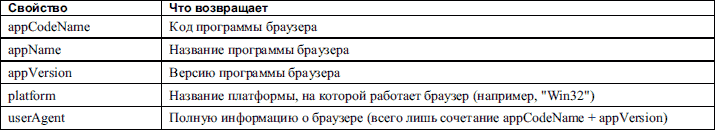
//Другие действия...

}

В каждую их приведенных выше функций помещен только код, сообщающий о работоспособности отдельного пункта меню.

Объект navigator

Глобальный объект navigator позволяет получить некоторую информацию о браузере, в котором происходит просмотр страницы со сценарием. Свойства объекта navigator, поддерживаемые большинством браузеров (по крайней мере, не только браузером Internet Explorer), приведены в табл. 13.5.

Таблица 13.5. Свойства объекта navigator

Часто использовать объект navigator нет необходимости, однако он может очень пригодиться при создании достаточно продвинутых и «живучих» веб?страниц, способных выбирать сценарии для выполнения в зависимости от браузера, в котором они открываются. Простейший код, позволяющий отличить браузер Internet Explorer, приведен ниже (пример 13.8).

Пример 13.8. Определение браузера

function do\_script(){

if (navigator.appName == "Microsoft Internet Explorer"){

//Код для Internet Explorer...

}

else{

//Код для другого браузера...

}

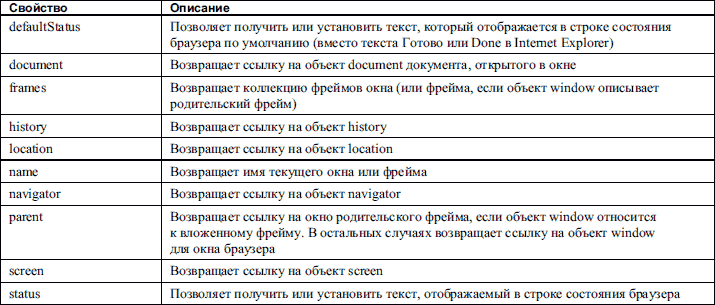
}

Объект window

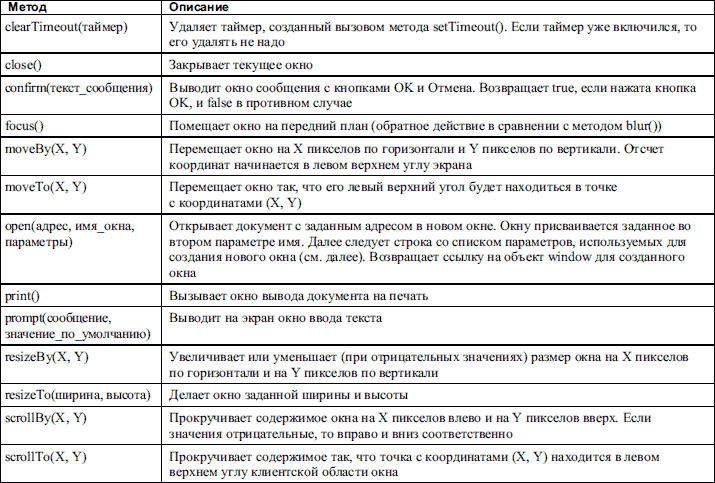
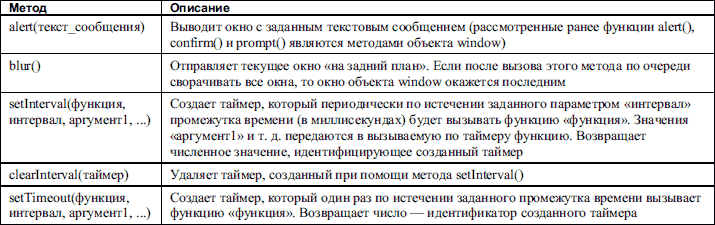
Глобальный объект window предоставляет возможности по манипулированию окном браузера или окном фрейма, в котором открыт документ со сценарием. Кроме того, при помощи объекта window можно открывать новые окна, манипулировать фреймами, создавать таймеры (что очень нужно для анимации) и делать еще много полезного.

Свойства и методы объекта window

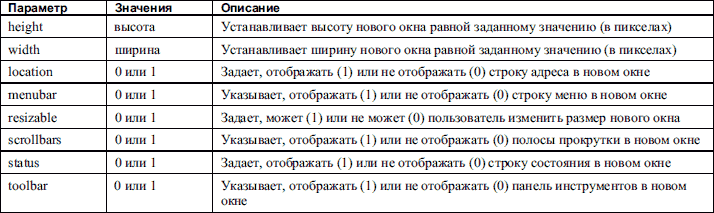
Основные свойства объекта window приводятся в табл. 13.6.

Таблица 13.6. Свойства объекта window

В табл. 13.7 приведены основные методы объекта window.

Таблица 13.7. Методы объекта window

Как было сказано при описании метода open() в табл. 13.7, для этого метода предусмотрено несколько дополнительных параметров. Основные параметры перечислены в табл. 13.8.

Таблица 13.8. Параметры метода open()

Каждый из приведенных в таблице параметров может добавляться в строку параметры в виде: имя\_параметра = значение. Так, для открытия документа в новом окне размером 300 ? 400 можно использовать следующий вызов метода open():

window.open(«13.6.html», "", «width = 300, height = 400»);

Вообще, по крайней мере в браузере Internet Explorer, разделителем параметров в строке необязательно может быть запятая. Этот браузер нормально воспринимает в качестве разделителя и пробел, и точку с запятой. Еще при испытаниях метода open() в Internet Explorer замечена следующая особенность: если в строке задан хотя бы один параметр, то значения всех остальных параметров, принимающих значения 0 или 1, сбрасываются в 0. Так, созданное приведенным выше вызовом метода open() окно будет отображаться без строки состояния, панели инструментов, строки меню, полос прокрутки, строки адреса и будет неизменяемого размера.

Как можно было заметить, в табл. 13.8 приведены параметры, позволяющие задать ширину нового окна, но не приведены параметры, задающие положение окна. Эти параметры в действительности есть, но они отличаются для различных браузеров. Для Internet Explorer это left и top, а для Navigator – screenX и screenY.

Примеры использования объекта window

Теперь рассмотрим, как можно использовать объект window для воспроизведения анимации в окне браузера.

Методы, позволяющие создавать таймеры, просто незаменимы при работе с анимацией средствами браузера. Суть данного примера состоит в последовательной загрузке изображений в элемент IMG. Интервал между сменами кадров выдерживается с помощью таймера. Всего кадров шесть (рис. 13.3).

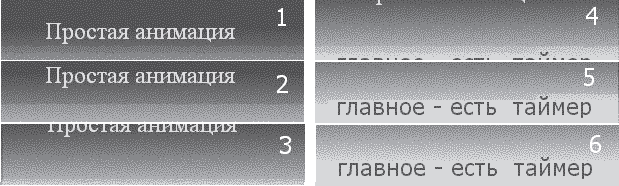


Рис. 13.3. Изображения-кадры

Кадры меняются от первого до шестого, а затем от шестого до первого. Текст HTML?документа со сценарием, реализующим смену кадров, приведен ниже (пример 13.9).

Пример 13.9. Анимация на странице

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Frameset//EN»

"http://www.w3.org/TR/html4/frameset.dtd">

<HTML>

<HEAD>

<TITLE>Страница с анимацией</TITLE>

</HEAD>

<BODY>

<SCRIPT type = "text/javascript">

//Назначаем функцию, вызываемую по таймеру и меняющую

//изображения window.setInterval(new\_frame, 300);

var inc = 1;

var curFrame = 1;

var maxFrame = 6;

//Функия смены кадров function new\_frame(){

//Покажем текущий кадр animate.src = "frames/" + curFrame + ".gif";

//Переход на следующий кадр curFrame += inc;

if (curFrame > maxFrame){

//Начинаем воспроизведение в обратном порядке curFrame = maxFrame;

inc = –1;

}

else if (curFrame == 0){

//Начинаем воспроизведение в прямом порядке curFrame = 1;

inc = 1;

}

}

</SCRIPT>

<IMG id = "animate" src = "frames/1.gif">

</BODY>

</HTML>

В коде сценария количество кадров задается в переменной maxFrames. Предполагается, что кадры помещаются в папке frames и имеют имена вида номер.gif.

Создание всплывающих окон

Иногда бывает удобно использовать дополнительные так называемые всплывающие окна, например, чтобы открывать в них список файлов для закачки, если речь идет о каком?то веб?архиве. Как вы уже догадались, в этом примере для открытия новых окон используется метод open() объекта window. Кроме открытия нового окна, в приведенном ниже примере 13.10 реализовано также его закрытие через 5 секунд.

Пример 13.10. Создание и закрытие всплывающего окна

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Frameset//EN»

"http://www.w3.org/TR/html4/frameset.dtd">

<HTML>

<HEAD>

<TITLE>Всплывающие окна</TITLE>

</HEAD>

<BODY>

<SCRIPT type = "text/javascript">

//Функция открывает окно\

function open\_window(){

wnd = window.open("13.9.html", "asd", "height = 200, width = 350");

//Функция закрытия окна вызывается через 5 секунд window.setTimeout(wnd.close, 5000);

}

</SCRIPT>

<P>Щелкните

<INPUT type = "button" value = "кнопку" onClick = "open\_window()">

чтобы открыть новое окно на 5 секунд.

</BODY>

</HTML>

Помните, что использовать всплывающие окна следует оправданно. Чаще всего пользователя очень раздражают появляющиеся неожиданно окна, например с какой?нибудь рекламой.

Объект style

Объект style, который для большинства элементов страницы поддерживается как свойство, предоставляет большие возможности по манипулированию стилем элементов. Эти возможности такие же, как и доступные при использовании таблиц стилей: можно получать и указывать значения тех же свойств, что и с использованием CSS. Сначала рассмотрим, как формируются имена свойств объекта style.

Свойства объекта style

Итак, имена свойств CSS, состоящие из одного слова, в таком же виде и используются как имена свойств объекта style, например:

el.style.width = 100;

el.style.color = "red";

Здесь el предоставляет доступ к элементу страницы со значением атрибута id, равным «el».

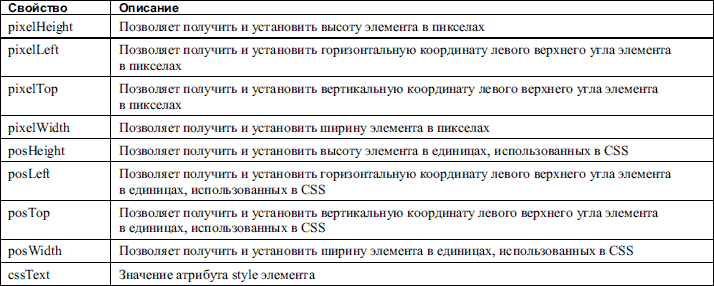
В приложении 2, где приведен список основных свойств CSS, можно увидеть, что названия многих свойств состоят из нескольких слов, разделенных символом –. Так вот, имена таких свойств CSS преобразуются в имена свойств объекта style следующим образом: первое слово имени записывается cо строчной буквы, остальные слова начинаются с прописной буквы, все символы – из имени свойства удаляются. Ниже приведен пример для того же элемента el:

el.style.borderStyle = «solid»; //свойство border-style

el.style.borderColor = "blue"; //свойство border-color

el.style.borderBottomWidth = "10mm"; //свойство border-bottom-width

Как видно, значения свойств могут быть в тех же единицах измерения, которые применяются для CSS. Однако это хорошо до тех пор, пока не нужно производить вычисления с использованием текущих параметров элемента. Так, в приведенном выше примере el.style.borderBottomWidth вернет строковое значение «10mm». Удобно ли производить вычисления с такими значениями? В табл. 13.9 приведен перечень дополнительных свойств, поддерживаемых только интерпретатором браузера Internet Explorer, но значительно облегчающих программирование таких вещей, как перемещение элементов страницы.

Таблица 13.9. Дополнительные свойства объекта style

Примечание

При использовании описанных в этом разделе свойств объекта style следует учитывать одну неприятную особенность: до того, как сценарий может получить значение свойства, это значение должно быть установлено также с использованием сценария.

Примеры использования объекта style

Теперь рассмотрим три примера, иллюстрирующих возможности элемента style. Первые два из них связаны с анимацией, а в последнем примере приводится реализация усовершенствованного меню на основе таблицы.

В первом примере реализуется перемещение изображений«шариков» внутри рамки (элемент DIV). Шарики имеют случайные первоначальные скорости и направления движения. Кроме того, при достижении стенок (рамки элемента DIV) они упруго он них отталкиваются. Страница примера выглядит так, как показано на рис. 13.4.

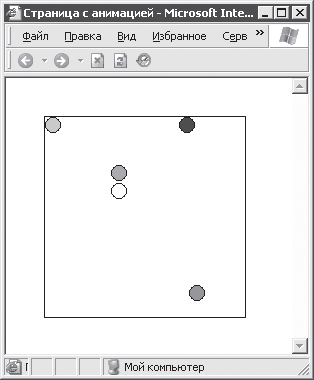


Рис. 13.4. Перемещение «шариков» внутри элемента DIV

Ниже приведен текст HTML?документа, который показан на рис. 13.4 (пример 13.11).

Пример 13.11. Перемещающиеся элементы

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Frameset//EN»

"http://www.w3.org/TR/html4/frameset.dtd">

<HTML>

<HEAD>

<TITLE>Страница с анимацией</TITLE>

<STYLE type = "text/css">

.ball {position: absolute}

.ballarea {border-style: solid; border-width:1px;

position: absolute; background-color: white}

</STYLE>

</HEAD>

<BODY>

<DIV class = "ballarea" id = "area">

<!–Изображения-шарики–>

<IMG src = "balls/ball1.bmp" class = "ball" id = "ball1">

<IMG src = "balls/ball2.bmp" class = "ball" id = "ball2">

<IMG src = "balls/ball3.bmp" class = "ball" id = "ball3">

<IMG src = "balls/ball4.bmp" class = "ball" id = "ball4">

<IMG src = "balls/ball5.bmp" class = "ball" id = "ball5">

<SCRIPT type = "text/javascript">

//Позиционируем элемент DIV

area.style.left = area.style.top = "10mm";

area.style.width = area.style.height = 200;

//Установим размер изображений-шариков ball1.style.pixelWidth = ball1.style.pixelHeight = 16;

ball2.style.pixelWidth = ball2.style.pixelHeight = 16;

ball3.style.pixelWidth = ball3.style.pixelHeight = 16;

ball4.style.pixelWidth = ball4.style.pixelHeight = 16;

ball5.style.pixelWidth = ball5.style.pixelHeight = 16;

</SCRIPT>

</DIV>

<SCRIPT type = "text/javascript" src = "balls.js"></SCRIPT>

</BODY>

</HTML>

Здесь предполагается использование пяти изображений из папки balls. Как видно из примера, чтобы сценарий нормально работал с координатами и размером изображений, соответствующие свойства пришлось установить не в CSS, а опять же с помощью сценария.

Во внешний файл вынесен сценарий, отвечающий за перемещение изображений (файл balls.js). Cодержимое файла balls.js выглядит следующим образом (пример 13.12).

Пример 13.12. Содержимое файла balls.js

//Массив со ссылками на объекты-изображения шариков var balls = [ball1, ball2, ball3, ball4, ball5];

//Массивы скоростей по горизонтали и вертикали (от –10 до 10)

var xSpeed = [rand(–10,10), rand(–10,10), rand(–10,10),

rand(–10,10), rand(–10,10)];

var ySpeed = [rand(–10,10), rand(–10,10), rand(–10,10),

rand(–10,10), rand(–10,10)];

//Минимальные значения координат изображений var minX = ball1.style.pixelLeft;

var minY = ball1.style.pixelTop;

//Максимальные значения координат изображений var maxX = area.style.pixelWidth + minX – balls[0].style.pixelWidth;

var maxY = area.style.pixelHeight + minY – balls[0].style.pixelHeight;

//Назначаем функцию обновления изображения, вызываемую по таймеру window.setInterval(redraw, 100);

//Функция генерации случайных значений function rand(min, max){

return Math.random()\*(max–min)+min;

}

//Функция обновления координат и перерисовки изображений function redraw(){

var i, newX, newY;

for (i=0; i<balls.length; i++){

//Вычисляем новое положение шарика newX = balls[i].style.pixelLeft + xSpeed[i];

newY = balls[i].style.pixelTop + ySpeed[i];

//Проверка столкновения с границами if (newX > maxX){

newX = maxX;

xSpeed[i] = –xSpeed[i];

}

else if (newX < minX){

newX = minX;

xSpeed[i] = –xSpeed[i];

}

if (newY > maxY){

newY = maxY;

ySpeed[i] = –ySpeed[i];

}

else if (newY < minY){

newY = minY;

ySpeed[i] = –ySpeed[i];

}

//Наконец, перемещаем изображение balls[i].style.pixelLeft = newX;

balls[i].style.pixelTop = newY;

}

}

В следующем примере рассматривается сценарий, позволяющий случайным образом изменять цвет текста, для которого задан определенный стилевой класс. Стилевой класс в примере имеет название colored. Пример разбит на две части: собственно сценарий (файл coloredtext.js) и HTML?документ, использующий возможности этого сценария.

Для начала рассмотрим сам сценарий, код которого приведен ниже (пример 13.13).

Пример 13.13. Содержимое файла coloredtext.js

//Поиск всех элементов, имеющих класс «colored»

var i, j = 0;

var elements = [];

for (i=0; i<document.all.length; i++){

if (document.all(i).className == "colored"){

elements[j] = document.all(i);

j++;

}

}

//RGB-составляющие цвета текста var R = 0, G = 0, B = 0; //По умолчанию цвет черный

//Назначаем функцию перерисовки window.setInterval(redraw, 50);

//Функция генерации случайных значений function rand(min, max){

return Math.random()\*(max–min)+min;

}

function redraw(){

//Вычисляем новые значения составляющих цвета

R += rand(–8, 8);

R = (R>=0) ? R : 0;

R = (R<=255) ? R : 255;

G += rand(–8, 8);

G = (G>=0) ? G : 0;

G = (G<=255) ? G : 255;

B += rand(–8, 8);

B = (B>=0) ? B : 0;

B = (B<=255) ? B : 255;

//Установка нового цвета для всех элементов for (i=0; i<elements.length; i++){

elements[i].style.color =

"rgb(" + R + "," + G + "," + B + ")";

}

}

Этот сценарий работает следующим образом. После загрузки содержимого документа (для этого файл должен подключаться к документу перед закрывающим тегом </BODY>) производится просмотр всей коллекции all документа, и в массив elements помещаются ссылки на все элементы, имеющие стилевой класс colored. Кроме обозначения элементов, цвет которых нужно изменять, этот стилевой класс больше ни для чего не используется.

Изменение цвета в примере осуществляется по таймеру. Значения RGB?составляющих цвета случайным образом изменяются на небольшие значения, что обеспечивает эффект этакого плавного перехода цвета. Еще следует отметить, что цвет всех отобранных элементов всегда одинаков. Это сделано для упрощения программы.

