Сергей Велихов

Справочник по HTML 4.0

ББК 32.973
УДК 681.3

Сергей Велихов

В80 Справочник по HTML 4.0. Серия книг «Руководство по работе: Советы, хитрости, трюки и секреты» — М.: Бук-пресс, 2006. — 412 с.

ISBN 5-88548-039-7

Справочное пособие по новой версии языка гипертекстовых документов HTML 4.0. В книге вы познакомитесь с основами HTML 4.0 (определения, новые функции, фреймы, мультимедиа, таблицы, формы, сценарии, таблицы стилей, определения типов документов, тэги), узнаете как создавать документы в формате HTML 4.0 и представлять документ в формате HTML, поймете глобальную структуру документа в формате HTML, найдете информацию о языке и направлении текста и самоучитель для начинающих web-дизайнеров, научитесь различать HTML 3.2 и HTML 4.0, а также сможете воспользоваться советами по web-дизайну, которые помогут вам профессионально создавать web-сайты в Internet и Intranet.

Налоговая льгота «Общероссийский классификатор ОК 005-93-ТОМ2 953000 — Книги и брошюры».

Подписано в печать 20.09.2006. Формат 70x100/32. Бумага газетная. Гарнитура «Newton». Кол-во п.л. 13. Печать офсетная. Тираж 3000 экз.

Содержание

Глобальная структура документа в формате HTML

Информация о языке и направление текста

Списки

Таблицы стилей

Приложения

Введение в HTML 4.0

Что такое World Wide Web?

World Wide Web (Web) — это сеть информационных ресурсов. Для того, чтобы сделать эти ресурсы доступными наиболее широкой аудитории, в Web используются три механизма:

- Единая схема наименования для поиска ресурсов в Web (например, URI).
- Протоколы для доступа к именованным ресурсам через Web (например, HTTP).
- Гипертекст для простого перемещения по ресурсам (например, HTML).

Введение в URI

Каждый ресурс в Web — документ HTML, изображение, видеоклип, программа и т.д. имеет адрес, который может быть закодирован с помощью универсального идентификатора ресурсов (Universal Resource Identifier), или URI.

URI обычно состоят из трех частей:

- Схема наименования механизма, используемого для доступа к ресурсу.
- Имя машины, на которой располагается ресурс.
- Имя собственно ресурса, заданное в виде пути.

Большинство читателей уже знакомо с термином «URL», но не знает термина «URI». URL образуют подмножество более общей схемы наименования URI.

Идентификаторы фрагментов

Некоторые URI указывают на местоположение внутри ресурса. Этот тип URI заканчивается символом #, за которым следует указатель (идентификатор фрагмента). Например, следующий URI указывает на фрагмент с именем section_2:

```
http://somesite.com/html/top.html#section_2
```

Относительные URI

Относительный URI не содержит информации о схеме наименования. Путь в нем указывает на ресурс на машине, на которой находится текущий документ. Относительные URI могут содержать компоненты относительного пути (например,

«..» означает один уровень выше в иерархии) и идентификаторы фрагментов.

Относительные URI приводятся к полным URI с помощью базового URI. В качестве примера приведения относительного URI предположим, что у нас имеется базовый URI «http://www.acme.com/support/intro.html». Относительный URI в следующей ссылке:

```
<A href="suppliers.html">Suppliers</A>
```

будет преобразован в полный URI «http://www.acme.com/support/suppliers.html», а относительный URI в следующем фрагменте

```
<IMG src="../icons/logo.gif" alt="logo">
```

будет преобразован в полный URI «http://www.acme.com/icons/logo.gif».

В HTML URI используются для:

- ссылки на другие документы или ресурсы.
- ссылки на внешние таблицы стилей или скрипты.
- включения в страницу изображений, объектов или апплетов.
- создания изображений-карт.
- отправки форм.

- создания документов с использованием кадров.
- ссылок на внешние источники.
- ссылок на соглашения о метаданных, описывающих документ.

Что такое HTML?

Чтобы представить информацию для глобального использования, нужен универсальный язык, который понимали бы все компьютеры. Языком публикации, используемым в World Wide Web, является HTML (HyperText Markup Language — язык разметки гипертекстов).

HTML дает авторам средства для:

- публикации электронных документов с заголовками, текстом, таблицами, списками, фотографиями и т.д.
- загрузки электронной информации с помощью щелчка мыши на гипертекстовой ссылке.
- разработки форм для выполнения транзакций с удаленными службами, для использования в поиске информации, резервировании, заказе продуктов и т.д.

- включения электронных таблиц, видеоклипов, звуковых фрагментов и других приложений непосредственно в документы.

Краткая история HTML

Язык HTML был разработан Тимом Бернерс-Ли во время его работы в CERN и распространен браузером Mosaic, разработанным в NCSA. В 1990-х годах он добился особенных успехов благодаря быстрому росту Web. В это время HTML был расширен и дополнен. В Web очень важно использование одних и тех же соглашений HTML авторами Web-страниц и производителями. Это явилось причиной совместной работы над спецификациями языка HTML.

HTML 2.0 (ноябрь 1995) был разработан под эгидой Internet Engineering Task Force (IETF) для упорядочения общепринятых положений в конце 1994 года. HTML+ (1993) и HTML 3.0 (1995) — это более богатые версии языка HTML. Несмотря на то, что в обычных дискуссиях согласие никогда не было достигнуто, эти черновики привели к принятию ряда новых свойств. Усилия Рабочей группы World Wide Web Consortium

по HTML в упорядочении общепринятых положений в 1996 привели к версии HTML 3.2 (январь 1997).

Большинство людей признают, что документы HTML должны работать в различных браузерах и на разных платформах. Достижение совместимости снижает расходы авторов, поскольку они могут разрабатывать только одну версию документа. В противном случае возникает еще больший риск, что Web будет представлять собой смесь личных несовместимых форматов, что в конечном счете приведет к снижению коммерческого потенциала Web для всех участников.

В каждой версии HTML предпринималась попытка отразить все большее число соглашений между работниками и пользователями этой индустрии, чтобы усилия авторов не были потрачены впустую, а их документы не стали бы нечитаемыми в короткий срок.

Язык HTML разрабатывался с той точки зрения, что все типы устройств должны иметь возможность использовать информацию в Web: персональные компьютеры с графическими дисплеями с различным разрешением и числом цветов,

сотовые телефоны, переносные устройства, устройства для вывода и ввода речи, компьютеры с высокой и низкой частотой и т.д.

Новая версия HTML — 4.0

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

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

Именно так и должно быть, если учесть, что HTML — это «официально утвержденная

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

В 1996 г. W3C выпустил стандарт HTML 3.2. Основная цель принятия данной версии — объединить и официально утвердить значительное число фирменных тэгов Microsoft и Netscape, поскольку они уже получили широкое распространение. Но совершенно ясно, что подобная мера была временной, рассчитанной на то, чтобы пользователи могли пережить период ожидания кардинально новой версии HTML. Эта версия, в течение нескольких месяцев носившая кодовое название Cougar (Пума), отчасти предназначалась для обуздания процесса появления расширений HTML от отдельных фирм. Конечно, абсолютного успеха этой версии достичь не удастся, поскольку Microsoft и Netscape слишком

ревностно стремятся быть первыми и единственными игроками на этом поприще с правом на собственные нововведения в HTML. Однако удалось добиться по крайней мере возможности контролировать дальнейшее расширение этого основного авторского инструмента Web.

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

Расширяемый HTML

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

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

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

Чтобы решить эту проблему, фирма Netscape (тогда имевшая название Mosaic Communications Corp.) стала время от времени предлагать свои собственные HTML-элементы; впервые они вошли в версию 2.0 ее браузера (сначала он назывался Netscape). Благодаря его популярности, эти управляющие коды стали все чаще применяться в Web, и помешать этому процессу уже не могли сторонники чистоты стандарта HTML. В свою очередь, Microsoft добавила в готовящийся к выпуску браузер Internet Explorer собственные тэги, и вскоре

стали появляться Web-узлы, где были предусмотрены «ветвления» для пользователей Internet Explorer и Navigator. В худшем случае для обращения к узлам годился только один из этих двух браузеров.

Короче говоря, расширение возможностей HTML всегда сопряжено с трудностями. Для этого нужно сначала разработать сами элементы (что не так уж трудно), а затем подготовить соответствующие модули для браузера или внести в него изменения, с тем чтобы он надлежащим образом воспроизводил эти тэги (что значительно сложнее). Эффективного решения этой задачи удалось добиться только для браузеров Microsoft и Netscape, поскольку за последние несколько лет именно им было отдано практически повсеместное предпочтение пользователей. Но ни Microsoft, ни Netscape не отвечают за утверждение стандартов, поэтому де-факто им приходится сначала вводить свои расширения, а затем вносить предложение в W3C об их включении в официальную спецификацию HTML. Официально принятая версия HTML 3.2 по сути дала лучшие из уже существующих тэгов.

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

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

воспроизведение этого элемента он берет на себя.

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

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

и

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

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

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



```
height="600">Click here to view the sheet
music in a Java applet</OBJECT >
```

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

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

Microsoft и Netscape будут совершенствовать такую конфигурацию в последующих версиях браузеров.

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

Тем не менее важно соблюдать меру. По расширяемости HTML очень далек от XML, который по сути дела описывает языки разметки, или Document Type Definitions (DTDs) (описания типов документов).

Фактически весь HTML — это всего лишь одна спецификация DTD. Присущая XML возможность задавать новые DTD означает, что этот язык обеспечивает гораздо более широкое разнообразие параметров, чем когда бы то ни было будет в тэге **OBJECT** языка HTML. Однако тэг OBJECT — это только начало, и он, есть надежда, ослабит поток фирменных, рассчитанных на конкретный браузер тэгов.

Полнофункциональные экранные формы

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

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

Конечно, такие функции, как верификация ввода, имелись и в предыдущих версиях HTML, однако раньше для этого выполнялся обмен данными с сервером. Приходилось щелкать на кнопке **Submit** и ждать отклика с сервера с подтверждением

корректности введенной строки. В HTML 4.0 проверка ввода в экранные формы проводится локально, тем самым ускоряется работа с формами и возникает меньше проблем при обработке.

Новые табличные функции

Таблицы в предыдущих версиях HTML, как и формы, значительно уступали по своим возможностям таблицам из систем управления базами данных или электронным таблицам. HTML 4.0 отличается от своих предшественников гораздо более гибкими и полезными средствами для компоновки таблиц. Например, атрибут **align** предусматривает не только стандартные варианты выравнивания — по правой и левой границе, по центру и по ширине, но также при использовании признака **char** обеспечивает выравнивание текста по отношению к заданному символу. Скажем, с применением **align char** можно выровнять все числа по знаку десятичных точек. Атрибуты **frame** и **rules** предназначены для управления внешним видом таблиц. С помощью первого можно задать рамку вокруг указанных сторон таблицы (скажем, ее правой и левой сторонами); тогда как, употребив атрибут

rules, вы проведете линии между ячейками, строками или отдельными группами ячеек.

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

Сценарии для автоматизации

Новый контейнер **<SCRIPT>** языка HTML 4.0 обеспечивает возможность составлять сценарии. В нем указывается URL-адрес сценария, который будет выполняться, и сообщается, какой язык сценариев должен использовать браузер. Предусматривается два основных языка: JavaScript и VBScript. С помощью элемента **META** можно задать язык

по умолчанию для всех сценариев в данном документе или для конкретного сценария. Сценарии исполняются либо при загрузке документа, либо, если применяются атрибуты intrinsic event, всякий раз, когда происходит определенное событие (например, при щелчке мышью).

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

Встроенные кадры

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

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

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

Другие возможности

Кроме перечисленных средств, HTML 4.0 содержит множество других. Например, акцент был сделан и на

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

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

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

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

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

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

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

Доступность

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

- усилено разделение структуры и представления документа, что побуждает использовать таблицы стилей вместо элементов и атрибутов представления языка HTML.
- улучшены формы, включена возможность назначения клавиш доступа, возможность семантической группировки управляющих элементов

формы, семантической группировки вариантов в тэге SELECT и активные метки.

- добавлена возможность разметки текстового описания включенного объекта (с помощью элемента **OBJECT**).
- введен новый механизм действия изображений-карт на стороне клиента (элемент **MAP**), который позволяет авторам интегрировать изображения и текстовые ссылки.
- альтернативный текст для изображений, включаемых с помощью элемента **IMG**, обязателен.
- добавлена поддержка атрибутов **title** и **lang** во всех элементах.
- добавлена поддержка элементов **ABBR** и **ACRONYM**.
- более широкий диапазон целевых устройств (телетайп, шрифт Бройля и т.д.) для использования в таблицах стилей.
- улучшены таблицы, включена поддержка заголовков, групп столбцов и механизмов для упрощения

невизуального представления документа.

- добавлены длинные описания таблиц, изображений, кадров и т.д.

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

Таблицы

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

Составные документы

В HTML теперь имеется стандартный механизм для внедрения объектов и приложений в документы HTML. Элемент **OBJECT** (а также более специфичные элементы, его преемники, **IMG** и **APPLET**)

обеспечивает механизм включения в документ изображений, видеофайлов, звуковых файлов, математических выражений, специализированных приложений и других объектов. Он также позволяет авторам указывать иерархию или альтернативный способ создания изображения для браузеров, не поддерживающих указанный способ создания изображения.

Таблицы стилей

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

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

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

До появления таблиц стилей возможности управления созданием изображения у авторов были ограничены. В HTML 3.2 был включен

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

Скрипты

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

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

Печать

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

Основы HTML 4.0

Определения

Автор

Автор — это человек или программа, пишущая или генерирующая документы в формате HTML. Средство разработки — это отдельный случай автора, а именно программа, генерирующая код HTML.

Пользователь

Пользователь — это человек, взаимодействующий с агентом пользователя для просмотра, прослушивания или другого использования сгенерированного документа в формате HTML.

Новые функции

Всего несколько лет назад браузер Mosaic произвел сенсацию воспроизведением простых веб-страниц. Те дни давно канули в прошлое. Сегодня веб-страницы становятся все более сложными, как и применяемые для их создания технологии. Спецификация HTML 4.0 наглядно демонстрирует это: ее

изложение занимает несколько сотен листов текста.

Если попытаться выделить многочисленные новые функции и особенности последней версии HTML, эту информацию будет трудно переварить. Однако в действительности многие «новые» средства не такие уж и новые, хотя HTML, эволюционируя от версии 3.2 к версии 4.0, претерпел значительные изменения. Но такие браузеры, как Internet Explorer и Netscape Navigator, уже поддерживают ключевые средства новой версии HTML, а ряд других новых функций представляет собой упрощенный или усовершенствованный вариант существующих структур HTML, облегчающих работу с современными веб-технологиями.

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

Фреймы

Одним из самых значительных усовершенствований в последней версии HTML стало распознавание стандартных и встроенных фреймов. Конечно, фреймы не

представляют собой ничего нового. На самом деле, стандартные фреймы появились еще в Netscape Navigator 2.0. С тех пор в разных версиях браузеров предлагались различные усовершенствования. К счастью, для совместимости с HTML 4.0 браузеры должны будут поддерживать фреймы, определенные в данной спецификации.

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

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

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

```
<iframe src="samples.html" width="325"
height="500" frameborder="0">
<P>Normally, a product sample is
displayed in this space. However, your
browser doesn't support HTML 4.0.</P>
</iframe>
```

Совместимые с HTML 4.0 браузеры будут игнорировать подобный текст, а несовместимые выводят его, игнорируя тэг **iframe**. И еще одно замечание по встроенным фреймам: присваивая фрейму имя, можно дать ссылки, которые выводят в этом фрейме

другие файлы. Такие гиперссылки для фрейма называются целевыми. Цель указывается атрибутом фрейма **name**. Имя же присваивается как значение атрибута **target** с соответствующей гиперссылкой.

Указание цели во встроенных фреймах с помощью значения атрибута name

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

```
M3500 Statistical Plus</A></P>
</DIV>
</BODY>
</HTML>
```

Мультимедиа

Еще один важный шаг вперед — элемент **object**, предусматривающий тэг общего назначения для работы с различными типами информации, такими, как Java-апплеты, встроенное видео, потоковые звук и изображение. В конечном счете он должен заменить тэги для разных типов информации мультимедиа. Элемент **object** можно будет использовать вместо тэгов **applet**, **img** и **embed**. Его удобно применять даже для вывода на экран в окне браузера обычных текстовых файлов. В этом случае он во многом напоминает встроенный фрейм.

