ИНТЕРНЕТ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

П.Б. Храмцов С.А. Брик
А.М. Русак А.И. Сурин

Введение в HTML

2003

Введение в HTML
Авторы: П.Б. Храмцов, С.А. Брик, А.М. Русак, А.И. Сурин

Информация о курсе
Курс посвящен изучению языка гипертекстовой разметки HTML.
Рассматриваются основные конструкции языка, приемы разметки и связь с другими инструментами разработки WEB-страниц.

Цель
Главная цель курса – научить студентов создавать свои собственные сайты в Internet.

Лекции

1.

Общие сведения

В этой лекции рассказывается о принципах построения гипертекстовых информационных систем, определяется роль языка гипертекстовой разметки HTML в построении сайтов глобальной компьютерной сети Internet, определяются формат и структура HTML-документов.

2.

Структура HTML-документа и элементы разметки заголовка документа

В этой лекции разбираются типовая структура HTML-документа и содержание его заголовка. Подробно описывается содержание элементов разметки, и разбираются формат и назначение каждого из элементов разметки заголовка.

3.

Контейнеры тела документа

В этой лекции рассказывается об элементах разметки тела HTML-документа, подробно разбирается их типизация, назначение и применение.

4.

Графика

В этой лекции рассматриваются принципы применения графических образов при HTML-разметке.

5.

Таблицы в HTML

В этой лекции подробно рассматриваются принципы применения таблиц в HTML-разметке. Это и табличная организация текста, и табличная координатная сетка, и организованная в таблицы графика.

6.

HTML-формы

В этой лекции разбирается возможность взаимодействия читателя HTML-страниц с сервером Web-узла. Подробно описывается подмножество HTML, которое носит название HTML-формы.

7.

Фреймы

В этой лекции рассматриваются различные способы фрагментирования содержания Web-узла при помощи механизма HTML-фреймов.

1. Лекция: Общие сведения

В этой лекции рассказывается о принципах построения гипертекстовых информационных систем, определяется роль языка гипертекстовой разметки HTML в построении сайтов глобальной компьютерной сети Internet, определяются формат и структура HTML-документов.

Общие принципы создания Web-узла

Вы решили создать и разместить в информационном пространстве WWW (World Wide Web, Всемирная паутина) собственный Web-узел. Какие же шаги надо предпринять, чтобы он был интересен, полезен и, что немаловажно, посещаем. Первый вопрос, на который необходимо дать четкий ответ: с какой целью создается Web-узел? От этого зависит многое: стиль оформления, необходимые для создания и последующего функционирования затраты, формат представления информации для размещения в Web, инструментарий и требования, предъявляемые к программному обеспечению Web-сервера и каналам связи с Internet. Здесь возможно несколько вариантов.

Если вы создаете Web-узел для компании, реализующей какой-то товар, то основной целью может быть распространение информации о фирме и реклама продукции, а также организация Web-магазина. При этом будут решены следующие задачи:

- изменение имиджа и поднятие престижа компании;
- продвижение торговой марки;
- доступность информации о продукции и ценах для клиентов;
- поддержка дилерской сети, доступность информации о продукции и ценах для дилеров;
- прямая продажа продукции в Internet, организация Web-магазина;
- доступность внутренней информации для сотрудников, работающих вне офиса.

Другой вариант — создание Web-узла научной или общеобразовательной организации, не занимающейся коммерцией в Internet, а распространяющей информацию. В этом случае речь пойдет о сборе, переработке и размещении на Web-узле больших массивов данных с организацией поиска и доступа к ним.

И последний рассматриваемый вариант — когда вы считаете необходимым разместить в Internet свою личную страницу.

Для того, чтобы правильно ответить на поставленные вопросы, необходимо сформировать категории пользователей, на которые рассчитан Web-узел. Исходя из их психологии, должна строиться информационная структура, которая будет привлекать и удерживать клиентов. В дальнейшем все вопросы о целесообразности каких-либо действий, связанных с Web-узлом, должны рассматриваться в соответствии с тем, как отреагируют на них посетители, и насколько они будут способствовать достижению главной цели.

После того, как сформулированы цели и определены категории пользователей, необходимо распределить подготовленную информацию по Web-документам, продумать связи между ними и предусмотреть дополнительные навигационные возможности, например, поисковую систему по содержимому Web-узла.

Типичная структура Web-узла фирмы обычно представлена так:

Информация о компании. Следует рассказать о целях и деловом облике фирмы, ее истории и т.д. Покажите, какую выгоду получат клиенты от сотрудничества именно с вами, а не с другими компаниями.

Информация о продукции и услугах. Разместите на Web-странице фотографии или рисунки своей продукции. Опишите ее свойства и преимущества, приведите примеры использования. Если имеется бумажный каталог продукции, то можно перенести его

структуру и содержание в Web-узел. Это облегчит создание и дальнейшее обновление электронного варианта каталога. Если планируется прием заказов на продукцию или услуги через Internet, то нужно разместить здесь бланк заказа, который будет поступать по электронной почте.

Информационная поддержка. В этом разделе публикуется дополнительная техническая информация, часто задаваемые вопросы, советы по устранению неисправностей и т.п.

Новости. Проинформируйте клиентов о новых товарах и услугах, предоставляемых фирмой, опубликуйте пресс-релизы и т.п.

Обратная связь. Сообщите, как с вами можно связаться, где вы находитесь. Поместите форму для отзыва, гостевую книгу, адреса электронной почты, на которые клиент может отправить запрос, и т.п.

При наполнении Web-узла всегда нужно помнить два принципа: уникальность и достоверность публикуемых материалов.

Уникальность является первоочередным требованием к содержанию. В **WWW** уже может существовать немало страниц с похожими материалами. Ваш Web-узел должен чем-то отличаться от серверов с аналогичной тематикой, хотя бы для того, чтобы привлечь к себе внимание. Наличие уникальных материалов на вашей странице увеличит ее посещаемость. Для того, чтобы создать уникальный информационный ресурс, не обязательно изобретать что-то принципиально новое, можно по-другому оформить уже существующие ресурсы, но при этом не заставлять клиента тратить много времени на их поиск. Проверить же ресурсы на уникальность можно с помощью поисковых серверов. Что касается авторитетности, то все зависит от того, насколько тщательно вы подберете информацию, проверите ее и будете своевременно обновлять.

При создании Web-узла необходимо помнить, что составляющие его отдельные документы должны быть объединены общим стилем оформления и средствами навигации. Единый стиль оформления — один из показателей, отличающих профессиональный Web-узел от любительского. Благодаря единообразно сделанным документам пользователи будут отличать ваш Web-узел от других и запомнят его. Это не значит, что документы должны быть похожи друг на друга как две капли воды, но общая идея, единый стиль, должны присутствовать непременно.

То же относится и к средствам навигации по страницам. Не стоит рассчитывать, что посетитель знает структуру Web-узла так же хорошо, как вы. Он должен без труда понимать, где он находится сейчас и как можно попасть в любое другое место. Необходимо предусмотреть возможность перехода к первому документу, программе поиска или к схеме Web-узла.

Кроме того, единство стиля позволяет использовать шаблоны — страницы, содержащие только общие элементы оформления и навигации (без информационного наполнения). С их помощью можно быстро и эффективно создавать новые страницы и распределять работу по их созданию между несколькими людьми. При использовании шаблона для получения готовой страницы достаточно лишь внести в него необходимую информацию. Последовательность, логичность, постоянство — вот необходимые качества хорошего Web-узла. Значительно упростят работу по формированию и изменению стиля вашего Web-узла каскадные таблицы стилей, появившиеся в HTML 4.0. О некоторых их возможностях будет рассказано ниже, а полностью им посвящена отдельная глава.

После того, как определены цели, задана структура и собрана текстовая и графическая информация, необходимо разработать внешний вид Web-узла. Он также зависит от тех целей, которые необходимо достичь. Спектр возможных решений здесь очень широк:

от просмотра уже существующих страниц и создания подобных, до обращения за помощью к профессиональным дизайнерам и художникам. В то же время, необходимо помнить о некоторых уже сложившихся правилах построения Web-документов, из которых состоит Web-узел.

Структура. На сегодня представление о структуре документа достаточно устоялось. Web-документ должен содержать в себе следующие разделы: заглавие, название компании, навигационную панель, собственно содержание, контактную информацию, дату и время обновления, авторские права и статус документа.

Логотип. Создавая Web-страницу, необходимо позаботиться о том, чтобы название фирмы всегда присутствовало на экране. Для этого в начале каждого Web-документа обычно помещается красочно оформленный логотип фирмы. Кроме того, название компании должно присутствовать и в выходных данных ко всем документам.

Навигационная панель. Одним из наиболее важных разделов Web-документа является навигационная панель или панель управления. **WWW** завоевала весь мир во многом благодаря тому, что гипертекстовые ссылки обеспечивают полную связность публикуемых материалов. Но эти же ссылки таят в себе опасность погружения в полный хаос, когда, пройдя цепочку из трех-четырех документов, вы уже не сможете вернуться обратно, запутавшись в обилии ссылок. Ваш Web-узел должен обеспечивать пользователю ясные и интуитивно понятные навигационные маршруты.

Многочисленные исследования показали, что посетители Web-серверов очень нетерпеливы и дальше, чем на два уровня документов, углубляться в содержание сервера не хотят. Поэтому, создавая Web-узел большого объема, следует предусмотреть промежуточные документы, обычно находящиеся на первом-втором уровнях, от которых любая информация находится не далее, чем в двух переходах.

Навигационная панель вашего Web-узла должна присутствовать в каждом документе. В первую очередь, она должна включать в себя направляющие ссылки типа "Вперед"-"Назад" ("Следующий"-"Предыдущий"), указывающие на соседние документы в структуре Web-узла. Далее от панели управления обязательно должны идти ссылки на все крупные разделы Web-узла — так называемые разделы первого уровня. И, наконец, пользователь всегда должен иметь возможность мгновенно вернуться на главную страницу Web-узла. Помимо ссылок следует указать путь к локальной поисковой системе и индексу.

Содержание. Прежде всего, следует отметить, что содержание Web-документов должно в полной мере отвечать всем требованиям, предъявляемым к обычным газетным или журнальным публикациям: грамматическая и орфографическая корректность, точность и достоверность предлагаемых материалов и многое другое. Кроме того, появляется целый ряд специфических требований, которым должен удовлетворять Web-документ.

Часто возникает вопрос о размерах документа: какое число страниц является оптимальным? Ответ на первый взгляд может показаться странным: одна экранная страница или вообще никаких ограничений. Многочисленные исследования показали, что пользователи не любят работать с полосами прокрутки браузеров. Больше всего им нравятся документы, которые размещаются на одной экранной странице. Так и в **WWW** — вы никоим образом не сможете дать пользователю больше информации, чем в концентрированном изложении на одной странице. Если все-таки вы не укладываетесь в эти рамки, создайте еще один документ.

Одна экранная страница оказалась подходящей мерой представления информации. Если размер документа превышает одну страницу, то в большинстве случаев он может быть поделен на несколько логических частей, каждая из которых будет занимать не более одной страницы. Если же логического деления информации произвести не

удается, то необходимо переработать стиль изложения, а может быть, и сами материалы. Сейчас выработалось единое мнение, что Web-сервер необходимо строить на основе одноэкранных документов. Есть только два исключения из этого правила. Оно не распространяется на статьи, публикуемые в **WWW**, и второе исключение — анкетные формы, которые, естественно, нельзя разрывать.

Графика. При разработке Web-страницы нужно очень внимательно выбирать оптимальное соотношение графических и текстовых материалов. Одна хорошая картинка может заменить тысячу строк текста, но и загружаться по сети она будет в тысячу раз дольше. Поэтому графикой нужно пользоваться осторожно. Можно исходить из того, что графики на странице должно быть чуть меньше, чем хочется Web-мастеру. Пользователям может просто не хватить терпения, и они закроют документ еще до того, как он полностью загрузится. Задержка отклика системы вызывает у пользователя раздражение. Все понимают, как тяжело сейчас обстоят дела с канальной инфраструктурой в Internet. Поэтому время задержки возрастает в зависимости от времени суток, по разным оценкам до 15-60 секунд. Теперь представьте, что у клиента только модем на 19200 бит/с. Большего на российских телефонных линиях достичь очень тяжело. Тогда за минуту, то есть до того, как клиент потеряет терпение, можно передать только около 170 Кбайт данных. Следовательно, размер документа не должен превышать этого значения.

Следует отметить, что обычно панель управления, логотип и название фирмы выполняются в виде графических элементов. После создания макета можно приступить к его реализации с помощью языка HTML и иных средств, предлагаемых современными технологиями **WWW**.

Завершив создание Web-узла, необходимо разместить его в Internet. Здесь возможны два варианта: первый — использовать компьютер, который вместе с Web-сервером и Web-узлом находится в вашем офисе и подключается к Internet по выделенной или коммутируемой линии; второй — воспользоваться для размещения Web-узла услугами специальных организаций.

Рассмотрим второй вариант. Правильный выбор провайдера, предоставляющего доступ к Web-странице, позволит вашим клиентам с максимальным удобством получать необходимую информацию. Кроме того, поддержка Web-сервером специальных возможностей значительно облегчит разработку Web-узла.

На что следует обратить внимание при выборе провайдера, размещающего ваш Web-узел на своем сервере?

Пропускная способность каналов. Чтобы вашим посетителям не пришлось слишком долго ждать загрузки страниц, провайдер должен обладать надежным высокоскоростным соединением порядка 1-2 Мбит в секунду.

Поддержка сервером провайдера SSI (Server Side Includes, вставки на стороне сервера). Использование SSI позволяет Web-серверу вставлять небольшие объемы динамических данных непосредственно в пересылаемый пользователю HTML-документ. Запрошенная HTML-страница "просматривается" в поисках элементов SSI. Обнаружив такой элемент, сервер вставляет требуемую динамическую информацию. С помощью SSI можно включать один файл в состав другого, исполнять CGI-сценарии и передавать другую информацию. Необходимо уточнить, какие именно функции SSI поддерживаются на сервере провайдера.

Поддержка сервером провайдера CGI-сценариев. CGI (Common Gateway Interface, общий шлюзовой интерфейс) — спецификация, позволяющая Web-серверу выполнять произвольные прикладные программы. В результате работы таких программ (сценариев, или "скриптов") создаются HTML-документы. С помощью CGI-сценариев могут приниматься данные от пользователя, они позволяют организовать диалог на

Web-страницах, запросы к базам данных и т.д. Создать CGI-сценарий можно с помощью любого популярного языка программирования: Perl, Basic, C, C++, Pascal и т.п.

Поддержка моментальной перекодировки. К сожалению, для русского языка в Internet при работе на разных платформах (Windows, Mac, Unix и т.д.) приняты различные кодировки. Чтобы пользователю было легко просматривать страницы, Web-сервер провайдера должен уметь автоматически перекодировать документы в зависимости от поступившего запроса. В противном случае либо содержание вашего Web-узла для некоторых посетителей будет нечитаемым, либо придется обеспечивать несколько копий Web-узла — по одной на каждую поддерживаемую кодировку.

Способ обновления страниц. Обычно страницы обновляются по протоколу FTP (File Transfer Protocol, протокол передачи файлов). Некоторые FTP-клиенты позволяют работать с файлами на компьютере провайдера так же, как с собственным диском, — копировать, удалять, переименовывать и т.п.

Как правило, возможность размещения Web-узла провайдер предоставляет своим пользователям за небольшую плату или бесплатно.

Существуют службы, которые предоставляют место под Web-узлы бесплатно вместе с адресом электронной почты и другими услугами. Как правило, условием такого "бесплатного" размещения является выделение на ваших страницах некоторого места под рекламу. Кроме того, накладываются ограничения на размер ваших файлов.

История развития HTML

В 1989 году Тим Бернерс-Ли предложил руководству Европейского Центра ядерных исследований (CERN) проект распределенной гипертекстовой системы, которую он назвал World Wide Web (WWW), Всемирная паутина. Первоначально идея системы состояла в том, чтобы при помощи гипертекстовой навигационной системы объединить все множество информационных ресурсов CERN в единую информационную систему. Технология оказалась настолько удачной, что дала толчок к развитию одной из самых популярных в мире глобальных информационных систем. Практически в сознании большинства пользователей глобальной компьютерной сети Internet сама эта сеть ассоциируется с тремя основными информационными технологиями:

- электронная почта (e-mail);
- файловые архивы FTP;
- World Wide Web.

Причем последняя технология постепенно перемещается на первое место.

Успех технологии World Wide Web определен двумя основными факторами: простотой и использованием протоколов межсетевого обмена семейства TCP/IP, (Transmission Control Protocol, протокол управления передачей/Internet Protocol, протокол Internet), которые являются основой Internet.

Практически все пользователи Сети одновременно получили возможность попробовать себя в качестве создателей и читателей информационных материалов, опубликованных во Всемирной паутине. Но и популярность самого Internet во многом вызвана появлением World Wide Web, так как это первая сетевая технология, которая предоставила пользователю простой современный интерфейс для доступа к разнообразным сетевым ресурсам. Простота и удобство применения привели к росту числа пользователей **WWW** и привлекли внимание коммерческих структур. Далее процесс роста числа пользователей стал лавинообразным, и так продолжается до сих пор.