В примере 13.14 показано, как используется рассматриваемый сценарий.

Пример 13.14. Страница с разноцветным текстом

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Frameset//EN»

"http://www.w3.org/TR/html4/frameset.dtd">

<HTML>

<HEAD>

<TITLE>Разноцветный текст</TITLE>

<STYLE type = "text/css">

.colored {}

</STYLE>

</HEAD>

<BODY>

<H1>Обычный заголовок</H1>

<H1 class = "colored">Разноцветный заголовок</H1>

<P class>Обычный текст

<P class = "colored">Разноцветный текст

<SCRIPT type = "text/javascript" src = "coloredtext.js"></SCRIPT>

</BODY>

</HTML>

Теперь создадим усовершенствованный вариант рассмотренного ранее меню. Сейчас это будет не просто набор пунктов, все время находящийся на странице, а настоящее меню, похожее на то, которое имеют многие Windows?приложения.

Пример реализован следующим образом: вверху окна организуется строка меню, в которой присутствуют два пункта, открывающих два различных меню. Первое меню выглядит так, как показано на рис. 13.5.

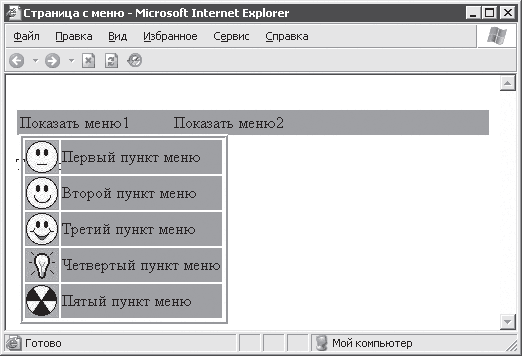


Рис. 13.5. Открыто первое меню

Второе меню, вызываемое при выборе второго пункта в строке меню, показано на рис. 13.6.

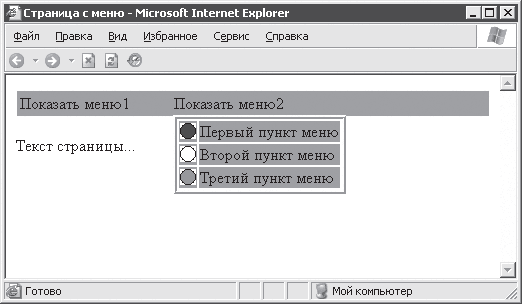


Рис. 13.6. Открыто второе меню

Как видно, оба меню появляются под соответствующими пунктами строки меню. Теперь рассмотрим, как реализован этот пример, а также какие существуют направления усовершенствования этого примера. Пример разбит на две части: HTML?документ и сценарий (файл popup\_menu.js). Сначала разберем текст HTML?документа (пример 13.15).

Пример 13.15. Документ со строкой меню

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Frameset//EN»>

<HTML>

<HEAD>

<TITLE>Страница с меню</TITLE>

<STYLE type = "text/css">

.item {background-color: rgb(170, 170, 170)}

.selected {background-color: magenta}

.menu {border-style: ridge; visibility: visible; position: absolute}

.hidden {display: none}

.menu\_line {border-style: solid; border-width: 1px;

background-color: rgb(170,170,170); width:100%}

</STYLE>

<SCRIPT src = "popup\_menu.js" type = "text/javascript"></SCRIPT>

</HEAD>

<BODY>

<!–Создаем первое меню (изначально оно скрывается)–>

<TABLE id = "menu1" class = "hidden">

<!–Первый пункт меню–>

<TR id = "menu1\_item1" class = "item" onClick = "menu1\_item1\_click()"

onMouseOver = "menu1\_item1.className = 'selected'"

onMouseOut = "menu1\_item1.className = 'item'">

<TD><IMG src = "icons/1.jpg"><TD>Первый пункт меню

<!–Второй пункт меню–>

<TR id = "menu1\_item2" class = "item" onClick = "menu1\_item2\_click()"

onMouseOver = "menu1\_item2.className = 'selected'"

onMouseOut = "menu1\_item2.className = 'item'">

<TD><IMG src = "icons/2.jpg"><TD>Второй пункт меню

<!–Третий пункт меню–>

<TR id = "menu1\_item3" class = "item" onClick = "menu1\_item3\_click()"

onMouseOver = "menu1\_item3.className = 'selected'"

onMouseOut = "menu1\_item3.className = 'item'">

<TD><IMG src = "icons/3.jpg"><TD>Третий пункт меню

<!–Четвертый пункт меню–>

<TR id = "menu1\_item4" class = "item" onClick = "menu1\_item4\_click()"

onMouseOver = "menu1\_item4.className = 'selected'"

onMouseOut = "menu1\_item4.className = 'item'">

<TD><IMG src = "icons/4.jpg"><TD>Четвертый пункт меню

<!–Пятый пункт меню–>

<TR id = "menu1\_item5" class = "item" onClick = "menu1\_item5\_click()"

onMouseOver = "menu1\_item5.className = 'selected'"

onMouseOut = "menu1\_item5.className = 'item'">

<TD><IMG src = "icons/5.jpg"><TD>Пятый пункт меню

</TABLE>

<!–Создаем второе меню (изначально оно также скрывается)–>

<TABLE id = "menu2" class = "hidden">

<!–Первый пункт меню–>

<TR id = "menu2\_item1" class = "item" onClick = "menu2\_item1\_click()"

onMouseOver = "menu2\_item1.className = 'selected'"

onMouseOut = "menu2\_item1.className = 'item'">

<TD><IMG src = "balls/ball1.bmp"><TD>Первый пункт меню

<!–Второй пункт меню–>

<TR id = "menu2\_item2" class = "item" onClick = "menu2\_item2\_click()"

onMouseOver = "menu2\_item2.className = 'selected'"

onMouseOut = "menu2\_item2.className = 'item'">

<TD><IMG src = "balls/ball2.bmp"><TD>Второй пункт меню

<!–Третий пункт меню–>

<TR id = "menu2\_item3" class = "item" onClick = "menu2\_item3\_click()"

onMouseOver = "menu2\_item3.className = 'selected'"

onMouseOut = "menu2\_item3.className = 'item'">

<TD><IMG src = "balls/ball3.bmp"><TD>Третий пункт меню

</TABLE>

<!–Вверху страницы организуется строка меню–>

<TABLE id = "main\_menu1" class = "menu\_line">

<COL span = "2" width = "150">

<COL width = "\*">

<TR>

<TD class = "item" id = "main\_item1"

onClick = "show\_menu(menu1, main\_menu1, main\_item1)"

onMouseOver = "main\_item1.className = 'selected'"

onMouseOut = "main\_item1.className = 'item'">Показать меню1

<TD class = "item" id = "main\_item2"

onClick = "show\_menu(menu2, main\_menu1, main\_item2)"

onMouseOver = "main\_item2.className = 'selected'"

onMouseOut = "main\_item2.className = 'item'">Показать меню2

<TD><!-Пустая ячейка, просто занимает место–>

</TABLE> <!–Далее идет остальное содержимое страницы–>

<P>Текст страницы...

</BODY>

</HTML>

Хотя документ практически не содержит текста, являющегося обычным содержимым страницы (ведь мы рассматриваем не наполнение страницы текстов, а меню), он все равно получился довольно объемным. Больше всего места в документе примера 13.5 занимают описания двух меню. Описание первого меню практически не отличается от рассмотренного ранее в примере 13.6. Второе же меню, обозначенное как menu2, создано по такому же шаблону.

В таблицу стилей пришлось добавить новый стиль menu\_line. После этого очень просто создавать любое количество строк меню на странице. При создании строки меню основной работой является настройка пунктов, которые будут открывать нужные меню (см. определение пунктов в таблице с id, равным main\_menu1 в примере 13.5).

При выборе пунктов из строки меню происходит вызов одной и той же функции сценария. Этой функции в качестве параметров передается ссылка на меню, которое нужно открыть, а также ссылки на строку меню и ссылка на пункт, к которому относится показываемое меню. При рассмотрении сценария станет ясно, зачем это нужно.

Ниже приводится текст сценария из файла popup\_menu.js (пример 13.16).

Пример 13.16. Содержимое файла popup\_menu.js

/\*

Функция показывает заданное всплывающее меню под заданным главным пунктом заданного меню

\*/

var lastMenu = null; //Предыдущее показанное меню function show\_menu(menu, main\_menu, item){

if (menu.className == "menu"){

//Закрываем открытое меню hide\_menu();

return;

}

if (lastMenu != null)

//Скрываем прошлое меню hide\_menu();

//Определяем положение меню menu.className = "menu";

menu.style.top = main\_menu.offsetTop + main\_menu.clientHeight;

menu.style.left = main\_menu.offsetLeft + item.offsetLeft;

lastMenu = menu;

}

//Функция скрывает меню, открытое ранее function hide\_menu(){

lastMenu.className = "hidden";

lastMenu = null;

}

/\*

Далее содержатся функции-обработчики для каждого пункта меню "menu1"

\*/

function menu1\_item1\_click(){

hide\_menu();

alert("Вы выбрали первый пункт в меню1");

//Другие действия...

}

function menu1\_item2\_click(){

hide\_menu();

alert("Вы выбрали второй пункт в меню1");

//Другие действия...

}

function menu1\_item3\_click(){

hide\_menu();

alert("Вы выбрали третий пункт в меню1");

//Другие действия...

}

function menu1\_item4\_click(){

hide\_menu();

alert("Вы выбрали четвертый пункт в меню1");

//Другие действия...

}

function menu1\_item5\_click(){

hide\_menu();

alert("Вы выбрали пятый пункт в меню1");

//Другие действия...

}

/\*

Далее содержатся функции-обработчики для каждого пункта меню "menu2"

\*/

function menu2\_item1\_click(){

hide\_menu();

alert("Вы выбрали первый пункт в меню2");

//Другие действия...

}

function menu2\_item2\_click(){

hide\_menu();

alert("Вы выбрали второй пункт в меню2");

//Другие действия...

}

function menu2\_item3\_click(){

hide\_menu();

alert("Вы выбрали третий пункт в меню2");

//Другие действия...

}

Первая функция приведенного в примере 13.16 сценария отвечает за правильное отображение меню. Первый параметр является ссылкой на показываемое меню (таблица в HTML?документе).

Второй и третий параметры используются для корректного позиционирования показываемого меню. Перед тем как будет показано новое меню, скрывается то, которое было показано ранее (если оно имеется). Для этого ссылка на отображаемое меню сохраняется в глобальной переменной lastMenu.

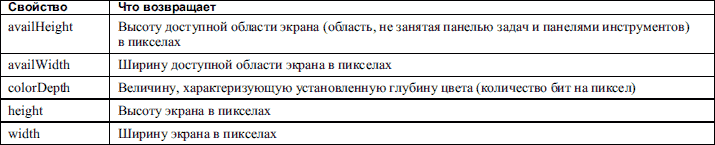
Если пользователь открыл меню, но потом передумал выбирать какой?либо пункт, он должен иметь возможность закрыть меню. В рассматриваемом примере для закрытия меню пользователь должен повторно выбрать тот же самый пункт строки меню.

Основной объем сценария составляют обработчики выбора пунктов меню. Думаю, принцип именования функций?обработчиков в зависимости от принадлежности в меню очевиден. Новой же деталью является наличие в каждом обработчике вызова функции hide\_menu(). Этим достигается закрытие меню после выбора одного из его пунктов.

Объект screen

Глобальный объект screen предоставляет набор свойств, которые сообщают сценарию некоторую информацию о возможностях видеосистемы компьютера пользователя.

Свойства объекта screen приводятся в табл. 13.10.

Таблица 13.10. Свойства объекта screen

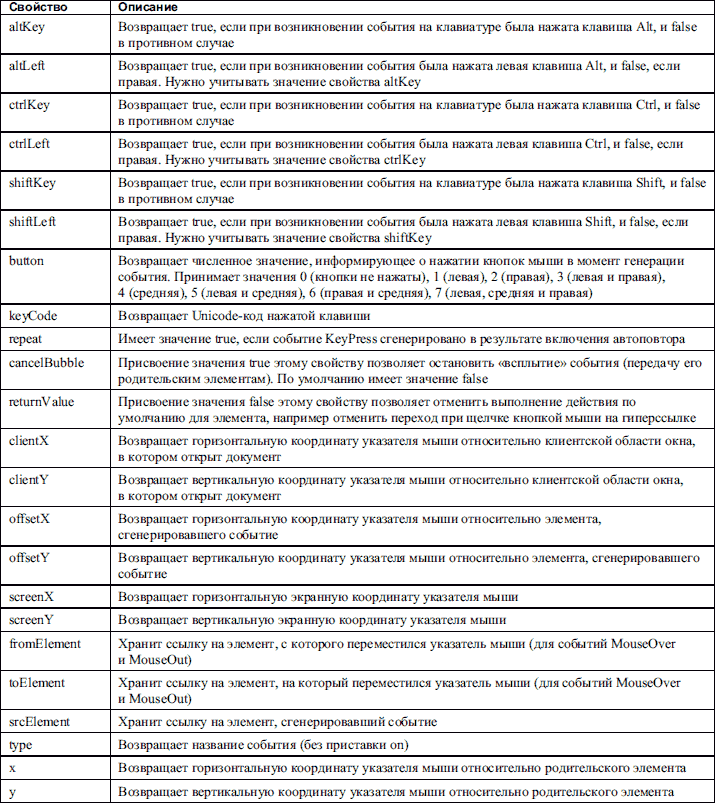
Объект screen может представлять большой интерес для тех, кто заботится о том, чтобы пользователь получал максимум комфорта при просмотре веб?страниц независимо, например, от разрешения своего монитора.

Объект event

Глобальный объект event является большим подарком для тех, кто хочет создавать сценарии, досконально отслеживающие и адекватно реагирующие практически на все действия пользователя. Поддержка этого объекта очень сильно отличается в различных браузерах. В этом разделе рассмотрено использование объекта event только для браузера Internet Explorer.

Свойства объекта event

Итак, используя браузер Internet Explorer, получить доступ к объекту event можно, как к свойству объекта window. Можно также указывать event как глобальный объект. Свойства объекта event приведены в табл. 13.11.

Таблица 13.11. Свойства объекта event  
Примеры использования объекта event

Для демонстрации работы с объектом event рассмотрены два небольших примера, использующие информацию о положении указателя и состоянии кнопок мыши.

В первом примере используется возможность получения координат указателя мыши относительно различных объектов документа. Текст HTML?документа примера приведен ниже. Обратите внимание, что благодаря «всплытию» событий отслеживать перемещение мыши можно, только написав обработчик события onMouseMove для элемента BODY (это самый верхний элемент, до которого доходит событие) (пример 13.17).

Пример 13.17. Отслеживание положения мыши

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.01 Transitional//EN»>

<HTML>

<HEAD>

<TITLE>Слежение за мышью</TITLE>

<STYLE type = "text/css">

.category {font-weight: bold}

.value {font-family: "courier new"}

</STYLE>

<SCRIPT type = "text/javascript">

function move(){

//Записываем в элементы страницы текущее значение

//координат указателя мыши scr.innerText = event.screenX + "x" + event.screenY;

wnd.innerText = event.clientX + "x" + event.clientY;

element.innerText = event.offsetX + "x" + event.offsetY;

prnt.innerText = event.x + "x" + event.y;

//Индикация состояния мыши switch (event.button){

case 0: mouse.innerText = "0|0|0"; break;

case 1: mouse.innerText = "1|0|0"; break;

case 2: mouse.innerText = "0|0|1"; break;

case 3: mouse.innerText = "1|0|1"; break;

case 4: mouse.innerText = "0|1|0"; break;

case 5: mouse.innerText = "1|1|0"; break;

case 6: mouse.innerText = "0|1|1"; break;

case 7: mouse.innerText = "1|1|1"; break;

}

}

</SCRIPT>

</HEAD>

<BODY onMouseMove = "move()">

<DIV style = "border-style: solid; border-width:1px; width: 50%;

position: absolute; left: 50%; background-color: yellow; padding: 2mm">

<SPAN class = "category">Экранные координаты: </SPAN>

<SPAN class = "value" id = "scr"></SPAN><BR>

<SPAN class = "category">Оконные координаты: </SPAN>

<SPAN class = "value" id = "wnd"></SPAN><BR>

<SPAN class = "category">Относительно элемента: </SPAN>

<SPAN class = "value" id = "element"></SPAN><BR>

<SPAN class = "category">Относительно родителя: </SPAN>

<SPAN class = "value" id = "prnt"></SPAN>

<P><SPAN class = "category">Кнопки мыши (л|c|п): </SPAN>

<SPAN class = "value" id = "mouse"></SPAN>

</DIV>

<H1>Заголовок</H1>

<P>Содержимое страницы...

</BODY>

</HTML>

Приведенный HTML?документ выглядит так, как показано на рис. 13.7.

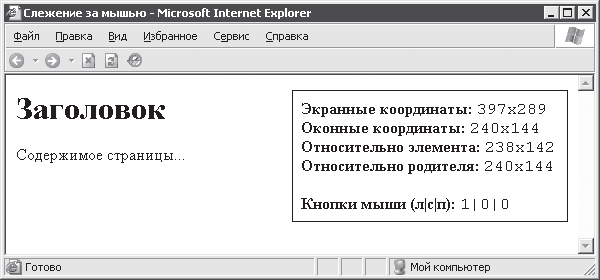


Рис. 13.7. Отслеживание указателя и состояния кнопок мыши

В следующем примере реализовано перетаскивание элементов страницы с помощью мыши. Перетаскивание элемента начинается при нажатии левой кнопки мыши и заканчивается при ее отпускании. Операции начала перемещения, самого перемещения и «бросания» элемента на новом месте реализованы в обработчиках событий элемента BODY (пример 13.18).

Пример 13.18. Перетаскивание элементов

<!DOCTYPE HTML PUBLIC «-//W3C//DTD HTML 4.0 Transitional//EN»>

<HTML>

<HEAD>

<TITLE>Перемещение элементов страницы</TITLE>

<SCRIPT type = "text/javascript">

var dX, dY; //Координаты точки, за которую "держат" элемент var element; //Перемещаемый элемент var fMoving = false; //==true, если перемещается элемент

//Функция начинает перемещение элемента function start(){

if (event)

fMoving = true;

element = event.srcElement;

//Сохраняем координаты "хватания" элемента dX = event.offsetX;

dY = event.offsetY;

//Для перемещения элемент должен свободно позиционироваться element.style.position = "absolute";

}

//Функция перемещения элемента function move(){

if (fMoving){

//Установим новые координаты для элемента element.style.pixelLeft = event.x – dX;

element.style.pixelTop = event.y – dY;

}

}

</SCRIPT>

</HEAD>

<BODY onMouseMove="move()" onMouseDown = "start()"

onMouseUp = "fMoving = false">

<IMG alt = "Перемещаемый рисунок" width = "100" height = "100">

<P>Перемещаемый текст

<H1>Перемещаемый заголовок</H1>

</BODY>

</HTML>

Документ, текст которого приведен в примере 13.18, после перетаскивания элементов показан на рис. 13.8.