Первоначально элемент **object** был предложен в качестве компромиссного варианта вместо тэгов, предназначенных для конкретных целей, таких, как **img** и **applet**. К сожалению, хотя разработчики браузеров видели необходимость стандартного способа включения мультимедиа в веб-страницы, они не всегда были единодушны в том, каким способом это нужно делать. Кроме того, в

течение какого-то времени тэг **object** не являлся частью официальной спецификации HTML. Наконец, в HTML 4.0 элемент **object** все же принят как решение для будущей реализации в веб-страницах объектов мультимедиа.

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

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

Таблицы

Большая часть нововведений в данной области, появившихся в HTML 3.0, сохранилась в HTML 4.0, но появились и дополнительные средства. Стало возможным

группировать столбцы и определять (в начале таблицы) их свойства. Кроме того, допускается определение заголовка, нижнего колонтитула и содержимого разделов таблицы.

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

Использовать такие элементы в таблице проще, чем кажется. Они поддерживают идентичный набор атрибутов, предоставляя достаточную свободу в определении структур группы. Атрибутам **width**, **cellhalign** и **cellvalign** присваивается заданная по умолчанию ширина, выравнивание ячейки по горизонтали и вертикали соответственно, а **span** задает число столбцов, совместно использующих атрибуты элемента. Чтобы сделать таблицы более динамичными, можно

применять атрибуты таблицы стилей и встроенные события.

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

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

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

Использование элементов COLGROUP и COL

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



```
//непосредственно в группах колонок
<COLGROUP span="3" width="100">
</COLGROUP>
<COLGROUP span="3" width="75">
</COLGROUP></xmp>
Применение заголовков и нижних
колонтитулов помогает браузеру
структурировать таблицу</P>
<xmp>
<TABLE BORDER=2 WIDTH=50%>
<COLGROUP>
<COL width="100">
<COL width="75">
<COL width="75">
<COL width="75">
</COLGROUP>
<THEAD>
<TR> <TH> </TH> <TH>1996</TH>
<TH>1997</TH> <TH>1998</TH> </TR>
</THEAD>
<TFOOT>
<TR><TD>Данные предоставлены компанией
только для статистических целей.</TD></TR>
</TFOOT>
<TBODY>
<TR> <TH>Неделя 1</TH> <TD>252</TD>
<TD>267</TD> <TD>289</TD> </TR>
<TR> <TH>Неделя 2</TH> <TD>194</TD>
<TD>197</TD> <TD>205</TD> </TR>
```

```
<TR> <TH>Неделя 3</TH> <TD>212</TD>
<TD>225</TD> <TD>234</TD> </TR>
<TR> <TH>Неделя 4</TH> <TD>145</TD>
<TD>176</TD> <TD>179</TD> </TR>
<TR> <TH>Неделя 5</TH> <TD>167</TD>
<TD>182</TD> <TD>193</TD> </TR>
<TR> <TH>Неделя 6</TH> <TD>185</TD>
<TD>201</TD> <TD>205</TD> </TR>
<TR> <TH>Неделя 7</TH> <TD>197</TD>
<TD>207</TD> <TD>213</TD> </TR>
. . .
<TR> <TH>Неделя 52</TH> <TD>203</TD>
<TD>221</TD> <TD>279</TD> </TR>
</TBODY>
</TABLE>
```

Формы

Если вы работали с предыдущими версиями HTML, то знаете, что формы с момента их появления в этом языке изменились незначительно, их нельзя было назвать «дружественными пользователю», и они уже давно нуждались в усовершенствовании. На этот раз спецификация HTML устраняет, наконец, их наиболее очевидные недостатки.

Теперь формы имеют немало новых элементов, но давайте сосредоточимся на

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

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

Когда пользователь нажимает на клавишу **Tab**, он попадает сначала в поле **userName**, затем в поле **userEmail** и т.д. Порядок табуляции может включать в себя любое число от 0 до 32767. Браузеры используют числовое значение для определения

следующего или предыдущего поля. Некоторым полям можно также присваивать и клавиши доступа. Клавиши доступа позволяют быстро перемещаться на определенные поля в форме. Например, если полю **userName** в предыдущем примере присвоена клавиша **N**, то, нажав ее, можно сразу оказаться в этом поле. Работа клавиш доступа зависит от операционной системы. В системах Microsoft Windows, кроме клавиши доступа, обычно нужно нажимать на клавишу **Alt**. Таким образом, чтобы попасть в поле **userName** в системе Windows, нужно нажать **Alt-N**. Браузеры должны выводить клавишу доступа на экран в поле ввода, причем

каким-то уникальным способом, например, подчеркнутым или жирным шрифтом.

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

атрибута **id** в этом поле. В следующем примере показаны метки с текстовыми полями ввода:

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

<input accesskey="R" type="reset"></P>
</form>
```

Еще одно полезное усовершенствование: элемент **button**. Он создает нажимаемую экранную кнопку, аналогичную кнопкам **Reset** и **Submit** в формах. Между тем, к этим новым кнопкам можно добавить содержимое (чего стандартные экранные кнопки не допускали). Таким образом, новые кнопки теперь могут содержать изображения, текст и другие навороты. Чтобы добавить их, достаточно вставить «наполнение» между открывающим тэгом **<button>** и

закрывающим тэгом **</button>**. Воспользовавшись сказанным, в предыдущем примере стандартные кнопки **Submit** и **Reset** можно заменить элементом **button**. Добавление текста и изображений позволит сделать эти элементы более привлекательными:

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

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

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


Сценарии

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

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

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

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

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

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

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

Еще одно важное изменение, касающееся сценариев, состоит в том, что большинство элементов HTML теперь поддерживают широкий спектр атрибутов событий. Событие происходит автоматически при определенном условии. Некоторые события вызываются

действием пользователя (например, нажатием клавиши или щелчком кнопкой мыши), другие — браузерами (например, когда браузер завершает загрузку изображения). Атрибуты событий позволяют задать условия, при которых должен обнаруживаться элемент. Например, можно создать в таблице ячейки, активизируемые щелчком мыши, абзацы текста, выделяемые при перемещении на них курсора, и многое другое.

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

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

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

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

Таблицы стилей

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

К сожалению, когда информация о представлении документа комбинируется с его содержательными материалами, получается излишне сложный и трудно обслуживаемый документ. Данное важное нововведение направлено на постепенное сокращение презентационных элементов и атрибутов в HTML-документах, поскольку таблицы стилей позволяют отделить представление документа от его фактического содержимого.

Элементы и атрибуты, которые могут исчезнуть из спецификации HTML, помечаются как выходящие из употребления. Один из таких элементов — **font**, применяемый для указания цвета текста, его гарнитуры и размера шрифта. В числе других **basefont** (для задания информации о назначаемом по умолчанию шрифте), **center** (центрирующий объекты на странице) и несколько других элементов, таких, как **u** (подчеркивание) и **s** (перечеркивание текста).

Кроме включения элементов в список выходящих из употребления, некоторые элементы в данной спецификации объявлены устаревшими, а другие — не рекомендуемыми к использованию. Устаревшие элементы отсутствуют в спецификации HTML, и нет никакой гарантии, что браузер будет их

поддерживать. Хотя ясно, что устаревшие элементы использовать не следует, не вполне понятно, что такое не рекомендуемые элементы, отсутствующие в списке устаревших и выходящих из употребления элементов. Спецификация просто не поощряет их применение.

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

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

информации, то, вероятно, лучше воспользоваться вместо него таблицей стилей.

Определения типов документов

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

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

Основная цель строгого определения DTD состоит в том, чтобы отделить визуальное оформление от фактического контента. Это делается с помощью таблиц стилей, управляющих представлением веб-страниц. Строгое определение DTD не включает в себя выходящих из употребления

элементов/атрибутов или структур, применяемых для создания фреймов. Таким образом, это наиболее ограничивающее определение DTD. Если ваш браузер совместим с HTML 4.0, и нужно протестировать веб-страницу со строгим DTD, воспользуйтесь следующим описанием типа:

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

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

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

Фреймовое определение DTD предназначено для веб-страниц с фреймами. Эта версия поддерживает все структуры, допустимые в свободном определении DTD, а также структуры, необходимые для

фреймов. Для спецификации такого DTD используйте описание:

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

Тэги

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

Атрибуты:

HREF

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

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

просмотра этот способ работает только внутри одного документа.

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

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

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

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

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

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

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


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

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

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

NAME

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

TITLE

Используется только для информационных целей (за исключением совместного использования с **mailto**). Значение данного

атрибута будет появляться как всплывающая подсказка, когда пользователь будет задерживать указатель мыши над данным элементом. Работает только в IE.

URN

В настоящее время не используется.

TARGET

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

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

- **Window_name** — название окна или фрейма, заданное явно при его создании. Если окна или фрейма с таким названием не существует ссылка будет открыта в новом окне и ему будет присвоено это название, по

которому к нему можно будет обращаться в дальнейшем.

- **_self** — открывает ссылку в том же окне
- **_parent** — открывает ссылку в родительском фрейме
- **_top** — открывает ссылку в самом «верхнем» окне.
- **_blank** — открывает ссылку в новом окне, при этом этому окну не присваивается имени как в первом случае.

ACCESSKEY

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

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

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

LANGUAGE

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

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

и запись

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

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

LANG

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

CLASS

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

ID

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

STYLE

Применяется в случае использовании встроенных стилей, например:

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

Другие атрибуты

REL

REV

METHOD

DATAFLD

DATASRC

Тэг <A>, использование в сценариях

Свойства тэга <A>

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

- className

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

Кроме них поддерживается еще ряд свойств, присущих только объекту **<A**

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

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

- **accessKey** — (Internet Explorer 4.0+)
- **datafld** — (Internet Explorer 4.0+)
- **datasrc** — (Internet Explorer 4.0+)
- **hash** — (Internet Explorer 3.0+, Netscape 2.0+)
- **host** — (Internet Explorer 3.0+, Netscape 2.0+)
- **hostname** — (Internet Explorer 3.0+, Netscape 2.0+)
- **href** — (Internet Explorer 3.0+, Netscape 2.0+)
- **Methods** — (Internet Explorer 4.0+)
- **name** — (Internet Explorer 3.0+, Netscape 2.0+)

- **pathname** — (Internet Explorer 3.0+, Netscape 2.0+)
- **port** — (Internet Explorer 3.0+, Netscape 2.0+)
- **protocol** — (Internet Explorer 3.0+, Netscape 2.0+)
- **rel** — (Internet Explorer 4.0+)
- **rev** — (Internet Explorer 4.0+)
- **search** — (Internet Explorer 3.0+, Netscape 2.0+)
- **target** — (Internet Explorer 3.0+, Netscape 2.0+)
- **urn** — (Internet Explorer 4.0+)

Методы тэга <A>

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

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

События тэга <A>

Помимо стандартных событий DHTML, таких как **onclick**, **ondblclick**, **ondragstart**,

onfilterchange, **onhelp**, **onkeydown**, **onkeypress**, **onkeyup**, **onmousedown**, **onmousemove**, **onmouseout**, **onmouseover**, **onmouseup**, **onselectstart**, объект **<A>** поддерживает следующие события:

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

Создание документов в формате HTML 4.0

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

Разделение структуры и представления

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

платформ, носителей и т.д. и упрощает изменение документов.

Универсальность доступа к Web

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

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

Помощь в последовательном создании изображений

При тщательной разработке таблиц и использовании новых возможностей HTML

4.0 авторы могут ускорить отображение документов браузером. Авторы могут прочесть здесь о том, как создавать таблицы для последовательного представления (элемент **TABLE**). Разработчики могут получить информацию об алгоритмах последовательного представления в замечаниях о таблицах в приложении.

SGML и HTML

Введение в SGML

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

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

Документ HTML состоит из *раздела заголовка* (здесь — между тэгами **<HEAD>** и

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

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

Объявлением SGML

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

Определением типа документа (DTD)

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

Спецификацией, описывающей семантику, используемую в разметке

Эта спецификация также налагает синтаксические ограничения, которые невозможно выразить при помощи DTD.

Экземпляры документа содержат данные (содержимое) и разметку. Каждый экземпляр

содержит ссылку на DTD, которое должно использоваться для интерпретации.

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

Конструкции SGML, используемые в HTML

Элементы

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

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

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


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

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

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

Имена элементов всегда учитывают регистр.

Информацию о правилах, управляющих элементами, (например, что они могут быть вложенными соответствующим образом,

конечный тэг закрывает все опущенные начальные тэги вплоть до соответствующего ему начального тэга находятся в стандарте SGML.

Например, следующий абзац:

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

можно перезаписать без конечного тэга:

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

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

```
<DIV>
<P>Это абзац.
</DIV>
```

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

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

Атрибуты

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

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

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

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

кавычек авторы могут также использовать **character entity reference "**.

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

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

Ссылки на символы

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

```
"&lt;"
```

представляет знак <.

```
"&gt;"
```

представляет знак >.


```
"&quot;"
```

представляет знак ".

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

(десятичное число) представляет букву @.

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

(десятичное число) представляет кириллическую букву «I».

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

(шестнадцатеричное число) представляет китайский знак воды.

Комментарии

Комментарии в HTML имеют следующий синтаксис:

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

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

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

Как читать HTML DTD

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

Комментарии DTD

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

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

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

Определения Parameter entity

HTML DTD начинается с ряда определений **parameter entity**. Определение **parameter entity** определяет макрос, на

который можно ссылаться в любом месте DTD. Эти макросы не отображаются в документах HTML, они отображаются только в DTD. Другие типы макросов, называемые ссылками на символы, могут использоваться в тексте документа в формате HTML или в значениях атрибутов.

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

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

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

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

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

«**%fontstyle;**», «**%phrase;**», «**%special;**» и «**%formctrl;**» **parameter entities**.

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

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

Объявления элементов

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

Имя элемента

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

Содержимое элемента

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

В этом примере:

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

объявляется тип элемента UL.

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

Модель содержимого для типа элемента

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

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

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

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

Определения модели содержимого

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

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

Модель содержимого элемента указывается с использованием следующего синтаксиса:

```
( ... )
```

Разделяет группы.


```
A | B
```

Происходит A или B, но не оба.

```
A , B
```

Происходят A и B в указанном порядке.

```
A & B
```

Происходят A и B в любом порядке.

```
A?
```

A происходит ноль или один раз.

```
A*
```

A происходит ноль или более раз.