При этом сама технология на начальном этапе была чрезвычайно проста. Дело в том, что при разработке различных компонентов технологии (языка гипертекстовой разметки HTML (HyperText Markup Language, язык разметки гипертекста), протокола обмена гипертекстовой информацией HTTP, спецификации разработки прикладного программного обеспечения CGI и др.) предполагалось, что квалификация авторов информационных ресурсов и их оснащенность средствами вычислительной техники будут минимальными.

Одним из компонентов технологии создания распределенной гипертекстовой системы World Wide Web стал язык гипертекстовой разметки HTML, разработанный Тимом Бернерсом-Ли на основе стандарта языка разметки печатных документов — SGML (Standard Generalised Markup Language, стандартный обобщенный язык разметки). Дэниел В. Конноли написал для него Document Type Definition — формальное описание синтаксиса HTML в терминах SGML.

Разработчики HTML смогли решить две задачи:

- предоставить дизайнерам гипертекстовых баз данных простое средство создания документов;
- сделать это средство достаточно мощным, чтобы отразить имевшиеся на тот момент представления об интерфейсе пользователя гипертекстовых баз данных.

Первая задача была решена за счет выбора теговой модели описания документа. Такая модель широко применяется в системах подготовки документов для печати. Примером такой системы может служить хорошо известный язык разметки научных документов TeX, который был создан Дональдом Кнутом и предложен Американским математическим обществом, и программы его интерпретации.

Язык HTML позволяет размечать электронный документ, который отображается на экране с полиграфическим уровнем оформления; результирующий документ может содержать самые разнообразные метки, иллюстрации, аудио- и видеофрагменты и так далее. В состав языка вошли развитые средства для создания различных уровней заголовков, шрифтовых выделений, различные списки, таблицы и многое другое.

Вторым важным моментом, повлиявшим на судьбу HTML, стало то, что в качестве основы был выбран обычный текстовый файл. Выбор был сделан под влиянием следующих факторов:

- такой файл можно создать в любом текстовом редакторе на любой аппаратной платформе в среде какой угодно операционной системы;
- к моменту разработки HTML существовал американский стандарт для разработки сетевых информационных систем — Z39.50, в котором в качестве единицы хранения указывался простой текстовый файл в кодировке LATIN1, что соответствует US ASCII.

Таким образом, гипертекстовая база данных в концепции **WWW** — это набор текстовых файлов, размеченных на языке HTML, который определяет форму представления информации (разметка) и структуру связей между этими файлами и другими информационными ресурсами (гипертекстовые ссылки). Гипертекстовые ссылки, устанавливающие связи между текстовыми документами, постепенно стали объединять самые различные информационные ресурсы, в том числе звук и видео; в результате возникло новое понятие — гипермедиа.

Такой подход предполагает наличие еще одного компонента технологии — интерпретатора языка. В World Wide Web функции интерпретатора разделены между Web-сервером гипертекстовой базы данных и интерфейсом пользователя. Сервер, кроме доступа к документам и обработки гипертекстовых ссылок, обеспечивает предпроцессорную обработку документов, в то время как интерфейс пользователя

осуществляет интерпретацию конструкций языка, связанных с представлением информации.

Первая версия языка (HTML 1.0) была направлена на представление языка как такового, где описание его возможностей носило скорее рекомендательный характер. Вторая версия языка (HTML 2.0) фиксировала практику использования его конструкций. Версия ++ (HTML++) представляла новые возможности, расширяя набор тегов HTML в сторону отображения научной информации и таблиц, а также улучшения стиля компоновки изображений и текста. Версия 3.2 смогла упорядочить все нововведения и согласовать их с существующей практикой. HTML 3.2 позволяет реализовать использование таблиц, выполнение кодов языка Java, обтекание графики текстом, а также отображение верхних и нижних индексов.

Сейчас World Wide Web Consortium (W3C) — международная организация, которая занимается подготовкой и распространением документации на описание новых версий HTML — уже опубликовала материалы спецификации HTML 4.01. Кроме возможностей разметки текста, включения мультимедиа и формирования гипертекстовых связей, уже существовавших в предыдущих версиях HTML, в версию 4.01 включены дополнительные средства работы с мультимедиа, языки программирования, таблицы стилей, упрощенные средства печати изображений и документов. Для управления сценариями просмотра страниц Website (гипертекстовой базы данных, выполненной в технологии World Wide Web) можно использовать языки программирования этих сценариев, например, JavaScript, Java и VBScript.

Усложнение HTML и появление языков программирования привело к тому, что разработка Web-узлов стала делом высокопрофессиональным, требующим специализации по направлениям деятельности и постоянного изучения новых Web-технологий. Но возможности Internet позволяют пользователям, владеющим основами HTML, создавать и размещать собственные Web-узлы без больших затрат. Именно на таких пользователей и рассчитан предлагаемый курс.

Принципы гипертекстовой разметки

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 HTML представляет собой не что иное, как обычный ASCII-файл, с добавленными в него управляющими HTML-кодами (тегами). Поскольку HTML произошел от SGML, в нем разрешено использовать только три управляющих символа: горизонтальную табуляцию, перевод каретки и перевод строки. Это облегчает взаимодействие с различными операционными системами.

Теги HTML-документов в большинстве своем просты и понятны, ибо они образованы с помощью общеупотребительных слов английского языка, понятных сокращений и обозначений. HTML-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, 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, выравнена 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 одним или несколькими знаками табуляции, пробелами или символами возврата к началу строки. Порядок записи атрибутов в теге значения не имеет. Значение атрибута, если таковое имеется, следует за знаком равенства, стоящим после имени атрибута. Если значение атрибута — одно слово или число, то его можно просто указать после знака равенства, не выделяя дополнительно. Все остальные значения необходимо заключать в одинарные или двойные кавычки, особенно если они содержат несколько разделенных пробелами слов. Длина значения атрибута ограничена 1024 символами. Регистр символов в именах тегов и атрибутов не учитывается, чего нельзя сказать о значениях атрибутов. Например, особенно важно использовать нужный регистр при вводе URL (Uniform Resource Locator, унифицированный указатель ресурса) других документов в качестве значения атрибута HREF.

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 <TITLE> закрывающей парой будет </TITLE>). Конечные теги никогда не содержат атрибутов. По своему значению теги близки к понятию скобок "begin/end" в универсальных языках программирования, которые задают области действия имен локальных переменных и т.п. Теги определяют область действия правил интерпретации текстовых документов.

При использовании вложенных элементов разметки в документе следует соблюдать особую аккуратность. Вложенные теги нужно закрывать, начиная с последнего. 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, теги <META> и <BASE>.

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 <P>. Поскольку он используется в документе очень часто, его обычно ставят только в начале каждого абзаца. Когда один абзац заканчивается, следующий тег <P> сигнализирует браузеру о том, что нужно завершить данный абзац и начать следующий. Большинство авторов тегом конца абзаца не пользуются.

Есть и другие конечные теги, без которых браузеры отлично работают, например, конечный тег </HTML>. Тем не менее, рекомендуется включать по возможности больше конечных тегов, чтобы избежать путаницы и ошибок при воспроизведении документа.

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 <, а символ ">" — на >. То есть, если указать в тексте HTML строку <BODY>, 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 &, и т. д.

CER легко обнаружить, если посмотреть на структуру любого документа HTML, поскольку каждый из них начинается с амперсанда "&". 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;	&	Амперсанд
&#060;	&lt;	<	Меньше
&#062;	&gt;	>	Больше
			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

Группы тегов HTML

Все теги HTML по их назначению и области действия можно разделить на следующие основные группы:

- определяющие структуру документа;
- оформление блоков гипертекста (параграфы, списки, таблицы, картинки);
- гипертекстовые ссылки и закладки;
- формы для организации диалога;
- вызов программ.

Структура гипертекстовой сети задается гипертекстовыми ссылками. Гипертекстовая ссылка — это адрес другого HTML-документа или информационного ресурса Internet, который тематически, логически или каким-либо другим способом связан с документом, в котором ссылка определена.

Естественно, при таких условиях очень важна схема адресации всех имеющихся информационных ресурсов.

Реальный механизм интерпретации идентификатора ресурса, опирающийся на URI (Uniform Resource Identifier, универсальный идентификатор ресурса), называется URL, и пользователи WWW имеют дело именно с ним.

Типичным примером использования такой записи можно считать следующий пример:

```
Этот текст содержит:
<A HREF="http://www.intuit.ru/help/index.html">
гипертекстовую ссылку</A>
```

Выглядеть это будет следующим образом:

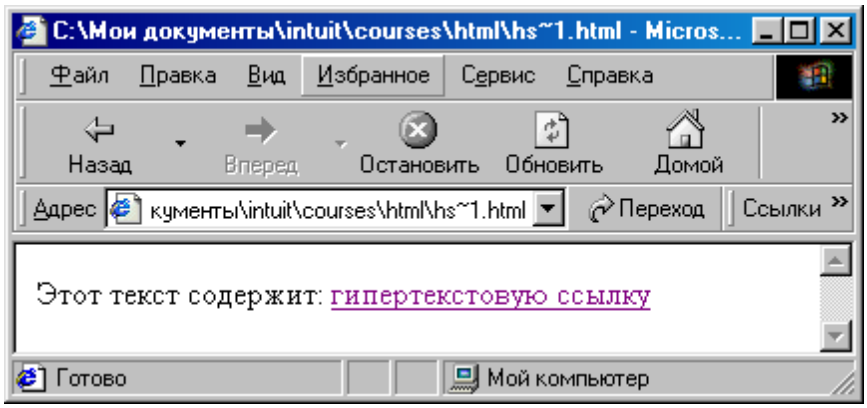


Рис. 1.1.

В приведенном выше примере тег "A", который в HTML называют якорем (anchor), использует атрибут HREF, обозначающий гипертекстовую ссылку (Hypertext Reference), для записи этой ссылки в форме URL. Данная ссылка указывает на документ с именем "index.html" в каталоге "help" на сервере "www.intuit.ru", доступ к которому осуществляется по протоколу HTTP.

Гипертекстовые ссылки в HTML делятся на два класса: контекстные гипертекстовые ссылки и общие. Контекстные ссылки вмонтированы в тело документа, как это было продемонстрировано в предыдущем примере, в то время как общие ссылки связаны со всем документом в целом и могут использоваться при просмотре любого фрагмента документа. Оба класса ссылок изначально присутствуют в стандарте языка, однако первое время наибольшей популярностью пользовались контекстные ссылки. Эта популярность привела к тому, что механизм использования общих ссылок практически полностью "атрофировался". В данном примере мы заключили URL в двойные кавычки. На самом деле, это необязательно. Кавычки (двойные или одинарные) применяются только тогда, когда внутри значения URL появляются символы-разделители (пробел, табуляция, неотображаемые символы). Но такого сорта URL следует всячески избегать.

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>`.

В заключение отметим, что при подготовке документов HTML используется идентификатор текста DTD (Document Type Definition, определение типа документа) в качестве первой строки. Это позволяет браузеру идентифицировать документ как соответствующий стандарту HTML. Обычно (но не обязательно) каждый документ HTML начинается со строки типа:

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

Здесь содержится информация о том, что документ соответствует версии HTML 4.0; разработанной W3C; используемый язык — английский.

2. Лекция: Структура HTML-документа и элементы разметки заголовка документа:
В этой лекции разбираются типовая структура HTML-документа и содержание его заголовка. Подробно описывается содержание элементов разметки, и разбираются формат и назначение каждого из элементов разметки заголовка.

HTML-документ — это один большой контейнер, который начинается с тега <HTML> и заканчивается тегом </HTML>:

```
<HTML>Содержание документа</HTML>
```

Контейнер HTML или гипертекстовый документ состоит из двух других вложенных контейнеров: заголовка документа (HEAD) и тела документа (BODY). Рассмотрим простейший пример классического документа.

```
<HTML>
<HEAD>
<TITLE>Простейший документ</TITLE>
</HEAD>
<BODY TEXT=#0000ff BGCOLOR=#f0f0f0>
<H1>Пример простого документа</H1>
<HR>
Формы HTML-документов
<UL>
<LI>Классическая
<LI>Фреймовая
</UL>
<HR>
</BODY>
</HTML>
```

Компания Netscape Communication расширила классическую форму документа возможностью организации фреймов (кадров), позволяющих разделить рабочее окно программы просмотра на несколько независимых фреймов. В каждый фрейм можно загрузить свою страницу HTML. Приведем пример документа с фреймами.

```
<HTML>
<HEAD>
<TITLE>Документ с фреймами</TITLE>
</HEAD>
<FRAMESET COLS="30%,*">
<FRAME SRC=frame1.htm NAME=LEFT>
<FRAME SRC=frame2.htm NAME=RIGHT>
</FRAMESET>
</HTML>
```

Назначение заголовка

Заголовок HTML-документа является необязательным элементом разметки. В HTML 2.0 предлагалось вообще отказаться от элементов HEAD и BODY. В то время в HTML не было элементов, которые использовались одновременно и в заголовке, и в теле документа. Современная практика HTML-разметки такова, что почти в каждом документе есть HTML-заголовок.

Первоначально существование заголовка определялось необходимостью именования окна браузера. Это достигалось за счет элемента разметки TITLE:

```
<HTML>
 <HEAD>
  <TITLE>Это заголовок</TITLE>
```

```
 ...
 </HEAD>
 <BODY>
 ...
 </BODY>
</HTML>
```

Отображение содержания элемента TITLE

Однако задумывался заголовок для несколько иных целей. Исходя из общих соображений, связанных с теорией и практикой разработки и эксплуатации гипертекстовых систем, все гипертекстовые связи информационных узлов принято разделять на контекстные и общие.

Контекстные гипертекстовые связи соответствуют определенному месту документа — контексту. В HTML такие связи реализованы в виде гипертекстовых ссылок (элемент A (anchor)). Фактически до реализации таблиц описателей стилей в современных браузерах это был единственный вид связей, которыми мог управлять автор HTML-документа.

Общие гипертекстовые связи определяются не частью документа (контекстом), а всем документом целиком. Например, быть предыдущим по отношению к другому документу или следующим — это общая гипертекстовая связь, которая позволяет организовать так называемый "линейный" просмотр информационных узлов гипертекстовой сети.

Реализация такого сорта ссылок уже давно является частью проектов W3C (Arena, Amaya). В коммерческих браузерах такой механизм реализован только для описателей стилей (элемент разметки LINK).

Важную роль заголовок HTML-документа играет в JavaScript. Существует принципиальная разница между заголовком и телом документа при использовании элемента разметки SCRIPT. Она заключается в определении зоны видимости функций и переменных. Переменные и функции, определенные в заголовке документа, относятся ко всему окну браузера. Это значит, что к ним можно обратиться из любого места документа и изменить их значения. Кроме того, к ним можно обратиться из другого окна или фрейма. Фактически, это глобальные переменные. При работе с многослойными документами переменные и функции тела относятся к слоям, что делает доступ к ним неудобным.

Еще одной функцией заголовка HTML-документа является управление HTTP-обменом через элемент разметки META. При современной практике размещения Web-узлов компаний на серверах провайдеров администраторы этих узлов могут не иметь возможности управлять программой-сервером. В этом случае для управления обменом остается только одна возможность — через заголовок HTML-документа.

Нельзя также не упомянуть еще об одном важном назначении заголовка HTML-документа — поисковом образе документа для индексирования роботами поисковых систем. Элемент META позволяет хранить списки ключевых слов и описания документа, которые будут использоваться для составления индекса поисковой системы и появляться в качестве описания документа в случае выдачи ссылки на него при поиске по ключевым словам.

Основные контейнеры заголовка

Основные контейнеры заголовка — это элементы HTML-разметки, которые наиболее часто встречаются в заголовке HTML-документа, т.е. внутри элемента разметки HEAD.

Мы рассмотрим только восемь элементов разметки, включая сам элемент разметки HEAD:

- HEAD (элемент разметки HEAD);
- TITLE (заглавие документа);
- BASE (база URL);
- **ISINDEX** (поисковый шаблон);
- META (метаинформация);
- LINK (общие ссылки);
- STYLE (описатели стилей);
- SCRIPT (скрипты).

Чаще всего применяются элементы TITLE, SCRIPT, STYLE. Использование элемента META говорит об осведомленности автора о правилах индексирования документов в поисковых системах и возможности управления HTTP-обменом данными. BASE и **ISINDEX** в последнее время практически не применяются. LINK указывают только при использовании внешних относительно данного документа описателей стилей.

Элемент разметки HEAD

Элемент разметки HEAD содержит заголовок HTML-документа. Данный элемент разметки не является обязательным. При наличии тега начала элемента разметки желательно использовать и тег конца элемента разметки. По умолчанию элемент HEAD закрывается, если встречается либо тег начала контейнера BODY, либо тег начала контейнера FRAMESET. Атрибутов у тега начала контейнера нет, хотя в DTD HTML один необязательный атрибут прописан. Синтаксис контейнера HEAD в общем виде выглядит следующим образом:

```
<HEAD profile="http://www.intuit.ru/help">
Это пример из документации по сайту Интернет-
Университета Информационных Технологий
</HEAD>
```

Контейнер заголовка служит для размещения информации, относящейся ко всему документу в целом. Необязательный атрибут PROFILE указывает на внешний файл META-тегов. В качестве значения этого атрибута указывается URL данного файла.

Элемент разметки TITLE

Элемент разметки TITLE служит для именования документа в World Wide Web. Более прозаическое его назначение — именование окна браузера, в котором просматривается документ. Состоит контейнер из тега начала, содержания и тега конца. Наличие тега конца обязательно. Тег начала элемента не имеет специфических атрибутов.