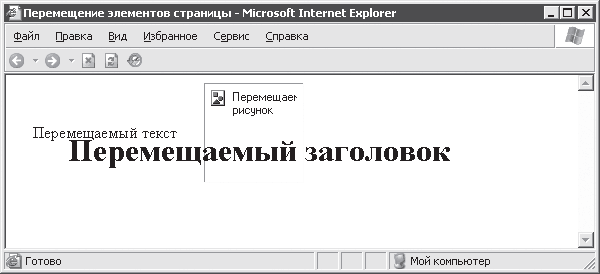


Рис. 13.8. Перетаскивание элементов страницы

Свободно позиционировать можно практически любой элемент страницы, однако наиболее эффектно выглядит перетаскивание изображений. Несмотря на кажущуюся ненужность, перетаскивание элементов может все же оказаться полезным при повышении «дружественности» интерфейса веб?страниц. Представьте себе, что вы, например, совершаете покупки в интернет?магазине. Понравился товар – перетаскиваете его изображение в свою корзину (определенная область окна). Хотите отказаться от покупки выбранного ранее товара – перетаскиваете его изображение за пределы корзины.

**Контрольные вопросы:**

1. Структура HTML-документа и элементы разметки заголовка документа
2. Основные контейнеры заголовка HTML-документа
3. Контейнеры тела документа HTML-документа
4. Использование графики в HTML
5. Таблицы в HTML
6. HTML-формы
7. Фреймы в HTML
8. Базовые типы данных в HTML
9. Общая структура документа HTML

## Задания для внеаудиторной самостоятельной работы

## *РАЗДЕЛ 4. ТЕХНОЛОГИИ СОЗДАНИЯ САЙТА*

## Принципы реализации Интернет-проекта

1. Определение целевой аудитории Web-сайта. Прогноз эффективности проекта

Попытайтесь спрогнозировать эффективность Вашего Интернет-проекта. Определите, будет ли он окупать себя? Будет ли Ваш Web-сайт приносить прибыль?

Эффективность Web-сайта во многом определяется масштабом его целевой аудитории. Целевая аудитория – это все потенциальные потребители продукции Вашего предприятия или услуг Вашей организации. Очевидно, что мужская аудитория в меньшей степени является посетителями сайтов косметических средств, тогда как сайты со спортивной тематикой в большей степени привлекают мужское населений. Определите размер целевой аудитории, проанализировав статистическую информацию на ведущих поисковых машинах в сети Интернет, серьезных рейтинговых маркетинговых агентств.

Используйте для исследования следующие адреса:

**Рейтинги:**

Rambler's Top 100 ([www.rambler.ru](http://www.rambler.ru)) ;

Ranker ([ranker.ru](http://ranker.ru));

Top List ([www.top.list.ru](http://www.top.list.ru));

SpyLog ([spylog.ru](http://spylog.ru));

Myweb (<http://www.myweb.ru/>);

**Маркетинговые исследования:**

АКиМ – Информационное агентство (<http://www.akm.ru/rus/akm/> );

Евро-азиатская маркетинговая исследовательская ассоциация (<http://www.emra-research.org/>);

Маркетинговые исследования в России – Журнал ( <http://www.marketingandresearch.ru/> );

РЕГИСТР – помощь принятия управленческих решений (<http://www.registr.sp.ru/services/marketing/>);

[Той-Опинион- Маркетинговые исследования- Санкт-Петербург](http://www.myweb.ru/cgi-bin/go/redir.cgi?u=http://www.opinion.spb.ru/&i=194975) (<http://www.opinion.spb.ru/>).

Определите эффективность продвижения проекта, используя для расчетов следующую формулу:



При результате менее 5% стоит задуматься об эффективности применения данной маркетинговой акции.

1. Оценка окупаемости и доходности Web-сайта

Предположим, что Ваша компания предполагает реализовать проект Интернет-магазина. Известно, что только 3% посетителей станут покупателями товаров в Вашем магазине. Определите, при известных цене товара ($5) и числе посетителей (1000000 чел.) возможный доход от реализации товара через Интернет-магазин?

1. Анализ сайтов типа Интернет-магазин

Осуществите поиск нескольких сайтов Интернет-магазинов.

1. определите цели и целевую аудиторию сайта соответствие содержания возможной цели;
2. оцените удобство навигации по сайту, соответствие дизайна содержанию, удобство оформления заказа.

Проведите ранжирование Интернет-магазинов по совокупности показателей.

1. Анализ сайтов типа Корпоративное представительство

Проанализируйте контент - информационное наполнение нескольких сайтов, являющихся корпоративным представительством одной из следующих компаний:

1. Компании по услугам в области Web-дизайна.
2. Компании по производству автомобилей.
3. Компании по оказанию консалтинговых услуг.
4. Образовательных учреждений.
5. Компаний, специализирующихся на выполнении маркетинговых исследований.
6. Предприятий по производству хлебобулочных и кондитерских изделий.

Отчет о выполненном задании должен содержать таблицу 1.1 выполненных исследований и выводы, сформулированные на основе анализа.

Таблица 1.1. Таблица исследований

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Адрес сайта** | **Название страниц** | **Соответствие содержания заявленному названию** | **Качество представленной информации** | **Дата последнего обновления информации** |
|  |  |  |  |  |
|  |  |  |  |

1. Выбор и регистрация доменного имени

Присвоение доменных имен имеет строго определенную систему, которая построена по иерархическому принципу. Домены первого уровня делятся на две группы: территориальные (ru, ua,uk, us) и внетерриториальные (com, org, net). Количество доменов первого уровня ограничены, решение о вводе новых принимается специальной международной организацией –ICANN. Однако на любом из доменов первого уровня каждый желающий ожжет зарегистрировать домен второго уровня. Например, Российская экономическая академия им. Г.В. Плеханова имеет домен Rea второго уровня в зоне ru. Доменное имя сайта организации или предприятия, а также любого on-line представительства играет значительную роль для успеха Интернет-проекта. Хорошо запоминающееся доменное имя, имеющее определенные ассоциативные связи позволяет не только легко его запоминать, но облегчить пользователю поиск сайта в сети Интернет.

Приведите примеры доменных имен, в которых содержится информация об области деятельности компании или они однозначно отражают принадлежность сайта к существующей в реальной жизни компании.

Попробуйте подобрать доменное имя для компаний, занимающихся следующей деятельностью:

* Рекламное агентство.
* Научно-исследовательский институт железнодорожного транспорта.
* Агентство по торговле с недвижимостью.
* Книжный магазин.
* Предприятие по производству хлебобулочных изделий.

Выбрав имя, проверьте, свободно ли оно. Это можно сделать по адресам [www.register.com](http://www.register.com) , <http://www.nic.ru>.

Убедившись, что выбранное имя свободно, можно приступить к его регистрации. В России регистрацией в зоне домена ru занимается RU-Center ( [www.nic.ru](http://www.nic.ru) ), а также значительное число посредников, которые помогут Вам зарегистрироваться сразу в нескольких зонах.

1. Размещение Web-сайта в сети Интернет

Не смотря на то, что Вы еще не создали своего сайта, изучите возможность размещения его в сети Интернет.

Размещение сайта на сервере фирмы-провайдера (хостера) называется Web-хостингом. Разместив свой сайт на сервере такой фирмы, Вы получите некоторое место на сервере и собственное доменное имя.

Ознакомьтесь с условиями провайдера, проанализировав информацию, предоставляемую в сети соответствующими фирмами. Заполните таблицу 1.2, анализ которой позволит вам сделать правильный выбор.

Таблица 1.2. Анализ рынка услуг

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Адрес провайдера** | **Дисковое пространство** | **Загрузка сайта** | **Реклама на сайте** | **Адрес Вашего сайта** | **Стоимость** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Проектирование Web-сайта средствами языка HTML на примере сайта информационного агентства.

1. Разработка структуры Web-сайта

Разработайте проект структуры сайта для одной из перечисленных ниже фирм:

* Рекламное агентство.
* Магазин канцтоваров.
* Мебельная фабрика.
* Консалтинговое агентство.
* Издательский дом.
* Туристическое агентство.
* Гостиничный комплекс.
* Строительная компания.

Web-сайт должен состоять не более чем из шести страниц и представлять следующую информацию: название фирмы, общую краткую информацию о фирме, перечень продуктов/услуг, краткие отзывы о вашей деятельности, список крупнейших клиентов со ссылками на их Web-сайты. Проект Web-сайта должен отвечать следующим требованиям: четкость формулировок, структурированность материалов, единство стиля, иметь собственное лицо. Отчет о выполненном задании содержать графический материал, отражающий структуру сайта, а также текстовые комментарии. Пример структура Web-сайта, реализующего информационную поддержку в сети в сети Интернет, представлена на рис. 2.1

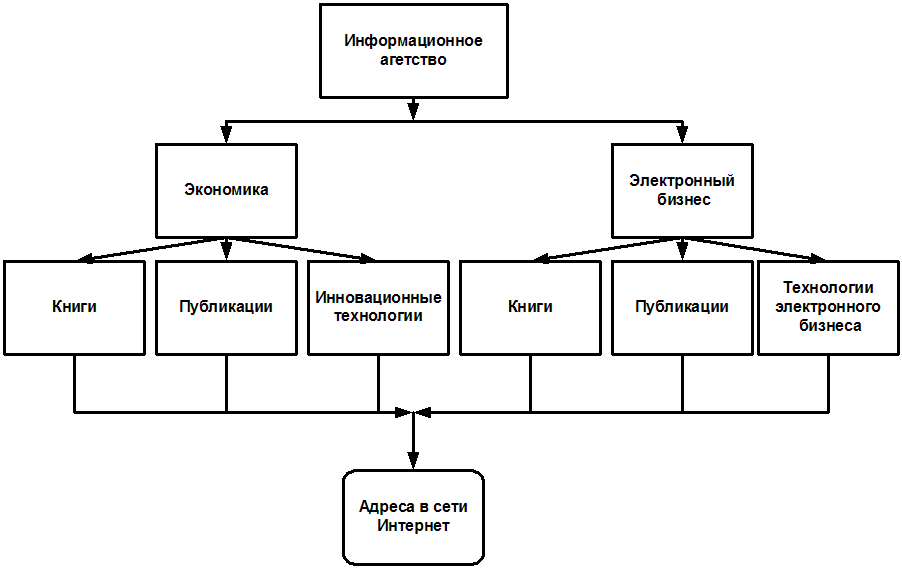


Рис. 2.1.Структура Web-сайта информационного агентства

1. Разработка простой Web-страницы

Большинство Web-страниц представляют собой гипертекстовые документы, отформатированные с помощью специального языка разметки документов HTML (HyperText Markup Language).

Язык разметки гипертекста состоит из набора элементов, которые описывают структуру документа и управляют его отображением. Разметка документов (markup) производится с помощью специальных флагов разметки, называемых тегами (tag – метка), которые интерпретируются Web-браузерами при загрузке HTML-документа. Теги являются инструкциями для браузера и определяют как внешний вид документа (форматирование символов, организации абзацев, таблиц и т.п.), так и связи с другими URL-ресурсами (HTML-страницами, изображениями, серверами).

Web-страница представляет собой обычный текстовый файл (то есть файл, содержащий только текст с символами, записанными в кодах ASCII[[3]](#footnote-3)), в котором собственно текст страницы размечен инструкциями для Web-браузера. На основании этих инструкций браузер определенным образом отображает страницу в своем окне и при активизации гиперссылки осуществляет загрузку указанного ресурса.

Любой HTML-документ состоит из двух логических частей: заголовка и тела. Заголовок документа содержит основную информацию о документе, в том числе название HTML-документа, которое отображается в строке заголовка Web-браузера. Тело документа интерпретируется браузером как содержимое Web-страницы.

HTML-документ состоит из размеченного тегами текста. Теги представляют собой определенную последовательность символов, заключенную между символами "<" и ">". Большинство тегов парные, закрывающий тег начинается с символа "/". Использование прописных или строчных букв в написании тегов значения не имеет.

Вот пример "минимальной" Web-страницы:

<HTML>

<HEAD>

<TITLE> Информационное агентство </TITLE>

</HEAD>

<BODY>

Привет! Это моя личная страница!

</BODY>

</HTML>

Здесь представлены парные теги:

<HTML>…</HTML> – начальный и конечный теги всего HTML-документа. Данный тег предназначен для того, чтобы браузер определил тип документа (HTML) и соответствующим образом его отобразил.

<HEAD>…</HEAD> – начальный и конечный теги заголовка документа. Тег заголовка должен быть указан сразу после тега <HTML> и не должен присутствовать в теле документа. Он отмечает общее описание документа и позволяет HTTP-серверу получить информацию о документе.

<TITLE>…</TITLE> – начальный и конечный теги названия страницы. Название, заключенное в теги <TITLE>…</TITLE>, должно размещаться внутри тегов <HEAD>…</HEAD>. Название страницы не является ее содержимым, а отображается в заголовке окна браузера. Название документа используется программами поиска информации в Web для построения индексов, поэтому должно быть информативным.

<BODY>…</BODY> - начальный и конечный теги тела документа.

Для создания HTML-документа можно воспользоваться любым текстовым редактором. Лучше всего использовать приложение Блокнот, которое создает файлы в текстовом формате. Редакторы WordPad и MS Word по умолчанию создают файлы в формате документа Word, который содержит не только текст, но и описание его форматирования. При разработке в них HTML-текста требуется особое внимание. В этом случае сохраняйте HTML-документ как текстовый файл, указав в имени файла одно из общепринятых расширений HTML-документов .htm или .html.

Загрузите текстовый редактор Блокнот. Введите HTML-текст Документа 2.1 и сохраните его как First.htm. Загрузив страницу в Internet Explorer, вы увидите ее изображение (рис.2.2).

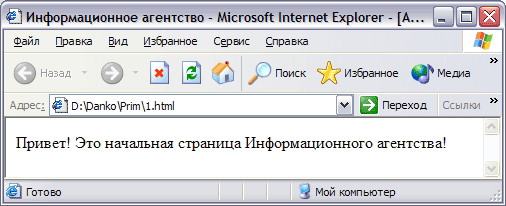


Рис. 2.2. Пример простой Web-страницы

Название страницы, заключенное в теги <TITLE>…</TITLE>, отобразилось в заголовке окна браузера, имя файла страницы – в строке адреса, а содержание страницы, ограниченное тегами <BODY>…</BODY>, – в окне просмотра.

1. Основы разработки Web-сайта

Выполните разработку Web-сайта информационного агентства. Первые шаги на этом пути Вам помогут сделать наши инструкции.

На первом этапе Вам необходимо разработать начальную, или как ее еще принято назвать домашнюю страницу. Эта страница должна содержать основную информацию, описывающую деятельность фирмы, а также ссылки на последующие страницы в соответствии со структурой сайта, представленной в Задании 7 (рис.1.1), а также контактную информацию для связи с представителями фирмы.

Следует помнить, что все теги относятся к одному из следующих трех форматов:

<ИМЯ\_ТЕГА>

<ИМЯ\_ТЕГА> … </ИМЯ\_ТЕГА>

<ИМЯ\_ТЕГА имя\_атрибута1=значение1 имя\_атрибута2=значение2 …>… </ИМЯ\_ТЕГА>

Первый формат используется для непарных тегов. Например, непарным является тег <BR> – тег, указывающий переход на новую строку.

Во втором формате используются открывающий и завершающий теги. Они ограничивают часть документа, в которой могут присутствовать и другие теги.

Третий формат используется для тегов, имеющих дополнительные атрибуты, задающие конкретные параметры. Атрибуты тега следуют за именем и отделяются от него и друг от друга пробелами или знаками табуляции. Последовательность атрибутов в теге значения не имеет. Если в таких тегах, а их большинство, атрибуты не указывать, то принимаются некоторые значения по умолчанию. Значения атрибутов заключаются в одинарные или двойные кавычки, кроме случаев, когда значением атрибута является одно слово или число. Некоторые атрибуты не принимают значения и присутствуют в теге в виде <ИМЯ\_ТЕГА имя\_атрибута>

Например, в теге <HR>, задающем вставку горизонтальной линии, можно использовать следующие атрибуты:

<HR ALIGN=right NOSHADE WIDTH=50%>

В этом примере атрибут ALIGN задает положение линии по горизонтали (справа), атрибут NOSHADE является указателем, что объемное затенение при отображении линии отсутствует, атрибут WIDTH задает значение длины линии в % от экранной ширины страницы.

Здесь и далее для облегчения чтения HTML-текста мы будем указывать имена тегов и атрибутов прописными буквами, хотя регистр при написании тегов значения не имеет.

Несколько тегов может быть размещено на одной строке, однако целесообразно выделять элементы, помещая их на разных строках для облегчения чтения и редактирования страницы. Еще раз отметим, что разбиение HTML-текста на строки, вставка пробелов и строк, а также символов табуляции не влияет на вид страницы, так как идущие подряд пробелы, символы перехода на новую строку и табуляции игнорируются

Загрузите текстовый редактор Блокнот и введите текст, представленный в Документе 1. Сохраните его в файле index.htm.

Документ 1. . Файл index.htm

<HTML>

<HEAD>

<TITLE>Информационное агентство</TITLE>

<HEAD>

<BODY>

<H3 ALIGN = RIGHT> ОАО "ИНФОРМЭкономика" </H3>

<H1 ALIGN = CENTER> Добро пожаловать! </H1>

<H3 ALIGN = CENTER> Вам нужна срочная поддержка при организации собственного бизнеса? </H3>

<H3 ALIGN = CENTER>Вы столкнулись с проблемами в производстве?</H3>

<H3 ALIGN = CENTER> Вы хотите узнать о новинках первыми?</H3>

<H3 ALIGN = CENTER> Этот сайт для Вас! </H3>

<P> Экономика <P> Электронный бизнес <P>

<P> Информация, которую мы Вам предоставляем, самая достоверная и актуальная! </P>

<H2> Контактная информация </H2>

<P> Телефон: (095) 345-67-89 </P>

<P> Почтовый адрес: 12-6, ул.Знаний, Москва, 123456 </P>

</BODY>

</HTML>

Запустите Web-браузеры MS Internet Explorer и Netscape Navigator. Откройте полученный документ в каждом из браузеров и сравните результаты с изображением страницы, представленной на рис.2.3.

Возможные отличия при отображении страницы могут быть вызваны различиями в настройках браузеров. Строки "Новости", "Продукция", "Услуги" мы вставили с тем, чтобы в дальнейшем описать их как ссылки на соответствующие страницы сайта.