```
A+
```

A происходит один или несколько раз.

Вот некоторые примеры HTML DTD:

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

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

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

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

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

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

Некоторые типы элементов HTML используют дополнительную функцию SGML

для исключения элементов из модели содержимого. Исключенным элементам предшествует символ переноса. Явные исключения имеют приоритет по отношению к допустимым элементам.

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

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

Помните, что тип элемента **A** является частью DTD **parameter entity «%inline;»**, но явно исключается выражением **-(A)**.

Точно так же следующее объявление типа элемента **FORM** запрещает вложенные формы:

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

Объявления атрибутов

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

Имя атрибута

Тип значения атрибута или явный набор допустимых значений. Значения, определенные явным образом при помощи DTD, учитывают регистр.

Является ли значение атрибута по умолчанию неявным (ключевое слово «**#IMPLIED**»), то есть значение по умолчанию устанавливается браузером (в некоторых случаях с использованием наследования от родительских элементов); всегда обязательным (ключевое слово «**#REQUIRED**»); или фиксированным данным значением (ключевое слово «#FIXED»). Некоторые определения атрибутов явным образом указывают значение атрибута по умолчанию.

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

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

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

В следующих примерах показано несколько определений атрибутов:

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

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

DTD entities в определениях атрибутов

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

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

```
<!ELEMENT LINK - O EMPTY
-- a media-independent link -->
<!ATTLIST LINK
```



```
   %attrs;
-- %coreattrs, %i18n, %events --
   charset        %Charset;          #IMPLIED
-- char encoding of linked resource --
   href             %URI;
#IMPLIED   -- URI for linked resource --
   hreflang       %LanguageCode; #IMPLIED
-- language code --
   type             %ContentType;    #IMPLIED
-- advisory content type --
   rel              %LinkTypes;
#IMPLIED   -- forward link types --
   rev              %LinkTypes;
#IMPLIED   -- reverse link types --
   media           %MediaDesc;       #IMPLIED
-- for rendering on these media --
   >

Start tag: required, End tag: forbidden
```

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

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

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

```
<!ENTITY % coreattrs
  "id               ID
#IMPLIED   -- document-wide unique id --
```

```
   class          CDATA
#IMPLIED   -- space separated list of
classes --
   style          %StyleSheet;     #IMPLIED
-- associated style info --
   title          %Text;
#IMPLIED   -- advisory title/amplification
--"
   >
```

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

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

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

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

Boolean attributes

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

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

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

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

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

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

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

```
<OPTION selected>
```

вместо:

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

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

Представление документа в формате HTML

Набор символов документа

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

Документ включает:

Репертуар

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

Коды

Набор целочисленных ссылок на символы репертуара.

Каждый документ SGML (включая каждый документ HTML) — это последовательность символов из репертуара. Компьютерные системы идентифицируют каждый символ по его коду; например, в

наборе символов ASCII коды 65, 66 и 67 означают символы «A», «B» и «C» соответственно.

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

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

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

использовались для преобразования потока символов документа в поток байт.

Кодировки символов

Кодировки символов в ряде спецификаций имеют другие названия (что может вызвать некоторую путаницу). Однако это понятие в Интернет означает примерно одно и то же. Одно и то же имя — «**charset** — набор символов» — используется в заголовках протоколов, атрибутах и параметрах, ссылающихся на символы и использующих одни и те же значения.

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

Простой техники кодировки «один байт — один символ» недостаточно для текстовых строк с широким репертуаром

символов. Кроме кодировок всего набора символов (например, UCS-4), имеются некоторые другие кодировки частей.

Выбор кодировки

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

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

Широко используемые в Web кодировки:

- ISO-8859-1 (также называется «Latin-1»; используется для большинства западноевропейских языков)
- ISO-8859-5 (с поддержкой кириллицы)
- SHIFT_JIS (японская кодировка)
- EUC-JP (еще одна японская кодировка)
- UTF-8 (вариант кодировки ISO 10646, использующий разное число байт для разных символов).

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

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

Замечания об определенных кодировках

Когда текст HTML передается в UTF-16 (charset=UTF-16), текстовые данные должны передаваться в сетевом порядке байт («big-endian», байт высшего

порядка — первый) в соответствии с ISO10646 и UNICODE.

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

Не следует использовать формат трансформации UTF-1

Указание кодировки символов

Как сервер определяет, какая кодировка символов применяется в документе? Некоторые серверы проверяют первые несколько байт документа или сверяются с базой данных известных файлов и кодировок. Многие современные серверы Web предоставляют администраторам больше возможностей управления конфигурацией набора символов, чем старые серверы.

Администраторы серверов Web должны при возможности использовать следующие механизмы для отправки параметра «**charset**», но должны позаботиться о том, чтобы не установить для документов ошибочное значение параметра «**charset**».

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

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

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

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

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

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

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

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

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

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

Кроме этого списка приоритетов, браузер может использовать эвристические установки и установки пользователя. Например, многие браузеры используют эвристику для распознавания различных кодировок, используемых для японского языка. Браузеры обычно имеют определяемую пользователем локальную кодировку по умолчанию, которую они используют, если нет указаний кодировки.

Браузеры могут обеспечивать механизм, позволяющий пользователям изменять некорректную информацию о наборе символов. Однако если браузер предлагает

такой механизм, он должен предлагать его только для просмотра, а не для изменения, во избежание создания Web-страниц с некорректным параметром «**charset**».

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

Ссылки на символы

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

Ссылки на символы в HTML могут принимать две формы:

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

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

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

В SGML можно в некоторых случаях не использовать заключительный символ «**;**» после ссылки на символы (например, в символе переноса строки или непосредственно перед тэгом). В других обстоятельствах их нельзя удалять (например, в середине слова). Мы предлагаем использовать «**;**» всегда во избежание проблем с браузерами, для которых этот символ обязателен.

Числовые ссылки на символы

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

- Синтаксис «**&#D;**», где **D** — десятичное число, указывает символ Unicode с десятичным номером D.

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

Вот некоторые примеры числовых ссылок на символы:

```
&#229;
```

(десятичное) представляет букву «а» с кружком сверху (используемую, например, в норвежском языке).

```
&#xE5;
```

(шестнадцатеричное) представляет тот же символ.

```
&#Xe5;
```

(шестнадцатеричное) представляет тот же символ.

```
&#1048;
```

(десятичное) представляет кириллическую заглавную букву «I».

```
&#x6C34;
```

(шестнадцатеричное) представляет китайский иероглиф «вода».

Комбинации ссылок на символы

Чтобы дать авторам более инициативный способ использования символов, HTML

предлагает набор **character entity references**. Комбинации ссылок на символы используют символические имена, так что авторам не придется запоминать коды. Например, комбинация **å** обозначает символ «а» нижнего регистра с кружком сверху; **å** легче запомнить, чем **å**.

HTML 4.0 не определяет **character entity reference** для каждого символа. Например, для кириллической буквы «**I**» нет **character entity reference**.

Комбинации ссылок на символы учитывают регистр. Так, **Å** указывает на другой символ (А с кружком верхнего регистра), а не на **å** (а с кружком нижнего регистра).

Четыре ссылки нужно упомянуть специально, поскольку они часто используются для указания специальных символов:

```
&lt;
```

представляет знак <.

```
&gt;
```

представляет знак >.

```
&amp;
```

представляет символ &.

"

представляет знак ".

Авторы, которые хотят поместить в текст символ <, должны использовать ссылку **<** (десятичный код ASCII 60) во избежание возможной путаницы с началом тэга (открывающий разделитель начального тэга). Точно так же следует использовать **>** (десятичный код ASCII 62) вместо >, чтобы избежать проблем со старыми версиями браузеров, некорректно принимающих их за окончание тэга (закрывающий разделитель тэга).

Авторам следует использовать **&** (десятичный код ASCII 38) вместо «**&**» во избежание путаницы со ссылками на символы (открывающий разделитель **entity reference**). Авторам также следует использовать **&** в значениях атрибутов, поскольку ссылки на символы внутри значений атрибута **CDATA** разрешены.

Некоторые авторы используют **character entity reference** «**"**» для кодирования экземпляров двойных кавычек (**"**), поскольку этот символ может использоваться для разделения значений атрибутов.

Неотображаемые символы

Возможно, браузер не сможет отобразить все символы в документе, например, из-за отсутствия соответствующего шрифта или если символ имеет значение, которое не может быть выражено во внутренней кодировке браузера и т.д.

Поскольку в этом случае есть несколько вариантов, этот документ не предписывает определенной тактики. В зависимости от применения непечатные символы могут также обрабатываться дополнительной системой отображения, а не самим приложением. В случае более сложного поведения, например, настроенного для определенного сценария или языка, рекомендуем следующее поведение для браузеров:

- Примите явно видимый, но незаметный механизм для предупреждения пользователя об отсутствующих ресурсах.
- Если отсутствующие символы представляются в другом числовом представлении, используйте шестнадцатеричную (не десятичную) форму, поскольку эта форма используется в стандартах наборов символов.

Основные типы данных HTML

Информация о регистре

Каждое определение атрибута включает информацию об учете регистра его значениями. Информация о регистре представляется следующими ключами:

CS

Значение учитывает регистр (то есть браузеры по-разному интерпретируют «a» и «A»).

CI

Значение не учитывает регистр (то есть браузеры одинаково интерпретируют «a» и «A»).

CN

Значение не зависит от регистра, например, потому что это число или символ из набора символов документа.

CA

Само определение элемента или атрибута дает информацию о регистре.

Если значением атрибута является список, ключи применяются к каждому значению в списке, если не указано обратное.

Основные типы SGML

В определении типа документа определяется синтаксис содержимого элемента HTML и значений атрибутов с использованием меток SGML (например, **PCDATA**, **CDATA**, **NAME**, **ID** и т.д.).

Вот обобщенная информация о ключах:

CDATA — это последовательность символов из набора символов документа, она может включать **character entities**. Браузеры должны интерпретировать значения атрибутов следующим образом:

- Заменять **character entities** на символы;
- Игнорировать перевод строки;
- Заменять каждый возврат каретки или табуляцию на один пробел.

Браузеры могут игнорировать пробелы в начале и в конце значений атрибута **CDATA** (например, « myval » интерпретируется как «myval»). Авторы не должны объявлять значения атрибутов с пробелами в начала или в конце.

На некоторые атрибутов HTML 4.0 со значениями атрибутов **CDATA** спецификация налагает дополнительные ограничения на множество допустимых значений атрибутов, не выраженные в DTD.

Хотя элементы **STYLE** и **SCRIPT** используют **CDATA** для своей модели данных, для этих элементов браузеры должны обрабатывать **CDATA** по-другому. Разметка и **entities** должны считаться текстом и передаваться в приложение как есть. Первое вхождение последовательности символов «</» (открывающий разделитель конечного тэга) считается концом содержимого элемента. В допустимых документах это будет конечный тэг элемента.

Метки **ID** и **NAME** должны начинаться с буквы (A-Z, a-z), за которой может следовать любое число букв, цифр (0-9), символов переноса (-), символов подчеркивания (_), двоеточий (:) и точек (.).

IDREF и **IDREFS** — это ссылки на метки **ID**, определенные другими атрибутами. **IDREF** — одиночная метка, а **IDREFS** — разделенный пробелами список меток.

Метки **NUMBER** должны содержать по крайней мере одну цифру (0-9).

Текстовые строки

Ряд атрибутов (**%Text;** в DTD) принимают текст, который предназначается для чтения людьми.

URI

URI включают URL. Относительные URI разрешаются до полных URI с использованием основного URI. URI представляются в DTD комбинацией символов **%URI;**.

URI вообще учитывают регистр. Могут быть URI, или части URI, в которых регистр не имеет значения (например, имена машин), но идентификация их может быть непроста. Пользователи должны всегда считать, что URI учитывают регистр (чтобы не ошибиться).

Цвета

Значение атрибута типа «**color**» (**%Color;**) относится к определениям цветов. Значение цвета может быть шестнадцатеричным числом (которому предшествует знак диеза) или одним из следующих шестнадцати

названий цветов. Названия цветов учитывают регистр.

Названия цветов и значения RGB

Black = #000000

Green = #008000

Silver = #C0C0C0

Lime = #00FF00

Gray = #808080

Olive = #808000

White = #FFFFFF

Yellow = #FFFF00

Maroon = #800000

Navy = #000080

Red = #FF0000

Blue = #0000FF

Purple = #800080

Teal = #008080

Fuchsia = #FF00FF

Aqua = #00FFFF

То есть, значения **#800080** и «**Purple**» оба означают пурпурный цвет.

Замечания об использовании цветов

Хотя цвета могут существенно добавлять информации в документ и повышать удобство чтения, при использовании цветов имейте в виду следующие основные принципы:

- Использование элементов и атрибутов HTML для указания цвета нежелательно. Вместо этого следует использовать таблицы стилей.
- Не используйте комбинации цветов, вызывающие проблемы у пользователей.
- Если вы используете изображение в качестве фона или устанавливаете цвет фона, не забудьте установить и цвета текста.

Цвета, указанные в элементах **BODY** и **FONT** и в **bgcolor** в таблицах выгладят по-разному на разных платформах (на рабочих станциях, Mac, Windows и на панелях LCD и CRT), поэтому не рассчитывайте на определенный эффект. В будущем поддержка цветовой модели вместе с цветовыми профилями ICC должна устранить эти проблемы.

При возможности принимайте общие соглашения.

Длины

HTML определяет три типа значений длины для атрибутов:

Пикселы

Значение (**%Pixels;** в DTD) — это целое, представляющее число пикселов (на экране, на бумаге). Таким образом, значение «50» означает пятьдесят пикселов.

Длина

Значение (**%Length;** в DTD) может быть **%Pixel;** или доля вертикального или горизонтального расстояния в процентах. Таким образом, значение «**50%**» означает половину доступного пространства.

МультиДлина

Значение (**%MultiLength;** в DTD) может быть **%Length;** или относительной длиной. Относительная длина имеет форму «**i***», где «**i**» — целое число. При распределении пространства между элементами, конкурирующими за это пространство, браузеры сначала отводят место для длин, определенных в пикселах и процентах, а затем делят оставшееся место между относительными длинами. Каждая относительная длина получает часть доступного пространства, пропорциональную

целому числу, предшествующему «*». Значение «*» эквивалентно «**1***». Таким образом, если имеется 60 пикселов пространства после того, как браузер распределит пространство для длин, определенных в пикселах и процентах, а конкурирующими относительными длинами являются **1***, **2*** и **3***; **1*** получит 10 пикселов, **2*** — 20 пикселов, а **3*** — 30 пикселов.

Значения длин не учитывают регистр.

Типы содержимого (типы MIME)

«Тип носителя» указывает природу связанного ресурса. Далее будет использоваться термин «тип содержимого» вместо «типа носителя» в соответствии с его использованием. Более того, «тип носителя» может означать носитель, на котором браузер генерирует документ.

Этот тип представлен в DTD с помощью **%ContentType;**.

Типы содержимого учитывают регистр.

Примеры типов содержимого включают «**text/html**», «**image/png**», «**image/gif**», «**video/mpeg**», «**audio/basic**», «**text/tcl**», «**text/javascript**» и «**text/vbscript**».

Тип содержимого «**text/css**», хотя он и не зарегистрирован в IANA, должен использоваться, если связываемым элементом является таблица стилей.

Коды языков

Значения атрибутов, типом которых является код языка (**%LanguageCode** в DTD), относится к коду языка. В кодах языков пробелы недопустимы.

Коды языков учитывают регистр.

Кодировки символов

Атрибуты «**charset**» (**%Charset** в DTD) относятся к кодировкам символов. Значениями должны быть строки (например, «**euc-jp**») из реестра IANA.

Имена кодировок символов учитывают регистр.

Отдельные символы

Определенные атрибуты вызывают отдельный символ из набора символов документа. Эти атрибуты имеют тип **%Character** в DTD.

Отдельные символы можно указать с помощью ссылок на символы (например, «**&**»).

Дата и время

Кодировка ISO позволяет много вариантов представления даты и времени. Один из таких форматов для определения допустимых строк дата/время (**%Datetime** в DTD) следующий:

```
ГГГГ-ММ-ДДТчч:мм:ссУЧП
```

где:

ГГГГ — год из четырех цифр

ММ — месяц из двух цифр (01 — январь и т.д.)

ДД — день из двух цифр (01 — 31)

чч — две цифры часов (00 — 23)

мм — две цифры минут (00 — 59)

сс — две цифры секунд (00 — 59)

УЧП = указатель часового пояса

Указатели часового пояса:

Z

означает **UTC** (Общее скоординированное время). «**Z**» должно быть в верхнем регистре.

+чч:мм

указывает, что местное время отстоит на **чч** часов и **мм** минут от **UTC** вперед.

-чч:мм

указывает, что местное время отстает на **чч** часов и **мм** минут от **UTC**.

Указанные компоненты должны присутствовать в точности, с точно такой же пунктуацией. Помните, что буква «T» отображается в строке литерально (она должна быть в верхнем регистре), для указания начала времени.

Если генерирующее приложение не знает времени с точностью до секунды, для секунд может использоваться значение «00» (при необходимости также для минут и для часов).

Типы ссылок

Авторы могут использовать следующие распознаваемые типы ссылок, перечисленные здесь вместе с условными интерпретациями. В DTD **%LinkTypes** означает список типов ссылок, разделенных пробелами. Символы пробелов в типах ссылок не допускаются.

Эти типы ссылок не учитывают регистр, т.е. «**Alternate**» означает то же, что и «**alternate**».

Браузеры, поисковые машины и т.д. могут интерпретировать эти типы ссылок несколькими способами. Например, браузеры могут предоставлять доступ к связанным документам с помощью навигационной панели.

Alternate

Обозначает альтернативные версии документа, в котором находится ссылка. Вместе с атрибутом **lang** означает переведенную версию документа. Вместе с атрибутом **media** означает версию, созданную для другого носителя.

Stylesheet

Обозначает внешнюю таблицу стилей. Используется вместе с типом ссылки «**Alternate**» для таблиц стилей, выбираемых пользователем.

Start

Обозначает первый документ в наборе. Этот тип ссылки сообщает поисковым машинам о том, какой документ автор считает началом набора.

Next

Обозначает следующий документ в линейной последовательности документов. Браузеры могут предварительно загружать

документ «**next**» для сокращения времени загрузки.

Prev

Обозначает предыдущий документ в упорядоченной серии документов. Некоторые браузеры также поддерживают синоним «**Previous**».

Contents

Обозначает документ, служащий содержанием. Некоторые браузеры также поддерживают синоним **ToC** (из «**Table of Contents**»).

Index

Обозначает документ, являющийся указателем текущего документа.

Glossary

Обозначает документ — глоссарий терминов, относящихся к текущему документу.

Copyright

Обозначает замечание об авторском праве для текущего документа.

Chapter

Обозначает документ, являющийся главой в наборе документов.

Section

Обозначает документ, являющийся разделом в наборе документов.

Subsection

Обозначает документ, являющийся подразделом в наборе документов.

Appendix

Обозначает документ, являющийся приложением в наборе документов.

Help

Обозначает документ, содержащий справку (более подробная информация, ссылки на другие информационные ресурсы и т.д.)

Bookmark

Обозначает закладку. Закладка — это ссылка на ключевую точку в расширенном документе. Атрибут **title** может использоваться, например, для пометки закладки. Помните, что в каждом документе можно определить несколько закладок.

Авторы могут определить дополнительные типы ссылок, не описанные в этой спецификации. При этом они должны использовать профиль для указания соглашений, используемых для определения типов ссылок.

Дескрипторы носителей

Ниже приведен список распознаваемых дескрипторов носителей (**%MediaDesc** в DTD).

screen

Предназначен для экранов компьютеров, не разделенных на страницы.

tty

Предназначен для носителя с фиксированной сеткой для символов, таких как телетайпы, терминалы или переносные устройства с ограниченными возможностями отображения.

tv

Предназначен для устройств типа телевизора (низкое разрешение, цвета, ограниченные возможности прокрутки).

projection

Предназначен для проекторов.

handheld

Предназначен для карманных устройств (небольшой экран, монохромный, растровая графика, ограниченный диапазон).

print

Предназначен для страничных, непрозрачных материалов и документов, просматриваемых на экране в режиме предварительного просмотра печати.

braille

Предназначен для тактильных устройств с алфавитом Бройля.

aural

Предназначен для синтезаторов речи.

all

Для всех устройств.

В будущих версиях HTML могут быть введены новые значения и разрешены параметризованные значения. Для упрощения введения этих расширений соответствующие спецификации браузеры должны иметь возможность анализировать значение атрибута **media** следующим образом:

Значение — это разделенный запятыми список элементов. Например,

```
media="screen, 3d-glasses, print and
resolution > 90dpi"
```

отображается в:

```
"screen"
"3d-glasses"
"print and resolution > 90dpi"
```

Каждый элемент усекается перед первым символом, не являющимся буквой кодировки US ASCII (a-z, A-Z) (десятичные коды Unicode 65-90, 97-122), цифрой (0-9) (шестнадцатеричные коды Unicode 30-39) или знаком переноса (45). В данном примере получается:

"screen"

"3d-glasses"

"print"

Затем с учетом регистра проводится сверка с набором определенных выше типов дескрипторов. Браузеры могут игнорировать несовпадающие элементы. В данном примере останутся только элементы **screen** и **print**.

Таблицы стилей могут включать вариации в зависимости от носителя (например, конструкция CSS **@media**). В таких случаях имеет смысл использовать «**media=all**».

Данные сценария

Данные сценария (**%Script;** в DTD) могут быть содержимым элемента **SCRIPT** и значением атрибутов внутренних событий. Браузеры не должны оценивать данные сценариев в разметке HTML, а должны передавать эти данные ядру сценариев.

Учет регистра в данных сценариев зависит от языка сценариев.

Помните, что данные сценариев, являющиеся содержимым элемента, не могут содержать ссылок на символы, но данные сценария, являющиеся значением атрибута, могут.

Данные таблиц стилей

Данные таблиц стилей (**%StyleSheet;** в DTD) могут быть содержимым элемента **STYLE** и значением атрибута **style**. Браузеры не должны оценивать данные стилей в разметке HTML.

Учет регистра данных стиля зависит от языка таблиц стилей.

Помните, что данные таблиц стилей, являющиеся содержимым элемента, не могут включать ссылки на символы, но данные таблиц стилей, являющиеся значением атрибута, могут включать их.

Целевые имена кадров

За исключением приведенных ниже зарезервированных имен, целевые имена кадров (**%FrameTarget;** в DTD) должны

начинаться с алфавитных символов (a-z, A-Z). Браузеры должны игнорировать все остальные имена.

Следующие **target names** зарезервированы и имеют специальные значения.

_blank

Браузеры должны загружать документ в новое окно без имени.

_self

Браузеры должны загружать документ в тот же кадр, в котором находится ссылающийся на него документ.

_parent

Браузеры должны загружать документ в непосредственный родительский кадр этого кадра во **FRAMESET**. Это значение эквивалентно **_self**, если текущий кадр не имеет родительского кадра.

_top

Браузеры должны загружать документ в полное окно (закрывая все остальные кадры). Это значение эквивалентно **_self**, если у текущего кадра нет родительского кадра.

Глобальная структура документа в формате HTML

Введение в структуру документа HTML

Документ в формате HTML 4.0 состоит из трех частей:

- строки, содержащей информацию о версии HTML
- раздела заголовков (определяемого элементом **HEAD**)
- тела, которое включает собственно содержимое документа. Тело может вводиться элементом **BODY** или элементом **FRAMESET**.

Перед каждым элементом или после каждого элемента может находиться пустое пространство (пробелы, переход на новую строку, табуляции и комментарии). Разделы 2 и 3 должны отделяться элементом HTML.

Вот пример простого документа HTML:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.0//EN"
     "http://www.w3.org/TR/REC-
html40/strict.dtd">
<HTML>
     <HEAD>
          <TITLE>Мой первый документ
HTML</TITLE>
     </HEAD>
     <BODY>
          <P>Всем привет!
     </BODY>
</HTML>
```

Информация о версии HTML

В документе HTML должна быть объявлена используемая в нем версия языка HTML. Объявление типа документа указывает определение типа документа (DTD), используемое в этом документе.

HTML 4.0 определяет три DTD, так что авторы должны включать в свои документы одно из следующих объявлений типов. Разница между DTD заключается в поддерживаемых ими элементах.

HTML 4.0 **Strict** DTD (строгое определение) включает все элементы и атрибуты, не являющиеся нежелательными и не использующиеся в документах с кадрами. Для документов, использующих это DTD, используйте такое объявление типа документа:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-tml40/strict.dtd">
```

HTML 4.0 **Transitional** DTD (переходное определение) включает все, что включено в строгое DTD, а также нежелательные элементы и атрибуты (большинство из которых относится к визуальному представлению). Для документов, использующих это DTD, используйте такое объявление типа документа:

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

HTML 4.0 **Frameset** DTD (определение для кадров) включает все, что включено в переходное DTD, а также кадры. Для документов, использующих это DTD, используйте такое объявление типа:

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

```
"http://www.w3.org/TR/REC-
html40/frameset.dtd">
```

URI в каждом объявлении типа документа позволяет браузерам загрузить DTD и все необходимые **entity sets**. Следующие URI указывают на DTD и **entity sets** для HTML 4.0, поддерживаемого W3C:

```
"http://www.w3.org/TR/REC-html40/strict.dtd"
```

— строгое DTD по умолчанию

```
"http://www.w3.org/TR/REC-html40/loose.dtd"
```

— переходное DTD

```
"http://www.w3.org/TR/REC-
html40/frameset.dtd"
```

— DTD для документов, использующих кадры

```
"http://www.w3.org/TR/REC-
html40/HTMLlat1.ent"
```

— Latin-1 entities

```
"http://www.w3.org/TR/REC-
html40/HTMLsymbol.ent"
```

— Symbol entities

```
"http://www.w3.org/TR/REC-
html40/HTMLspecial.ent"
```

— Special entities

Связь между общими идентификаторами и файлами можно указать с использованием файла каталога, за которым следует формат,

рекомендуемый Открытым консорциумом SGML.

Элемент HTML

```
<!ENTITY % html.content "HEAD, BODY">
<!ELEMENT HTML O O (%html.content;)
-- корневой элемент документа -->
<!ATTLIST HTML
   %i18n;
-- lang, dir -->
```

Начальный тэг: не обязательный, Конечный тэг: не обязательный.

Определения атрибутов

```
версия = cdata[CN]
```

Нежелателен. Значение этого атрибута указывает версию HTML DTD, которой подчиняется этот документ. Этот атрибут является нежелательным, поскольку он является избыточным при наличии информации о версии, указываемой в объявлении типа документа.

Атрибуты, определяемые в любом другом месте

```
lang (информация о языке),
dir (направление текста)
```

После объявления типа документа остальная часть документа HTML содержится

в элементе HTML. Таким образом, типичный документ HTML имеет такую структуру:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-
html40/strict.dtd">
<HTML>
...Здесь идут заголовок, тело и т.д...
</HTML>
```

Заголовок документа

Элемент HEAD

```
<!-- %head.misc;, определенный ранее как
"SCRIPT|STYLE|META|LINK|OBJECT" -->
<!ENTITY % head.content "TITLE & BASE?">
<!ELEMENT HEAD O O (%head.content;)
+(%head.misc;) - заголовок документа -->
<!ATTLIST HEAD
   %i18n;
-- lang, dir --
   profile        %URI;
#IMPLIED   -- каталог метаинформации -->
```

Начальный тэг: не обязателен. Конечный тэг: не обязателен.

Определения атрибутов

```
profile = uri [CT]
```

Этот атрибут указывает местоположение одного или нескольких профилей метаданных, отделяемых пробелами. Для расширения в будущем браузеры должны предполагать, что значение является списком, хотя в как правило значащим считается только первый URI.

Атрибуты, определяемые в любом другом месте

```
lang (информация о языке),
dir (направление текста)
```

Элемент **HEAD** содержит информацию о текущем документе, такую как заголовок, ключевые слова, которые могут использоваться поисковыми машинами, и другие данные, которые не считаются содержимым документа. Браузеры обычно не используют при генерации элементы из раздела **HEAD**. Однако они могут предоставлять пользователям информацию из раздела **HEAD** с помощью своих собственных механизмов.

Элемент TITLE

```
<!-- Элемент TITLE не считается частью
текста.
```

```
Он должен отображаться, например, в
качестве заголовка страницы или окна. В
документе должен быть ровно один
заголовок. -->
<!ELEMENT TITLE - - (#PCDATA) -
(%head.misc;) -- document title -->
<!ATTLIST TITLE %i18n>
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Атрибуты, определяемые в любом другом месте

```
lang (информация о языке),
dir (направление текста)
```

Каждый документ HTML должен иметь элемент **TITLE** в разделе **HEAD**.

Авторы должны использовать элемент **TITLE** для идентификации содержимого документа. Поскольку пользователи часто обращаются к документам за пределами контекста, авторам следует обеспечивать заголовки в широком контексте. Таким образом, вместо заголовков типа «Введение», ничего не говорящих о контексте, авторам следует использовать заголовки типа «Введение в средневековое пчеловодство».

Из соображений доступности браузеры всегда должны делать содержимое элемента **TITLE** доступным пользователям (включая

элементы **TITLE** в кадрах). Механизм этого зависит от браузера (например, в виде заголовка или произносимый).

Заголовки могут включать **character entities** (для символов со знаком ударения, специальных символов и т.д.), но не могут содержать другой разметки. Вот образец заголовка документа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.0//EN"
"http://www.w3.org/TR/REC-
html40/strict.dtd">
<HTML>
<HEAD>
<TITLE>Исследование динамики популяции
</TITLE>
... другие элементы заголовка...
</HEAD>
<BODY>
... тело документа...
</BODY>
</HTML>
```

Атрибут title

Определения атрибутов

```
title = text [CS]
```

Этот атрибут предлагает информацию об элементе, для которого он устанавливается.

В отличие от элемента **TITLE**, который предоставляет информацию обо всем документе и может присутствовать в тексте только один раз, атрибут **title** может сопровождать любое число элементов. Узнать, поддерживает ли элемент этот атрибут, можно в определении элемента.

Значения атрибута **title** могут использоваться браузерами в генерации изображения по-разному. Например, визуальные браузеры часто отображают заголовок как подсказку (краткое сообщение, которое появляется, если вы указываете на объект). Аудибраузеры могут проговаривать информацию заголовка. Например, установка этого атрибута для ссылки позволяет браузерам (визуальным и невизуальным) сообщить пользователям о природе связанного ресурса:

```
...текст...
Вот фотография
<A href="http://someplace.com/neatstuff.gif"
title="Me scuba diving">
как я нырял в прошлом году
</A>
...еще текст...
```

Атрибут **title** играет дополнительную роль при использовании с элементом **LINK** для назначения внешней таблицы стилей.

Для улучшения качества синтеза речи в случае плохой обработки стандартными механизмами будущие версии HTML могут включать атрибут для кодирования фонематической информации.

Метаданные

HTML позволяет авторам указывать метаданные — информацию о документе вместо содержимого документа — множеством способов.

Например, чтобы указать автора документа, можно использовать элемент **META** следующим образом:

```
<META name="Author" content="Иван Иванов">
```

Элемент **META** задает свойство (здесь «Author (Автор)») и назначает ему значение (здесь — «Иван Иванов»).

Значение свойства и набор допустимых значений этого свойства должны определяться в относительном словаре, называемом профилем. Например, профиль, разработанный для помощи в индексировании документов для поисковых машин может определять такие свойства как «**author**», «**copyright**», «**keywords**» и т.д.

Задание метаданных

В общем случае задание метаданных состоит из двух шагов:

- Объявление свойства и его значения. Это можно сделать двумя способами:
  - Из документа с помощью элемента **META**.
  - Не из документа с помощью ссылки на метаданные через элемент **LINK**.
- Сославшись на профиль, в котором определяются свойства и их допустимые значения. Для назначения профиля используйте атрибут профиль элемента **HEAD**.

Помните, что поскольку профиль определяется для элемента **HEAD**, этот профиль применяется ко всем элементам **META** и **LINK** в заголовке документа.

Браузеры не обязательно должны поддерживать механизмы метаданных.

Элемент META

```
<!ELEMENT  META  -  O  EMPTY
--  общая  метаинформация  -->
<!ATTLIST  META
   %i18n;
```



```
--  lang,  dir,  для  использования  с
содержимым  --
   http-equiv   NAME
#IMPLIED  --  имя  заголовка  ответа  HTTP  --
   name            NAME
#IMPLIED  --  имя  метаинформации  --
   content        CDATA
#REQUIRED  -  связанная  информация  --
   scheme          CDATA
#IMPLIED  --  выбор  формы  содержимого  --
   >
```

Начальный тэг: обязателен. Конечный тэг: запрещен.

Определения атрибутов

Для следующих атрибутов допустимые значения и их интерпретация зависят от профиля:

```
name  =  name  [CS]
```

Этот атрибут определяет имя свойства.

```
content  =  cdata  [CS]
```

Этот атрибут определяет значение свойства.

```
scheme  =  cdata  [CS]
```

Этот атрибут дает имя схеме, используемой для интерпретации значения свойства.

```
http-equiv = name [CI]
```

Этот атрибут может использоваться вместо атрибута **name**. Серверы HTTP используют этот атрибут для сбора информации для заголовков сообщений ответов HTTP.

Атрибуты, определяемые в любом другом месте

```
lang (информация о языке),
dir (направление текста)
```

Элемент **META** может использоваться для идентификации свойств документа (например, автора, срок истечения, список ключевых слов и т.д.) и назначения им значений.