В различных браузерах алгоритм отображения элемента TITLE может отличаться. Так, в некоторых руководствах предлагается создать бегущую строку в заголовке документа, указав несколько последовательных контейнеров TITLE:

```
<TITLE>И</TITLE>
<TITLE>Ин</TITLE>
<TITLE>Инт</TITLE>
<TITLE>Инте</TITLE>
```

```
<TITLE>Интер</TITLE>
...
<TITLE>Интернет-Университе</TITLE>
<TITLE>Интернет-Университет</TITLE>
```

Такой механизм в современных браузерах не работает. При этом следует учитывать, что в отличие от реализации "бегущей" строки средствами JavaScript, лидирующие пробелы в заголовке игнорируются.

При выборе текста для содержания контейнера TITLE следует учитывать, что отображается он системным шрифтом, так как является заголовком окна браузера. В нелокализованных версиях операционных систем и графических оболочек русский текст содержания элемента TITLE будет отображаться абракадаброй.

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.

Элемент разметки BASE

Элемент разметки BASE служит для определения базового URL для гипертекстовых ссылок документа, заданных в неполной (частичной) форме. Кроме того, BASE позволяет определить мишень (окно) загрузки документа по умолчанию при выборе гипертекстовой ссылки текущего документа.

Разметка гипертекстовых ссылок обычно выполняется как разметка в частично заданных (относительных) адресах, когда URL задается относительно текущего местоположения документа.

```
<A HREF=../next_level/document.html>...</A>
```

В этом случае в качестве базы по умолчанию выбирается каталог, в котором размещен HTML-документ (`../`). Такой стиль разметки удобен тем, что при переносе всего дерева документов в другое место не потребуется менять систему гипертекстовых ссылок внутри документов. Кроме того, распространению этого стиля способствует и сама архитектура World Wide Web. Наиболее тесные связи между документами задаются только в рамках одного Web-узла. Связей данного узла с остальными существенно меньше, и их можно прописать непосредственно в ссылках в полной форме.

Контейнер BASE можно использовать вне документа, в заголовке или теле документа. При этом область действия базового адреса определяется от места размещения контейнера до следующего контейнера BASE.

```
<BASE HREF=http://intuit.ru/start/>
<HTML>
<HEAD>
<BASE HREF=http://intuit.ru/cgi-bin/>
  ... </HEAD>
<BODY>
  <BASE HREF=http://intuit.ru/start/>
  ... </BODY>
</HTML>
```

Наиболее часто BASE встречается на страницах узлов, которые имеют "зеркала". Часть документов основного сервера по разным причинам на "зеркальный" сервер не переносится. В этом случае документ с принудительно заданным базовым URL всегда будет ссылаться на основной сервер. Он оказывается "белой вороной" среди прочих документов Web-узла. При этом такая схема часто используется в совокупности с запретом на кэширование данного документа как клиентом (браузером), так и proxy-серверами.

Существуют различия и при определении базового URL по умолчанию при обращении к страницам, которые различны по своей природе. Если для обычного файла базовым адресом по умолчанию является адрес каталога, где хранится данный файл, то для страниц, которые генерируются "на лету", возможны и другие базовые адреса по умолчанию. Например, для страниц, сгенерированных CGI-скриптом, адресом по умолчанию является URL данного скрипта. Если из такой страницы снова вызвать скрипт, как частично заданную ссылку, то имя скрипта будет передано в качестве параметра скрипту, который сгенерировал данную страницу.

```
<A HREF=http://intuit.ru/cgi-bin/script/intuit.ru?name=value>...</A>
```

Базовый адрес: `http://intuit.ru/cgi-bin/script/intuit.ru`

Если скрипт вызовет сам себя по частично заданной ссылке, то он себя не найдет.

Возможность определения мишени загрузки позволяет не указывать атрибут TARGET в теге начала контейнера A (anchor):

```
<A HREF=intuit.htm TARGET=left>intuit</A>
```

Потребность в этом возникает при организации постоянно отображаемых меню. Такое меню может быть реализовано либо во фрейме, либо в окне. При этом информационные страницы Web-узла, которые загружаются при активизации гипертекстовых ссылок, будут загружаться в другое окно или фрейм.

Особенно полезен атрибут `TARGET` на страницах с вызовом скриптов, если результат работы скрипта нужно загрузить в определенное окно (фрейм).

Тег начала контейнера содержит один обязательный атрибут `HREF`, и может содержать один необязательный атрибут `TARGET`. Синтаксис контейнера BASE в общем виде выглядит следующим образом:

```
<BASE HREF="http://www.intuit.ru/intro.html">
<BASE HREF=http://www.intuit.ru/intro.html
TARGET=new>
```

Применение BASE в современных документах ограничено в силу разных причин. В сложных случаях можно пользоваться указаниями URL в полной форме.

Элемент разметки ISINDEX

Элемент разметки **ISINDEX** используется для указания поискового шаблона и унаследован от ранних версий HTML. В HTML 4.0 этот контейнер не определен. Утрата данного контейнера объясняется широким применением форм и CGI-скриптов. Тем не менее все браузеры его поддерживают.

Шаблон ввода ключевых слов при наличии данного контейнера в заголовке HTML-документа отображается в виде дополнительного поля ввода рабочей области браузера, что нарушает компоновку HTML-страниц, выполненных с применением

современных средств разметки. Больше всего **ISINDEX** подходит для документов с компоновкой в стиле HTML 2.0.

```
<HTML>
 <HEAD>
  <ISINDEX>
 </HEAD>
 <BODY>
  ...
 </BODY>
</HTML>
```

Применение элемента ISINDEX

В классическом варианте при использовании **ISINDEX** список ключевых слов, которые вводятся в поисковом шаблоне и разделены символом "+", присоединяется к базовому адресу HTML-документа после символа "?".

```
http://intuit.ru/isindex.html?keyword+list
```

Очевидно, что сам HTML-документ не способен выполнить поиск. Это может сделать только поисковая программа.

Присоединение запроса к документу унаследовано от первого сервера CERN (Conseil Europeen pour la Recherche Nucleaire, Европейская организация по ядерным исследованиям), в котором оно использовалось по аналогии с поисковыми серверами Gopher. Современный подход, основанный на HTML-формах, позволяет указывать URL поисковой программы, что дает большую свободу при разметке страниц.

Современный синтаксис **ISINDEX** позволяет применить аналогичный формам подход. Для этой цели в теге начала контейнера **ISINDEX** можно указать атрибут `ACTION`.

```
<ISINDEX ACTION=/cgi-bin/search.cgi>
```

Однако и традиционная форма контейнера позволяет обращаться к внешним CGI-скриптам. Сделать это можно либо в совокупности с контейнером BASE, либо с использованием SSI.

В первом случае для всего документа устанавливается базовый URL поисковой программы. Все URL гипертекстовых ссылок на другие страницы задаются в полной форме или базовый адрес переназначается после **ISINDEX**. Это вполне оправдано, если данная страница ничего, кроме поискового критерия и ссылки на домашнюю страницу Web-узла, не содержит.

```
<HTML>
 <HEAD>
  <BASE HREF=http://intuit.ru/cgi-bin/search.cgi>
  <ISINDEX>
 </HEAD>
 <BODY>
  <BASE HREF=http://intuit.ru/>
 </BODY>
```

Во втором случае в документ встраивается обращение к CGI-скрипту, который реализует функции поисковой программы. Такое совмещение — свойство современного подхода к компоновке поисковых страниц. Как правило, и поисковый шаблон, и результаты поиска отображаются на одной странице, так как это позволяет корректировать запрос по мере получения результатов поиска. Встроенный в страницу

скрипт анализирует переменные окружения сервера, и в случае отсутствия запроса может вообще никак не обнаруживать свое присутствие внутри документа.

Тег начала элемента может содержать два необязательных атрибута: ACTION и PROMPT. Синтаксис элемента **ISINDEX** в общем виде выглядит следующим образом:

```
<ISINDEX [PROMPT=текст] [ACTION=URL]>
```

Первый необязательный атрибут тега начала **ISINDEX** — PROMPT. Он позволяет вместо стандартного приглашения к вводу ключевых слов задать приглашение, которое, по мнению автора документа, лучше отражает суть поискового шаблона. Например, можно задать приглашение к вводу ключевых слов на русском языке.

```
Введите ключевые слова:
```

Применение атрибута PROMPT

ISINDEX — отмирающий элемент разметки. Однако он определил формат обмена данными **ISINDEX**. Данные в этом формате передаются от браузера серверу в случае применения **ISINDEX** и в случае прямого указания дополнительных параметров после символа "?" в гипертекстовой ссылке.

Элемент разметки META

Это наиболее популярный элемент разметки заголовка, более распространен только элемент TITLE. Такое положение дел объясняется назначением данного элемента разметки. META содержит управляющую информацию, которую браузер использует для правильного отображения и обработки содержания тела документа.

Впервые контейнер META был задействован при принудительной перезагрузке документа браузером через заголовок HTTP-сообщения. В заголовке HTTP-сообщения можно указать оператор refresh. Время, заданное как параметр этого оператора, определяет интервал в секундах, после которого браузер загружает документ, определенный атрибутом URL данного оператора. Впервые этот механизм был реализован на сервере CERN, но наибольшую популярность приобрел при использовании сервера WN (Web-сервер, который был разработан для платформы Linux).

В контейнере META подобный механизм реализуется следующим образом:

```
<META HTTP-EQUIV="Refresh" CONTENT="1;
      URL=refresh.htm">
```

В данном случае через одну секунду после загрузки документа браузер должен инициировать загрузку страницы refresh.htm.

Используя этот механизм, можно построить автоматически перезагружаемую последовательность страниц. Для этого в заголовке каждой страницы из данной последовательности следует разместить соответствующий контейнер META.

```
<META HTTP-EQUIV="Refresh" CONTENT="1;
      URL=refreshX.htm">
```

Заглавная буква "X" в слове "refreshX.htm" — это цифра номера кадра. На странице нулевого кадра в этом месте следует указать на первый кадр (refresh1.htm), на странице первого кадра — на второй (refresh2.htm) и т.д.

В Windows 95 и Windows NT 4.0 с поддержкой таблиц UNICODE появилась возможность указывать тип кодировки документа — CHARSET. К сожалению, на многих Unix-платформах этот механизм не работает, что часто приводит к ошибкам, например в IRIX версий 6.2-6.4. Скептическое отношение поклонников Unix к этой возможности ничем не подкреплено, так как основная масса пользователей российской части Internet просматривает документы World Wide Web в Windows. Для перекодировки на стороне клиента (документ подготовлен в кодировке cp1251) в заголовок документа необходимо включить META-тег следующего вида:

```
<META HTTP-EQUIV="Content-type"
      CONTENT="text/html;
      CHARSET=windows-1251">
```

Приведенный выше пример показывает, как используются операторы заголовка HTTP-сообщения. Однако здесь тоже следует быть осторожным. Большинство российских Web-узлов используют в качестве HTTP-сервера Russian Apache. Эта модификация сервера поддерживает перекодировку документов "на лету" для правильного отображения на стороне клиента. Russian Apache сам вставляет в HTTP-заголовок (не путать с HEAD) директиву Content-type. Если в документе будет META-элемент с указанием типа кодировки, а Apache перекодировал содержание, то возможно несоответствие между указанным в META типом кодировки и реальной кодировкой содержания документа.

Кроме Content-type, можно указать и другие операторы. Например, запретить кэширование документа. Необходимость в этом возникает при частом обновлении документа или наличии в нем изменяющихся SSI-вставок. Для запрета кэширования достаточно вставить в заголовок META-тег вида:

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
```

Pragma — это наследие HTTP 1.0. В новой версии протокола HTTP (HTTP 1.1) управление кэшированием осуществляется через оператор Cache-Control. Для получения такого же эффекта, как в случае с Pragma, в заголовке HTML-документа достаточно указать:

```
<META HTTP-EQUIV="Cache-Control"
      CONTENT="no-cache">
```

Новый механизм управления кэшированием и хранением документа на стороне клиента гораздо более гибок, чем в HTTP 1.0. Например, можно запретить хранение документа после пересылки:

```
<META HTTP-EQUIV="Cache-Control"
      CONTENT="no-store">
```

Точно так же можно задать время последней модификации (Last-Modified) или дату истечения актуальности документа (Expire).

С появлением роботов поисковых машин на META-тег была возложена еще одна функция — описание поискового образа документа. Наиболее последовательно это было впервые реализовано в Webcrawler. До этого в качестве поискового образа документа использовался либо весь список слов документа, либо слова первого абзаца.

Собственно, для описания документа используется два META-тега. Один определяет список ключевых слов, а второй – реферат (краткое содержание документа), который отображается в качестве пояснения к ссылке на документ в отчете поисковой машины о выполненном запросе. Контейнер TITLE здесь также используется в качестве названия документа.

```
<TITLE>Основы Web-технологий</TITLE>

<META NAME="description"
http-equiv="description"
content="Учебный курс Основы Web-технологий.
Тема: Заголовок HTML-документа. Элемент
разметки META. Дается краткое описание
основных способов применения контейнера META
в заголовке HTML-документа. Рассматривается
управление HTTP-обменом и индексирование
документов.">

<META NAME="keywords" HTTP-EQUIV="keywords"
CONTENT="учебный курс; Web-технология; web;
технология; HTML; язык гипертекстовой разметки;
заголовок HTML-документа; заголовок; HTML;
документ; контейнер; META; элемент; HEAD;
пример; разметка; методика">
```

При индексировании такого документа содержимое контейнера TITLE и атрибутов CONTENT контейнеров META после фильтрации попадет в индекс поисковой машины и может быть использовано для составления запросов. Процесс фильтрации отбракует так называемые stop-слова и общие слова. Они не попадут в индекс поисковой машины. В частности, будут отбракованы предлоги или, если речь идет о тематическом поисковом индексе, например по технологиям World Wide Web, то в него не попадут: web, Web-технология и т.п.

META-тегом пользуются и программы подготовки документов. Они размещают в нем свой идентификатор. В общем случае контейнер META выглядит следующим образом:

```
<META [name=имя] [HTTP-EQUIV=имя_HTTP-оператора]
      CONTENT=текст>
```

Практика показывает, что при индексировании можно указывать одновременно и атрибут NAME, и атрибут HTTP-EQUIV с одинаковыми значениями. Это связано с тем, что одни роботы индексирования анализируют содержание META-элемента по атрибуту NAME, а другие — по атрибуту HTTP-EQUIV.

Элемент разметки LINK

Элемент разметки LINK – это результат давно предпринятой попытки придать HTML академический вид. Согласно теории гипертекстовых систем, все гипертекстовые связи разделяют на два типа: контекстные и общие. Такое деление чисто условное и определяется тем, что контекстную связь можно привязать к определенному месту документа, а общую — отнести только ко всему документу целиком. Если взглянуть на проблему связи чуть шире, то очевидной становится аналогия с отношениями. Гипертекстовая связь задает отношение на множестве информационных узлов.

Контекстная связь определяет отношение на паре узлов. При этом в модели World Wide Web один из узлов является источником, а второй — мишенью. Собственно, это и отражено в названии элемента разметки A (anchor), который определяет гипертекстовую ссылку (не путать с гипертекстовой связью). При этом в контекстной связи один и тот же термин может идентифицировать разные связи. Например, в контексте содержания конспекта данной темы слово "HEAD" определяет документ head.htm, который описывает контейнер HEAD и особенности его применения, а в контексте справочника по данной теме слово "HEAD" будет означать ссылку на описание синтаксиса этого контейнера.

Общие ссылки нельзя привязать по контексту. Например, два информационных узла находятся в отношении следования, т.е. при "линейном" просмотре одна Web-страница является следующей для другой Web-страницы. В этом случае речь идет о страницах целиком, а не об отдельных их частях. Такой же общей связью является принадлежность к Web-узлу, который ассоциируется со своей домашней страницей.

В информационно-поисковых системах поисковый термин определяет отношение "быть заиндексированным данным термином", которое также задает связь соответствующих документов.

В настоящее время в браузерах не существует единого способа программирования или определения общих гипертекстовых связей. В течение последних пяти лет W3C строит уже второй браузер, который должен продемонстрировать возможность программирования окон меню браузера (вперед, назад и т.п.). Однако производители наиболее популярных браузеров такой поддержки через HTML-разметку в своих программах не предлагают.

Существенный сдвиг в этом направлении произошел после реализации поддержки описателей стилей в Netscape Navigator и Internet Explorer четвертых версий. CSS (Cascade STYLE Sheets, каскадные таблицы стилей) позволяют определять для различных типов гипертекстовых связей вид гипертекстовых ссылок. При этом можно определять различные типы контекстных ссылок. Кроме того, впервые нашел осмысленное применение контейнер LINK. Он позволил загружать внешние описатели стилей:

```
<LINK REL=stylesheet href="../css/style.css"
      TYPE="text/css">
```

В данном случае речь идет о загрузке стилей из файла `style.css`. При этом стили задаются в нотации W3C, а не JavaScript, что определяется атрибутом `TYPE`. В сущности, атрибут `REL` определяет тип гипертекстовой связи, HREF (Hypertext REFerence) указывает адрес документа, идентифицирующего связь, а атрибут `TYPE` определяет тип содержания этого документа.

В общем случае контейнер LINK имеет следующий вид:

```
<LINK [REL=тип_отношения] [HREF=URL]
      [TYPE=тип_содержания]>
```

Для разных типов содержания действия по интерпретации элемента разметки будут различными. В настоящее время идет процесс разработки спецификаций описания метаданных, где возможно применение элемента разметки LINK.

Элемент разметки STYLE

Элемент разметки STYLE предназначен для размещения описателей стилей. При этом описание стиля из данного элемента разметки, если оно совпадает по имени класса и/или идентификатору подкласса со стилем, описанным во внешнем файле, заменяет описание стиля из внешнего файла. С точки зрения влияния на весь документ, описатели стилей задают правила отображения контейнеров HTML-документа для всей страницы.

В настоящее время контейнер используется только с одним атрибутом `TYPE`, который задает тип описателя стиля. Это может быть либо `text/css` , либо `text/javascript`. Если элемент разметки открыт тегом начала, то он должен быть закрыт тегом конца. В общем виде запись элемента STYLE выглядит так:

```
<STYLE TYPE=тип_описания_стилей>
описание стиля/стилей
</STYLE>
```

Применению стилей в HTML-разметке, а также проектированию Web-узлов с применением CSS посвящена отдельная глава "Применение каскадных таблиц и стилей".

Элемент разметки SCRIPT

Элемент разметки SCRIPT служит для размещения кода JavaScript, VBScript или JScript. Вообще говоря, SCRIPT можно использовать не только в заголовке документа, но и в его теле. В отличие от контейнера STYLE, ему не требуется дополнительный контейнер LINK для загрузки внешних файлов кодов. Это можно сделать непосредственно в самом контейнере SCRIPT:

```
<SCRIPT TYPE="text/javascript"
      SRC=script.code>
```

Если открыт тег начала, то нужно обязательно использовать тег конца контейнера. В противном случае, браузер может отобразить только символ "]". Если код не помещен в HTML-комментарии, то старые версии браузеров (до Mozilla 2) отображают программу перед текстом страницы. В ряде случаев страница вообще может не отображаться.

В общем виде запись контейнера выглядит следующим образом:

```
<SCRIPT [TYPE=тип_языка_программирования]>
JavaScript/VBScript-код
</SCRIPT>
```

и

```
<SCRIPT [TYPE=тип_языка_программирования]
      [SRC=URL]>
</SCRIPT>
```

Существует несколько скриптовых языков: JavaScript, VBScript, JScript. По умолчанию подразумевается JavaScript. Подробнее с JavaScript и контейнером SCRIPT можно ознакомиться в курсе "Введение в JavaScript".

3. Лекция: Контейнеры тела документа:
В этой лекции рассказывается об элементах разметки тела HTML-документа, подробно разбирается их типизация, назначение и применение.

Теги тела документа

Теги тела документа предназначены для управления отображением информации в программе интерфейса пользователя. Они описывают гипертекстовую структуру базы данных при помощи встроенных в текст контекстных гипертекстовых ссылок. Тело документа состоит из:

- иерархических контейнеров и заставок;
- заголовков (от `H1` до `H6`);
- блоков (параграфы, списки, формы, таблицы, картинки и т.п.);
- горизонтальных отчеркиваний и адресов;
- текста, разбитого на области действия стилей (подчеркивание, выделение, курсив);
- математических описаний, графики и гипертекстовых ссылок.

Тело документа – контейнер BODY

Описание тегов тела документа следует начать с тега `BODY`. В отличие от тега `HEAD`, тег `BODY` имеет атрибуты.

Атрибут `BACKGROUND` определяет фон, на котором отображается текст документа. Так, если источником для фона HTML- документа является графический файл image.gif, то в открывающем теге тела BODY появляется соответствующий атрибут:

```
<BODY BACKGROUND="image.gif">
```

Как видно из этого примера, в качестве значения данного атрибута используется адрес в сокращенной форме URL. В данном случае это адрес локального файла. Следует заметить, что разные интерфейсы пользователя поддерживают различные дополнительные атрибуты для тега `BODY`.

Таблица 3.1. Атрибуты

Атрибут	Значение
`BGCOLOR=#FFFFFF`	Цвет фона
`TEXT=#0000FF`	Цвет текста
`VLINK =#FF0000`	Цвет пройденных гипертекстовых ссылок
`LINK =#008000`	Цвет гипертекстовой ссылки

В данной таблице строка #XXXXXX определяет цвет в терминах RGB в шестнадцатеричной нотации. Также имеется возможность задавать цвета по названию. Далее в таблице приведены названия цветов, определенные в стандарте HTML 4 и соответствующие им RGB-коды. Отметим, что многие современные браузеры выходят за рамки стандартов и поддерживают гораздо больше названий цветов.

Таблица 3.2. Цвета

Название	Код	Название	Код
aqua	#00FFFF	navy	#000080
black	#000000	olive	#808000
blue	#0000FF	purple	#800080

fuchsia	#FF00FF	red	#FF0000
gray	#808080	silver	#C0C0C0
green	#008000	teal	#008080
lime	#00FF00	white	#FFFFFF
maroon	#800000	yellow	#FFFF00

Так, значения атрибутов в таблице 3.1 определяют цвет текста как синий, фона — белый, пройденные ссылки красные, а новые ссылки зеленые. Если в качестве атрибутов тега BODY указать

```
<BODY BGCOLOR=#FFFFFF TEXT=#0000FF
      VLINK=#FF0000 LINK=#00FF00>,
```

то цвет фона будет белым, текст будет синим, ссылки — зелеными, а пройденные ссылки станут красными. Однако пользоваться этими атрибутами следует крайне осторожно, так как у пользователя может оказаться другой интерфейс, который эти параметры не интерпретирует.

Microsoft Internet Explorer и Netscape Navigator допускают применение атрибутов LEFTMARGIN=n и TOPMARGIN=n в теге <BODY>. Атрибут LEFTMARGIN= задает левое поле для всей страницы. TOPMARGIN= определяет верхнее поле. Число n показывает ширину поля в пикселах. Например, тег <BODY LEFTMARGIN ="40"> создаст на всей странице левое поле шириной 40 пикселов. При n, равном 0, левое поле отсутствует.

Теги управления разметкой

Заголовки

Заголовок обозначает начало раздела документа. В стандарте определено 6 уровней заголовков: от H1 до H6. Текст, окруженный тегами <H1></H1>, получается большим — это основной заголовок. Если текст окружен тегами <H2></H2>, то он выглядит несколько меньше (подзаголовок); текст внутри <H3></H3> еще меньше и так далее до <H6></H6>. Некоторые программы позволяют использовать большее число заголовков, однако реально более трех уровней встречается редко, а более 5 — крайне редко.

Ниже на рисунке показан результат использования следующих заголовков:

```
<H1>Заголовок 1</H1>
<H2>Заголовок 2</H2>
```

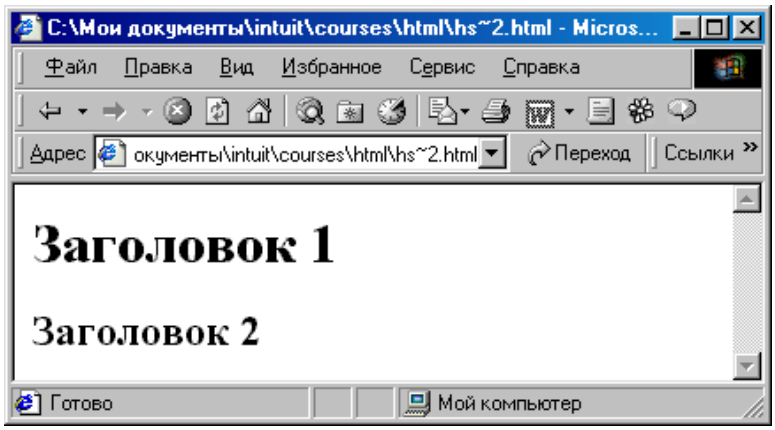


Рис. 3.1.

Тег <P>

Тег <P> применяется для разделения текста на параграфы. В нем используются те же атрибуты, что и в заголовках.

Атрибут ALIGN

Атрибут ALIGN позволяет выровнять текст по левому или правому краю, по центру или ширине. По умолчанию текст выравнивается по левому краю. Данный атрибут применим также к графике и таблицам.

Далее приведены возможные значения атрибута ALIGN:

ALIGN=justify выравнивание по левому и правому краям. Реализовано не во всех программах интерпретации.

ALIGN=left выравнивание по левому краю. По умолчанию текст HTML выравнивается по левому краю и не выравнивается по правому, то есть начало строк находится на одном уровне по вертикали, а концы — на разных. Чаще всего, получающийся при этом текст с равными промежутками между словами выглядит лучше. Поскольку выравнивание по левому краю задается автоматически, атрибут ALIGN=left можно опустить.

ALIGN=right выравнивание по правому краю. Текст, выравненный по правому краю и не выравненный по левому — концы строк находятся на одном уровне, а начало на разных, — часто используется с целью создать оригинальный дизайн. Для этого задается атрибут ALIGN=right в обычных тегах, например в теге <P>.

ALIGN=center центрирование текста и графики. Есть несколько способов отцентрировать текст или графику. В спецификациях HTML 3.0 предлагается пользоваться тегом <ALIGN=center>. Однако этот тег применим не ко всем объектам HTML-страницы, поэтому разработчики Netscape добавили тег <CENTER>, который центрирует любые объекты и поддерживается браузерами Netscape Navigator 3.0, Microsoft Internet Explorer 3.0 и другими. К тегу <CENTER> нужно относиться с осторожностью. Какой-нибудь браузер может его вообще проигнорировать, и на странице окажется текст, выравненный по левому краю.

Обтекание графики текстом. С помощью атрибута ALIGN можно заставить текст "обтекать" графический объект. Для этого следует поместить тег <IMG SRC="/путь/файл.gif"> туда, где должен быть графический объект, и добавить атрибут ALIGN=left, ALIGN=right или ALIGN=center. Кроме того, с помощью атрибутов HSPACE и VSPACE (они описаны ниже) задается ширина горизонтальных и вертикальных полей, отделяющих изображение от текста. Можно также создать рамку вокруг картинки или обрамление таблицы текстом. Чтобы текст не "обтекал" графику, а прерывался, необходимо применить тег
 с атрибутом CLEAR.

Использование тега

Принудительный перевод строки используется для того, чтобы нарушить стандартный порядок отображения текста. При обычном режиме интерпретации программа интерфейса пользователя отображает текст в рабочем окне, автоматически разбивая его на строки. В этом режиме концы строк текста игнорируются. Иногда для большей выразительности требуется начать печать с новой строки. Для этого и нужен тег BR. Атрибут CLEAR в теге
 используется для того, чтобы остановить в указанной точке обтекание объекта текстом и затем продолжить текст в пустой области за объектом.

Продолжающийся за объектом текст выравнивается в соответствии со значениями LEFT, RIGHT или ALL атрибута CLEAR:

```
<BR CLEAR=left> Текст будет продолжен, начиная
с ближайшего пустого левого поля.
<BR CLEAR=right> Текст будет продолжен, начиная
с ближайшего пустого правого поля.
<BR CLEAR=all> Текст будет продолжен, как
только и левое, и правое поля окажутся пустыми.
```

Элемент разметки <NOBR>

Тег <NOBR> (No Break, без обрыва) дает браузеру команду отображать весь текст в одной строке, не обрывая ее. Если текст, заключенный в <NOBR>, не поместится на экране, браузер добавит в нижней части окна документа горизонтальную полосу прокрутки. Если вы хотите оборвать строку в определенном месте, поставьте там тег <BR>.

Теги управления отображением символов

Все эти теги можно разбить на два класса: теги, управляющие формой отображения (font style), и теги, характеризующие тип информации (information type). Часто внешне разные теги при отображении дают одинаковый результат. Это зависит главным образом от настроек интерпретирующей программы и вкусов пользователя.

Теги, управляющие формой отображения

Курсив, усиление, подчеркивание, верхний индекс, нижний индекс, шрифт большой, маленький, красный, синий, различные комбинации — все это делает страницы более интересными. Microsoft Internet Explorer и Netscape Navigator позволяют определить шрифт с помощью тега FONT. Теперь можно объединять на одной странице несколько видов шрифтов, вне зависимости от того, какой из них задан по умолчанию в браузере пользователя.

Теги <BIG> и <SMALL> — изменение размеров шрифта

Текст, расположенный между тегами <BIG></BIG> или <SMALL> </SMALL>, будет, соответственно, больше или меньше стандартного.

Верхние и нижние индексы

С помощью тегов <SUP> и <SUB> можно задавать верхние и нижние индексы, необходимые для записи торговых знаков, символов копирайта, ссылок и сносок. Рассматриваемые теги позволяют создать внутри текстовой области верхние или нижние индексы любого размера. Чтобы они казались меньше окружающего текста, можно использовать теги <SUP> и <SUB> с атрибутом FONT SIZE=-1, уменьшающим размер шрифта.

Атрибут SIZE

Атрибут SIZE тега <FONT> позволяет задавать размер текста в данной области. Если вы не пользуетесь тегом <BASEFONT SIZE=n> для задания определенного размера шрифта на всей странице, то по умолчанию принимается 3. Некоторые браузеры тег <FONT> не поддерживают, поэтому желательно употреблять его только внутри текстовой области. В других случаях лучше использовать теги <H1>, <H2>, <H3> и т.д. Главное преимущество тега <FONT> состоит в том, что после окончания действия он не

разбивает строку, как теги <Hn>. Поэтому тег <FONT> бывает очень полезен для изменения размера шрифта в середине строки.

Атрибут COLOR

Если вы хотите сделать свою страницу более красочной, можете воспользоваться атрибутом COLOR в теге FONT, и тогда единственным ограничением будет цветовая палитра на компьютере пользователя.

Теги, управляющие формой отображения, приведены в таблице.

Таблица 3.3. Теги, управляющие формой отображения

Тег	Значение
<I>...</I>	Курсив (Italic)
<B>...</B>	Усиление (Bold)
<TT>...</TT>	Телетайп
<U>...</U>	Подчеркивание
<S>...</S>	Перечеркнутый текст
<BIG>...</BIG>	Увеличенный размер шрифта
<SMALL>...</SMALL>	Уменьшенный размер шрифта
_{...}	Подстрочные символы
^{...}	Надстрочные символы

Таблица 3.4. Теги, характеризующие тип информации

Тег	Значение
<EM>...</EM>	Типографское усиление
<CITE>...</CITE>	Цитирование
<STRONG>...</STRONG>	Усиление
<CODE>...</CODE>	Отображает примеры кода (например, "коды программ")
<SAMP>...</SAMP>	Последовательность литералов
<KBD>...</KBD>	Пример ввода символов с клавиатуры
<VAR>...</VAR>	Переменная
<DFN>...</DFN>	Определение
<Q>...</Q>	Текст, заключенный в двойные кавычки

Эти теги допускают вложенность, поэтому все они имеют тег начала и конца. При использовании таких тегов следует помнить, что их отображение зависит от настроек программы-интерфейса пользователя, которые могут и не совпадать с настройками программы-разработчика гипертекста.

Блоки цитат — элемент <BLOCKQUOTE>

Тег добавляет поля слева и справа от текста. Это полезный тег, поскольку он позволяет компактно расположить текст в центре страницы. При неоднократном использовании <BLOCKQUOTE> текст все больше сжимается к центру.

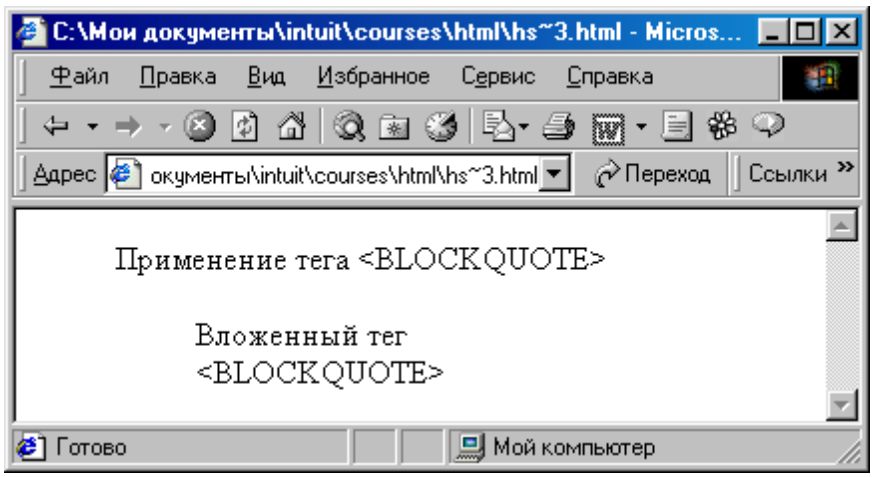


Рис. 3.2.

Создание списков в HTML

Списки являются важным средством структурирования текста и применяются во всех языках разметки. В HTML имеются следующие виды списков: ненумерованный список (неупорядоченный) (Unordered Lists <UL>), нумерованный список (упорядоченный) (Ordered Lists <OL>) и список определений. Теги для ненумерованных и нумерованных списков — это основа HTML. HTML 3.2 добавляет несколько атрибутов к тегам списков для выбора разных типов маркеров в ненумерованных списках и разных схем нумерации в нумерованных. Можно включать такие атрибуты и в сами теги элементов списка (List Item <LI>), чтобы сменить тип маркера в середине списка. После появления нового атрибута все последующие маркеры в списке будут иметь такой же вид.

Неупорядоченные списки — тег <UL>

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>второй элемент списка
<LI>третий элемент списка
</UL>
```

Теги <UL> и </UL> — это теги начала и конца ненумерованного списка, тег <LI> (List Item) задает тег элемента списка. Помимо этих тегов, существует тег, позволяющий именовать списки — <LH> (List Header).