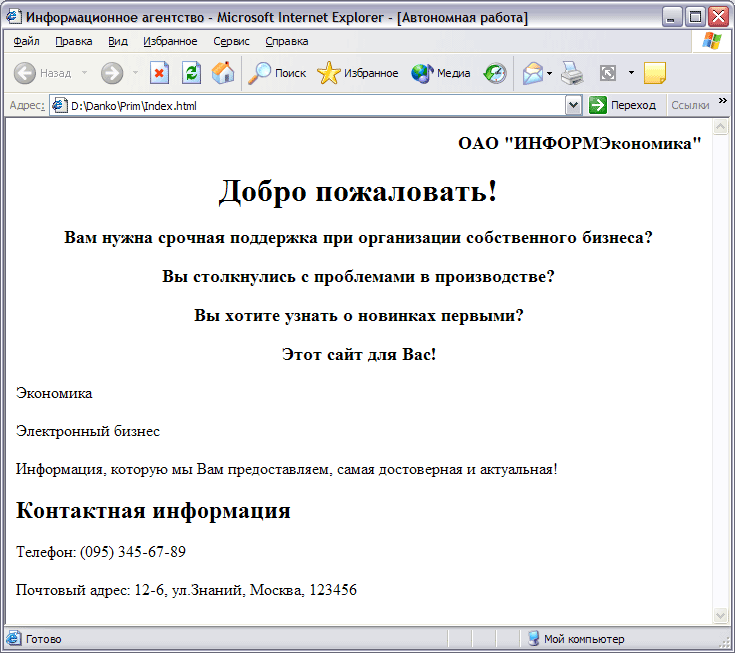


Рис. 2.3. Внешний вид Домашней страницы

Здесь появились новые теги: теги заголовков первого, второго и третьего уровня <H1> … </H1>, <H2> … </H2>, <H3> … </H3> и тег абзаца <P> … </P>.

В языке HTML поддерживается 6 уровней заголовков, отличающихся размерами текста. Самый крупный заголовок размечается тегами <H1> … </H1>, самый мелкий – тегами <H6> … </H6>. Следует иметь в виду, что в большинстве браузеров мелкие заголовки (H4 и ниже) отображаются очень мелким шрифтом и могут оказаться мельче основного текста. В нашем примере тег <H1 ALIGN = CENTER> Добро пожаловать! </H1> означает, что текст "Добро пожаловать!" оформлен как заголовок первого уровня, выровненный по центру. Атрибут тега заголовка ALIGN указывает способ выравнивания текста на строке (возможные значения: LEFT, CENTER, RIGHT). По умолчанию используется выравнивание по левому краю.

При создании абзацев в HTML-документах необходимо учитывать, что:

* нажатие клавиши Enter не создает новый абзац;
* последовательность нескольких пробелов, символов табуляции и символов новой строки сжимается при отображении документа в браузере в одиночный пробел.

Текст, который надо представить на странице в виде отдельного абзаца, заключается в теги <P>…</P>.

Информацию на странице можно визуально разделить, вставив между логически разными частями текста горизонтальную линию. Для этого используется непарный тег горизонтальной линейки <HR>.

Текст на странице можно отформатировать различными стилями. Для этого используются теги физического и логического форматирования (Таблица 2.1). Теги логического форматирования определяют лишь тип стиля, конкретный браузер интерпретирует их в соответствии со своими возможностями. Физические стили дают браузеру явные указания о том, как изображать тот или иной символ. Если браузер не поддерживает заданный физический стиль, то он его проигнорирует. Для форматирования текста целесообразно применять именно логические стили. Web-браузеры MS Internet Explorer и Netscape Navigator одинаково интерпретируют как логические, так и физические теги.

|  |  |  |
| --- | --- | --- |
| **Таблица 2.1. Теги форматирования текста** | | |
| **Тип шрифта** | **Логические** | **Физические** |
| Полужирный стиль | <STRONG> | <B> |
| Курсив | <EM> | <I> |
| Моноширинный шрифт | <CODE> | <TT> |

Размер шрифта для всего документа или последующей его части задается тегом <BASEFONT> (базовый шрифт), имеющим единственный атрибут SIZE. Значение размера задается в диапазоне от 1 до 7. По умолчанию размер базового шрифта равен 3.

Для установки размера, цвета и гарнитуры участка текста используется тег <FONT> с атрибутами SIZE, COLOR и FACE соответственно. Атрибут размера задается в диапазоне от 1 до 7 или в виде +n -n относительно размера, заданного в теге <BASEFONT>. Например, теги <FONT SIZE=-1 FACE=”Arial” COLOR=”#FF0000”> </FONT> означают, что заключенный между ними текст имеет размер на 1 меньший, чем это установлено для базового шрифта, выводится красным цветом шрифтом Arial.

Символы табуляции, множественные пробелы, вставляемые в текст страницы, игнорируются браузерами при выводе документа. Единственный способ отобразить текст в таком виде, как он был предварительно отформатирован, - это использование тега ручного форматирования <PRE>. Заключенный в теги <PRE>…</PRE> текст будет отображаться в окне браузера без обработки, с точным соблюдением переносов строк и интервалов. Именно этим способом можно формировать простые документы, таблицы. Однако в текст, отформатированный тегом <PRE>, нельзя вводить другие теги оформления, создавать рамки, задавать стили, цвета и т.д. Поэтому он не так часто используется при разработке Web-страниц.

Цвет в атрибутах задания цвета различных элементов страницы указывается в виде RGB-модели, то есть в виде соотношения красной (Red), зеленой (Green) и синей (Blue) составляющих цвета. Именно в этой модели отображаются точки на экранах цветных мониторов. Каждая составляющая описывается двузначным шестнадцатеричным числом. Значение 00 означает, что данная компонента отсутствует, а FF (255) – что присутствует с максимальной интенсивностью. Таким образом, код цвета всегда можно задать шестью шестнадцатеричными цифрами. Перед кодом цвета ставится символ #. Например, ярко-синий цвет задается в виде #4169F1:

<FONT COLOR=”#4169F1”>

В стандарте HTML для 16 основных цветов в атрибутах, описывающих цвет, вместо кодов можно использовать названия Таблица 2.2). Например, можно с одинаковым результатом использовать как тег <FONT COLOR=”Maroon”> , так и <FONT COLOR=”#800000”>. Регистр в названии цвета значения не имеет.

|  |  |  |
| --- | --- | --- |
| **Таблица 2.2. Коды цветов стандартной палитры Windows** | | |
|  | **Цвет** | **Шестнадцатеричное значение для**  **RGB-модели** |
|  | Белый (WHITE) | FFFFFF |
|  | Черный (BLACK) | 000000 |
|  | Светло-серый (Silver) | C0C0C0 |
|  | Серый (GRAY) | 808080 |
|  | Синий (BLUE) | 0000FF |
|  | Темно-синий (Navy) | 000080 |
|  | Оливковый-(Olive) | 808000 |
|  | Зеленовато-голубой (Aqua) | 00FFFF |
|  | Темный циан (Teal) | 008080 |
|  | Светло-зеленый (Lime) | 00FF00 |
|  | Зеленый (GREEN) | 008000 |
|  | Желтый (YELLOW) | FFFF00 |
|  | Красный (RED) | FF0000 |
|  | Темно-красный (Maroon) | 800000 |
|  | Фуксин (Fuchsia) | FF00FF |
|  | Фуксин темный (Purple) | 800080 |

Браузеры MS Internet Explorer и Netscape Navigator позволяют указывать имена вместо кодов для гораздо большего количества цветов, однако эти цвета не входят в стандарт и могут не распознаваться другими браузерами.

Установка цвета фона и текста для всего документа производится с помощью атрибутов тега <BODY>. Атрибут BGCOLOR=цвет задает цвет фона страницы, атрибут TEXT=цвет – цвет основного текста страницы.

Например, для заливки всей страницы голубым фоном и вывода текста темно-синим нужно использовать тег:

<BODY BGCOLOR="#00FFFF " TEXT= "#000080">

Дополните свою Домашнюю страницу тегами оформления, как это показано в Документе 2.

Документ 2. Файл index.html

<HTML>

<HEAD>

<TITLE> ИНФОРМКнига </TITLE>

</HEAD>

<BODY BGCOLOR="#C0C0C0">

<H3 ALIGN = RIGHT> ОАО "ИНФОРМКнига" </H3>

<H1 ALIGN = CENTER> <FONT COLOR = "#804040"> Добро пожаловать!</FONT> </H1>

<H3 ALIGN = CENTER> Вам нужна срочная поддержка при организации собственного бизнеса? </H3>

<H3 ALIGN = CENTER>Вы столкнулись с проблемами в производстве?</H3>

<H3 ALIGN = CENTER>Вы хотите узнать о новинках первыми?</H3>

<H3 ALIGN = CENTER><EM>Этот сайт для Вас!</EM></H3>

<P> Экономика <P> Электронный бизнес <P>

<P> Информация, которую мы Вам предоставляем, самая достоверная и актуальная! </P>

<H2> Контактная информация </H2>

<P> Телефон: (095) 345-67-89 </P>

<P> Почтовый адрес: 12-6, ул.Знаний, Москва, 123456 </P>

</BODY>

</HTML>

Теперь ваш документ выводится браузером на светло-сером фоне, с базовым размером шрифта 3 (по умолчанию), строка "Добро пожаловать!" имеет темно-бордовый цвет. Строки "Новости", "Продукция", "Услуги" выводятся без пропуска строки. Строка "Мы рады, что Вы посетили наш сайт!" отображается курсивным стилем шрифтом Arial размером на 1 больше, чем символы основного текста (размер 4). Тег < ! > является тегом комментария. Текст, который вводится внутри тега комментария не отображается браузером. Контактная информация отделена от основного содержания страницы горизонтальной линией.

Создайте страницы второго уровня: "”Экономика", "Электронный бизнес", Название фирмы целесообразно повторять в заголовке страницы и в первом заголовке тела страницы, чтобы пользователь всегда видел, на каком сайте он находится. Фон и оформление заголовков должно быть единым для всех страниц. На каждой странице вставим строку "Добро пожаловать!" как заготовку для ссылки на Домашнюю страницу.

На странице "Экономика" разместим информацию о последних достижениях фирмы (Документ 3), а также каталог разделов сайта по тематике страницы.

Документ 3.Файл economics.html

<HTML>

<HEAD>

<TITLE> ИНФОРМКнига - "ЭКОНОМИКА </TITLE>

</HEAD>

<BODY BGCOLOR="#C0C0C0">

<H3 ALIGN = RIGHT> ОАО "ИНФОРМЭкономика" </H3>

<H1 ALIGN = CENTER> <FONT COLOR = "#804040"> ЭКОНОМИКА </FONT> </H1>

<HR>

<P> Добро пожаловать!

<H2> Мы хотим Вам сказать, что: </H2>

<P> Количество клиентов компании увеличилось за последний месяц в два раза и составило 1000 000 человек.

<P> Поставщиками продукции для нашей фирмы стали 100 лучших издательств России.

<P>Посетите наши тематические разделы:</P>

<PRE>

КНИГИ

ПУБЛИКАЦИИ

ИННОВАЦИОННЫЕ ТЕХНОЛОГИИ

</PRE>

</BODY>

</HTML>

Результат отображения данного документа в браузере представлен на рис. 4.

Страницу «Электронный бизнес» оформите самостоятельно по аналогии со страницей «Экономика», сохранив ее в файле e\_busines.html.

После того, как мы разработали основное оформление страниц, можно организовать между ними связи. Для этого предусмотрительно включили в текст Домашней страницы названия страниц следующего уровня:

<P> Экономика <P> Электронный бизнес

Кроме того, на подчиненных страницах расположен текст «Добро пожаловать!», который будет использован для перехода на Домашнюю страницу.

Для организации гиперссылки используется тег <A> … </A>. Включение в документ гиперссылки производится с помощью вида тега <A HREF=…> , в котором атрибут HREF является обязательным элементом.

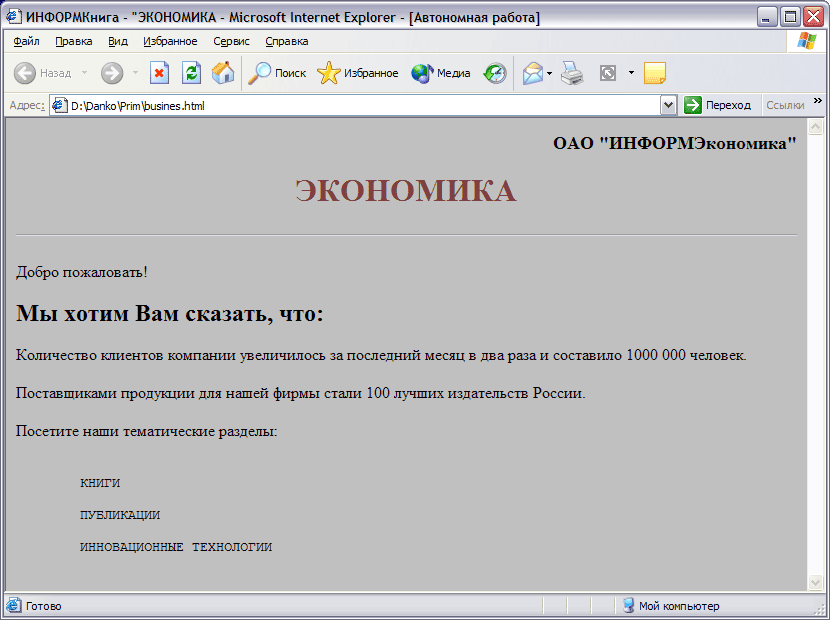


Рис. 2.4. Внешний вид страницы Экономика

Атрибут HREF="URL" задает URL-адрес объекта гиперссылки. Наличие спецсимволов в адресе требует заключения его в кавычки. С помощью атрибута HREF можно задать ссылку на другую страницу сайта, на определенный участок страницы, включить адрес электронной почты, задать ссылку на сетевой ресурс.

При просмотре в браузере текст, заключенный между тегами <A HREF=”URL”> и </A> , изображается в браузере с подчеркиванием и выделяется цветом в соответствии с настройками браузера. Пользователь, щелкнув по выделенному как гиперссылка тексту мышкой, может загрузить документ с указанным в ссылке адресом. Например, тегом

<A HREF="http://www.magazine.soft.ru"> Наши партнеры </A>

описывается гиперссылка на Домашнюю страницу партнеров. В этой ссылке задается полный URL-адрес.

Гиперссылки, указывающие на локальные документы, также содержат обычные URL-адреса, но в этом случае не нужно указывать протокол и полный адрес. При создании ссылки, указывающей на Web-страницу, расположенную на том же сервере, что и первоначальный HTML-документ, можно использовать относительный URL-адрес. Например, тег

<A HREF= economics.html > Экономика </A>

описывает ссылку на страницу news.htm, находящуюся на том же сайте. Здесь атрибут HREF= economics.html задает относительный адрес.

До сих пор мы рассматривали ссылки формата <A HREF=”URL”>, указывающие на отдельные файлы или страницы. Можно организовать ссылку на помеченный фрагмент внутри данного документа. Для этого в адресе используется имя метки со знаком #. Так, тег <A HREF = "#index1"> Подробнее... </A> описывает переход к фрагменту, находящемуся на той же странице, и помеченному меткой "index1".

В более общем случае можно ссылаться на помеченные фрагменты внутри других документов, используя формат <A HREF=”URL#метка”>. Так можно ссылаться на отдельные участки других страниц своего сайта. Для размещения меток в документе используется второй тип тега гиперссылки <A NAME=”метка”>. Атрибут NAME= метка задает имя метки (якоря) в некоторой точке документа. Например, мы можем пометить некоторую область на странице с большим количеством текста как <A NAME=”part2”> Аннотация </A>,а затем организовать переход на нее из начальной области страницы с помощью тега <A HREF=”#part2”> Подробнее </A>.

Отметим, что тег <A NAME=”метка”> … </A> не создает гиперссылки, и помещенный внутри него текст никак не выделяется в браузере. При переходе на метку страница прокручивается так, чтобы отмеченный текст оказался в верхней области окна просмотра.

Вставим гиперссылки на страницы нашего сайта. Для облегчения чтения теги гиперссылок и меток выделены в HTML-тексте жирным шрифтом.

Создадим на Домашней странице ссылки на страницы следующего уровня, включив в HTML-документ фрагмент, представленный в Документе 4.

Документ 4. Файл index.htm

…

<H3 ALIGN = CENTER>Вы хотите узнать о новинках первыми?</H3>

<H3 ALIGN = CENTER> <EM>Этот сайт для Вас!</EM> </H3>

<P> <A HREF=economics.html> Экономика </A><BR>

<P> <A HREF=”e\_business.html”>Электронный бизнес </A> <P>

<P> Информация, которую мы Вам предоставляем, самая достоверная и актуальная! </P> …

Для обеспечения возврата на домашнюю страницу вставим на нее ссылку на подчиненных страницах:

<A HREF="index.htm"> Добро пожаловать! </A>

Эти ссылки являются локальными, более того, мы предполагаем, что все наши страницы будут располагаться на сервере в одном каталоге, что позволяет вместо полного URL-адреса указать имя HTML-файла.

Добавим к контактной информации на Домашней странице адрес электронной почты менеджера по продажам (Документ 5). Для включения в качестве гиперссылки адреса электронной почты следует указать в URL-адресе mailto:.

Документ 5.Файл index.htm

<H2> Контактная информация </H2>

Телефон: (095) 345-67-89 <BR>

Почтовый адрес: 12-6, ул.Знаний, Москва, 123456 <BR> Адрес электронной почты:

<A HREF= "mailto: manager@ibook.ru"> manager@ibook.ru </A>

Теперь можно выполнить просмотр Домашней страницы в браузере и протестировать вставленные гиперссылки. При активизации ссылок на адрес электронной почты должно открыться окно создания сообщения почтовой программы, зарегистрированной в настройках браузера. Адрес, указанный в ссылке, появится в поле "Кому" (To:).

Для проверки работы внешней ссылки вставьте вместо несуществующего адреса партнеров ссылку на книжный Internet-магазин: http://www.ozon.ru.

1. Списки в HTML-документе

Информацию на HTML-странице можно представлять в виде маркированных и нумерованных списков, а также списков определений. Списки определений применяются для создания словарей или глоссариев, в которых после каждого элемента термина следует описательный абзац.

Использование тегов списков как элементов форматирования абзацев создают висячие отступы. Для описания списков существуют специальные теги (Таблица 2.3).

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица 2.3. Теги списков** | | | |
| **Вид списка** | **Формат тега** | **Атрибуты** | **Назначение** |
| Маркированный | <UL>список</UL> | TYPE=стиль маркера  Значения:  DISK, CIRCLE , SQUARE | Заключают весь маркированный список |
| Нумерованный | <OL>список</OL> | TYPE=формат нумерации  Значения:  A, a, I, i, 1  START=n  - начало отсчета | Заключают весь нумерованный список |
| Маркированный и нумерованный | <LI>элемент </LI> |  | Ограничивают каждый элемент списка |
| Список определений | <DL>список </DL> |  | Заключают весь список определений |
|  | <DT>термин</DT |  | Ограничивают элемент-термин |
|  | <DD>определение  </DD> |  | Ограничивают элемент-определение |

Тег <UL> имеет атрибут TYPE, назначающий стиль маркера. Его значениями могут быть: CIRCLE (незакрашенный кружок), DISC (диск, используется по умолчанию) или SQUARE (квадрат).