Каждый элемент **META** задает пару свойство/значение. Атрибут **name** определяет свойства, а атрибут **content** — значение.

Например, в следующем объявлении устанавливается значение свойства **Author**:

```
<META name="Author" content="Дэйв Рэггетт">
```

Атрибут **lang** может использоваться с элементом **META** для указания языка значения атрибута **content**. Это позволяет синтезаторам речи использовать правила произношения для разных языков.

В этом примере имя автора объявляется на французском языке:


```
<META name="Author" lang="fr"
content="Arnaud Le Hors">
```

Элемент **META** — это общий механизм задания метаданных. Однако некоторые элементы и атрибуты HTML уже обрабатывают некоторые части метаданных и могут использоваться авторами вместо элементов **META** для указания этих частей: элементы **TITLE**, **ADDRESS**, **INS** и **DEL**, атрибут **title** и атрибут **cite**.

Если свойство, заданное с помощью элемента **META**, принимает значение URI, некоторые авторы предпочитают указывать метаданные с помощью элемента **LINK**. Таким образом, следующее объявление:

```
<META name="DC.identifier"
content="ftp://ds.internic.net/rfc/rfc
1866.txt">
```

можно также записать следующим образом:

```
<LINK rel="DC.identifier"
              type="text/plain"
href="ftp://ds.internic.net/rfc/rfc1866.txt"
>
```

META и заголовки HTTP

Атрибут **http-equiv** может использоваться вместо атрибута **name**; он особенно важен, если документы загружаются по протоколу

передачи гипертекста (HTTP). Серверы HTTP могут использовать имя свойства, указанное в атрибуте **http-equiv** для создания заголовка в ответе HTTP.

В следующем примере объявление **META**:

```
<META http-equiv="Expires" content="Tue, 20
Aug 1996 14:25:27 GMT">
```

вернет следующий заголовок HTTP:

```
Expires: Tue, 20 Aug 1996 14:25:27 GMT
```

Это может использоваться кэш-памятью для определения того, когда следует загрузить новую копию связанного документа.

Некоторые браузеры поддерживают использование элемента **META** для обновления текущей страницы по истечении указанного числа секунд с возможностью замены на другой URI.

```
<META http-equiv="refresh"
content="3,http://www.acme.com/intro.html">
```

content — это число, указывающее задержку в секундах, за которым следует URI, который нужно загрузить по прошествии этого времени. Этот механизм широко используется для создания кратковременных заставок. Однако поскольку некоторые браузеры не поддерживают этот механизм, авторам следует включить в заставку возможность перейти на следующую

страницу (чтобы они на «зависли» на заставке).

META и поисковые машины

Основной способ использования элемента **META** — задание ключевых слов, которые поисковые машины могут использовать для улучшения результатов поиска. Если информация о документе представлена в нескольких элементах **META** в зависимости от языка, поисковые машины могут фильтровать атрибут **lang** и отображать результаты поиска с использованием выбранного пользователем языка. Например,

```
<-- Для говорящих на американском
английском -->
<META name="keywords" lang="en-us"
content="vacation, Greece, sunshine">
<-- Для говорящих на британском
английском -->
<META name="keywords" lang="en"
content="holiday, Greece, sunshine">
<-- для русскоязычных пользователей -->
<META name="keywords" lang="fr"
content="отпуск, Греция, солнце">
```

Эффективность поисковых машин можно также повысить с использованием элемента **LINK** для задания ссылок на переводы документа на другие языки, ссылки на

версии документа для другого носителя (например, PDF), и, если документ является частью набора, ссылки на соответствующую начальную точку для просмотра набора.

META и PICS

Platform for Internet Content Selection (Платформа для выбора содержимого Интернет) — это инфраструктура для связывания меток (метаданных) с содержимым Интернет. Созданная для помощи родителям и учителям в управлении доступом детей к Интернет, она также упрощает другое использование меток, включая управление подписью кодов, секретностью и правами интеллектуальной собственности.

Этот пример иллюстрирует использование объявления **META** для включения метки PICS 1.1:

```
<HEAD>
  <META http-equiv="PICS-Label"
content='
  (PICS-1.1 "http://www.gcf.org/v2.5"
  labels on "1994.11.05T08:15-0500"
  until "1995.12.31T23:59-0000"
  for "http://w3.org/PICS/Overview.html"
ratings (suds 0.5 density 0 color/hue 1))
'>
```



```
<TITLE>... название документа ...</TITLE>
</HEAD>
```

META и информация по умолчанию

Элемент **META** может использоваться для указания информации по умолчанию для документа в следующих случаях:

- Язык сценариев по умолчанию.
- Язык таблиц стилей по умолчанию.
- Кодировка символов документа.

В следующем примере для документа указывается кодировка символов ISO-8859-5:

```
<META http-equiv="Content-Type"
content="text/html; charset=ISO-8859-5">
```

Профили метаданных

Атрибут профиль элемента **HEAD** указывает местоположение профиля метаданных. Значением атрибута **profile** является URI. Браузеры могут использовать этот URI двумя способами:

- Как глобально уникальное имя. Браузеры могут распознавать имя (не загружая в действительности профиль) и выполнять некоторые действия на базе известных соглашений для этого профиля. Например, поисковые

машины могут обеспечивать интерфейс для поиска в каталогах документов HTML, где все эти документы используют один и тот же профиль для представления записей каталога.

- Как ссылку. Браузеры могут разыменовывать URI и выполнять некоторые действия на базе определений из профиля (например, авторизовать использование профиля в текущем документе HTML).

В этом примере используется гипотетический профиль, определяющий полезные свойства для индексирования документов. Для свойств, определяемых этим профилем — включая «**author**», «**copyright**», «**keywords**» и «**date**» — значения устанавливаются с помощью последовательных объявлений **META**.

```
<HEAD
profile="http://www.acme.com/profiles/core">
<TITLE>How to complete Memorandum cover
sheets</TITLE>
   <META name="author" content="John Doe">
   <META name="copyright" content="&copy;
1997 Acme Corp.">
   <META name="keywords"
content="corporate,guidelines,cataloging">
```



```
   <META name="date" content="1994-11-
06T08:49:37+00:00">
  </HEAD>
```

Атрибут scheme позволяет авторам предоставлять браузерам дополнительный контекст для корректной интерпретации метаданных. Иногда такая дополнительная информация может иметь важное значение, например, если метаданные указаны в другом формате. Например, автор может указать дату в формате «10-9-97» (неоднозначно); означает ли это 9 октября 1997 г. или 10 сентября 1997 г.? Значение атрибута **scheme "Month-Date-Year"** устранит неоднозначность.

В других случаях атрибут **scheme** может предоставлять браузерам полезную, но не столь важную информацию.

Например, следующее объявление **scheme** поможет браузерам определить, что значение свойства «**identifier**» — номер кода ISBN:

```
<META scheme="ISBN"
name="identifier" content="0-8230-2355-9">
```

Значения атрибута **scheme** зависят от свойства **name** и связанного профиля.

Примером профиля является **Dublin Core**. Этот профиль определяет набор рекомендуемых свойств для электронных библиографических описаний и предназначен

для обеспечения интероперабельности в несопоставимых моделях описаний.

Тело документа

Элемент BODY

```
<!ELEMENT BODY O O (%block;|SCRIPT)+
+(INS|DEL) - тело документа -->
<!ATTLIST BODY
   %attrs;
-- %coreattrs, %i18n, %events --
   onload                %Script;
#IMPLIED   -- документ загружен --
   onunload              %Script;     #IMPLIED
-- документ удален --
   >
```

Начальный тэг: не обязателен. Конечный тэг: не обязателен.

Определения атрибутов

```
background = uri[CT]
```

Нежелателен. Значение этого атрибута — URI, указывающий на изображение. Это изображение является фоном (для визуальных браузеров).

```
text = color[CI]
```

Нежелателен.Этот атрибут устанавливает цвет текста (для визуальных браузеров).


```
link = color [CI]
```

Нежелателен. Этот атрибут устанавливает цвет текста гипертекстовых ссылок, по которым вы не переходили (для визуальных браузеров).

```
vlink = color [CI]
```

Нежелателен. Этот атрибут устанавливает цвет текста ссылок, по которым вы переходили (для визуальных браузеров).

```
alink = color [CI]
```

Нежелателен. Этот атрибут устанавливает цвет текста ссылок, когда они выбраны пользователем (для визуальных браузеров).

Атрибуты, определяемые в любом другом месте

```
id, class (идентификаторы в пределах
документа)
lang (информация о языке),
dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле)
bgcolor (цвет фона)
onload, onunload (внутренние события)
onclick, ondblclick, onmousedown,
onmouseup, onmouseover, onmousemove,
onmouseout, onkeypress, onkeydown, onkeyup
(внутренние события)
```

В теле документа располагается содержание документа. Содержимое может представляться браузером несколькими способами. Например, для визуальных браузеров вы можете считать тело документа полотном, на котором отображается содержимое: текст, изображения, цвета, рисунки и т.д. Для аудиобраузеров оно может произноситься. Поскольку предпочтительным способом задания представления документов теперь являются таблицы стилей, атрибуты представления в тэге **BODY** являются нежелательными.

Пример нежелательного использования

В следующем фрагменте кода HTML показано использование нежелательного атрибута. Он устанавливает белый цвет фона, черный цвет текста и красный цвет гиперссылок изначально, цвет фуксии при активизации ссылок и коричневый для ссылок, по которым вы переходили.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.0 Transitional//EN"
     "http://www.w3.org/TR/REC-
html40/loose.dtd">
<HTML>
<HEAD>
  <TITLE>Динамика популяции</TITLE>
```



```
</HEAD>
<BODY bgcolor="white" text="black"
   link="red" alink="fuchsia"
vlink="maroon">
   ... тело документа...
</BODY>
</HTML>
```

Используя таблицы стилей, того же эффекта можно достичь следующим образом:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-
html40/strict.dtd">
<HTML>
<HEAD>
  <TITLE>Динамика популяции</TITLE>
  <STYLE type="text/css">
  BODY { background: white; color: black}
    A:link { color: red }
    A:visited { color: maroon }
    A:active { color: fuchsia }
  </STYLE>
</HEAD>
<BODY>
   ... тело документа...
</BODY>
</HTML>
```

Использование внешних (связываемых) таблиц стилей обеспечивает гибкость при

изменении представления без пересмотра источника документа HTML:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-
html40/strict.dtd">
<HTML>
<HEAD>
  <TITLE>Динамика популяции</TITLE>
  <LINK rel="stylesheet" type="text/css"
href="smartstyle.css">
</HEAD>
<BODY>
   ... тело документа...
</BODY>
</HTML>
```

Кадры и тела документов HTML. В документах, содержащих кадры, элемент **BODY** заменяется элементом **FRAMESET**.

Идентификаторы элементов: атрибуты id and class

Определения атрибутов

```
id = name [CS]
```

Этот атрибут назначает элементу имя. Имя в пределах документа должно быть уникальным.


```
class = cdata-list [CS]
```

Этот атрибут назначает элементу имя класса или набор имен классов. Одно и то же имя или имена классов могут быть назначены любому числу элементов. Несколько имен классов должны быть разделены пробелами.

Атрибут **id** назначает элементу уникальный идентификатор (который может проверяться синтаксическим анализатором SGML). Например, следующие абзацы распознаются по значениям их атрибутов **id**:

```
<P id="myparagraph"> Этот абзац имеет
уникальное имя.</P>
<P id="yourparagraph"> Этот абзац тоже
имеет уникальное имя.</P>
```

Атрибут **id** имеет в HTML несколько ролей:

- Способ выбора таблицы стиля.
- Назначение цели (якорь) для гипертекстовых ссылок.
- Средство ссылки на определенный элемент сценария.
- Имя объявленного объекта **OBJECT**.
- В целях обработки браузерами (например, для полей идентификации при извлечении данных из страниц HTML в базу данных, преобразовании

документов HTML в другие форматы и т.д.).

Атрибут **class**, с другой стороны, назначает одно или несколько имен классов элементу; при этом элемент может называться принадлежащим к этим классам. Имя класса может использоваться несколькими экземплярами элемента. Атрибут **class** имеет в HTML несколько ролей:

- Способ выбора таблицы стиля (когда автор хочет назначить информацию о стиле набору элементов).
- Для общей обработки браузерами.

Далее элемент **exampleSPAN** используется вместе с атрибутами **id** и **class** для пометки сообщений документа. Сообщения отображаются в английской и русской версиях.

```
<!-- английские сообщения -->
<P><SPAN id="msg1" class="info"
lang="en">Variable declared twice</SPAN>
<P><SPAN id="msg2" class="warning"
lang="en">Undeclared variable</SPAN>
<P><SPAN id="msg3" class="error"
lang="en">Bad syntax for variable
name</SPAN>
<!-- русские сообщения -->
<P><SPAN id="msg1" class="info"
```



```
lang="fr">Переменная объявлена
дважды</SPAN>
<P><SPAN id="msg2" class="warning"
lang="fr">Переменная не объявлена</SPAN>
<P><SPAN id="msg3" class="error"
lang="fr">Синтаксическая ошибка в имени
переменной</SPAN>
```

Следующие правила стиля CSS сообщат браузерам о необходимости отображения информации зеленым цветом, предупреждений — желтым, а сообщений об ошибках — красным:

```
SPAN.info      { color: green }
SPAN.warning { color: yellow }
SPAN.error     { color: red }
```

Помните, что русское «**msg1**» и английское «**msg1**» не могут отображаться в одном документе, поскольку они используют одно и то же значение атрибута **id**. Авторы могут извлечь дальнейшую пользу, используя атрибут **id** для усовершенствования представления отдельных сообщений, указания их в качестве целей (якорей) и т.д.

Почти каждому элементу HTML может быть назначен идентификатор и информация о классе.

Предположим, мы пишем документ о языке программирования. Этот документ

должен включать ряд отформатированных примеров. Для форматирования примеров мы используем элемент **PRE**. Мы также назначаем цвет фона (зеленый) всем экземплярам элемента **PRE**, принадлежащим классу «**example**».