Атрибуты маркеров в ненумерованном списке

Чтобы не применять одни и те же маркеры на разных уровнях вложенности, можно использовать атрибут TYPE. Вы можете задать любой тип маркера в произвольном месте

списка. Можно даже смешивать разные типы маркеров в одном списке. Ниже перечислены теги с атрибутами стандартных маркеров:

```
<UL TYPE=DISC>Тег создает сплошные маркеры
такого типа, как в списках первого уровня по
умолчанию.
<UL TYPE=CIRCLE>Тег создает маркеры в виде
окружностей.
<UL TYPE=SQUARE>Тег создает сплошные квадратные
маркеры.
```

Упорядоченные списки — тег <OL>

Нумерованные списки. Тег <OL> вместе с атрибутом TYPE= в HTML 3.2 позволяет создавать нумерованные списки, используя в качестве номеров не только обычные числа, но и строчные и прописные буквы, а также строчные и прописные римские цифры. При необходимости можно даже смешивать эти типы нумерации в одном списке:

```
<OL TYPE=1> Тег создает список с нумерацией
в формате 1., 2., 3., 4. и т.д.
<OL TYPE=A> Тег создает список с нумерацией
в формате A., B., C., D. и т.д.
<OL TYPE=a> Тег создает список с нумерацией
в формате a., b., c., d. и т.д.
<OL TYPE=I> Тег создает список с нумерацией
в формате I., II., III., IV. и т.д.
```

Список определений — тег <DL>

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

```
<DL><DT>Термин</DT> <DD>Определение</DD></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.

Горизонтальные линейки — тег <HR>

Горизонтальное отчеркивание (Horizontal Rule) применяется для разделения документа на части. С помощью одного лишь тега <HR> можно придать странице оригинальный вид. Попробуйте поэкспериментировать с тегом <HR>, и вы получите линии, совсем не похожие на те, которыми обычно пользуетесь.

Преформатированный вывод — тег <PRE>

Применение этого тега позволяет отобразить текст "как есть" (без форматирования), теми же символами и с тем же разбиением на строки.

Применение тега <BLINK>

Текст, помещенный между тегами <BLINK> и </BLINK>, мерцает. Данный тег поддерживается только браузером Netscape Navigator. Пользоваться им следует с большой осторожностью.

Комментарии в языке HTML

При разметке документов HTML возникает необходимость в использовании комментариев, которые браузер не выводит на экран, но другой специалист, редактирующий данный документ, может прочитать. В таких примечаниях можно найти информацию о том, кто является автором документа, где и почему используется конкретный элемент HTML и т.п. Комментарии HTML начинаются с символа "<!--" и оканчиваются символом "-->". Можно вставлять текст с любыми символами. Комментарии могут состоять из нескольких строк текста. В общем и целом они ничем не отличаются от аналогичных комментариев в других языках программирования, так как видимы только тогда, когда это необходимо. Например, браузер игнорирует их. При создании файла HTML можно разместить в нем комментарии о его структуре. Кроме того, там можно размещать информацию о том, какие сложные операции способен выполнять данный документ.

Гипертекстовые ссылки

Все рассмотренные выше средства управления отображением текста, безусловно, важны, но они только дополняют основной тег HTML-документа — гипертекстовую ссылку. Для записи гипертекстовой ссылки используется тег <A>, который называют "якорь" (anchor). Якорь имеет несколько атрибутов, главным из которых является HREF. Простую ссылку можно записать в виде

```
<A HREF="http://www.intuit.ru/index.html">
Отображаемое название гипертекстовой ссылки
</A>,
```

где значение атрибута HREF — адрес документа "index.htm" на машине "www.intuit.ru", доступ к которой осуществляется по протоколу HTTP. Форма записи этого адреса называется универсальным локатором ресурсов URL и является составной частью технологии WWW.

Согласно схеме HTTP нотации URI, полный адрес информационного ресурса, доступного по протоколу HTTP, надлежит записывать следующим образом:

```
http://user:password@domain.ru:port/path/
some.html?query_string,
```

где http — протокол обмена данными; user — идентификатор пользователя; password – пароль; domain.ru — доменное имя сервера; port — номер TCP-порта, на котором ведет обслуживание сервер; path — путь в корневом каталоге сервера к файлу ресурса; some.html — файл ресурса; query_string — поисковое предписание.

Заданный в таком виде адрес ресурса называется абсолютным или полным адресом ресурса. На практике редко используют все компоненты полного адреса схемы HTTP. Чаще всего первые компоненты опускают. Например, обращение к документу в том же каталоге в гипертекстовой ссылке будет записано просто как имя данного файла. Обращение к CGI-скрипту может выглядеть следующим образом:

```
<A HREF=../scripts/my_script?query_string>
```

Имя протокола, имя домена, номер порта и другие компоненты начала URL опущены. В этом случае говорят, что ссылка задана частично определенной или неполной формой URL.

Естественно, что браузер при обращении к серверу будет восстанавливать полную форму URL, опираясь на некоторую схему по умолчанию, которая называется базовым URL. Иногда неполную форму URL называют относительным URL, подразумевая, что адрес задается относительно некоторого базового адреса.

По умолчанию в качестве базового используется URL каталога, в котором находится текущий документ. Если URL начинается с символа "." или "..", то это означает исчисление от текущего каталога. Если URL начинается с символа "/", то относительный URL берется от корня каталогов сервера.

В HTML есть элемент разметки BASE (рассмотренный ранее), который позволяет задать или переопределить базовый адрес. Первоначально этот контейнер использовался только в заголовке HTML-документа. Сейчас его применяют как за пределами документа (например, при создании документов HTML-редакторами), так и в теле документа.

Содержание контейнера гипертекстовой ссылки, заключенное между тегом начала и тегом конца, выделяется в тексте цветом, определенным для контекстных гипертекстовых ссылок. В атрибутах тега <BODY>:

Таблица 3.5.

Атрибут	Значение
TEXT=#000000	Цвет текста (черный)
ALINK=#FF0000	Цвет "активных" гипертекстовых ссылок (красный)
VLINK=#FF00FF	Цвет пройденных гипертекстовых ссылок (пурпурный)
LINK=#0000FF	Цвет гипертекстовой ссылки (синий)

Одна из особенностей создания Web-сайта состоит в том, что представленную на нем информацию желательно разбить на отдельные части, которые могут быть выведены на экран без необходимости его прокрутки. Организация связей между отдельными частями осуществляется с помощью гипертекстовых ссылок.

```
<A HREF="http://www.intuit.ru/help/index.html">
Помощь</A>
```

При нажатии на ссылку в окно браузера будет загружен новый документ.

Другой формой использования тега <A> является определение точек внутри текста, на которые можно сослаться. Такой метод применяется в том случае, когда документ нельзя поделить на части и необходимо быстро перемещаться из оглавления в текст:

```
<A NAME="point">
```

Для ссылки на такую точку используют следующую форму URL:

```
<A HREF="http://www.intuit.ru/
index.html#point">Ссылка на точку "point" в
документе "index.html"</A>
```

На описании простых гипертекстовых ссылок обзор средств языка HTML, ориентированных на текстовое представление информации и организацию гипертекстовых баз данных, можно закончить.

4. Лекция: Графика:
В этой лекции рассматриваются принципы применения графических образов при HTML-разметке.

Использование графики в HTML

Для того чтобы вставить в Web-страницу изображение, необходимо либо нарисовать его, либо взять уже готовое. В любой программе рисования можно создать простое изображение и сохранить его в нужном формате. Если программа этот формат не поддерживает, необходимо преобразовать файл в требуемый формат. Существует множество программ, предназначенных для преобразования одного графического формата в другой. Позаимствовать же картинки можно из различных программных пакетов или с других Web-страниц в Internet, содержащих библиотеки свободного доступа художественных изображений. Когда браузер выводит Web-страницу с изображением, соответствующий графический файл временно хранится в памяти компьютера. В большинстве браузеров есть команда, позволяющая сохранить файл на локальном диске. Существует также множество других вариантов получения графических файлов.

Изображения могут нести определенную информацию, да и просто придают Web-странице привлекательный вид. Приведем наиболее распространенные случаи применения изображений:

- логотип компании на деловой странице;
- графика для рекламного объявления;
- различные рисунки;
- диаграммы и графики;
- художественные шрифты;
- подпись автора страницы;
- применение графической строки в качестве горизонтальной разделительной линии;
- применение графических маркеров для создания красивых маркированных списков.

Теперь рассмотрим как вставить изображение в Web-страницу. Тегом HTML, который заставляет браузер выводить изображение, является <IMG> с обязательным атрибутом SRC (SouRCe, источник). Имя файла представляет собой имя выводимого графического файла. Замыкающего тега не требуется.

Пример вставки изображения:

```
<IMG SRC="image.gif" ALT="ИЗОБРАЖЕНИЯ">
```

Изображения на Web-странице могут использоваться в качестве гипертекстовых ссылок, как и обычный текст. Читатель щелкает на изображении и отправляется на другую страницу или переходит к другому изображению. Для обозначения изображения как гипертекстовой метки используется тот же тег <A>, что и для текста, но между <A> и </A> вставляется тег изображения <IMG> :

```
<A HREF="адрес файла или изображения"><IMG SRC="image.gif"></A>
```

При этом изображение, используемое в качестве гипертекстовой ссылки, обводится дополнительной рамкой.

Атрибуты и их аргументы

Тег изображения имеет один обязательный атрибут SRC и необязательные: ALT, ALIGN, USEMAP, HSPACE, VSPACE, BORDER, WIDTH, HEIGHT.

Атрибут SRC

Указывает файл изображения и путь к нему; изображение должно быть загружено в браузер и размещено в том месте документа, где расположен тег изображения.

Атрибут ALT

Позволяет указать текст, который будет выводиться вместо изображения браузерами, неспособными представлять графику. В некоторых случаях при недостаточной пропускной способности линий связи пользователи отключают отображение графики. Наличие названий вместо картинок облегчает восприятие Web-страниц в таком режиме.

Атрибут ALIGN

Определяет положение изображения относительно окружающего его текста. Возможные значения аргумента — ["top" | "middle" | "bottom"] (соответственно, "вверху", "посередине", "внизу").

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.

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

Дополнительные возможные значения аргумента — ["left" | "right" | "top" | "texttop" | "middle" |"absmiddle" | "baseline" | "bottom" | "absbottom"].

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.

ALIGN="top" выравнивает верх изображения по верхнему краю самого высокого элемента в строке окружающего текста точно так же, как при использовании стандартного набора атрибутов.

ALIGN="texttop" 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="middle" выравнивает центр изображения по базовой линии строки окружающего текста точно так же, как при использовании стандартного набора атрибутов.

ALIGN="absmiddle" выравнивает центр изображения по центру строки окружающего текста.

ALIGN="baseline" выравнивает нижний край изображения по базовой линии строки окружающего текста, то есть производит такое же действие, как и `ALIGN="bottom"`.

ALIGN="bottom" выравнивает нижний край изображения по базовой линии строки окружающего текста точно так же, как при использовании стандартного набора атрибутов.

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

Атрибут USEMAP

Если присутствуют атрибут `USEMAP` и тег `<MAP>`, изображение становится чувствительной картой, или "графическим меню". Если щелкнуть кнопкой мыши на активной области изображения, для которого определен атрибут `USEMAP`, произойдет гипертекстовый переход к информационному ресурсу, установленному для этой области. Более подробно этот вопрос будет рассматриваться в следующем разделе.

Атрибут BORDER

Целочисленное значение аргумента определяет толщину рамки вокруг изображения. Если значение равно нулю, рамка отсутствует. Чтобы не вводить пользователей в заблуждение, не стоит задействовать `BORDER=0` в изображениях, которые представляют собой часть элемента якоря, поскольку рисунки, применяемые в качестве гиперссылок, обычно выделяются цветной рамкой.

Атрибут HSPACE

Целочисленное значение этого атрибута задает горизонтальное расстояние между вертикальной границей страницы и изображением, а также между изображением и огибающим его текстом.

Атрибут VSPACE

Целочисленное значение этого атрибута задает вертикальное расстояние между строками текста и изображением.

Атрибуты WIDTH и HEIGHT

Оба атрибута задают целочисленные значения размеров изображения по горизонтали и по вертикали соответственно. Это позволяет уменьшить время загрузки страницы с графикой. Браузер сразу отводит рамку для изображения и продолжает загружать текст на страницу. Пока загружается графика, пользователь может начать читать текст. Определить размер изображения нетрудно, для этого достаточно воспользоваться любой программой просмотра графических файлов, например ACDSee или графическим редактором Corel PhotoPaint или Adobe Photoshop. Откройте файл в графическом

редакторе и определите размер картинки в пикселах. В теге изображения следует указать ширину и высоту картинки.

```
<IMG SRC="image.gif" ALT="изображение" WIDTH="100" HEIGHT="200" HSPACE="10"
VSPACE="10"
BORDER="2" ALIGN="left">
```

Форматы графических файлов

Самыми распространенными графическими форматами в Web являются GIF и JPEG. GIF — наиболее подходящий формат для обмена изображениями между системами. Архивы с изображениями в формате GIF можно найти на многих серверах Internet. Данный формат поддерживают многие графические приложения, в том числе все программы просмотра графики World Wide Web.

Однако у этого формата есть одно серьезное ограничение: он не поддерживает изображения с глубиной цвета больше восьми бит на пиксел. Обычно восьми бит на пиксель оказывается достаточно для контурных изображений типа комиксов и рисунков, где используется ограниченное количество цветов, или для небольших картинок, где для цветопередачи хватает 256 оттенков. Однако для больших изображений фотографического качества больше подходит формат JPEG.

Формат GIF использует один из лучших алгоритмов сжатия LZW, который изначально не предназначался специально для графики. Он не очень подходит для работы с двухцветными (черно-белыми) или фотографическими изображениями.

С развитием аппаратного обеспечения, поддерживающего высокое разрешение и богатую цветовую гамму, графические файлы значительно увеличились в размерах. Профессиональные художники теперь, как правило, работают с файлами, содержащими 10 и более мегабайт данных на каждое изображение. Даже пользователи с более скромными запросами подчас имеют дело с изображениями 640 на 480 пикселей в 256 цветах (а это более 300 килобайт). Кроме того, многие сейчас начинают работать с полноцветными изображениями 1024 на 768 пикселей (это более 2,3 мегабайт данных). Так как высококачественные изображения встречаются все чаще, ограничения, накладываемые традиционными методами сжатия (например, LZW), становятся все более ощутимыми.

Для поиска оптимального способа сжатия изображений фотографического качества две международные организации по стандартизации, International Telecommunications Union (ITU, Международный союз телекоммуникаций) и International Organization for Standartization (ISO, Международная организация по стандартизации), создали Joint Photographic Experts Group (JPEG, объединенная экспертная группа по фотографии). С тех пор сокращение "JPEG" используется как название этой техники сжатия. Кроме того, оно входит в названия некоторых использующих ее файловых форматов.

Имя JPEG указывает на метод сжатия, но не на формат файла. На самом деле метод сжатия JPEG используют как многочисленные мало различающиеся форматы, зачастую известные, например "JPEG", так и единичные радикально отличающиеся форматы, такие как TIFF и Quick Time. К счастью, все же большинство форматов, известных под именем "JPEG", очень похожи, и, скорее всего, у вас не возникнет с этим проблем, однако знать о возможных осложнениях не помешает.

Формат JPEG отличается от других графических форматов прежде всего тем, что он использует метод сжатия "с потерями". JPEG частично идентифицирует и удаляет ту информацию, которая несущественна для восприятия изображения. В результате JPEG может достигать высокого уровня сжатия без заметных потерь в качестве изображения.

Метод сжатия "с потерями" имеет много реализаций. JPEG достигает существенного сжатия за счет отбрасывания той графической информации, которая обычно не проявляется в реальных изображениях. Однако при сжатии с помощью JPEG изображений с четкими контурами линии начинают заметно "дрожать". Так, например, если изображение содержит какие-либо подписи, подобный эффект может возникнуть вокруг символов. Этот эффект можно свести к минимуму, задав очень высокие значения параметра качества, однако при этом нельзя достичь приемлемых показателей сжатия.

Так как JPEG предполагает сжатие с потерями, при создании файлов необходимо быть внимательным. Большинство программ, создающих такие файлы, позволяют задавать значение параметра качества изображения. Обычно оно варьирует от нуля до ста. Нижние значения позволяют при сжатии JPEG отбрасывать больше информации, в результате чего получаются файлы меньшего размера. В свою очередь, высокие значения ограничивают количество информации, которой можно пренебречь во время сжатия.

Одна из наиболее распространенных ошибок заключается в интерпретации значения параметра качества от нуля до ста как процента сохраняемых данных. Чтобы развеять это заблуждение, некоторые современные программные продукты JPEG просто используют шкалу "лучшее сжатие" — "лучшее качество".

Хитрость заключается в том, чтобы при наименьшей величине параметра качества получить изображение без видимого его ухудшения. Лучше начинать со средних значений и внимательно оценивать результат. Если вы отмечаете некоторое ухудшение, попробуйте увеличить значение параметра, если нет — попытайтесь его уменьшить. При просмотре изображения обращайте внимание на следующие моменты: четкость очертаний и углов, например вокруг текста, или контур детали изображения, выделяющейся на общем фоне. Часто бывает заметно, что контур "смазан" или линия "дрожит".

Сжатие JPEG использует мозаику размером восемь на восемь пикселей. Если задаются слишком низкие значения качества, ее границы могут стать заметны. Если у вас уже есть изображения в GIF или другом восьмиразрядном формате, возможно, вы захотите попробовать конвертировать их в JPEG. Несмотря на то, что иногда это все же приводит к уменьшению необходимого для хранения файлов пространства, в большинстве случаев игра не стоит свеч. Если вы все же хотите попытаться, сначала выясните, сколько цветов использует изображение GIF. Если в нем только 64 цвета, то конверсия вряд ли себя оправдает, так как изображение с такой бедной цветовой палитрой не имеет тех плавных цветовых переходов, которые хорошо сжимает JPEG. В результате вы просто ухудшите качество изображения, не освободив места.

Одна из серьезных проблем конверсии изображений GIF в JPEG заключается в том, что изображения в формате GIF, лимитированные набором из 256 (или менее) цветов, часто используют клиширование (dithering) и полутона (halftoning), в результате чего пиксели двух цветов смешиваются для получения эффекта третьего тона. В результате использования этой техники образуются шаблоны, крайне плохо сжимаемые с помощью JPEG. Отдельные программы позволяют усреднять значения этих шаблонов, "смягчая" таким образом изображение до преобразования, в результате чего сжатие с помощью JPEG оказывается более эффективным.

Активные изображения

Активные изображения (image maps), или изображения, чувствительные к щелчкам мыши, позволяют создать на узле графические меню произвольной формы. Активное изображение — это изображение с так называемыми активными областями (hot spots), которые ссылаются на URL других страниц или узлов.

Есть два метода формирования активных изображений: на сервере и у клиента. Изображения первого типа используют сервер для того, чтобы найти соответствующий данной активной области URL и передать на браузер нужную страницу. Активные изображения, работающие на клиентской машине, задают информацию об активной области на HTML-странице, так что браузер сам выясняет, какие области являются активными, и запрашивает с сервера соответствующую страницу.

Активные изображения, работающие у клиента, имеют несколько преимуществ. Во-первых, страницы с ними можно перенести на другой сервер. Во-вторых, серверу не приходится выполнять лишнюю работу (например, просматривать всю информацию об активных областях), то есть нагрузка на сервер уменьшается. При использовании работающих на сервере активных изображений в каталоге cgi-bin сервера должен быть соответствующий сценарий. Из соображений безопасности многие системные администраторы не записывают сценарии в каталог cgi-bin. Поэтому более подробно мы рассмотрим создание активных изображений у клиента.

Создание активного изображения. Процесс создания активного изображения состоит из двух этапов. Сначала необходимо определить на картинке области, которые нужно сделать активными, а потом соотнести их со ссылками на другие URL. Активные области задаются перечислением их координат (в пикселах). Все это можно сделать вручную, определив координаты углов активных областей, но гораздо проще воспользоваться какой-нибудь программой, например MapEdit.

Определить карту легко. Нужно открыть в MapEdit HTML-файл, содержащий изображение, на котором требуется создать активные области, после чего выбранное изображение будет загружено в рабочее окно. Затем следует выбрать тип активной области (квадрат, треугольник и круг), щелкнуть и потянуть мышкой, обозначив границу области. Программа автоматически производит запись в HTML-файл, описывающий границы активной области. Затем этой области нужно приписать URL. В любых местах изображения можно нарисовать активные области и определить для каждой из них URL. Важно оставлять между областями немного места, чтобы при чтении быть уверенным, что активизируется правильная ссылка. Границы активных областей задаются координатами углов прямоугольника и многоугольника или центра и радиуса круга. Если вы решили делать активное изображение у клиента, Map Edit поставляет данные только для тегов <MAP>. Вам придется самим задать тег изображения с атрибутом USEMAP и поместить его после тега </MAP>. Не забудьте перед именем карты в атрибуте USEMAP записать символ "#" следующим образом:

```
<IMG SRC="mymap.gif" USEMAP="#sitemap">
```

Активные изображения у клиента работают независимо от программного обеспечения сервера и не перестанут функционировать, даже если файлы будут перенесены на другой сервер. Таким изображениям требуются только две вещи: браузер, поддерживающий HTML 3.0, и информация о карте, записанная в HTML-файле. Приведем пример активных изображений.

```
<IMG SRC="image.gif" ALT="Изображения" USEMAP="#imap">
<MAP NAME="imap">
<AREA SHAPE="rect" COORDS="0,0,100,100"
HREF="http://www.intuit.ru/help/index.html">
<AREA SHAPE="rect" COORDS="100,0,200,100"
HREF="http://www.intuit.ru/shop/index.html">
<AREA SHAPE="default" nohref>
</MAP>
```

Изображения в миниатюре

Часто для иллюстрации какой-то темы требуются изображения большого размера, загружаться они будут достаточно долго. В том месте, где требуется разместить большой рисунок, можно поместить маленькую его копию и сделать ссылку на полномасштабное изображение. Те посетители, которым это действительно интересно, смогут посмотреть изображение полностью, а все остальные пролистнут страницу, не задерживаясь. Такая методика особенно хороша для обложек книг, фотографий, рекламных листков, которые не все читатели захотят изучить в деталях.

5. Лекция: Таблицы в HTML:
В этой лекции подробно рассматриваются принципы применения таблиц в HTML-разметке. Это и табличная организация текста, и табличная координатная сетка, и организованная в таблицы графика.

Средства описания таблиц в HTML

По мере развития WWW стало ясно, что средств, которые заложены в HTML, недостаточно для качественного отображения различного типа документов. Недостатком HTML было отсутствие в его составе средств отображения таблиц. Для этой цели обычно использовался предформатированный текст (тег <PRE>), в котором таблица обрисовывалась символами ASCII. Но такая форма представления таблиц была недостаточно высокого качества и выбивалась из общего стиля документа. После введения таблиц в HTML у Web-мастеров появился не просто инструмент для размещения текстовых и числовых данных, а мощное средство дизайна для размещения в нужном месте экрана графических образов и текста.

Создание таблиц в HTML

Для описания таблиц используется тег <TABLE>. Тег <TABLE>, как и многие другие, автоматически переводит строку до и после таблицы.

Создание строки таблицы - тег <TR>

Тег <TR> (Table Row, строка таблицы) создает строку таблицы. Весь текст, другие теги и атрибуты, которые требуется поместить в одну строку, должны размещаться между тегами <TR></TR>.

Определение ячеек таблицы - тег <TD>

Внутри строки таблицы обычно размещаются ячейки с данными. Каждая ячейка, содержащая текст или изображение, должна быть окружена тегами <TD></TD>. Число тегов <TD></TD> в строке определяет число ячеек

```
<HTML>
<BODY>
  <H1 ALIGN=center>Таблица</H1>
  <CENTER>
  <TABLE BORDER>
    <TR>
      <TD COLSPAN=3>Если в таблице два
          тега TR, то в ней две строки.</TD>
    </TR>
    <TR>
      <TD>Если в строке три тега TD,</TD>
      <TD>то в ней</TD>
```

```
      <TD>три столбца.</TD>
    </TR>
  </TABLE>
  </CENTER>
</BODY>
</HTML>
```

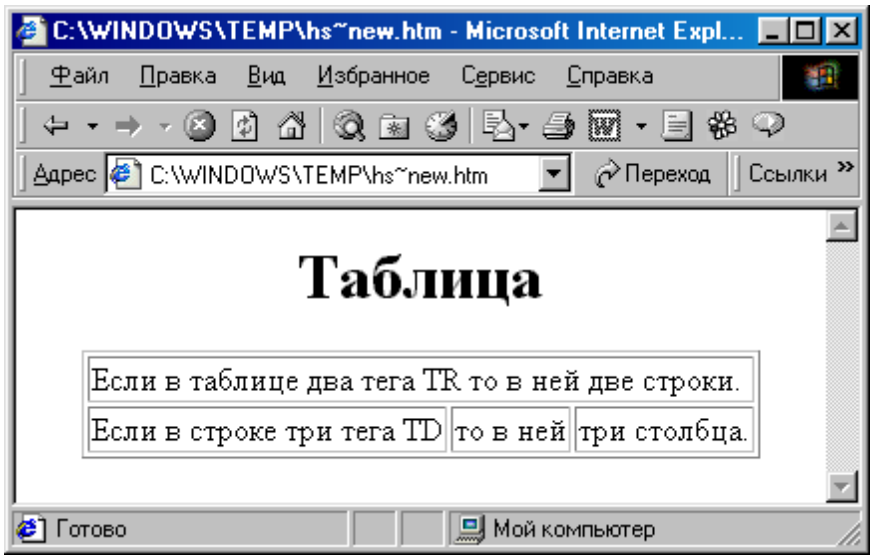


Рис. 4.1. Использование таблиц в дизайне

Заголовки столбцов таблицы - тег <TH>

Заголовки для столбцов и строк таблицы задаются с помощью тега заголовка <TH></TH> (Table Header, заголовок таблицы). Эти теги подобны <TD></TD>. Отличие состоит в том, что текст, заключенный между тегами <TH></TH>, автоматически записывается жирным шрифтом и по умолчанию располагается посередине ячейки. Центрирование можно отменить и выровнять текст по левому или правому краю. Если воспользоваться <TD></TD> с тегом <B> и атрибутом <ALIGN=center>, текст тоже будет выглядеть как заголовок. Однако следует иметь в виду, что не все браузеры поддерживают в таблицах жирный шрифт, поэтому лучше задавать заголовки таблиц с помощью <TH>.

```
<HTML>
<BODY>
  <TABLE BORDER>
    <TR>
      <TH>Заголовок центрирован по умолчанию
      </TH>
      <TH COLSPAN=2>Заголовок может объединять
                    столбцы</TH>
    </TR>
    <TR>
      <TH>Заголовок может быть расположен
          перед столбцами</TH>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
    <TR>
      <TH ROWSPAN=3>Заголовок может объединять
                    строки</TH>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
    <TR>
      <TD>Текст или данные</TD>
```

```
      <TD>Текст или данные</TD>
    </TR>
    <TR>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Использование заголовков таблицы - тег <CAPTION>

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

```
<HTML>
<BODY>
  <TABLE BORDER>
  <CAPTION ALIGN=top>Заголовок над таблицей
  </CAPTION>
    <TR>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
  </TABLE>
  <TABLE BORDER>
  <CAPTION ALIGN=bottom>Заголовок под таблицей
  </CAPTION>
    <TR>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Атрибут NOWRAP

Обычно любой текст, не помещающийся в одну строку ячейки таблицы, переходит на следующую строку. Однако при использовании атрибута NOWRAP с тегами <TH> или <TD> длина ячейки расширяется настолько, чтобы заключенный в ней текст поместился в одну строку.

Атрибут COLSPAN

Теги <TD> и <TH> модифицируются с помощью атрибута COLSPAN (Column Span, соединение столбцов). Если вы хотите сделать какую-нибудь ячейку шире, чем верхняя или нижняя, можно воспользоваться атрибутом COLSPAN, чтобы растянуть ее над любым количеством обычных ячеек.

```
<HTML>
```

```
<BODY>
  <CENTER>
  <TABLE BORDER=3>
    <TR>
      <TD>Если вы хотите сделать какую-нибудь
          ячейку шире, чем верхняя или нижняя,
      </TD>
      <TD>можно воспользоваться атрибутом
          COLSPAN=2, </TD>
    </TR>
    <TR>
      <TD BGCOLOR=white COLSPAN=2>чтобы
          растянуть ее над любым количеством
          обычных ячеек.</TD>
    </TR>
  </TABLE>
  </CENTER>
</BODY>
</HTML>
```

Атрибут ROWSPAN

Атрибут ROWSPAN, используемый в тегах <TD> и <TH>, подобен атрибуту COLSPAN=, только он задает число строк, на которые растягивается ячейка. Если вы указали в атрибуте ROWSPAN=s число, большее единицы, то соответствующее количество строк должно находиться в растягиваемой ячейке. Внизу таблицы ее поместить нельзя.

Атрибут WIDTH

Атрибут WIDTH применяется в двух случаях. Можно поместить его в тег <TABLE>, чтобы дать ширину всей таблицы, а можно использовать в тегах <TD> или <TH>, чтобы задать ширину ячейки или группы ячеек. Ширину можно указывать в пикселах или в процентах. Например, если вы задали в теге <TABLE> WIDTH=250, вы получите таблицу шириной 250 пикселей независимо от размера страницы на мониторе. При задании WIDTH=50% в теге <TABLE> таблица будет занимать половину ширины страницы при любом размере изображения на экране. Так что, указывая ширину таблицы в процентах, имейте в виду, что если у пользователя узкая область просмотра, ваша страница может выглядеть несколько странно. Если вы пользуетесь пикселами, и таблица оказывается шире области просмотра, внизу появится полоса прокрутки для перемещения вправо и влево по странице. В зависимости от поставленных задач и тот, и другой способ задания ширины таблицы может оказаться полезным.

```
<HTML>
<BODY>
  <TABLE BORDER WIDTH=100%>
    <TR>
      <TD ALIGN=center>Текст или данные -
          ширина 100%</TD>
    </TR>
  </TABLE>
или<BR>
  <TABLE BORDER WIDTH=50%>
    <TR>
      <TD ALIGN=center>Текст или данные -
          ширина 50%</TD>
    </TR>
  </TABLE>
или<BR>
  <TABLE BORDER WIDTH=200>
    <TR>
```

```
      <TD ALIGN=center>Текст или данные -
          ширина 200 пикселов</TD>
    </TR>
  </TABLE>
или<BR>
  <TABLE BORDER WIDTH=100>
    <TR>
      <TD ALIGN=center>Текст или данные -
          ширина 100 пикселов</TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

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 <BR>. 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.

Атрибут CELLPADDING

Данный атрибут определяет ширину пустого пространства между содержимым ячейки и ее границами, то есть задает поля внутри ячейки.

```
<HTML>
<BODY>
  <CENTER>
  <TABLE BORDER CELLPADDING=20>
    <TR>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
    <TR>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
  </TABLE>
<BR>
  <TABLE BORDER CELLPADDING=0>
    <TR>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
    <TR>
      <TD>Текст или данные</TD>
        <TD>Текст или данные</TD>
        <TD>Текст или данные</TD>
    </TR>
  </TABLE>
  </CENTER>
</BODY>
</HTML>
```

Атрибуты ALIGN и VALIGN

Теги <TR>, <TD> и <TH> можно модифицировать с помощью атрибутов ALIGN и VALIGN. Атрибут ALIGN определяет выравнивание текста и графики по горизонтали, то есть по левому или правому краю, либо по центру. Горизонтальное выравнивание может быть задано несколькими способами:

ALIGN=bleedleft прижимает содержимое ячейки вплотную к левому краю.

ALIGN=left выравнивает содержимое ячейки по левому краю с учетом отступа, заданного атрибутом CELLPADDING.

ALIGN=center располагает содержимое ячейки по центру.

ALIGN=right выравнивает содержимое ячейки по правому краю с учетом отступа, заданного атрибутом CELLPADDING.

```
<HTML>
<BODY>
  <TABLE BORDER WIDTH=100%>
    <TR>
      <TD ALIGN=left>Текст или данные</TD>
        <TD ALIGN=center>Текст или данные</TD>
        <TD ALIGN=right>Текст или данные</TD>
    </TR>
    <TR>
      <TD ALIGN=right>Текст или данные</TD>
      <TD ALIGN=center>Текст или данные</TD>
      <TD ALIGN=left>Текст или данные</TD>
    </TR>
    <TR>
      <TD ALIGN=right>Текст или данные</TD>
      <TD ALIGN=right>Текст или данные</TD>
      <TD ALIGN=right>Текст или данные</TD>
    </TR>
    <TR>
      <TD ALIGN=center>Текст или данные</TD>
      <TD ALIGN=center>Текст или данные</TD>
      <TD ALIGN=center>Текст или данные</TD>
    </TR>
    <TR>
      <TD ALIGN=left>Текст или данные</TD>
      <TD ALIGN=left>Текст или данные</TD>
      <TD ALIGN=left>Текст или данные</TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Атрибут VALIGN осуществляет выравнивание текста и графики внутри ячейки по вертикали. Вертикальное выравнивание может быть задано несколькими способами:

VALIGN=top выравнивает содержимое ячейки по ее верхней границе.

VALIGN=middle центрирует содержимое ячейки по вертикали.

VALIGN=bottom выравнивает содержимое ячейки по ее нижней границе.