Наличие завершающего элемент тега </LI> не является обязательным, в то время как завершающий тег </UL> необходим для обозначения конца списка.

Тэг нумерованного списка <OL> может иметь атрибуты TYPE и START. Атрибут TYPE задает формат нумерации:

A - прописные латинские буквы (A,B,C...)

a - строчные латинские буквы (a,b,c...)

I - большие римские цифры (I,II,III...)

i - маленькие римские цифры (i,ii,iii...)

1 - арабские цифры (1,2,3...)

По умолчанию используются арабские цифры.

Атрибут START задает начало отсчета нумерации. Например, тег <OL TYPE=I START=3> задает нумерацию списка в виде больших римских цифр, первый элемент списка будет иметь номер III.

Пример использования нумерованных и маркированных списков приведен в Документе 6.

Документ 6. Файл economics.html

<HTML>

<HEAD>

<TITLE> ИНФОРМКнига - "ЭКОНОМИКА </TITLE>

</HEAD>

<BODY BGCOLOR="#C0C0C0">

<H3 ALIGN = RIGHT> ОАО "ИНФОРМЭкономика" </H3>

<H1 ALIGN = CENTER> <FONT COLOR = "#804040"> ЭКОНОМИКА </FONT> </H1>

<HR>

<A HREF="index.html"> Добро пожаловать! </A>

<H2> Мы хотим Вам сказать, что: </H2>

<P> Количество клиентов компании увеличилось за последний месяц в два раза и составило 1000 000 человек.

<P> Поставщиками продукции для нашей фирмы стали 100 лучших издательств России.

<P>Посетите наши тематические разделы:</P>

<P>КНИГИ </P>

<P>ПУБЛИКАЦИИ </P>

<P>ИННОВАЦИОННЫЕ ТЕХНОЛОГИИ </P>

<HR>

<UL TYPE=square>

<LI>Список КНИГ </LI>

<OL>

<LI> Куртер Дж., Маркви А.Microsoft Office 2000: учебный курс – СПб: Питер,2000.-640с.: ил.</LI>

<LI> Новиков Ф.А., Яценко А.Д. Microsoft Office 2000 в целом. - СПб.: БХВ – Санкт-Петербург, 2000. - 728 с. ил.</LI>

<LI> Проектирование интерактивных Web-приложений: Учебное пособие — М.: Горячая линия,2001.—272 с.: ил.</LI>

</OL>

<LI> Список ПУБЛИКАЦИЙ </LI>

<LI> ИННОВАЦИОННЫЕ ТЕХНОЛОГИИ </LI>

</UL>

</BODY>

</HTML>

При просмотре файла economics.html с помощью браузера вы увидите страницу, представленную на рис. 2.5.

Введите список ПУБЛИКАЦИЙ и ИННОВАЦИОННЫЕ ТЕХНОЛОГИИ в текст документа economics.html и просмотрите полученные результаты.

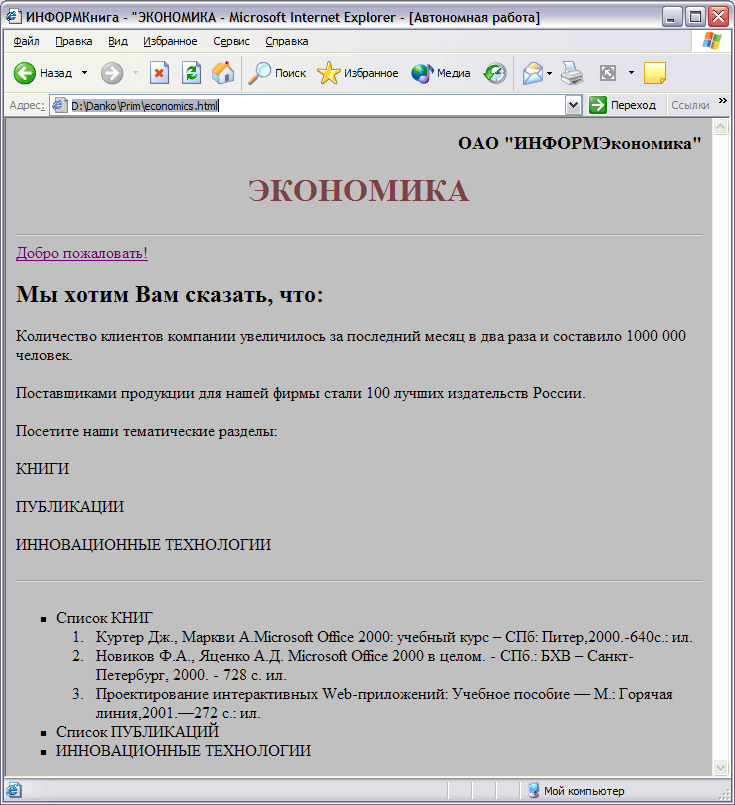


Рис. 2.5. Списки на странице Экономика

1. Таблицы в HTML-документе

Для создания таблиц в HTML-документах используются специальные теги таблиц. Их применяют как собственно для создания таблицы в HTML-документе, так и для организации размещения элементов (графики, гиперссылок, отдельных фрагментов текста) на странице.

Представление таблиц в HTML-документах базируется на использовании тега <TABLE>…</TABLE> и внутренних элементов, описывающих ее содержание :

<CAPTION>…</CAPTION> описывает заголовок таблицы;

<TR>…</TR> задает строки таблицы;

<TH>…</TH> ограничивает ячейки заголовков колонок;

<TD>...</TD> описывает стандартную ячейку.

Для включения таблицы в документ используется тег <TABLE>…</TABLE>. Он имеет множество атрибутов. Сейчас мы рассмотрим только один: BORDER, который задает обрамление таблицы. Тег <TABLE BORDER> задает вывод таблицы с рамкой. Ширина и вид рамки определяется браузером. Тег <TABLE BORDER=3> задает рамку шириной 3 пиксела. После тега <TABLE> можно вставить тег <CAPTION>, задающий заголовок таблицы.

Таблица формируется построчно. Каждая строка заключается в теги <TR>…</TR>. Внутри этих тегов размещаются описания ячеек таблицы. Стандартная ячейка описывается с помощью тега <TD>...</TD>. Ячейки заголовков колонок размечаются тегами <TH>…</TH>. Любая ячейка таблицы может быть описана только внутри строки таблицы. Сводка табличных тегов и их основных атрибутов дается в таблице 4.

Дополните Домашнюю страницу (файл index.html) таблицей, содержащей информацию о выставках (Документ 7).

Документ 7.Файл index.html

…

<TABLE BORDER ALIGN=CENTER>

<CAPTION> В выстовочном зале нашей компании проходят тематические выставки </CAPTION>

<TR><TH>Дата</TH> <TH>Название выставки</TH></TR>

<TR> <TD> 21.07.2002 </TD> <TD> Книга в образовании </TD> </TR>

<TR> <TD> 04.11.2002 </TD> <TD> Интернет-книга </TD> </TR>

</TABLE>

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Таблица 2.4. Основные атрибуты тегов таблиц** | | | | | | | | |
| **Тег** | **Атрибут** | **Возможные значения** | | **Значение по умолчанию** | | **Описание** | | |
| TABLE  Описание  таблицы | ALIGN=тип | | LEFT  CENTER  RIGHT | | LEFT | | Выравнивание таблицы по горизонтали | | |
| BORDER=n | |  | |  | | | Ширина рамки таблицы в пикселах. Число может быть опущено. | | |
| WIDTH=n | |  | |  | | | Ширина таблицы в пикселах или % | | |
| CELLPADDING=n | |  | |  | | | Область свободного пространства в пикселах между границами ячейки и ее содержимым | | | |
| CELLSPACING=n | |  | |  | | | Интервал в пикселах между ячейками таблицы | | | |
| TR  Описание строки | ALIGN=тип | | LEFT  CENTER  RIGHT | | LEFT | | Горизонтальное выравнивание данных в ячейках строки | | |
| VALIGN=тип | | TOP  CENTER  BOTTOM  BASELINE | | CENTER | | | Вертикальное выравнивание данных в ячейках строки | | |
| TH  Описание ячейки заголовка  TD  Описание стандартной ячейки | ALIGN=тип | | LEFT  CENTER  RIGHT | | CENTER–для ячеек заголовка (тег TH)  LEFT – для стандартных ячеек (тег TD) | | Выравнивание данных в ячейке по горизонтали | | |
| COLSPAN=n | |  | |  | | | Ячейка занимает n соседних столбцов | | |
| ROWSPAN=n | |  | |  | | | Ячейка занимает n соседних строк | | | |
| VALIGN | | TOP  CENTER  BOTTOM  BASELINE | | CENTER | | | Вертикальное выравнивание данных в ячейке | | | |
| WIDTH=n | |  | |  | | | Ширина ячейки в пикселах или % | | | |

Теги ячеек <TH> и <TD> имеют одинаковые с тегом строки <TR> атрибуты выравнивания. Значения атрибутов, установленные для отдельных ячеек таблицы, будут иметь приоритет над значениями этих же атрибутов, установленных в целом для строки.

Важное назначение тегов таблиц – это их творческое использование для компоновки элементов на странице. HTML имеет ограниченные возможности структурирования элементов страницы, например, нельзя расположить картинку и текст рядом на одной строке с помощью тегов форматирования. Поэтому очень часто таблицы используются именно для организации объектов на странице.

Нужно отметить следующие важные моменты:

* В ячейки таблиц можно помещать изображения, при этом полностью контролируя их ширину, высоту и расположение с помощью атрибутов тега IMG.
* Изображения и другие данные, помещенные в ячейки таблиц, одновременно могут являться гиперссылками. Объекты в таблице можно выравнивать с помощью атрибута CELLPADDING.

Разместим на Домашней странице рядом с названием компании ее логотип (графический gif-файл с именем logo.gif), создав его любом графическом редакторе. Вначале просто вставим логотип в документ с помощью тега IMG - Документ 8).

Документ 8.Файл index.htm

…

<IMG SRC="logo.gif">

<H3 ALIGN=RIGHT> ОАО "ИНФОРМКнига" </H3>

…

Атрибут SRC тега IMG является обязательным и задает графический файл, который нужно отобразить на странице. Ниже мы подробно рассмотрим вопросы, связанные с включением изображений в HTML-документы.

При просмотре страницы в браузере на экране появится сначала логотип, а под ним справа - название компании. Для размещения этих объектов рядом включим их как элементы данных в однострочную таблицу из двух ячеек (см. документ 2.19). Табличные теги далее выделяются жирным шрифтом.

Документ 9.Файл index.htm

…

<TABLE WIDTH=100%> <TR>

<TD WIDTH=30% ALIGN=LEFT> <IMG SRC= logo.gif></TD>

<TD WIDTH=70% ALIGN=CENTER> <H3 ALIGN = RIGHT> ОАО "ИНФОРМКнига" </H3> </TD>

</TR> </TABLE>

…

Целесообразно размещать ссылки на другие страницы либо в верхней части страницы, сразу же за заголовком, либо слева от основного текста страницы. Способ размещения должен быть единым для всех страниц Web-сайта, например таким, как это показано в Документе 10 и отображено на рис. 2.6.

Документ 10. Файл index.htm

…

</TR> </TABLE>

<TABLE width=100%> <TR>

<TD WIDTH=30%> <P> <A HREF=economics.html> Экономика </A>

<P> <A HREF=”e\_business.html”>Электронный бизнес </A> </P>

</TD>

</TR> </TABLE>

…

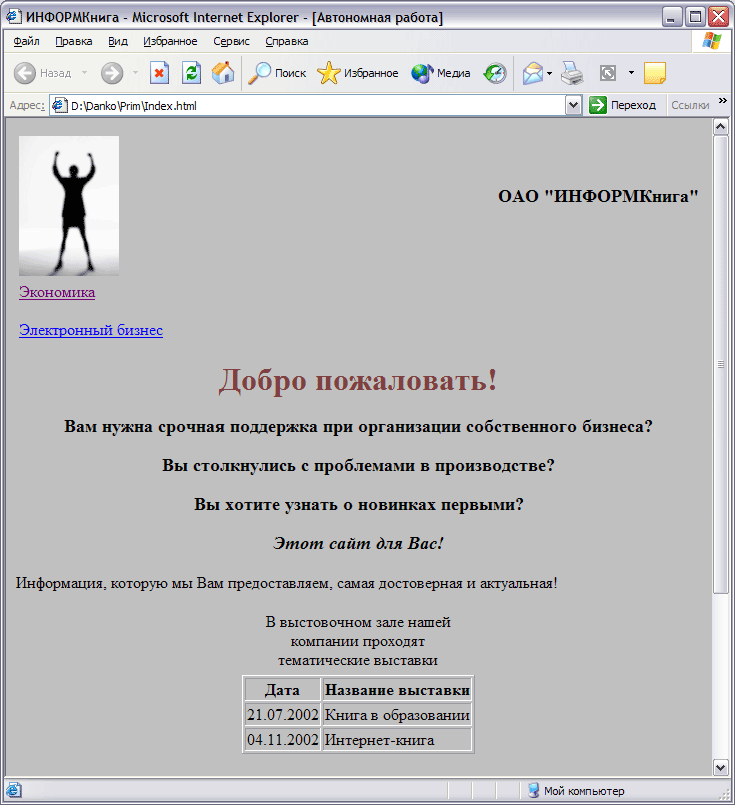


Рис. 2.6. Представление Домашней страницы при включении в нее логотипа

1. Изображения в HTML-документе

Несмотря на то, что существует множество графических форматов, для публикаций в сети Интернет, наиболее широкое применение находит два формата – GIF (Graphic Interchange Format) и JPEG (Joint Photographic Experts Group). Большинство Web-браузеров могут непосредственно отображать изображения, представленные в этих форматах. Такие изображения называются встроенными. Для отображения файлов других форматов может потребоваться запуск дополнительных программ. Изображения, для вывода которых требуются дополнительные к браузеру средства, называются внешними. Далее мы рассматриваем только встроенные изображения.

Встроенные изображения включаются в HTML-документ с помощью тега <IMG SRC=графический файл … дополнительные атрибуты>. В Таблице 2.5 приведены основные атрибуты тега <IMG>.

Таблица 2.5. Основные атрибуты тега <IMG>

|  |  |
| --- | --- |
| **Атрибут** | **Назначение** |
| SRC=url | Указывается исходный URL-адрес изображения, подлежащего воспроизведению.  Является обязательным. |
| ALT=текст | Задание альтернативного изображению текста. |
| ALIGN=тип выравнивания | Выравнивание изображения на странице относительно строки текста. Возможны значения:  LEFT, RIGHT - по правому или левому краю строки;  TOP, MIDDLE, BOTTOM - по высоте строки текста: по верху, по середине, по низу строки. |
| HEIGHT=n | Высота изображения в пикселах |
| WIDTH=n | Ширина изображения в пикселах |
| VSPACE=n | Задается свободное пространство в пикселах над и под изображением |
| HSPACE=n | Задается свободное пространство в пикселах слева и справа от изображения |
| BORDER=n | Установка толщины обрамления изображения в пикселах. |

Зададим альтернативный текст для фала логотипа компании logo.gif, включенного на домашней странице:

<IMG SRC="logo.gif" ALT="[Логотип]">

Проверьте присутствие альтернативного текста, отключив вывод графических объектов в настройках Internet Explorer.

Любое встроенное изображение может быть использовано в качестве гиперссылки на другой файл или HTML-документ. В том случае, когда изображение представляет собой гиперссылку, особенно важно задавать альтернативный текст. Например, мы задали гиперссылку на страницу о руководителях компании в виде графического файла:

<A HREF="http://www.person.infobook.ru/">

<IMG SRC="person.gif" ALT="Руководители"> </A>

Можно продублировать ссылку-изображение ссылкой-текстом:

<A HREF="http://www.person.infobook.ru/">

<IMG SRC="person.gif" ALT="Руководители">

Руководители </A>

В этом случае гиперссылками являются и изображение, и текст. Переход по гиперссылке выполняется как при загрузке изображения, так и при наличии на странице только альтернативного текста.

Несмотря на широкие возможности вставки изображений в HTML-документы, этим не следует злоупотреблять, так как загрузка файлов изображений, особенно больших иллюстраций, происходит медленно. Чтобы все же представить на странице всю желаемую графическую информацию, можно использовать следующий способ. Вместо иллюстрации большого размера на страницу вставляется ее миниатюрная копия (Thumbnail – с ноготок), которая загружается быстро. Эта копия оформляется как изображение-ссылка, а в теге гиперссылки указывается ссылка на исходное большое изображение. Теперь пользователь при желании может рассмотреть иллюстрацию, щелкнув на миниатюрной копии, загрузить ее в окно браузера.

Например, на Домашней странице можно карту проезда к выставочному залу. Для этого следует подготовить две серии графических изображений: исходные фото-изображения в формате JPEG и их уменьшенные копии в формате GIF. Попробуйте выполнить эту работу, предварительно изучив Документ 11.

Документ 11. Пример включения миниатюры

<HTML>

<HEAD>

<TITLE> Наша продукция </TITLE>

</HEAD>

<BODY>

<H2 ALIGN = RIGHT> Карта проезда </H2>

<H4>Щелкните по картинке, если хотите просмотреть фотографию образца</H4>

<A HREF = "bigpic.jpg"> <IMG SRC="pic.gif"> </A>

</BODY>

</HTML>

Файлы-изображения могут служить фоном для HTML-страницы. Для этого в теге BODY вместо задания цветового фона атрибутом BGCOLOR следует использовать атрибут BACKGROUND, задающий URL-адрес фонового изображения, например:

<BODY BACKGROUND="bgpic.gif">

При использовании изображений в качестве фона нужно помнить о том, что фон не должно затруднять чтение страницы. Лучше всего использовать небольшие изображения, которые браузер размножит на все поле страницы. Совсем маленькие изображения (в несколько пикселей) также не желательны, так как процесс «размножения» в этом случае займет много времени.

## Создание Web-сайтов средствами MS Office

1. Разработка Web-сайта средствами MS Word

Разработка Web-страниц в MS Word может быть выполнена двумя способами. Первый из них ориентирован на создание новой страницы, а второй – на сохранение ранее подготовленного текстового документа в формате Web-страницы.

Создание новой страницы осуществляется с помощью команды **Файл/Создать,** которая открывает окно **Создание документа**. На вкладке **Web-страницы** этого окна выбирается шаблон, соответствующий вашей задаче. Для создания Web-страниц Word располагает несколькими шаблонами. Наиболее часто используются шаблоны **Личная страница**, **Типичные вопросы**, **Простая страница** и **Оглавление**.

Для создания Web-страницы из текста, подготовленного в редакторе MS Word, необходимо выполнить команду **Файл/Сохранить как …**, а затем в стандартном диалоговом окне **Сохранение документа** задать тип файла (Веб-страница) и имя файла -рис.3.1.