```
<HEAD>
<TITLE>... название документа...</TITLE>
<STYLE type="text/css">
PRE.example { background : green }
</STYLE>
</HEAD>
<BODY>
<PRE class="example" id="example-1">
...код примера...
</PRE>
</BODY>
```

Установив атрибут **id** для этого примера, мы можем:

- создать на него гиперссылку;
- использовать информацию о стиле, отличную от определенной в таблицы, для одного экземпляра информации о стиле.

Атрибут **id** использует одно пространство имен с атрибутом **name**, если он используется для имен якорей.

Элементы уровня блока и встроенные элементы

Некоторые элементы HTML, которые могут присутствовать внутри тэга **BODY**, называются элементами «уровня блока», в то время как другие — «встроенными» (также называемыми элементами «уровня текста»).

Модель содержимого

Обычно элементы уровня блока могут содержать встроенные элементы и другие элементы уровня блока. Обычно встроенные элементы могут содержать только данные и другие встроенные элементы. Этому структурному различию свойственна идея о том, что элементы блока создают «большие» структуры, чем встроенные элементы.

Форматирование

По умолчанию элементы уровня блока форматируются иначе, чем встроенные элементы. Обычно элементы уровня блока начинаются с новой строки, а встроенные элементы — нет.

Направление

По техническим причинам, затрагивающим алгоритм двунаправленного текста (UNICODE), элементы уровня блока и встроенные элементы различаются способами

наследования информации о направлении. Таблицы стилей обеспечивают средства задания отображения произвольных элементов, включая и то, генерируется ли элемент как блочный или встроенный. В некоторых случаях, например, в случае встроенного стиля для элементов списка, это может быть полезным, но вообще говоря, авторам следует избегать такого переопределения интерпретации элементов языка HTML.

Изменение традиционных выражений представления для элементов уровня блока и встроенных элементов влияет на алгоритм двунаправленного текста.

Группировка элементов: элементы DIV и SPAN

```
<!ELEMENT DIV - - (%flow;)*
-- общий контейнер языка/стиля -->
<!ATTLIST DIV
   %attrs;
-- %coreattrs, %i18n, %events --
   >
<!ELEMENT SPAN - - (%inline;)*
-- общий контейнер языка/стиля -->
<!ATTLIST SPAN
   %attrs;
-- %coreattrs, %i18n, %events --
   >
```


Начальный тэг: обязателен. Конечный тэг: обязателен.

Определения атрибутов

```
href = uri [CT]
```

Этот атрибут указывает ресурс, предоставляющий дополнительную информацию о содержимом элемента **DIV** или **SPAN**.

Атрибуты, определяемые в любом другом месте

```
id, class (идентификаторы в пределах
документа)
lang (информация о языке),
dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле
information)
align (выравнивание)
onclick, ondblclick, onmousedown,
onmouseup, onmouseover, onmousemove,
onmouseout, onkeypress, onkeydown, onkeyup
(внутренние события)
```

Элементы **DIV** и **SPAN** вместе с атрибутами **id** и **class** обеспечивают общий механизм добавления в документы структуры. Эти элементы определяют встраиваемую информацию (**SPAN**) или информацию уровня блока (**DIV**), но не налагают никаких

других выражений для представления контекста. Таким образом, авторы могут использовать эти элементы с таблицами стилей, атрибутами **lang** и т.д.

Предположим, вы хотите сгенерировать документ в формате HTML на основе базы данных информации о клиентах. Поскольку HTML не включает элементов для идентификации таких объектов как «клиент», «номер телефона», «адрес электронной почты» и т.д., мы используем элементы **DIV** и **SPAN** для достижения нужных эффектов структуры и представления. Для структурирования информации мы могли использовать элемент **TABLE** следующим образом:

```
<!-- Пример данных из базы данных о
клиентах: -->
<!-- Имя: Ivan Ivanov, Тел.: (095) 185-
3332, Email: II@oo.org -->
<DIV id="client-ivanov" class="client">
<P><SPAN class="client-title">Информация о
клиенте:</SPAN>
<TABLE class="client-data">
<TR><TH>Фамилия:<TD>Ivanov</TR>
<TR><TH>Имя:<TD>Ivan</TR>
<TR><TH>Тел:<TD>(095) 185-3332</TR>
<TR><TH>Email:<TD>ii@oo.org</TR>
</TABLE>
```



```
</DIV>
<DIV id="client-petrov" class="client">
<P><SPAN class="client-title">Информация о
клиенте:</SPAN>
<TABLE class="client-data">
<TR><TH>Фамилия:<TD>Petrov</TR>
<TR><TH>Имя:<TD>Petr</TR>
<TR><TH>Тел:<TD>(253) 115-5420</TR>
<TR><TH>Email:<TD>pp@coucou.com</TR>
</TABLE>
</DIV>
```

Затем мы легко сможем добавить объявление таблицы стиля для настройки представления этих записей.

Визуальные браузеры обычно помещают символ перевода строки перед и после элементов **DIV**, например:

```
<P>ааааааааа<DIV>ббббббббб</DIV><DIV>ввввв<P
>ввввв</DIV>
```

что обычно генерируется как:

```
ааааааааа
ббббббббб
ввввв

ввввв
```

Заголовки: Элементы H1, H2, H3, H4, H5, H6

```
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
<!--
   Существует шесть уровней заголовков -
с H1 (наиболее важный) до H6 (наименее
важный).
-->
<!ELEMENT (%heading;)   - - (%inline;)* --
заголовки -->
<!ATTLIST (%heading;)
   %attrs;
-- %coreattrs, %i18n, %events --
   >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Атрибуты, определяемые в любом другом месте

id, class (идентификаторы в пределах документа)
lang (информация о языке),
dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле)
align (выравнивание)
onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события)

Заголовок кратко описывает содержание раздела, которому он предшествует. Информация из заголовка может использоваться браузерами, например, для автоматического построения оглавления документа.

В языке HTML существует шесть уровней заголовков: **H1** — наиболее важный — и **H6** — наименее важный. Визуальные браузеры обычно отображают более важные заголовки более крупным шрифтом.

В следующем примере показано, как использовать элемент **DIV** для того, чтобы связать заголовок с последующим разделом документа. Это позволит вам определить стиль для раздела (цвет фона, шрифт и т.д.) с использованием таблиц стилей.

```
<DIV class="section" id="forest-elephants"
>
<H1>Лесные слоны</H1>
<P>В этом разделе мы обсуждаем менее
известных лесных слонов.
...продолжение раздела...
<DIV class="subsection" id="forest-habitat"
>
<H2>Ариал</H2>
<P>Лесные слоны не живут в деревьях, а
среди них.
...продолжение раздела...
```

```
</DIV>
</DIV>
```

Эту структуру можно представить с использованием информации о стиле, например:

```
<HEAD>
<TITLE>... название документа ...</TITLE>
<STYLE type="text/css">
DIV.section { text-align: justify;
font-size: 12pt}
DIV.subsection { text-indent: 2em }
H1 { font-style: italic; color: green }
H2 { color: green }
</STYLE>
</HEAD>
```

Пронумерованные разделы и ссылки

Язык HTML не генерирует номера разделов из заголовков. Это может выполняться браузерами. Вскоре языки описания таблиц стилей, такие как CSS, предоставят авторам возможность управления генерацией номеров разделов.

Некоторые люди считают пропуск уровней заголовков дурным тоном. Они принимают порядок заголовков H1 H2 H1, но не принимают порядок H1 H3 H1, поскольку пропущен уровень H2.

Элемент ADDRESS

```
<!ELEMENT ADDRESS - - (%inline;)* --
информация об авторе -->
<!ATTLIST ADDRESS
   %attrs;
-- %coreattrs, %i18n, %events --
   >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Атрибуты, определяемые в любом другом месте

```
id, class (идентификаторы в пределах
документ)
lang (информация о языке),
dir (направление текста)
onclick, ondblclick, onmousedown,
onmouseup, onmouseover, onmousemove,
onmouseout, onkeypress, onkeydown, onkeyup
(внутренние события)
```

Элемент **ADDRESS** может использоваться авторами для указания контактной информации или основной части документа, такой как форма. Этот элемент часто находится в начале или в конце документа.

Например, страница на сервере W3C, относящаяся к языку HTML, может включать следующую контактную информацию:

```
<ADDRESS>
<P><A  href="../People/Raggett/">Dave
Raggett</A>,
     <A  href="../People/Arnaud/">Arnaud  Le
Hors</A>,
contact  persons  for  the  <A
href="Activity">W3C  HTML  Activity</A><BR>
$Date:  1997/12/16  05:38:14  $
</ADDRESS>
```


Информация о языке и направление текста

Задание языка содержимого: атрибут lang

Определения атрибутов

`lang = код языка [CI]`

Этот атрибут указывает основной язык значений атрибутов элементов и секстового содержимого. По умолчанию значение этого атрибута не установлено.

Информация о языке, указанная с помощью атрибута **lang**, может использоваться браузером для управления генерацией изображения различными способами. Некоторые ситуации, в которых указываемая автором информация о языке, может быть полезна:

- Помощь поисковым машинам
- Помощь синтезаторам речи
- Помощь браузерам в выборе вариантов глифов для высококачественной типографии

- Помощь браузеру в выборе набора кавычек
- Помощь браузеру в вопросах переноса, лигатур и интервалов
- Помощь программа проверки грамматики и орфографии

Атрибут **lang** указывает код содержимого элемента и значений атрибутов; относится ли он к данному атрибуту, зависит от синтаксиса и семантики атрибута и от операции.

Атрибут **lang** предназначен для того, чтобы позволить браузерам более осмысленно генерировать изображение на основе принятой культурной практики для данного языка. Это не подразумевает, что браузеры должны генерировать символы, не являющиеся типичными для конкретного языка, менее осмысленным способом; браузеры должны пытаться сгенерировать се символы, независимо от значения, указанного в атрибуте **lang**.

Например, если в русском тексте должен появиться символ греческого алфавита:

```
<P><Q lang="ru">"Эта супермощность была
результатом &gamma;-радиации,</Q> объяснил
он.</P>
```


Браузер:

- должен попытаться сгенерировать русский текст соответствующим образом (например, в соответствующих кавычках)
- попытаться сгенерировать символ ?, даже если это не русский символ.

Коды языков

Значением атрибута **lang** является код языка, идентифицирующий естественный разговорный язык, который устно, письменно или иным образом используется для передачи информации между людьми. Компьютерные языки явным образом исключены из кодов языков.

Кратко говоря, коды языков состоят из первичного кода и ряда подкодов, который может быть пустым:

```
код-языка = первичный-код ( «-» подкод )*
```

Вот несколько примеров кодов языков:

en
: английский

en-US
: американская версия английского.

en-cockney
: кокни (диалект английского).

i-navajo

навахо (язык американских индейцев).

x-klingon

Первичный код «x» обозначает экспериментальный код языка

Двухбуквенные первичные коды зарезервированы для сокращений языков по стандарту ISO639. Сюда входят коды **fr** (французский), **de** (немецкий), **it** (итальянский), **nl** (голландский), **el** (греческий), **es** (испанский), **pt** (португальский), **ar** (арабский), **he** (иврит), **ru** (русский), **zh** (китайский), **ja** (японский), **hi** (хинди), **ur** (урду) и **sa** (санскрит).

Любой двухбуквенный подкод считается кодом страны в стандарте ISO3166.

Наследование кодов языков

Элемент наследует информацию о коде языка в следующем порядке старшинства (от высшего к низшему):

Атрибут **lang**, установленный для самого элемента.

Самый близкий родительский элемент, для которого установлено значение атрибута **lang** (то есть атрибут **lang** наследуется).

Заголовок HTTP «**Content-Language**» может конфигурироваться на сервере. Например:

```
Content-Language: en-cockney
```

В примере, показанном ниже, первичным языком документа является французский («fr»). Один абзац объявлен на испанском языке («es»), после чего язык снова становится французским. В следующий абзац включена японская фраза («ja»), после чего язык опять изменяется на французский.

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-
html40/strict.dtd">
<HTML lang="fr">
<HEAD>
<TITLE>Un document multilingue</TITLE>
</HEAD>
<BODY>
...текст интерпретируется как
французский...
<P lang="es">... текст интерпретируется
как испанский...
<P>... текст опять интерпретируется как
французский...
<P>...французский текст, в котором
попадается <EM lang="ja">фрагмент на
японском</EM>, а здесь опять начинается
```

```
французский...
</BODY>
</HTML>
```

Ячейки таблицы могут наследовать значения атрибута **lang** не от родителя, а из первой ячейки объединения.

Интерпретация кодов языков

В контексте HTML код языка должен интерпретироваться браузерами как иерархия знаков, а не один знак. Если браузер генерирует изображение в соответствии с информацией о языке (скажем, сравнивая языковые коды в таблицах стилей и значения атрибута **lang**), он всегда должен находить точное соответствие, но должен также принимать во внимание первичные коды. Таким образом, если значение атрибута **lang** «**en-US**» установлено для элемента HTML, браузер должен сначала выбрать информацию о стиле, совпадающую с «**en-US**», а затем сгенерировать более общее значение «**en**».

Иерархия кодов языков не гарантирует понимания всех языков с общими префиксами людьми, бегло говорящими на одном или нескольких из этих языков. Она помогает пользователю запросить эту общность, когда для пользователя она является истинной.

Указание направления текста и таблиц: атрибут dir

Определения атрибутов

```
dir = LTR | RTL [CI]
```

Этот атрибут задает основное направление нейтрального в смысле направления текста (например, текста, который не наследует направленность, как определено в UNICODE) и направление таблиц. Возможные значения:

- **LTR**: Слева направо.
- **RTL**: Справа налево.

Кроме задания языка документа с помощью атрибута **lang**, авторы могут указать основное направление (слева направо или справа налево) частей текста, таблицы и т.д. Это делается с помощью атрибута **dir**.

UNICODE назначает направление символам и определяет (сложный) алгоритм для определения соответствующего направления текста. Если документ не содержит отображаемых справа налево символов, браузер не должен использовать двунаправленный алгоритм UNICODE. Если документ содержит такие символы, и если

браузер и отображает, он должен использовать двунаправленный алгоритм.

Хотя в Unicode определены специальные символы, отвечающие за направление текста, HTML предлагает конструкции разметки высшего уровня, выполняющие те же функции: атрибут **dir** (не спутайте с элементом **DIR**) и элемент **BDO**. Таким образом, чтобы привести цитату на иврите, проще написать

```
<Q lang="he" dir="rtl">...цитата на
иврите...</Q>
```

чем с эквивалентными ссылками Unicode:

```
&#x202B;&#x05F4;...цитата на
иврите...&#x05F4;&#x202C;
```

Браузеры не должны использовать атрибут **lang** для определения направления текста.

Атрибут **dir** наследуется, и его можно переопределить.

Введение в двунаправленный алгоритм

В следующем примере проиллюстрировано ожидаемое поведение двунаправленного алгоритма. В нем показаны английский текст слева направо и текст на иврите справа налево.

Рассмотрите следующий текст:

```
английский1 ИВРИТ2 английский3 ИВРИТ4
английский5 ИВРИТ6
```

Символы в этом примере (и во всех реплицированных примерах) хранятся в компьютере в том же виде, в каком они отображаются здесь: первый символ — «а», второй — «н», последний «б».

Предположим, для содержащего этот абзац документа определен английский язык. Это означает, что основным направлением является направление слева направо. Корректное представление этой строки:

```
английский1 2ТИРВИ английский3 4ТИРВИ
английский5 6ТИРВИ
<------     <------     <------
   Н           Н           Н
------------------------------>
               Е
```

Строки точек указывают структуру предложения: основным языком является английский, но встроены некоторые элементы на иврите. Для получения корректного представления не нужно никакой дополнительной разметки, поскольку фрагменты на иврите корректно обращаются браузерами, применяющими двунаправленный алгоритм.

С другой стороны, если для документа определен язык иврит, основным будет направление справа налево. Корректное представление, таким образом, будет:

```
6ТИРВИ английский5 4ТИРВИ английский3
2ТИРВИ английский1
------->     ------->     ------->
   E            E            E
<---------------------------------
                H
```

В этом случае все предложение представляется справа налево, а фрагменты на английском языке обращаются двунаправленным алгоритмом.

Наследование информации о направлении текста

Для двунаправленного алгоритма Unicode необходимо основное направление текста для текстовых блоков. Чтобы указать основное направление элементов уровня блока, установите для этого элемента атрибута **dir**. Значением атрибута **dir**, устанавливаемым по умолчанию, является «**ltr**» (слева направо).

Если атрибут **dir** установлен для элемента уровня блока, он действует на протяжении всего элемента и для всех вложенных элементов уровня блока. Установка атрибута

dir для вложенного элемента имеет приоритет по отношению к наследуемому значению.

Чтобы установить основное направление текста для всего документа, установите атрибут **dir** в элементе HTML.

Например:

```
<!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-
html40/strict.dtd">
<HTML dir="RTL">
<HEAD>
<TITLE>...заголовок справа
налево...</TITLE>
</HEAD>
...текст справа налево...
<P dir="ltr">...текст слева направо...</P>
<P>...опять текст справа налево...</P>
</HTML>
```

Встроенные элементы, с другой стороны, не наследуют атрибут **dir**. Это означает, что встроенный элемент без атрибута **dir** не открывает дополнительного уровня внедрения в соответствии с двунаправленным алгоритмом. (Здесь элементом считается элемент уровня блока или встроенный элемент на основе представления по умолчанию. Помните, что элементы **INS** и

DEL могут быть элементами уровня блока или встраиваемыми элементами в зависимости от контекста.)

Установка направления внедренного текста

Двунаправленный алгоритм автоматически обращает последовательности внедренных символов в соответствии с наследуемым направлением (как показано в предыдущих примерах). Однако в общем в расчет принимается только один уровень внедрения. Для того, чтобы изменения направления достигали дополнительных уровней, используйте атрибут **dir** во встроенном элементе.