```
<HTML>
```

```
<BODY>
  <CENTER>
  <TABLE BORDER WIDTH=90%>
    <TR>
      <TD WIDTH=100>Атрибут VALIGN осуществляет
          выравнивание текста и графики внутри
          ячейки по вертикали.</TD>
      <TD VALIGN=top>верх,</TD>
      <TD VALIGN=middle>середина,</TD>
      <TD VALIGN=bottom>низ.</TD>
    </TR>
    <TR VALIGN=top>
      <TD VALIGN=top>Выравнивает содержимое
          ячейки по ее верхней границе.</TD>
      <TD>верх,</TD>
      <TD>верх,</TD>
      <TD>верх.</TD>
    </TR>
    <TR VALIGN=middle>
      <TD VALIGN=middle>Центрирует содержимое
          ячейки по вертикали.</TD>
      <TD>середина,</TD>
      <TD>середина,</TD>
      <TD>середина.</TD>
    </TR>
    <TR VALIGN=bottom>
      <TD VALIGN=bottom>Выравнивает содержимое
          ячейки по ее нижней границе.</TD>
      <TD>низ,</TD>
      <TD>низ,</TD>
      <TD>низ.</TD>
    </TR>
  </TABLE>
  </CENTER>
</BODY>
</HTML>
```

Атрибут BORDER

В теге <TABLE> часто определяют, как будут выглядеть рамки, то есть линии, окружающие ячейки таблицы и саму таблицу. Если вы не зададите рамку, то получите таблицу без линий, но пространство под них будет отведено. Того же результата можно добиться, задав <TABLE BORDER=0>. Иногда хочется сделать границу потолще, чтобы она лучше выделялась. Можно для привлечения внимания к рисунку или тексту задать исключительно жирные границы. При создании вложенных таблиц приходится делать для разных таблиц границы различной толщины, чтобы их легче было различать.

Атрибут CELLSPACING

Атрибут CELLSPACING определяет ширину промежутков между ячейками в пикселах. Если этот атрибут не указан, по умолчанию задается величина, равная двум пикселам. С помощью атрибута CELLSPACING= можно размещать текст и графику там, где вам нужно. Если вы хотите оставить пустое место, можно вписать в ячейку пробел.

```
<HTML>
<BODY>
  <CENTER>
  <TABLE BORDER CELLSPACING=20>
    <TR>
      <TD>Текст или данные</TD>
```

```
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
    <TR>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
  </TABLE>
  <TABLE BORDER CELLSPACING=10>
    <TR>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
    <TR>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
  </TABLE>
  <TABLE BORDER CELLSPACING=0>
    <TR>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
      <TD>Текст или данные</TD>
    </TR>
    <TR>
      <TD>Текст или данные</TD>
      <TD></TD>
      <TD>Текст или данные</TD>
    </TR>
  </TABLE>
  </CENTER>
</BODY>
</HTML>
```

Атрибут BGCOLOR

Данный атрибут позволяет установить цвет фона. В зависимости от того, с каким тегом (TABLE, TR, TD) он применяется, цвет фона может быть установлен для всей таблицы, для строки или для отдельной ячейки. Значением данного атрибута является RGB-код или стандартное название цвета.

```
<HTML>
<BODY>
    <CENTER>
    <TABLE BORDER BGCOLOR=yellow>
      <TR BGCOLOR=blue>
        <TD>Текст или данные</TD>
        <TD BGCOLOR=red>Текст или данные
        </TD>
        <TD>Текст или данные</TD>
      </TR>
      <TR BGCOLOR=green>
        <TD>Текст или данные</TD>
        <TD>Текст или данные</TD>
        <TD BGCOLOR=lime>Текст или данные
        </TD>
      </TR>
    </TABLE>
    </CENTER>
```

```
</BODY>
</HTML>
```

Атрибут BACKGROUND

Данный атрибут задает фоновое изображение для таблиц. Применим к тегам TABLE и TD. Его значением является URL файла с фоновым изображением. Применение этого атрибута рассматривается ниже.

Использование таблиц в дизайне страницы

Таблицы хороши тем, что при желании можно сделать их границы невидимыми. Это позволяет с помощью тега <TABLE> красиво размещать на странице текст и графику. Пока тег <TABLE> остается единственным мощным средством форматирования в HTML. Дизайнеры Web-страниц сейчас обладают практически той же свободой в отношении использования "пустого пространства", что и создатели печатных страниц. Таблицы лучше всего помогают отойти от иерархического размещения текста на Web-страницах.

Если браузер поддерживает таблицы, он обычно правильно отображает наиболее интересные эффекты, полученные с их помощью

```
<HTML>
<BODY>
  <CENTER>
  <TABLE CELLPADDING="10" CELLSPACING="0"
         BORDER="16">
    <TR>
      <TD ALIGN="center">
        <H2>Интернет-Университет
            Информационных Технологий</H2>
        <H3>Добро пожаловать!</H3>
        <TABLE BORDER WIDTH="100%">
          <TR>
            <TD ALIGN="center"><I>Учебный курс
             "Основы Web-технологий"</I></TD>
          </TR>
        </TABLE>
      </TD>
    </TR>
  </TABLE>
  </CENTER>
</BODY>
</HTML>
```

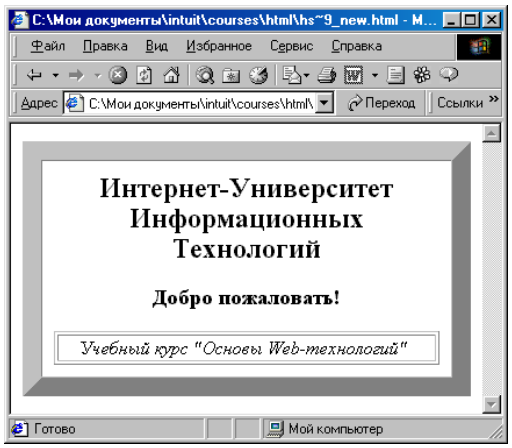


Рис. 4.2. Использование таблиц в дизайне

Создание разноцветных таблиц

Есть несколько способов раскрасить таблицу, в основном они зависят от используемого браузера.

Цветные границы в Netscape Navigator. Вы не только можете окружить таблицу красивой рамкой, но еще и задать для нее цвет, отличный от цветов текста и фона. Создайте простой серый GIF (или любой GIF, который вы хотели бы иметь в качестве фона) и определите его в теге <BODY> как фон страницы. Затем задайте цвет фона страницы. В результате ваш тег <BODY> будет выглядеть примерно так:

```
<BODY BACKGROUND="image.gif" BGCOLOR="#FF0000">
```

Вы создали двойной фон - GIF и заданный цвет. В результате фоновый цвет будет виден на всех границах таблиц и горизонтальных линиях (<HR>). Вне зависимости от того, является ваш фоновый GIF серым или нет, цветные линии и границы таблиц будут заметно выделяться. Если фоновый GIF устроен не слишком сложно, время загрузки страницы возрастет лишь немного.

6. Лекция: HTML-формы:

В этой лекции разбирается возможность взаимодействия читателя HTML-страниц с сервером Web-узла. Подробно описывается подмножество HTML, которое носит название HTML-формы.

HTML-формы

Формы были созданы и используются в WWW для получения отклика пользователя на предоставленную информацию и сбора данных о пользователе. После заполнения пользователем формы и запуска процесса ее обработки, информация из нее попадает к программе, работающей на сервере. Простота использования тега <MAILTO:> в формах позволяет даже владельцам небольших страниц получать отклик от своих читателей. Для обработки большого количества откликов используются программы, поддерживающие Common Gateway Interface (CGI) и расположенные на сервере, в адрес которого поступают отклики. Таким образом пользователь может интерактивно взаимодействовать с Web-сервером через Internet.

Задание формы — элемент FORM

Элемент FORM обозначает документ как форму и определяет границы использования других тегов, размещаемых в форме. Тег <FORM> определяется последовательностью тегов <INPUT>, размещенных внутри пары <FORM> и </FORM>. В форме используется как метод (method), так и действие (action) для описания обработки данных, вводимых пользователем в форму. Метод (GET или POST) определяет, как должны обрабатываться входные данные из формы, а действие указывает на URI (Uniform Resource Identifier) программы, ответственной за обработку этих данных.

```
<FORM METHOD=post
 ACTION=mailto:yourname@your.email.address>
```

Определение элементов управления формы — тег <INPUT>

Данный тег используют для определения области внутри формы, куда вводятся данные. Он формирует поле для ввода информации пользователем. Это может быть текстовое поле, опция, изображение или кнопка. Вид поля ввода определяется значением атрибута TYPE.

Атрибут TYPE=text

Когда пользователю необходимо ввести небольшое количество текста (одну или несколько строк), используется тег <INPUT>, и атрибут TYPE устанавливается в значение text. Это значение принято по умолчанию и указывать его необязательно. Кроме того, задается атрибут NAME для определения наименования переменной поля.

```
Ваше имя <INPUT NAME=Name SIZE=35>
```

Имеется еще три дополнительных атрибута, которые можно использовать. Первый называется MAXLENGTH, он ограничивает число символов, вводимых пользователем в текущее поле. По умолчанию данное число не ограничено. Вторым атрибутом является SIZE, определяющий размер видимой на экране области, занимаемой текущим полем. Значение по умолчанию определяется типом браузера. Если значение MAXLENGTH больше, чем SIZE, браузер будет прокручивать данные в окне. Последним из дополнительных атрибутов является атрибут VALUE, обеспечивающий начальное значение поля ввода.

Атрибут TYPE=checkbox

Для создания независимых флагов в формах HTML используется тег <INPUT> со значением атрибута TYPE=checkbox. В зависимости от содержания формы пользователь может отметить несколько флагов. Когда форма использует тег <INPUT> со значением атрибута CHECKBOX, в нем должны присутствовать и атрибуты NAME, и VALUE. Атрибут NAME указывает на наименование данного поля (флага) ввода. В атрибуте VALUE будет содержаться значение поля.

```
<BR>Россия<INPUT NAME="Страна" TYPE=checkbox
VALUE="Россия">
Страны СНГ<INPUT NAME="Страна" TYPE=checkbox
VALUE="СНГ">
```

В некоторых случаях необходимо инициализировать данный флаг, как уже отмеченный. В таких случаях тег <INPUT> должен содержать атрибут CHECKED.

Атрибут TYPE=radio

В некоторых случаях требуется организовать выбор одного из нескольких возможных значений. Для создания формы ввода при выборе пользователем одного значения из нескольких возможных необходимо использовать тег <INPUT> с атрибутом TYPE=radio. Когда в форме применяется данный атрибут, в теге <INPUT> должны быть указаны атрибуты NAME и VALUE. Атрибут NAME указывает наименование соответствующего поля (кнопки). Атрибут VALUE содержит значение поля.

```
<BR>Пол мужской<INPUT NAME="Пол" TYPE=radio
VALUE="Мужской">
Пол женский<INPUT NAME="Пол" TYPE=radio
VALUE="Женский">
```

Атрибут TYPE=image

В зависимости от содержимого формы может случиться так, что пользователю потребуется щелкнуть мышью на изображении, чтобы завершить работу с формой. Для этого программисты используют тег <INPUT> с атрибутом TYPE=image. Когда пользователь щелкает мышью по изображению, браузер сохраняет координаты соответствующей точки экрана. Далее он "обрабатывает" введенную в форму информацию. Когда форма использует атрибут image, тег <INPUT> должен содержать также атрибуты NAME и SRC. NAME указывает наименование поля ввода формы. Атрибут SRC содержит URI файла — источника изображения. Атрибут ALIGN является дополнительным и используется аналогично тому же атрибуту тега <IMG>.

```
<BR>Выберите точку<INPUT TYPE=image NAME=point
SRC=image.gif>
```

Атрибут TYPE=password

Если в форме требуется организовать ввод пароля, то атрибут TYPE можно установить в значение password (TYPE=password). Используя данный тип, можно организовать ввод пароля без вывода на экран составляющих его символов. При этом следует помнить, что введенные данные передаются по незащищенным каналам связи и могут быть перехвачены.

```
<BR>Подпись<INPUT NAME=login>Пароль
<INPUT TYPE=password NAME="Слово">
```

Атрибут TYPE=reset

Когда пользователь заполняет форму, ему может потребоваться начать все сначала. На такой случай существует кнопка Reset, по которой пользователь может щелкнуть мышью, чтобы вернуться к первоначальным значениям полей. Когда пользователь выбирает данную кнопку, форма восстанавливает первоначальные значения всех элементов, в которых присутствует атрибут TYPE=reset. Для создания кнопки Reset используется тег <INPUT> с атрибутом TYPE=reset. Браузер в свою очередь будет выводить изображение данной кнопки. Если в форме используется атрибут reset, тег <INPUT> может дополнительно содержать атрибут VALUE. Данный атрибут определяет надпись на изображении кнопки.

```
<INPUT TYPE=reset VALUE="Очистить форму">
```

Атрибут TYPE=submit

Используя форму HTML для ввода информации от пользователя, необходимо обеспечить пользователю возможность завершить ввод данных. Для этого используется тег <INPUT> с атрибутом TYPE=submit. Браузер, в свою очередь, выводит данный элемент, как кнопку, по которой пользователь может щелкнуть, чтобы завершить процесс редактирования. Когда в форме используется тег <INPUT> с атрибутом submit, данный элемент может содержать два дополнительных атрибута: NAME и VALUE. Атрибут NAME хранит название переменной поля в вашей форме. Атрибут VALUE — определяет значение элемента формы, которое будет отправлено на сервер или получено с помощью клиентских скриптов.

```
<BR><INPUT TYPE=submit
VALUE="Отправить сообщение">
```

Атрибут TYPE=hidden

Скрытые поля. Добавление в тег INPUT атрибута TYPE=hidden позволит включить в отправляемую форму значения атрибутов NAME и VALUE, которые пользователь изменить не может. Такие метки полезны при наличии нескольких форм для дальнейшей обработки данных.

Создание многострочных областей ввода текста — тег <TEXTAREA>

В зависимости от типа формы может потребоваться организовать ввод большого количества текста. В таких случаях используется тег <TEXTAREA> для создания текстового поля из нескольких строк. Данный тег использует три атрибута: COLS, NAME и ROWS.

Атрибут COLS

Указывает (число символов) число колонок, содержащихся в текстовой области.

Атрибут NAME

Определяет наименование поля.

Атрибут ROWS

Задает количество видимых строк текстовой области.

```
<BR><TEXTAREA NAME="Тема" COLS="38" ROWS="3">
</TEXTAREA>
```

Использование списков в форме — тег <SELECT>

Когда формы HTML становятся более сложными, в них часто включают списки с прокруткой и выпадающие меню. Для этого используют тег SELECT с атрибутом TYPE=select. Для определения списка пунктов используют тег <OPTION>. Тег <SELECT> поддерживает три необязательных атрибута: MULTIPLE, NAME и SIZE.

Атрибут MULTIPLE

Позволяет выбрать более чем одно наименование.

Атрибут NAME

Определяет наименование объекта.

Атрибут SIZE

Определяет число видимых пользователю пунктов списка. Если в форме установлено значение атрибута SIZE=1, то браузер выводит на экран список в виде выпадающего меню. В случае SIZE > 1 браузер представляет на экране обычный список.

В форме может использоваться тег <OPTION> только внутри тега <SELECT>. Эти теги поддерживают два дополнительных атрибута: SELECTED и VALUE.

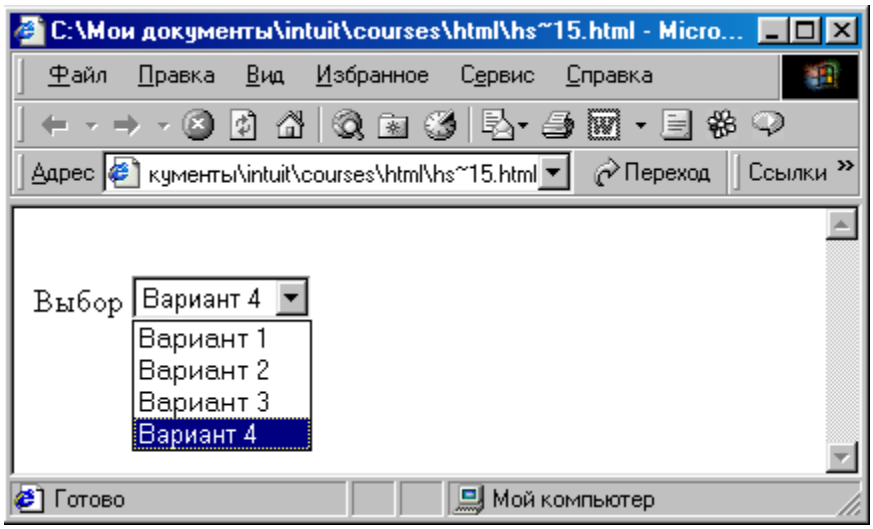
Атрибут SELECTED

Используется для первоначального выбора значения элемента по умолчанию.

Атрибут VALUE

Указывает на значение, возвращаемое формой после выбора пользователем данного пункта. По умолчанию значение поля равно значению тега <OPTION> (открыть).