Кнопка **Изменить** позволяет ввести заголовок Web-страницы, которая по умолчанию определяется как первая фраза самого документа. Однако, перед сохранением, если эта операция выполняется первый раз, необходимо указать параметры Web-документа. Используйте для задания параметров команду **Сервис/Параметры/**вкладка **Общие/** Кнопка **Параметры документа.** Онаоткрывает диалоговое окно **Параметры** **Веб-документа –** рис.3.2 которое содержит пять вкладок: **Обозреватели**, **Файлы**, **Картинки,** **Кодировка** и **Шрифты.**

|  |
| --- |
| 9_11 |
| Рис. 3.1. Окно Сохранение документа |

На вкладке **Обозреватели** определяется уровень средств, используемых для оформления документов. Способ хранения созданной Web-страницы, имена и структура папок для графических иллюстраций, мультимедийных объектов и других дополнительных файлов устанавливаются на вкладке **Файл**. Форматы графических файлов, а также выбор параметров, управляющих оптимизацией Web-страницей для вывода на конкретный монитор, устанавливаются на вкладке **Картинки**. На вкладке **Кодировка** определяется кодировка символов при сохранении Web-страницы. Выбор кодировки зависит от аудитории, для которой предназначена информация. Для англоязычной страницы можно использовать заподноевропейский набор символов, для страницы на русском языке – кириллицу или стандарт UNICODE(UFT-8).

При создании и редактировании сайта возможно применение стилевого форматирования и шаблонов, включение таблиц, графиков и рисунков, причем используемые приемы нечем не отличаются от аналогичных операций при оформлении текстовых документов.

|  |
| --- |
| 9_12 |
| Рис. 3.2 . Диалоговые окна для настройки параметров Web-документа |

Создайте Web-сайт образовательного центра «Интернет маркетинг». Домашняя страница , сохраните его в файле index.html. При просмотре страницы в Internet Explorer страница должна иметь вид, представленный на рис. 3.3.

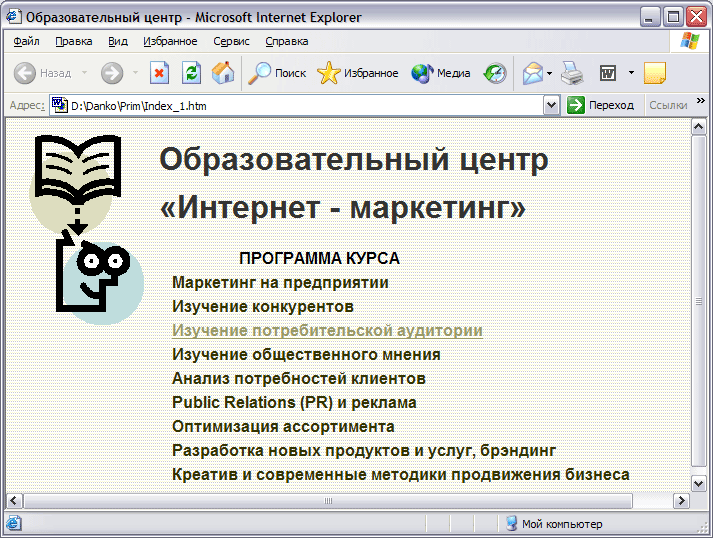


Рис. 3.3. Домашняя страница сайта Образовательного центра «Интернет-маркетинг»

Создайте еще несколько страниц, связанных с Домашней страницей гиперссылками, в которых разместите тематические планы по каждой из заявленных тем со следующим содержанием:

**Маркетинг на предприятии:**

1. Помощь в выходе на новые рынки.
2. Постановка системы статистики продаж с целью увеличения доходности бизнеса.
3. Разработка маркетинговой стратегии/ план маркетинга.
4. Организация клиентской базы данных.
5. Поиск новых рынков сбыта.
6. Организация системы прямых продаж.
7. Маркетинговый аудит.
8. Постановка системы маркетинговой логистики.
9. Поиск баз данных потенциальных клиентов.
10. Предложение продукции потенциальным клиентам.
11. Оценка воздействия рекламы и эффективности рекламы.

**Изучение потребительской аудитории:**

1. Оценка потенциала рынка, оценка объема рынка, исследование тенденций бизнеса.
2. Оценка потребностей, которые удовлетворяют покупатель, приобретая продукт и степени его удовлетворенности.
3. Изучение процесса принятия решения покупателем.
4. Изучение поведения потребителей.
5. Определение ценовых диапазонов, приемлемых для покупателя (плюс анализ эластичности спроса от цены).
6. Оценка спроса на товары/услуги.
7. Сегментация рынка.
8. Исследование конкретных целевых групп.

**Анализ конкурентной среды:**

1. Мониторинг цен конкурентов.
2. Изучение ценовой политики конкурентов.
3. Изучение товарной политики конкурентов.
4. Изучение рекламной политики.
5. S. W. O. T. – анализ.
6. Портфолио-анализ, BCG и другие виды анализа.
7. Анализ сильных и слабых сторон конкурирующего товара с точки зрения.
8. Разработка Web-сайта средствами MS Excel

Как и в текстовом процессоре Word, сохранение рабочей книги, подготовленной в Excel, с помощью команды **Файл/Сохранить как …**. Эта команда открывает диалоговое окно **Сохранение документа** – рис.3.4, котором указывается имя и тип файла, а также место сохранения файла.

|  |
| --- |
| 9_14 |
| Рис. 3.4. Диалоговое окно Сохранение документа |

Для выполнения публикации в диалоговом окне **Сохранение документа** следует нажать кнопку **Опубликовать** и в появившемся диалоговом окне **Публикация Веб-страницы** выбрать публикуемые элементы, установить параметры просмотра (Работа с электронными таблицами или Работа со сводными таблицами) и вновь нажать кнопку **Опубликовать**. Если в окне **Сохранение документа** была установлена опция **Добавить интерактивность**, то при просмотре Web-страницы вы увидите данные, а также полосы прокрутки и инструменты, позволяющие выполнить редактирование опубликованных данных.

Web-страницу, созданную средствами Excel, можно просмотреть с помощью браузера, а также в Excel или Word. Для открытия страницы из Excel необходимо выполнить команду **Файл/Открыть,** указав тип файла и его имя.

Данные в WWW, опубликованные средствами MS Excel, могут быть скопированы на листы книги. Для этого следует открыть Web-страницу в окне браузера, с которой будут извлекаться данные, выполнить копирование содержимого ячеек в буфер промежуточного хранения, а затем, прейдя на рабочий лист, вставить эти данные из буфера. Перетаскивание данных из одного приложения в другое возможно лишь в том случае, когда окна обоих приложений размещены рядом.

На рис.3.5 показана Web-страница, созданная средствами MS Excel на основе подготовленных в табличном процессоре данных. В процессе создания использовалась опция **Добавить интерактивность**, что позволило работать с данными в сети Интернет.

Выполните разработку аналогичной страницы, используя исходные данные, представленные в Таблице 3.1, а затем включите ее в сайт «Образовательный центр Интернет-маркетинг». При включении опции **Добавить интерактивность** данные можно вводить в таблицу уже на Web-странице.

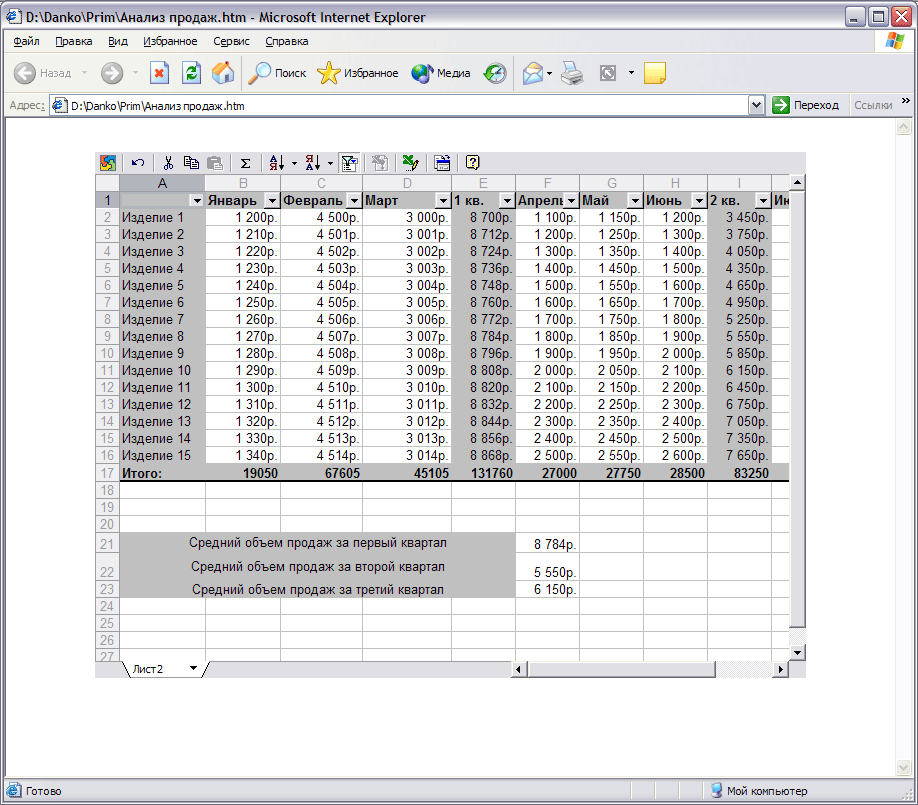


Рис. 3.5. Web-страница, подготовленная средствами Excel

Таблица 3.1. Анализ популярности программ курса

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Январь** | **Фвраль** | **Март** | **Апрель** | **Май** | **Июнь** |
| Маркетинг на предприятии |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Изучение конкурентов |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Изучение потребительской аудитории |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Изучение общественного мнения |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Анализ потребности клиентов |  |  |  |  |  |  |  |
| Изделие 10 |  |  |  |  |  |  |  |
| PR и реклама |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Оптимизация ассортимента |  |  |  |  |  |  |  |
| Изделие 15 |  |  |  |  |  |  |  |
| **Итого:** |  |  |  |  |  |  |  |

1. Создание Web-страниц средствами PowerPoint

PowerPoint позволяет сохранить презентацию в виде набора Web-страниц. Для выполнения этой операции необходимо создать презентацию по наиболее актуальной для вас теме.

Для сохранения презентации как Web-страницы необходимо выполнить команду **Файл/Сохранить как Веб-страницу,** в открывшемся диалоговом окне **Сохранение документа** установить тип файла, имя и папку назначения, а затем щелкнуть кнопку **Опубликовать**. На экране откроется диалоговое окно **Публикация веб-страницы – р**ис.3.6, которое позволяет выполнить ряд установок. В группе **Объект для публикации** необходимо указать содержимое Web-страницы, выбрать браузер для просмотра презентации потенциальными посетителями вашего сайта в группе **Поддержка обозревателя**. В группе **Опубликовать копию как…** определяются дополнительные аспекты записи Web-страницы.

|  |
| --- |
| 9_15 |
| Рис. 3.6. Диалоговое окно Публикация Web-страницы |

Кнопка **Веб-параметры** открывает диалоговое окно, в котором устанавливаются параметры Web – страницы.

Параметры вкладки **Общие** позволяют включить в состав Web-страницы содержание презентации и кнопки для перехода от слада к слайду, установить цветовую схему, которую Power Point использует при смене слайдов, обеспечить просмотр анимации.

При просмотре публикации в окне браузера в нижней части экрана расположена кнопка **Полноэкранная презентация**, нажатием на которую обеспечивается просмотр презентации.

Выполните создание простой презентации, в которой отразите Основные вопросы автоматизации маркетинговой деятельности. Сохраните эту презентацию как Web-страницу и осуществите просмотр в окне Internet Explorer.

## Создание Web-сайтов средствами MS FrontPage

1. Создание Web-сайтов средствами MS FrontPage

Выполняя предыдущие задания, вы познакомились с возможностями языка HTML для создания Web-сайтов. Хотя освоение этого языка не очень трудное дело, однако создание действительно эффективного Web-сайта –кропотливая задача, требующая значительных навыков и знаний. Упростить подготовку материалов для публикации в сети Интернет может использование средств MS Office. Однако для больших сайтов этот инструмент не всегда оказывается удобным, поскольку в нем отсутствуют возможности редактирования сайта в целом, нет сервисных средств проверки корректности подготовленных документов и т.д. Кроме того, форматирование документов средствами MS Office, рассмотренными выше, происходит по принципу WYSIWYG – соответствия его реального образа содержанию. Однако технические средства пользователей очень отличаются друг от друга и разработчик сайта может видеть его иначе, чем пользователь сети, то есть, принцип соответствия нарушается. Все это является существенным ограничением применения Ms Office (Word, Excel, Power Point) для разработки сайтов.

Вместе с тем, работу по созданию Web-сайтов можно автоматизировать, и одним из таких инструментов является MS FrontPage — специализированный редактор Web-страниц.

Начиная работу с программой FrontPage, следует помнить, что в результате получается документ HTML, построенный по тем же правилам, что и создаваемый вручную. Приложение FrontPage ориентировано на применение Internet Explorer, и создаваемый им код HTML наиболее адекватно отражается именно в этом браузере.

Создание Web-сайта состоит из трех основных этапов:

* проектирование структуры узла;
* создание страниц, входящих в структуру;
* определение связей между страницами.

MS FrontPage позволяет выполнить проектирование структуры сайта и его отдельных страниц можно выполнять любыми техническими средствами, в том числе и с помощью.

Предположим, что реально существующая фирма по производству хлебобулочных изделий пытается с помощью сети Интернет расширить свою клиентскую базу, а также найти новых поставщиков сырья.

С самого начала следует определить, из каких разделов будет состоять такой сайт, какая информация будет представлена в этих разделах, каким образом будет осуществляться навигация по сайту, а также продумать дизайн каждой страницы.

Допустим, что разрабатываемый нами сайт будет состоять из трех страниц:

Домашняя страница "Русский хлеб", сохраняемая в файле index.htm, должна содержать краткую информацию об истории фирмы и общую характеристику производимой продукции.

Страница "Прайс-лист" (файл price.htm) представляет сведения о товаре такие как название, цена и характеристика.

Страница "Заказ" (файл order.htm) должна обеспечить возможность оформить заказ и внести свои предложения производителю.

Схематично структуру такого сайта и потоки информации между страницами можно представить в виде рис. 4.1.

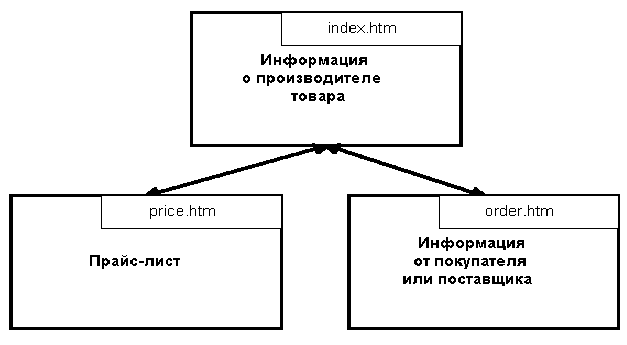


Рис. 4.1. Структура сайта

Для создания структуры Web-сайта необходимо осуществить загрузку приложения MS FrontPage и выполнить команду **File/New/Web** (Файл/Новый/Web)**,** а затем в появившемся диалоговом окне **New** (рис. 4.2) указать тип сайта **(One Page Web)** и место его расположения.

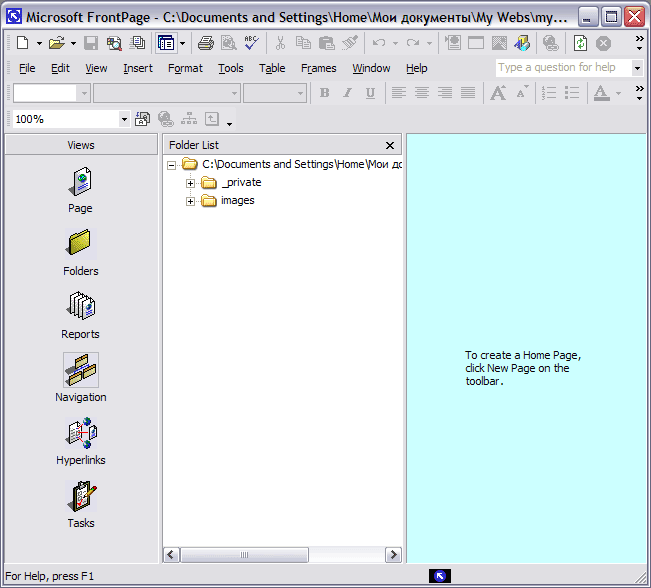


Рис. 4.2. Структура окна MS FrotnPage

Созданный сайт сохраняется в новой папке (по умолчанию My Webs), которая помещается на локальный диск или на сервер Web. Его можно впоследствии открыть с помощью команды **File/Open Web…** или выбрать из списка последних открытых узлов (команда **File/Reсent Webs**).

Окно приложения FrontPage содержит три части:

* панель **Views** (Представление) содержит ярлыки для отображения различных представлений Web-сайта;
* панель **Folder List** отображает дерево файлов и папок, составляющих физическую структуру узла;
* окно для работы с выбранным представлением.

Панель **Views** позволяет выполнить просмотр Web-сайта в следующих режимах:

* **Page view** (просмотр страницы);
* **Folders view** (просмотр папок);
* **Reports view**(просмотр отчета);
* **Navigator view** (просмотр в режиме навигации);
* **Hyperlinks view** (просмотр гиперссылок);
* **Тasks view** (просмотр задач).

Для сохранения станицы Web-сайта, расположенной в активном окне FrontPage, используется команда **File/Save** (Файл/Сохранить). Если Web-страница ни разу не сохранялась, появится диалоговое окно **Save As** (Сохранить как).

Новые Web-страницы можно создавать несколькими способами. Например, для создания страницы "Прайс-лист" установите курсор в список папок **Folder List** и выполните команду **File/New/Page** (Файл/Создать/Страница). Используя контекстное меню и команду **Rename** (Переименовать), введите имя файла price.htm. Для создания следующей страницы "Заказ", предназначенной для ввода информации от покупателей и поставщиков воспользуйтесь кнопкой **New Page** на панели инструментов. Определите имя нового файла как order.htm.

Для включения страниц в структуру сайта перетащите файлы из панели **Folder List** в окно работы в режиме **Navigation**.

Встроенный редактор FrontPage становится доступным в режиме **Page** и позволяет работать с Web- страницей одним из трех способов:

* Normal - редактирование страницы в режиме WYSIWYG, т.е. когда страница представлена так как она будет выглядеть Internet Explorer;
* HTML - редактирование страницы в текстовом представлении;
* Preview - просмотр и тестирование страницы.