Рассмотрите текст предыдущего примера:

```
английский1 ИВРИТ2 английский3 ИВРИТ4
английский5 ИВРИТ6
```

Предположим, основным языком для документа, содержащего этот абзац, является английский. В этом английском предложении содержится фрагмент на иврите, продолжающийся от ИВРИТ2 до ИВРИТ4, и в нем содержится англоязычный фрагмент (английский3). Таким образом, желаемое представление текста:

```
английский1 4ТИРВИ английский3 2ТИРВИ
английский5 6ТИРВИ
```



```
       ------->
          A
   <----------------------
          И
-----------------------------------------
          A
```

Для изменения направления текста двух внедренных фрагментов необходимо задать дополнительную информацию, что мы и делаем, явно разделяя второе внедрение. В этом примере мы используем для разметки текста элемент **SPAN** и атрибут **dir**:

```
английский1 <SPAN dir="RTL">ИВРИТ2
английский3 ИВРИТ4</SPAN> английский5
ИВРИТ6
```

Авторы также могут использовать для изменения направления нескольких внедренных фрагментов символы Unicode. Для указания направления слева направо во внедряемом фрагменте окружите текст символами **LEFT-TO-RIGHT EMBEDDING** («**LRE**», шестнадцатеричный код 202A) и **POP DIRECTIONAL FORMATTING** («**PDF**», шестнадцатеричный код 202C). Для указания направления справа налево во внедряемом фрагменте окружите текст символами **RIGHT-TO-LEFT EMBEDDING** («**RTE**», шестнадцатеричный код 202B) и **PDF**.

Использование разметки направленности HTML с символами Unicode

Авторы и разработчики средств создания HTML-документов должны знать о возможных конфликтах, возникающих при использовании атрибута **dir** со встроенными элементами (включая **BDO**) одновременно с соответствующими символами форматирования UNICODE. Предпочтительнее использовать только один метод. Метод с использованием разметки гарантирует структурную целостность документа и устраняет некоторые проблемы с редактированием двунаправленного текста HTML в простых текстовых редакторах, но некоторое программное обеспечение может лучше использовать символы UNICODE. Если используются оба метода, следует хорошо позаботиться о правильном вложении разметки и символов, иначе результаты могут быть непредсказуемыми.

Приоритет над двунаправленным алгоритмом: элемент BDO

```
<!ELEMENT BDO - - (%inline;)*
-- I18N BiDi over-ride -->
<!ATTLIST BDO
   %coreattrs;
-- id, class, style, title --
```



```
   lang            %LanguageCode; #IMPLIED
-- код языка --
   dir             (ltr|rtl)
#REQUIRED - направление --
 >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Определения атрибутов

```
dir = LTR | RTL [CI]
```

Этот обязательный атрибут указывает основное направление текстового содержимого элемента. Это направление имеет приоритет по отношению к наследуемому направлению символов, как определено в UNICODE. Возможные значения:

- **LTR**: Направление слева направо.
- **RTL**: Направление справа налево.

Атрибуты, определяемые в любом другом месте

```
lang (информация о языке)
```

Двунаправленного алгоритма и атрибута **dir** обычно достаточно для управления изменением направления внедренного текста. Однако в некоторых ситуациях двунаправленный алгоритм может привести к некорректному представлению. Элемент **BDO**

позволяет авторам отключать двунаправленный алгоритм для выбранных фрагментов текста.

Рассмотрите документ с тем же текстовым фрагментом:

```
английский1 ИВРИТ2 английский3 ИВРИТ4
английский5 ИВРИТ6
```

но предположите, что этот текст уже представлен в нужном порядке. Одной причиной этого может быть то, что стандарт MIME благоприятствует визуальному порядку, то есть последовательности с направлением справа налево вставляются в байтовый поток с направлением справа налево. В электронной почте это может форматироваться, включая перевод строки, например:

```
английский1 2ТИРВИ английский3
4ТИРВИ английский5 6ТИРВИ
```

Это конфликтует с двунаправленным алгоритмом UNICODE, поскольку этот алгоритм инвертирует 2ТИРВИ, 4ТИРВИ и 6ТИРВИ во второй раз, так что слова на иврите отображаются слева направо, а не справа налево.

В этом случае решением будет переопределить действие двунаправленного алгоритма, поместив выдержку Email в

элемент **PRE** (для сохранения переводов строки) и каждую строку, для которой атрибут **dir** установлен в **LTR**, в элемент **BDO**:

```
<PRE>
<BDO dir="LTR">английский1 2ТИРВИ
английский3</BDO>
<BDO dir="LTR">4ТИРВИ английский5
6ТИРВИ</BDO>
</PRE>
```

Двунаправленному алгоритму выдается команда «Я должен быть слева направо!», что приведет к нужному представлению:

```
английский1 2ТИРВИ английский3
4ТИРВИ английский5 6ТИРВИ
```

Элемент **BDO** следует использовать в сценариях, где необходим абсолютный контроль над последовательностью (например, многоязыковые номера частей). Атрибут **dir** для этого элемента является обязательным.

Авторы могут также использовать специальные символы Unicode для того, чтобы избежать использования двунаправленного алгоритма — **LEFT-TO-RIGHT OVERRIDE** (202D) или **RIGHT-TO-LEFT OVERRIDE** (202E). Символ **POP DIRECTIONAL FORMATTING**

(шестнадцатеричный код 202C) заканчивает любую последовательность, используемую для обхода двунаправленного алгоритма.

Помните, что при использовании атрибута **dir** во встроенных элементах (включая **BDO**) одновременно с соответствующими символами форматирования, могут возникать конфликты.

Существуют специальные соглашения относительно использования значений параметра «**charset**» для указания обработки двунаправленности в почте MIME, в частности для отличия визуальной, явной и неявной направленности. Значение параметра «ISO-8859-8» (для иврита) обозначает визуальную кодировку, «ISO-8859-8-i» обозначает неявную двунаправленность, а «ISO-8859-8-e» обозначает явную направленность.

Поскольку HTML использует двунаправленный алгоритм Unicode, соответствующие документы, использующие кодировку ISO 8859-8, должны помечаться как «ISO-8859-8-i». Явное управление направлением в HTML также возможно, но его нельзя выразить в ISO 8859-8, поскольку не следует использовать «ISO-8859-8-e».

Значение «ISO-8859-8» подразумевает, что документ отформатирован визуально, и некоторая разметка будет использоваться неправильно (например, **TABLE** с выравниванием по правому краю без разбивки строк), чтобы гарантировать правильное отображение для более старых браузеров, не поддерживающих двунаправленность. При необходимости документы можно изменить (и одновременно они будут корректно отображаться в старых версиях браузеров), добавив, где нужно, разметку **BDO**. Вопреки сказанному, кодировка ISO-8859-6 (арабская) не представляет визуального порядка.

Ссылки на символы для управления направлением и объединением

Поскольку иногда возникает двусмысленность относительно некоторых символов (например, символов пунктуации), UNICODE включает символы для правильного определения назначения. Unicode также включает некоторые символы для управления объединением при необходимости (например, в некоторых ситуациях с арабскими символами). HTML 4.0 включает для этих символов ссылки на символы.

Следующее **DTD** определяет представление некоторых объектов направления:

```
<!ENTITY zwnj CDATA "&#8204;"--=нулевая
ширина без объединения -->
<!ENTITY zwj  CDATA "&#8205;"--
=объединитель нулевой ширины-->
<!ENTITY lrm  CDATA "&#8206;"--=метка
слева направо-->
<!ENTITY rlm  CDATA "&#8207;"--=метка
справа налево-->
```

Объект **zwnj** используется для блокировки объединения в тех контекстах, где объединение произойдет, но оно происходить не должно. Объект **zwj** имеет обратное действие; он производит объединение в случае, когда оно не предполагается, но должно произойти. Например, арабская буква «НЕН» используется для сокращения «Hijri», названия исламской системы летоисчисления. Поскольку отдельный иероглиф «НЕН» в арабской письменности выглядит как цифра пять, для того, чтобы не путать букву «НЕН» с последней цифрой пять в годе, используется исходная форма буквы «НЕН». Однако, нет последующего контекста (например, буквы для объединения), с которым можно объединить «НЕН». Символ **zwj** предоставляет такой контекст.

Точно так же в персидских текстах буква может иногда объединяться с последующей буквой, в то время как в рукописном тексте этого быть не должно. Символ **zwnj** используется для блокировки объединения в таких случаях.

Символы порядка — **lrm** и **rlm**, используются для определения направления нейтральных по отношению к направлению символов. Например, если двойные кавычки ставятся между арабской (справа налево) и латинской (слева направо) буквами, направление кавычек неясно (относятся ли они к арабскому или к латинскому тексту?). Символы **lrm** и **rlm** имеют свойство направления, но не имеют свойств ширины и разделения слов/строк.

Отражение глифов символов

Вообще двунаправленный алгоритм не отражает глифы символов и не влияет на них. Исключением являются такие символы как скобки. Если отражение желательно, например, для египетских иероглифов, греческих знаков или специальных эффектов дизайна, можно сделать это с помощью стилей.

Таблице стилей и двунаправленность

Вообще использование таблиц стилей для изменения визуального представления элемента с уровня блока до встроенного и наоборот используется в прямом направлении. Однако, поскольку двунаправленный алгоритм использует различия встроенных элементов/элементов уровня блока, во время преобразования нужно быть внимательным.

Если встроенный элемент, не имеющий атрибута **dir**, преобразуется в стиль элемента уровня блока с помощью таблицы стилей, для определения основного направления блока он наследует атрибут **dir** от ближайшего родительского элемента блока.

Если элемент блока, не имеющий атрибута **dir**, преобразуется к стилю встроенного элемента с помощью таблицы стилей, результирующее представление должно быть эквивалентным, в терминах двунаправленного форматирования, форматированию, получаемому путем явного добавления атрибута **dir** (которому назначено унаследованное значение) преобразованному элементу.

Списки

Введение в списки

Язык HTML предлагает авторам несколько механизмов создания списков информации. В каждом списке должен быть один или несколько элементов списков. Списки могут содержать:

- Неупорядоченную информацию.
- Упорядоченную информацию.
- Определения.

Предыдущий список, например, не упорядочен, он создан с помощью элемента **UL**:

```
<UL>
<LI>Неупорядоченную информацию.
<LI>Упорядоченную информацию.
<LI>Определения.
</UL>
```

Упорядоченный список, создаваемый с помощью элемента **OL**, может содержать информацию, в которой важен порядок, например, рецепт:

1. Тщательно смешать сухие ингредиенты.
2. Влить жидкость.
3. Смешивать 10 минут.
4. Выпекать в течение часа при температуре 300 градусов.

Списки определений, создаваемые с помощью элемента **DL**, могут содержать ряд пар термин/определение (хотя списки определений могут иметь и иные применения). Например, список определений можно использовать в рекламе изделия:

Низкая цена
Новая модель этого изделия существенно дешевле предыдущей!
Проще работа
Мы изменили изделие, так что с ним теперь легко работать!
Безопасно для детей
Вы можете оставить своих детей в комнате, и изделие не причинит им вреда (не гарантируется).

На языке HTML он определяется следующим образом:

```
<DL>
<DT><STRONG>Низкая цена</STRONG>
<DD> Новая модель этого изделия
существенно дешевле предыдущей!
<DT><STRONG>Проще работа</STRONG>
<DD>Мы изменили изделие, так что с ним
```



```
теперь легко работать!
<DT><STRONG> Безопасно для детей </STRONG>
<DD> Вы можете оставить своих детей в
комнате, и изделие не причинит им вреда
(не гарантируется).
</DL>
```

Списки могут быть вложенными, разные типы списков можно использовать вместе, как в следующем примере, где список определений содержит неупорядоченный список (ингредиенты) и упорядоченный список (процедуру):

Ингредиенты:

- 100 г муки
- 10 г сахара
- 1 стакан воды
- 2 яйца
- соль, перец

Процедура:

1. Тщательно смешайте сухие ингредиенты.
2. Влейте жидкие ингредиенты.
3. Смешивайте 10 минут.
4. Выпекайте в течение часа при температуре 300 градусов.

Примечания:
Можно добавить изюм.

Точное представление трех типов списков зависит от браузера. Не стоит использовать списки для создания отступов в тексте. Это делается с помощью таблиц стилей.

Неупорядоченные списки (UL), упорядоченные списки (OL) и элементы списков (LI)

```
<!ELEMENT UL - - (LI)+
-- неупорядоченный список -->
<!ATTLIST UL
   %attrs;
-- %coreattrs, %i18n, %events --
   >
<!ELEMENT OL - - (LI)+
-- упорядоченный список -->
<!ATTLIST OL
   %attrs;
-- %coreattrs, %i18n, %events --
   >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

```
<!ELEMENT LI - O (%flow;)*
-- элемент списка -->
<!ATTLIST LI
   %attrs;
```



```
-- %coreattrs, %i18n, %events --
   >
```

Начальный тэг: обязателен. Конечный тэг: не обязателен.

Определения атрибутов

```
type = информация о стиле [CI]
```

Нежелателен. Этот атрибут устанавливает стиль элемента списка. Доступные в настоящее время значения предназначены для визуальных браузеров. Возможные значения описаны ниже (включая информацию о регистре).

```
start = число [CN]
```

Нежелателен. Только для **OL**. Этот атрибут задает начальный номер первого элемента в упорядоченном списке. По умолчанию начальный номер — «1». Помните, что, хотя значением этого атрибута является целое число, соответствующая метка может быть нецифровая. Если в качестве стиля выбраны латинские буквы верхнего регистра (A, B, C, ...), start=3 означает «C». Если в качестве стиля выбраны римские цифры нижнего регистра, start=3 означает «iii» и т.д.

```
value = число [CN]
```

Нежелательно. Только для **LI**. Этот атрибут устанавливает номер текущего элемента списка. Помните, что, хотя

значением атрибута является целое число, соответствующая метка может быть нечисловая.

compact [CI]

Нежелателен. Если этот логический атрибут установлен, он сообщает визуальным браузерам о том, что генерировать список нужно более компактно. Интерпретация этого атрибута зависит от браузера.

Атрибуты, определяемые в любом другом месте

id, class (идентификаторы в пределах документа)
lang (информация о языке),
dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле)
onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события)

Упорядоченные и неупорядоченные списки генерируются одинаково за исключением того, что визуальные браузеры нумеруют упорядоченные списки. Браузеры могут представлять эти номера несколькими способами. Элементы неупорядоченного списка не нумеруются.

Оба эти типа списков состоят из последовательностей элементов списков, определяемых элементом **LI** (конечный тэг которого можно опустить).

В этом примере показана общая структура списка.

```
<UL>
     <LI> ... первый элемент списка...
     <LI> ... второй элемент списка...
     ...
</UL>
```

Списки могут быть вложенными.

Пример нежелательного использования

```
<UL>
<LI> ... Уровень один, номер один...
<OL>
<LI> ... Уровень два, номер один...
<LI> ... Уровень два, номер два...
<OL start="10">
<LI> ... Уровень три, номер один...
</OL>
<LI> ... Уровень два, номер три...
</OL>
<LI> ... Уровень один, номер два...
</UL>
```

Информация о порядке номеров

В упорядоченных списках невозможно продолжать нумерацию автоматически из предыдущего списка или убрать нумерацию для некоторых элементов. Однако авторы могут пропустить несколько элементов списка, установив для них атрибут **value**. Нумерация для последующих элементов списка продолжается с нового значения. Например:

```
<ol>
<li value="30"> элемент списка номер 30.
<li value="40"> элемент списка номер 40.
<li> элемент списка номер 41.
</ol>
```

Списки определений: элементы DL, DT и DD

```
<!-- списки определений - DT - термин,
DD   - его определение -->

<!ELEMENT DL - - (DT|DD)+
-- список определений -->
<!ATTLIST DL
   %attrs;
-- %coreattrs, %i18n, %events --
   >
```


Начальный тэг: обязателен. Конечный тэг: обязателен.

```
<!ELEMENT DT - O (%inline;)*
-- термин -->
<!ELEMENT DD - O (%flow;)*
-- определение -->
<!ATTLIST (DT|DD)
   %attrs;
-- %coreattrs, %i18n, %events --
   >
```

Начальный тэг: обязателен. Конечный тэг: не обязателен.

Атрибуты, определяемые в любом другом месте

id, class (идентификаторы в пределах документа)
lang (информация о языке), dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле)
onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup (внутренние события)

Списки определений незначительно отличаются от других типов списков — тем, что элементы состоят из двух частей: термина и определения. Термин обозначается с помощью элемента **DT** и может иметь только

встроенное содержимое. Описание указывается с помощью элемента **DD**, имеющего содержимое уровня блока.