```
<BR>Выбор
<SELECT NAME="Выбор">
<OPTION>Вариант 1
<OPTION>Вариант 2
<OPTION VALUE="Вариант 3">Вариант 3
<OPTION SELECTED>Вариант 4
</SELECT>
```



7. Лекция: Фреймы:
В этой лекции рассматриваются различные способы фрагментирования содержания Web-узла при помощи механизма HTML-фреймов.

В каком-то смысле фрейм — это именно то, что означает данное слово: рамка вокруг картинки, окошко или страница. Вводя тег <FRAME>, дизайнер HTML-страницы разделяет экран браузера на части. В результате человек, просматривающий страницу, может изучать только одну ее часть, независимо от остального содержимого. Фактически браузер, распознающий фреймы, загружает разные страницы в разные секции, или фреймы, экрана. Например, вы можете построить страницу таким образом, что фирменный знак будет зафиксирован в верхней части экрана, в то время как остальную часть страницы пользователь пролистывает обычным способом. Можно расположить сбоку кнопки навигации, которые не перемещаются, когда читатель щелкает по ним мышкой, так что изменяется только часть экрана, а сама полоска навигации остается неподвижной.

Как работают фреймы

На первый взгляд, фреймы — это нечто сложное, но их легче понять, если провести аналогию с ячейками таблицы. Расположение фреймов на экране и ячеек в таблице

задается почти одинаково: теги и атрибуты работают так же, как их табличные "родственники". Однако, хотя аналогия между единичным фреймом на странице и ячейкой таблицы верна, нужно помнить, что есть и отличия. Содержимое ячейки задано в коде HTML-страницы с таблицей. Текст или графика, составляющие содержимое таблицы, фактически вводятся на той же странице HTML, что и тег или атрибут, описывающие таблицу. Напротив, экран с фреймами описывается в HTML-странице, в контейнере FRAMESET. Содержимое же фрейма — это отдельная HTML-страница, которая может находиться где угодно: в другом каталоге, на локальном сервере или на удаленном узле где-то в сети. Фреймовая структура определяет только способ организации экрана с фреймами и указывает, где находится начальное содержимое каждого фрейма. Для всех фреймов задаются URL, описывающие местонахождение их данных. Как правило, на странице с фреймовой структурой содержимого фреймов нет. Такая страница обычно невелика — она описывает только кадровую структуру экрана. Когда документ загружается во фрейм, вы можете щелкать мышкой на ссылке в этом документе, чтобы увидеть связанные документы в других кадрах, заданных во фреймовой структуре.

Создание простой страницы с фреймами

Построим страницу с двумя фреймами. Зададим слева фрейм оглавления с заголовками статей, а справа поместим страницу с самими статьями. Сделаем так, что когда пользователь щелкает мышкой на ссылке в той части экрана, где находится оглавление, сама статья появляется в правом фрейме. Это основной, наиболее распространенный способ использования фреймов.

Задание фреймовой структуры

Для начала мы должны представить себе общий вид страницы – где расположить фреймы и какого они будут размера. Затем можно подумать об их содержании. Ниже приводится код простой фреймовой структуры с использованием тега <FRAMESET>. Обратите внимание: страница с фреймовой структурой не содержит тега <BODY>.

```
<HTML>
<HEAD>
<TITLE>Пример фреймов</TITLE>
</HEAD>
<FRAMESET COLS="25%, 75%">
<FRAME SRC="menu.html">
<FRAME SRC="main.html" NAME="main">
</FRAMESET>
</HTML>
```

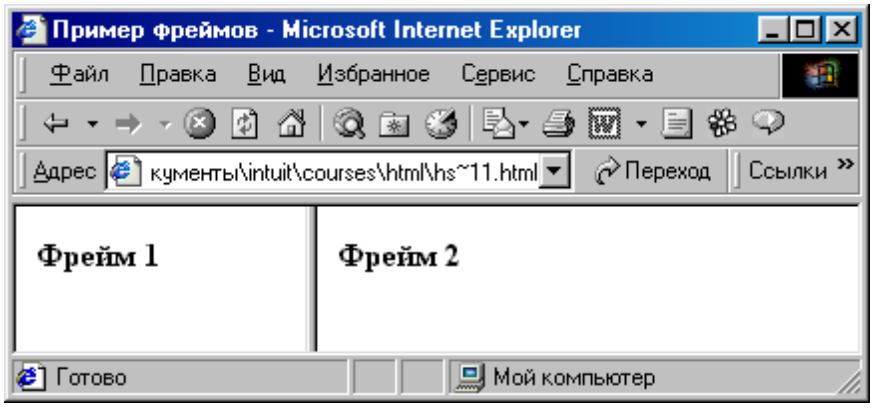


Рис. 7.1.

Вот и весь код, необходимый для того, чтобы задать фреймовую структуру. В результате мы получили экран, разделенный на два окна. Левое окно занимает 25%

экрана и содержит страницу с названием menu.html. Окно справа займет 75% экрана и содержит файл main.html. Пока у нас их нет, так что вы увидите страницу с двумя пустыми фреймами. Прежде чем она появится, нам придется пару раз щелкнуть мышкой в ответ на сообщения об ошибках, потому что браузер будет пытаться найти несуществующие страницы. Заметьте, что правую страницу мы назвали "main" (<главная>) с помощью строки:

```
<FRAME SRC="main.html" NAME="main">
```

Это означает, что фрейм под именем main будет содержать страницу main.html. Заметим, что поскольку мы не собираемся показывать в левом фрейме никаких страниц, кроме menu.html, нам не нужно его называть.

Подготовка содержимого фрейма

Теперь загрузим фреймы с содержимым. Зададим страницу menu.html в левом фрейме, где мы собираемся щелкать мышью, переключаясь между двумя страницами в правом фрейме. menu.html — это обычная HTML-страница, построенная как оглавление. На самом деле мы можем взять готовую страницу с оглавлением и использовать ее. Имейте в виду, что этот фрейм узкий и высокий, так что страница, которая будет в него загружаться, должна быть спроектирована соответствующим образом. Теперь мы должны определить, где будут появляться другие страницы при щелчке мышкой на ссылке. Поскольку мы хотим, чтобы они отображались в правом фрейме, добавим атрибут TARGET (TARGET="main") в тег ссылки. Это означает, что, когда пользователь щелкает на ссылке, вызываемая страница появляется в фрейме main. Мы отображаем все страницы в фрейме main, поэтому давайте добавим атрибут TARGET="main" во все теги ссылок в оглавлении. Если мы не определим атрибут TARGET, то страница появится там, где мы щелкнули мышкой, — в левом фрейме.

Подготовка фрейма main

Правый фрейм main будет содержать сами HTML-страницы. Ваша задача — спроектировать их так, чтобы они хорошо смотрелись в меньшем, чем обычно, окне, потому что часть экрана будет занята левым кадром оглавления. Но больше эти страницы ничем не примечательны.

Использование тега <NOFRAMES>

У некоторых пользователей еще остались браузеры, не умеющие обращаться с фреймами. По этой причине разумно предоставить доступ к версии ваших основных страниц без фреймов. Если читатель с устаревшим браузером окажется на вашей странице с фреймовой структурой, все, что находится на ней между тегами <NOFRAMES> и </NOFRAMES>, будет выглядеть отлично — браузер просто проигнорирует фреймы. Вот почему обязательно нужно использовать теги <BODY> </BODY>. Возможно, экран без фреймов придется организовать иначе.

Пример страницы с фреймовой структурой с добавленным в конце разделом <NOFRAMES>.

```
<HTML>
<HEAD>
<TITLE>Пример фреймов</TITLE>
</HEAD>
<FRAMESET COLS="25%, 75%">
<FRAME SRC="menu.html">
<FRAME SRC="main.html" NAME="main">
<NOFRAMES>
Вы просматриваете эту страницу с помощью
```

```
браузера, не поддерживающего фреймы.
</NOFRAMES>
</FRAMESET>
</HTML>
```

Имейте в виду, что браузер проигнорирует все, что находится между тегами <NOFRAMES> и </NOFRAMES>. Код без фреймов можно поместить и в начало, и в конец страницы.

Макетирование фреймов — тег <FRAMESET>

Теги <FRAMESET> обрамляют текст, описывающий компоновку фреймов. Здесь размещается информация о числе фреймов, их размерах и ориентации (горизонтальной или вертикальной). У тега <FRAMESET> только два возможных атрибута: ROWS, задающий число строк, и COLS, задающий число столбцов. Между тегами <FRAMESET> не требуется указывать тег <BODY>, но его можно поместить между тегами <NOFRAMES> в конце фреймовой структуры. Между тегами <FRAMESET> не должно быть никаких тегов или атрибутов, которые обычно используются между тегами <BODY>. Единственными тегами, которые могут находиться между тегами <FRAMESET> и </FRAMESET>, являются теги <FRAME>, <FRAMESET> и <NOFRAMES>. Это упрощает задачу. В основном все связано с тегами <FRAME> и их атрибутами. Если же вы хотите поэкспериментировать, можно создать вложенные друг в друга теги <FRAMESET> аналогично тегам <TABLE>.

Атрибуты ROWS и COLS

Для каждой строки и столбца, упомянутых в теге <FRAMESET>, необходим свой набор тегов <FRAME>.

Атрибут ROWS

Атрибут ROWS тега <FRAMESET> задает число и размер строк на странице. Количество тегов <FRAME> должно соответствовать указанному числу строк. Справа от знака "=" можно определить размер каждой строки в пикселах, процентах от высоты экрана или в относительных величинах (обычно это указание занять оставшуюся часть места). Следует пользоваться кавычками и запятыми, а также оставлять пробелы между значениями атрибутов. Например, следующая запись формирует экран, состоящий из трех строк: высота верхней — 20 пикселов, средней — 80 пикселов, нижней — 20 пикселов:

```
<FRAMESET ROWS="20, 80, 20">
```

Следующий тег — <FRAMESET> — создает экран, на котором верхняя строка занимает 10% высоты экрана, средняя — 60%, а нижняя — оставшиеся 30%:

```
<FRAMESET ROWS="10%, 60%, 30%">
```

Можно задать относительные значения в комбинации с фиксированными, выраженными в процентах или пикселах. Например, следующий тег создает экран, на котором верхняя строка имеет высоту 20 пикселов, средняя — 80 пикселов, а нижняя занимает все оставшееся место:

```
<FRAMESET ROWS="20, 80, *">
```

Атрибут COLS

Столбцы задаются так же, как строки. Для них применимы те же атрибуты.

Задание содержимого фрейма — элемент FRAME

Тег <FRAME> определяет внешний вид и поведение фрейма. Этот тег не имеет закрывающего тега, поскольку в нем ничего не содержится. Вся суть тега <FRAME> в его атрибутах. Их шесть: NAME, MARGINWIDTH, MARGINHEIGHT, SCROLLING, NORESIZE и SRC.

Атрибут NAME

Если вы хотите, чтобы при щелчке мышью на ссылке соответствующая страница отображалась в определенном фрейме, необходимо указать этот фрейм, чтобы страница "знала", что куда загружать. В предыдущих примерах мы назвали большой правый фрейм main, и именно в нем появлялись страницы, выбранные из оглавления в левом фрейме. Фрейм, в котором отображаются страницы, называется целевым (target). Фреймы, которые не являются целевыми, именовать не обязательно. Например, можно записать такую строку:

```
<FRAME SRC="my.html" NAME="main">
```

Имена целевых фреймов должны начинаться с буквы или цифры. Одни и те же имена разрешается использовать в нескольких фреймовых структурах. По щелчку мыши соответствующие страницы будут отображаться в именованном фрейме.

Атрибут MARGINWIDTH

Атрибут MARGINWIDTH действует аналогично атрибуту таблиц CELLPADDING. Он задает горизонтальный отступ между содержимым кадра и его границами. Наименьшее значение этого атрибута равно 1. Нельзя указать 0. Можно не присваивать ничего — по умолчанию атрибут равен 6.

Атрибут MARGINHEIGHT

Атрибут MARGINHEIGHT действует так же, как и MARGINWIDTH. Он задает поля в верхней и нижней частях фрейма.

Атрибут SCROLLING

Атрибут SCROLLING дает возможность пользоваться прокруткой во фрейме. Возможные варианты: SCROLLING=yes, SCROLLING=no, SCROLLING=auto. SCROLLING=yes означает, что во фрейме всегда будут полосы прокрутки, даже если это не нужно. Если задать SCROLLING=no, полос прокрутки не будет, даже когда это необходимо. Если документ слишком большой, а вы задали режим без прокрутки, документ просто будет обрезан. Атрибут SCROLLING=auto предоставляет браузеру самому решать, требуются полосы прокрутки или нет. Если атрибут SCROLLING отсутствует, результат будет таким же, как при использовании SCROLLING=auto.

Атрибут NORESIZE

Как правило, пользователь может, перемещая границу фрейма мышкой, изменить его размер. Это удобно, но не всегда. Иногда требуется атрибут NORESIZE. Помните: все границы фрейма, для которого вы задали NORESIZE, становятся неподвижными –

соответственно, может оказаться так, что размеры соседних фреймов тоже станут фиксированными. Пользуйтесь этим атрибутом с осторожностью.

Атрибут SRC

Атрибут SRC применяется в теге FRAME при разработке фреймовой структуры для того, чтобы определить, какая страница появится в том или ином кадре. Если вы зададите атрибут SRC не для всех фреймов, у вас возникнут проблемы. Даже если страницы, отображаемые во фрейме, выбираются в соседнем фрейме, вы должны по крайней мере задать для каждого фрейма начальную страницу. Если вы не укажете начальную страницу и URL, фрейм окажется пустым, а результаты могут быть самыми неожиданными.

Атрибут TARGET

Чтобы разобраться с атрибутом TARGET, необходимо вернуться к простому примеру с кадром оглавления. Когда пользователь щелкает мышкой на одной из ссылок в левом фрейме, соответствующая страница должна появиться в правом фрейме, а оглавление остается неизменным. Чтобы этого добиться, нужно определить целевой фрейм TARGET, в котором будет отображаться страница для каждого пункта оглавления. Целевые фреймы задаются в ссылках левого фрейма. Вот зачем всем кадрам во фреймовой структуре были присвоены имена. Правый фрейм называется main, так что нужно в каждой ссылке добавить атрибут TARGET="main", в результате чего соответствующая страница появится во фрейме main. Обратите внимание: каждая ссылка содержит атрибут TARGET="main", который по щелчку мыши отображает страницу во фрейме main.

Атрибут TARGET можно задавать для нескольких различных тегов. При использовании в теге <BASE> он направляет все ссылки в определенный целевой фрейм, если в дальнейшем не предусмотрено другое. Можно задать атрибут TARGET в теге <AREA> в активном изображении или в теге <FORM>. Фреймы полезны для организации форм. Пользователи будут видеть одновременно и форму, и результат своего выбора. Обычно при щелчке мышью кнопки Submit форма исчезает, и появляется страница с результатами выбора. Сочетание форм и фреймов может оказаться удобным способом навигации.

Вложенные и множественные кадровые структуры

Вложенные фреймы не очень способствуют навигации. И все же бывают случаи, когда возникает потребность разместить одни фреймы внутри других. Фреймы сами по себе — необычное средство навигации, и незачем еще более усложнять свои страницы. Но если вам все же нужны вложенные фреймы, то они не вызывают проблем.

В основном вложенные фреймы действуют так же, как вложенные таблицы. Задайте кадровую структуру, а внутри какого-нибудь фрейма в ней — еще одну структуру. Необходимо помнить, что тег <FRAME> не имеет закрывающего тега. Вы, наверное, заметили, что при работе с фреймами не используются атрибуты <COLSPAN> и <ROWSPAN>. Их роль играют множественные, или вложенные, фреймы. Задав внутри одной объемлющей фреймовой структуры две независимых **подструктуры**, можно поместить в левой части экрана столбец из двух, а в правой — из трех фреймов.

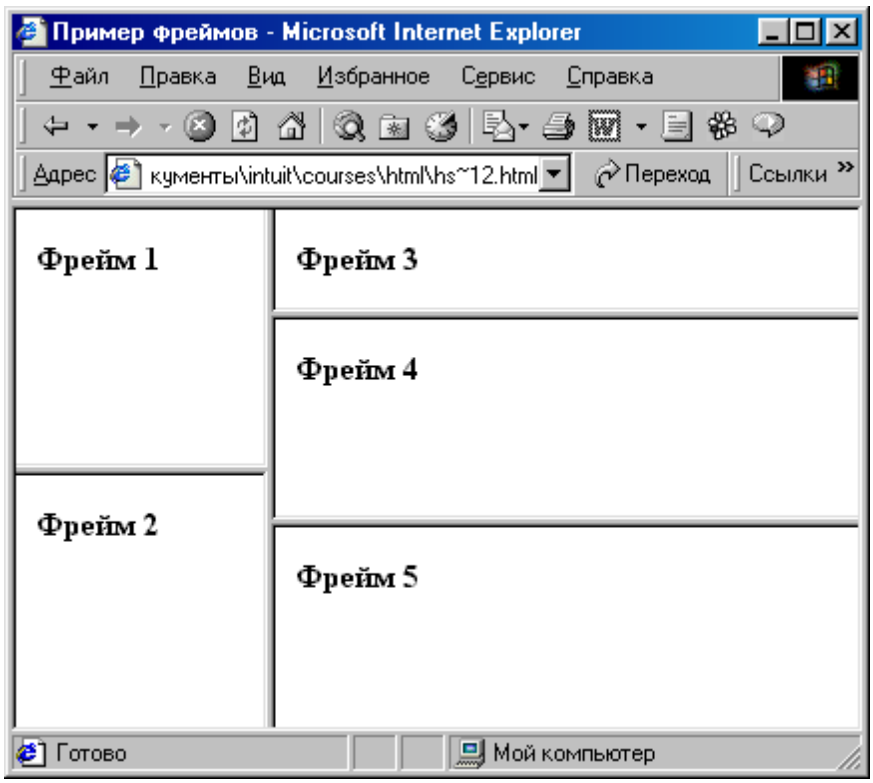

Пример фреймов - Microsoft Internet Explorer
Файл Правка Вид Избранное Сервис Справка
Адрес кументы\intuit\courses\html\hs~12.html
Переход
Ссылки
Фрейм 1
Фрейм 3
Фрейм 4
Фрейм 2
Фрейм 5
Готово
Мой компьютер

Рис. 7.2.