Чтобы представить принципы редактирования Web-страниц, выполним открытие страницы index.htm. Существует несколько способов открытия страниц в FrontPage. Можно использовать команду **File/Open** (Файл/Открыть) или щелкнуть на кнопке **Open** (Открыть) панели инструментов **Стандартная**. Попробуйте открыть страницу, дважды щелкнув на имени файла страницы в одном из режимов - F**olders, Hyperlinks, Navigator** или Page. Web-страница откроется в режиме **Page/Normal** и будет доступна для редактирования. Введите текст, соответствующий содержанию Домашней страницы (рис.4.3). Обратите внимание, что эффектным средством структуризации документа является применение заголовков для различных абзацев текста.

Для создания заголовка установите курсор на абзаце, для которого вы хотите применить стилевое форматирование, а затем выберите один из шести уровней заголовков в списке **Style** (Стили) на панели инструментов форматирования.

Для создания логических разделов на Web-странице применяются горизонтальные линии, установка которых производится с помощью команды **Insert/Horizontal Line** (Вставка/Горизонтальная линия). Для изменения внешнего вида горизонтальной линии необходимо:

* выделить горизонтальную линию;
* из контекстного меню выбрать пункт **Horizontal Line Properties** (Свойcтва горизонтальной линии) и установить в открывшемся диалоговом окне ширину (**Width**), высоту (**Height**),цвет (**Color**) и расположение на странице (**Alignments**).

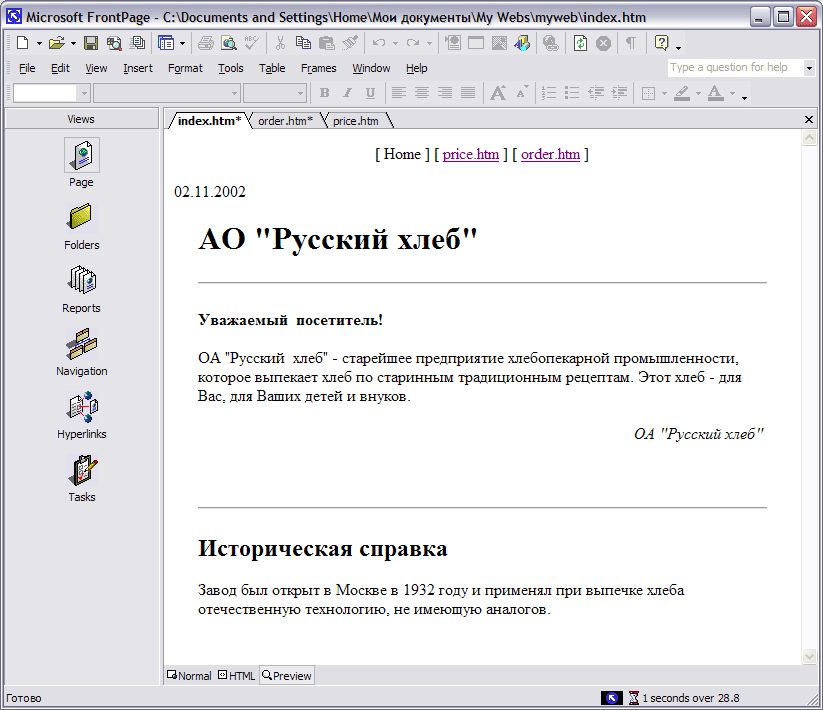


Рис. 4.3. Домашняя страница

Выполните предварительный просмотр результатов форматирования страницы, щелкнув на ярлыке **Preview**. Это режим не может воспроизвести все нюансы отредактированной страницы, как не может этого сделать никакой редактор. Окончательный просмотр Web-страницы следует производить Web-браузером. Для запуска браузера используйте команду **File/Preview in Browser** (Файл/Просмотреть браузером). В открывшемся диалоговом окне **Preview in Browser** выберите один из браузеров, установленных на вашем компьютере, установите размер окна браузера в группе параметров **Window Size** (Размер окна) и нажмите кнопку **Preview**. Если страница не была ранее сохранена, то появится диалоговое окно **Save As** (Сохранить как).

Существенное преимущество, которое дает FrontPage – это возможность управления сайтом как в процессе его создания так и при посещении страниц пользователями. Такая возможность обеспечивается общими полями и навигационными панелями. Навигационные панели предоставляют собой наборы гиперссылок, расположенные на общих полях Web-страницы. Общие поля располагаются по выбору разработчика у любого края страницы. Кроме основной – навигационной функции, общие поля и панели навигации придают Web-страницам единство стиля.

Создать навигационные панели можно следующим образом:

* Выберите команду **Format/Shared Borders** (Формат/Общие поля) в одном из режимов - **Folders**, **Hyperlinks**, **Navigator** или **Page**.
* В диалоговом окне **Shared Borders** (рис. 4.4) укажите страницы, на которых требуется отображать общие поля (на всех или на выделенных страницах), определите размещение общих полей и кнопок навигации.

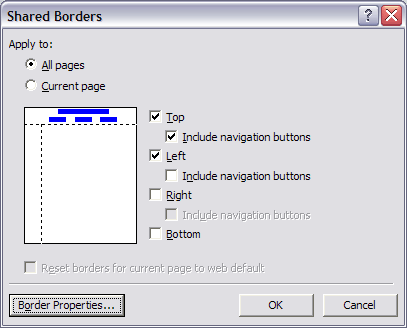


Рис. 4.4. Диалоговое Shared Borders (

Добавление панели навигации осуществляется командой **Insert/Navigation Bar**.

По мере развития Интернет оформление страниц и Web-сайтов становится все более изысканным и по сути своей превращается в отдельный вид искусства. FrontPage предоставляет широкие возможности для реализации Web-дизайна: Web-сайты можно создавать с помощью мастеров и шаблонов, а также использовать стилистические темы дизайна. Использование тем значительно упрощает процесс оформления сайта и обеспечивает единообразие в оформлении страниц, кроме того, тема может быть в любой момент изменена, удалена, применена к отдельным страницам или всему сайту в целом.

Для применения темы необходимо выполнить команду **Format/Themes** (Формат/Тема), что открывает диалоговое окно Themes для выбора темы и настройки ее свойств – рис. 4.5. Так, установка опций **Vivid Colors** (Яркие цвета) делает цвет некоторых блоков текста или графических объектов более яркими, **Active** **Graphics** (Активная графика) создает анимацию некоторых элементов Web-страницы, **Background** **Image** (Фоновый узор) добавляет текстуру фона на Web-страницу, **Apply** **Using** **CSS** (С использованием таблицы стилей) позволяет использовать каскадные таблицы стилей.

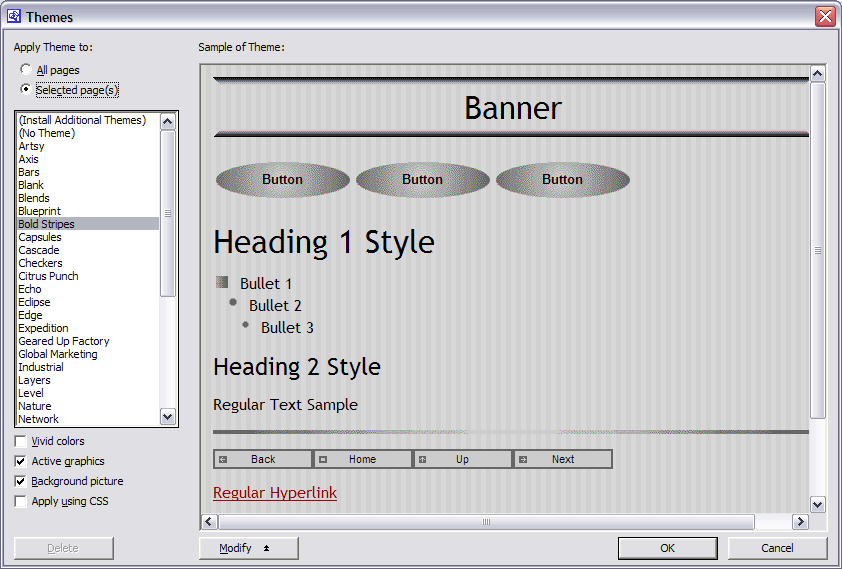


Рис. 4.5. Окно Themes (Тема)

Примените к сайту, над которым мы работаем, одну из тем и сравните полученные результаты с рис. 4.6.

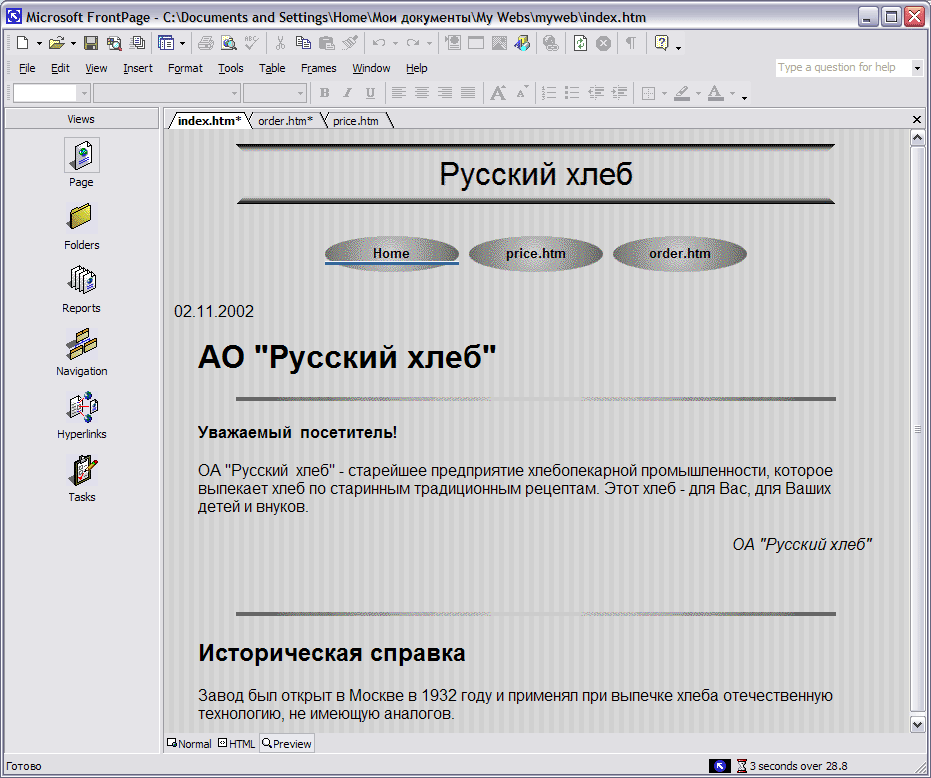


Рис. 4.6. Внешний вид Домашней страницы с использованием темы

1. Оформление таблиц средствами MS FrontPage

Использование таблиц на Web-страницах рассмотрим на примере оформления страницы "Прайс-лист", назначением которой является представлении информации о товарах, производимых фирмой.

При разработке Web-сайтов применение таблиц обеспечивает структурирование информации любого типа: числовой, тестовой и даже графической. В зависимости от типа представляемой информации таблица может иметь традиционный вид, то есть с линиями обрамления вокруг таблицы и составляющих ее ячеек, или без линий обрамления, что часто используется для представления графической информации. Разместим на странице "Прайс-лист" таблицу, содержащую название и изображение внешнего вида товара. Для этого:

* в режиме **Page** выберем файл price.htm и установим курсор в то место Web-страницы, где мы предполагаем разместить левый угол таблицы;
* создадим на странице "Прайс-лист" таблицу, состоящую из четырех строк и трех столбцов, используя команду **Table/Insert/Table** (Таблица/Вставить/Таблица). При появлении диалогового окне Insert Table (Вставить таблицу) в поле **Rows** (Число строк) и в поле **Columns** (Число столбцов) введем число 3.
* заполним таблицу информацией в соответствии с рис. 4.7. Остальную информацию, представленную на странице, введем несколько позже.

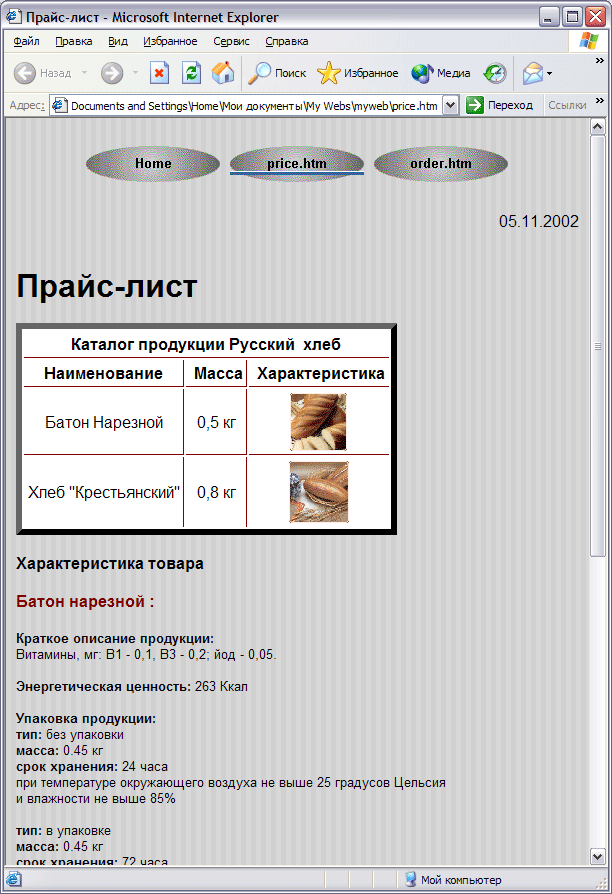


Рис. 4.7. Web-страница Прайс-лист

После создания таблицы вы можете изменить ее вид в диалоговом окне **Table** **Properties** (Свойства таблицы). Выберите для этого любую ячейку таблицы и выполните команду **Table/Properties/Table** (Таблица/Свойства/Таблица). В диалоговом окне **Table** **Properties** (рис. 4.8) установке доступны следующие опции:

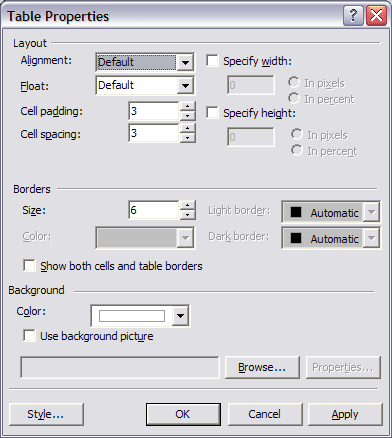


Рис. 4.8. Диалоговое окно Table Properties (файл F\_Table\_Properties)

**Группа Layout (расположение)**:

**Alignment** (Выравнивание) – устанавливает вид выравнивания таблицы по горизонтали относительно ширины Web-страницы;

**Float** (Обтекание) – позволяет расположить текст, не входящий в таблицу слева или справа от таблицы.

**Cell padding** (Поля ячеек) – определяет размер отступов в пикселях от краев ячейки до заполняющих ее элементов ( по умолчанию принимается равным 1);

**Call spacing** (Шаг ячейки) – определяет расстояние в пикселях между прилегающими ячейками ( по умолчнию принимается равным 2).

**Группа Borders (границы)** изменяет толщину линий и цвета обрамления таблицы. Обрамление состоит из двух линий – светлой и темной и применение разных цветов к ним создает эффект трехмерного изображения:

**Light Border** (Светлая граница) – определяет цвет светлой линии;

**Dark Border** (Темная граница) – определяет цвет темной линии;

**Border** (Граница) – определяет цвет обрамления, если опции **Light Border** и **Dark** **Border** определены по умолчанию (Default)/

**Группа Background (Фон):**

**Color** (Цвет) - позволяет установить цвет фона таблицы;

**Use background picture** (Использовать фоновый рисунок) – позволяет выбрать рисунок, который будет выбран в качестве фона в таблице.

При желании вы можете разместить заголовок перед таблицей. С этой целью:

* Установите курсор в любую ячейку таблицы и выполните команду **Table/Insert/Caption** (Таблица/Вставить/Подпись). Указатель вставки текста переместится над таблицей по центру.
* Введите текст подписи, например, дату обновления таблицы.

1. Создание ссылок и форм на страницах средствами MS FrontPage

Напомним, что гиперссылка – это связь, ведущая с одной Web-страницы на другую страницу, картинку, адрес электронной почты или файл, расположенный за пределами сайта, который вы создаете или просматриваете. Это позволяет подразделить гиперссылки следующим образом:

* + Локальные (или внутренние) гиперссылки - указывают на ресурс текущего сайта.
  + Внешние гиперссылки – указывают на ресурсы прочих сайтов, находящихся в сети Интернет, ссылки на закладки внутри страницы, ссылки на адреса электронной почты и т. д.

Для создания гиперссылок необходимо понимать, что этот процесс состоит из двух частей. Первый заключается в выборе текста или объекта на странице, который определяется как гиперссылка , а второй – в указании целевого ресурса, то есть файла или страницы (места на странице), куда осуществляется переход при щелчке по гиперссылке.

Просмотр всех ссылок, относящиеся к выбранной странице, можно осуществить в режиме **Hyperlinks**.

Рассмотрим некоторые способы создания гиперссылок на примере дальнейшего оформления Web-страницы "Прайс-лист". Сначала введите всю исходную информацию на редактируемую страницу в соответствии с рис. 4.7. Затем для создания ссылки на закладку необходимо:

* Создать закладки с использованием команды **Insert/Bookmark** (Вставить/Закладка).
* Выделить текст или объект, при указании на который требуется перейти по гиперссылке и выполнить команду **Insert/Hyperlink** (Вставка/Гиперссылка).
* В появившемся диалоговом окне – рис. 4.9, выделить страницу, на которой размещены закладки, и выбрать нужную закладку из списка **Bookmark** (Закладка).

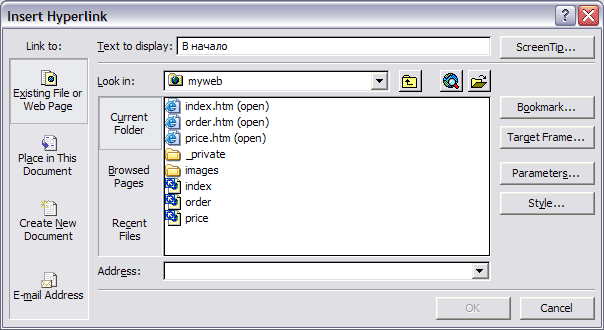


Рис. 4.9. Диалоговое окно Insert Hyperlinks ( создание гиперссылки)

Выполните операцию создания гиперссылок, обеспечивающих переход от рисунков из столбца Характеристики собственно к тексту страницы, где помещены характеристики товара.

Для создания гиперссылок, осуществляющих переход к другим целевым ресурсам, внимательно изучите возможности диалогового окна **Insert Hyperlink**.

Кроме обыкновенных ссылок в виде текста или рисунков, приложение FrontPage позволяет создавать так называемые карты – графические изображения, которым назначена ссылка. Щелчок по активной зоне позволяет перейти по указанной ссылке. Можно назначить ссылку целому рисунку или отдельному фрагменту. Рассмотрим эту операцию на примере оформления Web-страницы "Заказ". Подготовьте небольшой рисунок, сохранив его в папке **Images** вашего сайта, и только потом вставьте его на страницу "Заказ". Далее создание карты необходимо производить в следующей последовательности:

* выберите рисунок на странице. На экране отобразится панель инструментов **Pictures**;
* используя инструмент рисования прямоугольника, эллипса, многоугольника, нарисуйте контур на активной зоне изображения;
* вставьте гиперссылку, используя появившееся диалоговое окно **Edit** **Hyperlink**.

При этом будем считать, что та часть рисунка, в которой расположен текст "Отдел маркетинга" должна иметь гиперссылку на форму заказа товара, а текст "Web-администратор" – гиперссылку на адрес электронной почты web-администратора. Форма заказа может размещаться на этой же странице. Введите заголовок формы в нижней части страницы и установите перед ним закладку, к которой будет осуществляться переход при нажатии на текст "Отдел маркетинга". Страницы order.htm представлен на рис.4.10.

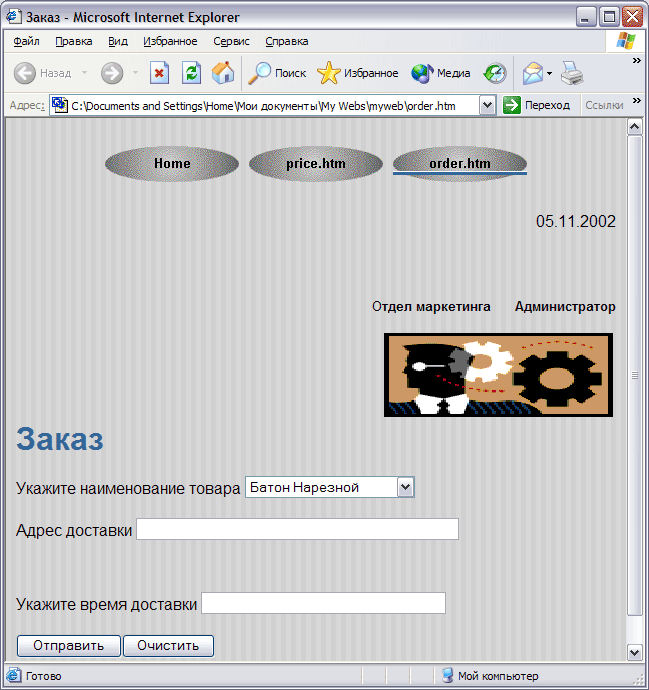


Рис. 4.10. Фрагмент страницы оrder.htm

Кроме гиперссылок на этой странице представлено использование – распространенного и эффективного способа установления контактов между пользователем и владельцем Web-сайта. Формы могут создаваться несколькими способами. Один из них – использование шаблонов, когда разработчик Web-страницы выбирает подходящую форму из набора имеющихся шаблонов. В случае, когда ни один из предлагаемых шаблонов вас не устраивает, можно воспользоваться мастером форм **Form** **Page** **Wizard**. Для этого при создании новой страницы из списка предлагаемых шаблонов выбирается шаблон **Form** **Page** **Wizard**. Проектирование формы выполняется в этом случае в диалоговом режиме в соответствии с запросами мастера.

Некоторые разработчики предпочитают разрабатывать формы самостоятельно. Для этого с помощью панели инструментов **Form** (Форма) создается новая форма, в которую затем помещаются элементы управления, текст и графические объекты.

Процесс создания формы состоит из нескольких этапов. Сначала необходимо определить, какие данные мы хотим получить от потенциальных посетителей нашей страницы. На следующем этапе следует разместить элементы управления, выбор которых определяется свойствами данных. Последний этап состоит в настройке обработчика полученных данных.

Каждый элемент управления формы имеет набор свойств, определяющих название элемента, установки по умолчанию, внешний вид, тип принимаемых данных. Для открытия диалогового окна, в котором задаются свойства, необходимо выбрать элемент управления, вызвать контекстное меню, нажатием правой кнопки мыши, и выполнить команду **From Field Properties** (Свойства поля формы).

Для ввода фамилии, имени, отчества и другой текстовой информации целесообразно выбрать однострочное текстовое поле On-line Text Box. Это поле имеет следующие свойства, отраженные в диалоговом окне ( рис. 4.11):

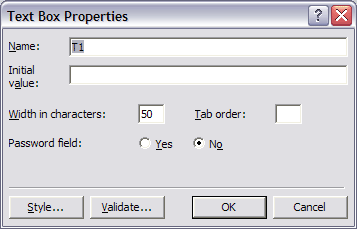


Рис. 4.11. Диалоговое окно Text Box Properties

**Name** (Имя) – имя элемента управления, служащее для идентификации содержащихся в нем данных.

**Initial value** (Начальное значение) – символы, отражающиеся после открытия формы.

**Width in characters** (Количество символов) – длина текстового поля в символах.

**Tab order** (Порядок перебора) – определяет порядок перебора элементов управления при нажатии клавиши **Tab**.

**Password field** (Пароль) – устанавливается **Yes** при необходимости ввода пароля.

Для фиксации заказа товара следует использовать поле **Check Box** (Флажок). Свойства этого поля состоят описания его имени, пересылаемого на сервер значения и указания и необходимости его установления при открытии формы.

Поле формы, позволяющее выбрать вид доставки товара, может быть представлено раскрывающимся списком, Диалоговое окно свойств этого элемента управления представлено на рис. 4.12.

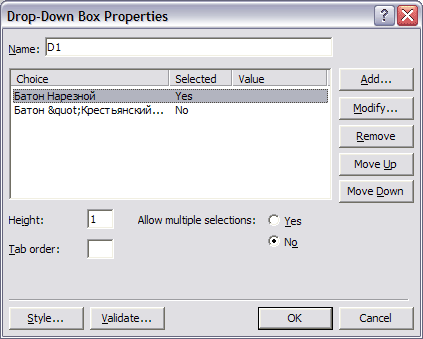


Рис. 4.12. Диалоговое окно Drop-Down Menu

В этом окне предоставляется возможность заполнить или изменить раскрывающийся список:

* кнопка **Add** (Добавить) - добавление элементов списка;
* кнопка **Modify** (Изменить) –изменение имени, значения или начального состояния записи;
* кнопка **Remove** (Удалить) – удаление строки из списка;
* **Move Up** (Вверх) и **Move Down** (Вниз) –изменение расположения записей.

По умолчанию в диалоговом окне установлена опция, позволяющая выбрать только одну запись из списка. Для разрешения множественного выбора в строке **Allow** **multiple** **selections** (разрешить множественный выбор) необходимо установить значение переключателя в позицию **Yes** (Да).

При самостоятельном создании формы необходимо указать средство для обработки данных, пересылаемых на сервер. С этой целью откройте контекстное меню, щелкнув правой кнопкой мыши в любом месте формы, и выберите строку **Form** **Properties** (Свойства формы) - рис. 4.13.

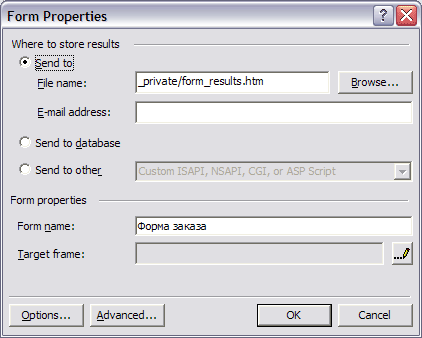


Рис. 4.13. Диалоговое окно Form Properties

В этом диалоговом окне предлагаются различные средства обработки формы:

* **Send to File name** (Направить в файл);
* **Send to E-mail** (Направить по электронной почте);
* **Send to database** (Сохранить в базе данных);
* **Send to other** (Другое).

После выбора способа обработки данных, например, направления данных в файл, необходимо установить опции в диалоговом окне **Options for Saving Results of Form** (Параметры сохранения результатов формы) – рис.4.14. Доступ к этому окну осуществляется нажатием кнопки Options (Параметры) диалогового окна **Form Properties** (Свойства формы).

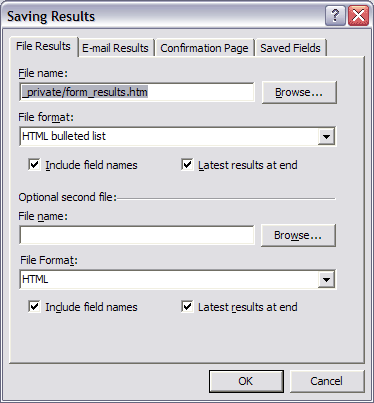


Рис. 4.14. Диалоговое окно Options of Form

Данное окно имеет четыре вкладки:

* **File Results** (Запись результатов в файл);
* **E-mail Results** (Отправка результатов по электронной почте);
* **Confirmation Page** (Страница подтверждения);
* **Saved Fields** (Сохраняемые поля).

При пересылке данных в файл на соответствующей вкладке устанавливаются имя файл для сохранения данных формы (**File name**), формат файла (**File format**), а также опции, позволяющие пересылать данные в паре с именем поля (**Include field names**) и добавлялись в конец файла HTML (**Latest results at and**). Если направление данных происходит в текстовый файл, то они всегда дописываются в конец.

После создания формы, откройте Web-сайт в браузере. Перейдя на страницу "Заказы", вы можете очистить форму или ввести в нее свою информацию. Данные формы сохраняются сервером в формате, который вы определили в окне **Options for Saving Results of Form** (Параметры сохранения результатов формы).

1. Контроль над разработкой Web-сайта

Создание Web-сайта – кропотливая планомерная работа, которая требует держать под контролем не только большой объем информации, предполагаемый разместить на Web-страницах, но и этапы создания каждой страницы, качество подготовленного материала, корректность гиперссылок и т.д. Программа FrontPage содержит средства, позволяющие управлять процессом создания и сопровождения Web-сайтов.

Для получения информации о создаваемом узле и ее анализе, можно воспользоваться режимом Reports view (просмотр отчета) на панели **Views**. При нажатии на значок **Reports** (Отчеты) на панели инструментов появится панель **Report** (Отчеты), на которой в раскрывающемся списке следует выбрать тип отчета. Стандартный режим –**Site** **Summary** (Обзор сайта) – рис. 4.15.

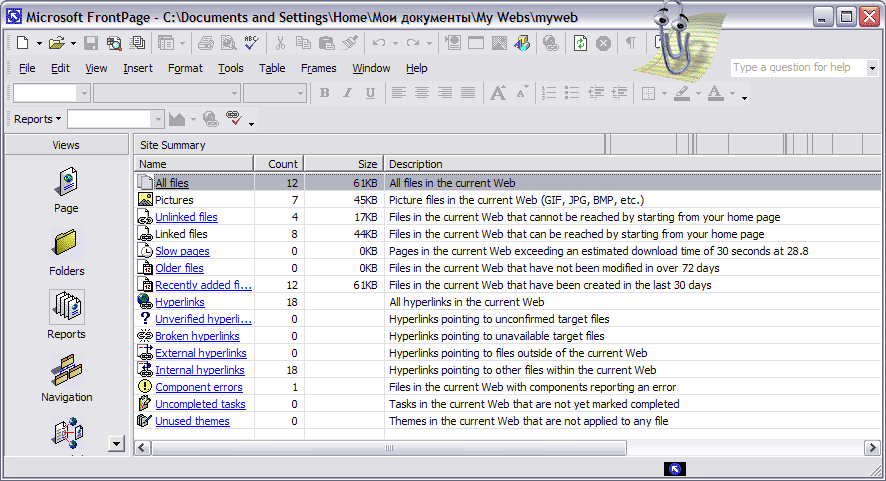


Рис. 4.15. Общий отчет о содержании Web-сайта

В этом режиме приводится общая информация о страницах сайта. В отчете **All** **Files** (Все файлы) указывается общее число файлов и их суммарный размер. Отчет **Pictures** (Изображения) дает информацию о числе и размере изображений. Сроки **Linked** **Files** (Связанные файлы) и **Unlinked** **Files** (Несвязанные файлы) показывают, к каким файлам есть доступ по гиперссылкам с начальной страницы, а к каким - отсутствует. Строка **Slow** **pages** (Медленные страницы) дает информацию о медленно загружающихся страницах. Задание ограничения на время загрузки страницы выполняется с использованием команды **Tools**/**Options**/вкладка **Reports** **View** (Сервис/Параметры/ вкладка Отчеты). Страница считается медленной, если время ее загрузки превосходит значение счетчика "**Slow** **page**" **take** **at** **least**… ("Медленные страницы загружаются как минимум…"). Расчет времени загрузки производится с учетом предполагаемой скорости соединения, выбрать которое можно из списка **Assume connection speed** (Считать скорость соединения равной).

Для просмотра отчетов, представляющих информацию о корректности гиперссылок на сайте, используется строка **Broken hyperlinks** ("Мертвые" ссылки). Этот режим позволяет выявить наличие ссылок на несуществующий ресурс.

Кроме анализа содержимого сайта, очень важно следить за ходом работ по созданию, модификации и обновлению Web-страниц. Для этого руководитель проекта составляет список задач, с указанием наименования работы и сроках ее выполнения, а также информации об ответственном за результат лице.

Во FrontPage предусмотрена автоматизация ведения такого списка. Для ввода новой задачи и просмотра списка уже созданных задач на панели **Views** (представление) используется режим **Тasks view** (просмотр задач). Новая задача в этом режиме включается с помощью команды **File/New/Tasks** (Файл/Создать/Задача), выполнение которой приводит к открытию диалогового окна **New Task** (Создание задачи). Ввод информации в соответствующие поля этого диалогового окна позволяет ввести информацию, необходимую для контроля за ходом выполнения работ.

Доступ к задаче осуществляется двойным щелчком на строке с описанием задачи в окне **Tasks**.

Тестирование и публикация Web-узла

Поскольку конечной целью создания Web-сайта является его представление на суд многомиллионной аудитории пользователей сети Интернет, очень важным является этап его тестирования. Задача тестирования Web-сайта с одной стороны состоит в оценке корректности представленной на нем информации, а с другой стороны - в проверке работоспособности сайта не зависимо от браузера, выбранного пользователем для просмотра информации в WWW, и характеристик технических средств, которыми он располагает.

При проверке корректности информации следует обращать внимание на следующее:

* достоверность публикуемого материала, его стилистику и орфографию;
* наличие ссылок на публикации при цитировании материалов других авторов;
* правильность работы гиперссылок, присутствующих на вашем сайте;
* существование информационных ресурсов при ссылках на URL-адреса в сети Интернет;
* корректность работы форм на Web-страницах и наличие возможности связаться по электронной почте не только с web-мастером, но и с представителями фирмы или предприятия, чья информация представлена на сайте.

При тестировании сайта важно учитывать технические характеристики компьютеров потенциальных посетителей. Страницы следует просмотреть вне зависимости от их дизайна на экранах с разной разрешающей способностью с помощью команды редактора **FrontPage File/Preview in Browser** (Файл/Просмотр браузером).

После завершения всех этапов создания и тестирования Web-сайта его можно опубликовать как в сети Интернет так и в локальной сети. Самый лучший способ - разместить сайт на сервере своего провайдера или корпоративном сервере. В этом случае вам потребуется консультация администратора Web-сервера по вопросам способов и параметров связи при публикации, а также принятой системы адресов новых ресурсов. Публикация сайта может выполняться с помощью Мастера издания Web (кнопка Пуск/Программы/Стандартные/Средства Интернет) или средствами, встроенными во FrontPage, выполняя команду **File/Publish Web** (Файл/ Опубликовать Web-сайт).

Для размещения Web-узла на домашнем персональном компьютере или в корпоративной сети, а также для разработки и тестирования Web-сайта перед его размещением у поставщика услуг Интернет.можно использовать сервер **MS Personal Web Server**.

Одним из наиболее доступных способов публикации своего сайта является его размещение на публичном российском сервере CHAT.RU. Этот способ включает несколько этапов:

* **Регистрация.**

Откройте страницу <http://www.chat.ru>/user и зарегистрируйтесь на сервере, последовательно заполняя поля регистрационной карточки пользователя. После успешной регистрации ваша страница будет иметь URL-адрес вида: [http://www.char.ru/`Имя\_пользователя](http://www.char.ru/%60Имя_пользователя). При каждом посещении сервера CHAT.RU вам потребуется ввести имя пользователя и пароль в окно ввода сетевого пароля.

* **Загрузка файлов на сервер.**

Помните, что в соответствии с требованиями администратора сервера CHAT.RU, все загружаемые файлы должны быть собраны в архивный файл, имена файлов должны состоять из строчных символов. Перед началом загрузки страницы или файла в поля предлагаемой формы следует ввести информацию о полном имени загружаемого файла и варианте кодировки страниц. Загрузка файлов осуществляется с помощью кнопок UpLoad+UnZip (загрузка файла архива) или UpLoad(загрузка Web-страницы).

* **Контроль.**

Для контроля щелкните на гиперссылке File List и изучите отчет сервера о состоянии файлов пользователя.

* Просмотр Web-cайта.

Введите в строку адреса браузера URL-адрес Домашней страницы и выполните тестирование сайта.

## *СПИСОК ЛИТЕРАТУРЫ*

1. Михеева, Е. В. Практикум по информационным технологиям в профессиональной деятельности: учеб. пособие для студ. сред. про. образования/Е. В. Михеева. – 7-е изд., стер. – М.: Издательский центр «Академия», 2011.
2. Фуфаев, Е. В. Пакеты прикладных программ: Учеб. пособие для сред. проф. образования/Э. В. Фуфаев, Л. И. Фуфаева. – М.: Издательский центр «Академия», 2011.
3. Петров, М.Н. Компьютерная графика : учебник / М.Н. Петров, В.П. Молочков – СПб. : Питер, 2011.
4. Баканов, В.М. Программирование мультимедиа-систем : учебное пособие / В.М. Баканов. – М. : МГАПИ, 2010
5. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин – М. : ДИАЛОГ-МИФИ, 2008. – 384 с.
6. Карлащук, В.И. Презентация на компакт-диске / В.И. Карлащук – М. : СОЛОН-Р, 2012.
7. Кречман, Д.Л. Мультимедиа своими руками/ Д.Л. Кречман, А.И. Пушков – СПб. : BHV–Санкт-Петербург, 2014.

1. \* Содержимое головной части не выводится в окне браузера, а содержит данные о документе. [↑](#footnote-ref-1)
2. CGI-скриптом называют программу, работающую на сервере (на котором лежит www-страничка).Эта программа получает с машины клиента данные. Данные посылаются, например, из формы по нажатию кнопки **Обработать заказ**. Данные поступают на сервер в программу CGI, обрабатываются и результат посылается клиенту. Гипертекстовая страничка на машине клиента выводит на экран надпись (например: «Заказ обработан и принят»). [↑](#footnote-ref-2)
3. ASCII – Американский стандартный код по обмену информацией [↑](#footnote-ref-3)