Пример:

```
<DL>
   <DT>Dweeb
   <DD>young excitable person who may
mature
      into a <EM>Nerd</EM> or
<EM>Geek</EM>
   <DT>Cracker
   <DD>hacker on the Internet
   <DT>Nerd
   <DD>male so into the Net that he
forgets
      his wife's birthday
</DL>
```

Вот пример с несколькими терминами и определениями:

```
<DL>
     <DT>Center
     <DT>Centre
     <DD> A point equidistant from all
points on the surface of a sphere.
     <DD> In some field sports, the
player who holds the middle position on
the field, court, or forward line.
</DL>
```


Другим применением элемента **DL**, например, может быть разметка диалогов, где каждый элемент **DT** означает говорящего, а в каждом элементе **DD** содержатся его слова.

Визуальное отображение списков

Таблицы стилей предоставляют большие возможности управления форматированием списков (например, в отношении нумерации, соглашений, используемых в разных языках, отступов и т.д.).

Визуальные браузеры обычно сдвигают вложенные списки соответственно уровню вложенности.

Для элементов **OL** и **UL** атрибут **type** определяет параметры генерации для визуальных браузеров.

Для элемента **UL** возможными значениями атрибута **type** являются **disc**, **square** и **circle**. Значение, используемое по умолчанию, зависит от уровня вложенности текущего списка. Эти значения не учитывают регистр.

Представление каждого значения зависит от браузера. Браузеры должны пытаться представлять «disc» в виде небольшого заполненного кружка, «circle» — в виде окружности, а «square» в виде небольшого квадрата.

Графические браузеры могут генерировать их как:

- для значения «disc»
- для значения «circle»
- для значения «square»

Для элемента **OL** возможные значения атрибута **type** приведены в следующей таблице (они учитывают регистр):

Type	Стиль нумерации	
1	арабские цифры	1, 2, 3, ...
a	буквы нижнего регистра	a, b, c, ...
A	буквы верхнего регистра	A, B, C, ...
i	римские цифры в нижнем регистре	i, ii, iii, ...
I	римские цифры в верхнем регистре	I, II, III, ...

Помните, что использование атрибута **type** нежелательно, и стили списков должны определяться с помощью таблиц стилей.

Например, с помощью **CSS** можно указать, что стиль нумерации для элементов списка в нумерованном списке — римские цифры нижнего регистра. В приведенном ниже примере каждый элемент **OL**,

принадлежащий классу «withroman», обозначается римской цифрой.

```
<STYLE type="text/css">
OL.withroman { list-style-type: lower-roman
}
</STYLE>
<BODY>
<OL class="withroman">
<LI> Шаг один ...
<LI> Шаг два ...
</OL>
</BODY>
```

Генерация списка определений также зависит от браузера. Например, список:

```
<DL>
   <DT>Dweeb
   <DD>young excitable person who may
mature
      into a <EM>Nerd</EM> or
<EM>Geek</EM>
   <DT>Cracker
   <DD>hacker on the Internet
   <DT>Nerd
   <DD>male so into the Net that he
forgets
      his wife's birthday
</DL>
```

может генерироваться следующим образом:

```
Dweeb
        young excitable person who may
        mature into a Nerd or Geek
Cracker
        hacker on the Internet
Nerd
        male so into the Net that he
        forgets his wife's birthday
```

Элементы DIR и MENU

Использование элементов **DIR** и **MENU** нежелательно.

Атрибуты, определяемые в любом другом месте

id, class (идентификаторы в пределах документа)
lang (информация о языке),
dir (направление текста)
title (заголовок элемента)
style (встроенная информация о стиле)
onclick, ondblclick, onmousedown,
onmouseup, onmouseover, onmousemove,
onmouseout, onkeypress, onkeydown, onkeyup
(внутренние события)

Элемент **DIR** предназначался для создания многостраничных списков каталогов.

Элемент **MENU** предназначался для использования в списках меню, состоящих из одного столбца. Оба эти элемента имеют ту же структуру, что и элемент **UL**, различаясь только при генерации. На практике браузеры генерируют списки **DIR** или **MENU** точно так же, как список **UL**.

Настоятельно рекомендуется использовать вместо этих элементов элемент **UL**.

Таблицы стилей

Введение в таблицы стилей

Таблицы стилей представляют наибольшее достижение для дизайнеров Web-страниц, расширяя возможности улучшения внешнего вида страниц. В научных средах, в которых и зародилась Web, люди более сосредоточены на содержании документов, нежели на их представлении. По мере открытия Web прочими людьми ограничения HTML стали источником разочарований, и авторам пришлось уклоняться от стилистических ограничений HTML. Хотя намерения и были добрыми — улучшение представления Web-страниц, — технологии имели нежелательные побочные эффекты. Эти технологии работали только для некоторых, только иногда, но не для всех и не всегда. Сюда включаются:

- Использование собственных расширений HTML
- Преобразование текста в изображения
- Использование изображений для управления пустым пространством

- Использование таблиц для размещения объектов на станице
- Написание программ вместо использования HTML

Эти технологии существенно усложняют Web-страницы, ограничивают гибкость, создают проблемы взаимодействия и создают сложности для людей с физическими недостатками.

Таблицы стилей решают эти проблемы, одновременно превосходя ограниченные механизмы представления в HTML. Таблицы стилей упрощают определение интервалов между строками текста, отступов, цветов, используемых для текста и фона, размера и стиля шрифтов и другой информации.

Например, следующая таблица стилей CSS (хранящаяся в файле «special.css») зеленый устанавливает цвет текста абзаца и окружает его сплошной красной рамкой:

```
P.special {
color : green;
border: solid red;
}
```

Авторы могут связывать таблицы стилей с исходным документом HTML с помощью элемента **LINK**:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.0//EN"
     "http://www.w3.org/TR/REC-html40">
<HTML>
   <HEAD>
      <LINK href="special.css"
rel="stylesheet" type="text/css">
   </HEAD>
   <BODY>
      <P class="special">В этом абзаце
текст должен быть зеленым.
   </BODY>
</HTML>
```

HTML 4.0 обеспечивает поддержку следующих функций таблиц стилей:

Гибкое размещение информации о стиле

Помещение таблиц стилей в отдельные файлы упрощает их повторное использование. Иногда полезно включать инструкции по представлению в документ, к которому они применяются, в начало документа или в атрибуты элементов в теле документа. Для упрощения управления стилем сайта применяется использование заголовков HTTP для установки таблиц стилей, применяемых к документу.

Независимость от языков таблиц стилей

HTML не привязан к конкретному языку таблиц стилей. Это позволяет использовать широкий диапазон таких языков, например, простые языки для большинства пользователей и более сложные для более специализированных случаев. Во всех примерах, приведенных ниже, используется язык CSS (Каскадные таблицы стилей), но можно использовать и другие языки.

Каскады

Эта возможность обеспечивается некоторыми языками таблиц стилей, такими как **CSS**, для объединения информации о стиле из нескольких источников. Это может быть, например, корпоративные положения о стиле, стили, общие для группы документов, а также стили, специфичные для одного документа. С использованием раздельного хранения эти таблицы стилей могут использоваться повторно, что упрощает работу авторов и повышает эффективность сетевого кэширования. Каскад определяет упорядоченную последовательность таблиц стилей, в которой правила более поздних таблиц имеют приоритет над более ранними. Не все языки таблиц стилей поддерживают каскады.

Зависимость от устройств

HTML позволяет авторам разрабатывать документы независимо от устройств. Это позволяет пользователям обращаться к Web-страницам с использованием различных устройств, например, графических дисплеев для компьютеров под управлением Windows, Macintosh OS и X11, телевизионных устройств, специальным образом адаптированных телефонов и портативных устройств на базе PDA, речевых браузеров и тактильных устройств на базе азбуки Бройля.

Таблицы стилей, напротив, применяются к конкретным устройствам или группам устройств. Таблица стилей, предназначенная для экрана, может применяться при печати, но бесполезна для речевых браузеров. Это позволяет вам определить широкие катэгории устройств, к которым применяется конкретная таблица стилей и позволяет браузерам избежать загрузки ненужных таблиц стилей. Языки таблиц стилей могут включать функции описания зависимости от устройств в одной таблице.

Альтернативные стили

Авторы могут предлагать читателям несколько способов просмотра документа. Например, таблица стилей для представления компактных документов с мелким шрифтом,

или таблица, задающая крупные шрифты для удобства чтения. Авторы могут указать предпочитаемую таблицу стилей, а также альтернативные таблицы для определенных пользователей или устройств. Браузеры должны предоставлять пользователям возможность выбора одной из альтернативных таблиц или отключать все таблицы стилей.

Вопросы производительности

Некоторые пользователи высказывали сомнения относительно скорости работы таблиц стилей. Например, загрузка внешней таблицы стилей может привести к задержке общего представления материала для пользователя. Подобные ситуации возникают и в том случае, если в заголовок документа включен длинный набор правил относительно стиля.

Эти проблемы решаются путем предоставления авторам возможности включать инструкции по представлению в каждый элемента HTML. Благодаря этому информация о представлении всегда доступна ко времени представления элемента браузером.

Во многих случаях авторы воспользуются преимуществами использования общей

таблицы стилей для группы документов. В этом случае распределение правил стиля в документе приведет к снижению производительности по сравнению с использованием связанной таблице стилей, поскольку для большинства документов таблицы стилей уже будет находиться в локальном кэше. К этому эффекту приведет общедоступность хороших таблиц стилей.

Как добавить стиль в HTML

Документы в формате HTML могут содержать правила таблиц стилей непосредственно или могут импортировать таблицы стилей.

В HTML можно использовать все языки таблиц стилей. Простого языка таблиц стилей может быть достаточно для большинства пользователей, в то время как другие языки могут подходить для более специализированных задач. В примерах ниже используется язык «Каскадные таблицы стилей», сокращенно CSS.

Синтаксис данных стиля зависит от языка таблицы стилей.

Установка языка таблицы стилей по умолчанию

Авторы должны указывать язык таблицы стилей для информации о стиле, связанной с документом HTML.

Для установки языка таблицы стилей для документа по умолчанию следует использовать элемент **META**. Например, чтобы установить по умолчанию язык **CSS**, следует поместить в раздел **HEAD** следующее объявление:

```
<META  http-equiv="Content-Style-Type"
content="text/css">
```

Язык таблиц стилей по умолчанию можно также установить с помощью заголовков HTTP. Показанное выше объявление с использованием тэга **META** эквивалентно заголовку HTTP:

```
Content-Style-Type:  text/css
```

Браузеры должны определять язык таблиц стилей по умолчанию для документа в соответствии со следующими шагами (от высшего приоритета к низшему):

- Если в объявлении **META** задается «**Content-Style-Type**», язык таблиц стилей определяет последнее объявление в потоке символов.

- В противном случае, если «**Content Style-Type**» задается в заголовках HTTP, язык таблиц стилей определяет последний заголовок в потоке символов.
- В противном случае по умолчанию используется язык «**text/css**».

Документы, включающие элементы, в которых устанавливается атрибут **style**, но не определяется язык таблиц стилей по умолчанию, являются некорректными. Средства разработки должна генерировать информацию о языке таблиц стилей по умолчанию (обычно с помощью объявлений **META**), чтобы браузеры не полагались на язык по умолчанию «**text/css**».

Встроенная информация о стиле

Определения атрибутов

```
style = style [CN]
```

Этот атрибут определяет информацию о стиле текущего элемента.

Атрибут **style** определяет информацию о стиле одного элемента. Язык таблиц стилей встроенных правил стиля определяется языком таблиц стилей по умолчанию. Синтаксис данных стиля зависит от языка таблиц стилей.

В данном примере устанавливается информация о цвете и размере шрифта текста определенного абзаца.

```
<P style="font-size: 12pt; color:
fuchsia">Что за прелесть эти таблицы
стилей!
```

В **CSS** объявления свойств имеют форму «имя: значение» и разделяются точкой с запятой.

Атрибут **style** может использоваться для применения определенного стиля к отдельному элементу HTML. Если стиль повторно используется для нескольких элементов, авторы должны использовать элемент **STYLE** для перегруппировки этой информации. Для оптимальной гибкости авторам следует определять стили во внешних таблицах стилей.

Информация о стиле в заголовке: элемент STYLE

```
<!ELEMENT STYLE - - %StyleSheet
-- информация о стиле -->
<!ATTLIST STYLE
   %i18n;
-- lang, dir, для использования с
заголовком --
   type            %ContentType;
#REQUIRED -- тип содержимого языка
```

```
стилей --
   media          %MediaDesc;      #IMPLIED
-- для использования с этими
устройствами --
   title          %Text;
#IMPLIED   -- рекомендуемый заголовок --
   >
```

Начальный тэг: обязателен. Конечный тэг: обязателен.

Определения атрибутов

```
type = content-type [CI]
```

Этот атрибут определяет язык таблиц стилей для содержимого элемента и имеет приоритет над языком таблиц стилей, используемых. Язык таблиц стилей указывается как тип содержимого (например, «**text/css**»). Авторы должны указать значение для этого атрибута; для него нет значения по умолчанию.

```
media = дескрипторы устройств [CI]
```

Этот атрибут задает целевое устройство для информации о стиле. Это может быть один дескриптор устройства или список дескрипторов, разделенных запятыми. По умолчанию устанавливается значение «**screen**».

Атрибуты, определяемые в другом месте

```
lang (информация о языке),
dir (направление текста)
```

Элемент **STYLE** позволяет авторам помещать правила таблиц стилей в раздел **head** документа. В HTML допустимо любое число элементов **STYLE** в разделе **HEAD**.

Браузеры, не поддерживающие таблицы стилей или не поддерживающие определенный язык таблиц стилей, используемый в элементе **STYLE**, не должны показывать элемент **STYLE**. Ошибкой будет генерировать его содержимое как часть текста документа. Некоторые языки таблиц стилей поддерживают синтаксис для того, чтобы не показывать содержимое несоответствующим браузерам.

Синтаксис данных стиля зависит от языка таблицы стилей.

Некоторые реализации таблиц стилей могут поддерживать большее разнообразие правил для элемента **STYLE**, чем в атрибуте **style**. Например, в **CSS** правила могут объявляться в элементе **STYLE** для:

- Всех экземпляров определенного элемента языка HTML (например, для всех элементов **P**, всех элементов **H1** и т.д.)

- Всех экземпляров элемента HTML, принадлежащих определенному классу (т.е. для атрибута **class** в котором установлено определенное значение).
- Отдельных экземпляров элемента языка HTML (т.е. для атрибута **id** которого установлено определенное значение).

Правила приоритета и наследования правил таблиц стилей зависят от языка таблиц.

Следующее объявление **CSS STYLE** приводит к появлению границы вокруг всех элементов **H1** в документе и центрированию их на странице.

```
<HEAD>
  <STYLE type="text/css">
     H1 {border-width: 1; border: solid;
text-align: center}
  </STYLE>
</HEAD>
```

Чтобы указать, что эта информация о стиле должна применяться только к элементам **H1** определенного класса, можно изменить определение следующим образом:

```
<HEAD>
  <STYLE type="text/css">
     H1.myclass {border-width: 1; border:
```



```
solid; text-align: center}
  </STYLE>
</HEAD>
<BODY>
  <H1 class="myclass"> Наш стиль влияет на
этот заголовок уровня H1</H1>
  <H1> А на этот заголовок наш стиль не
влияет</H1>
</BODY>
```

И, наконец, для ограничения области действия информации о стиле единственным экземпляром элемента **H1**, установите атрибут **id**:

```
<HEAD>
  <STYLE type="text/css">
     #myid {border-width: 1; border:
solid; text-align: center}
  </STYLE>
</HEAD>
<BODY>
  <H1 class="myclass"> На этот заголовок
H1 стиль не влияет</H1>
  <H1 id="myid"> А на этот влияет </H1>
  <H1> На этот опять не влияет </H1>
</BODY>
```

Хотя информация о стиле может устанавливаться почти для всех элементов HTML, два элемента, **DIV** и **SPAN**, особенно полезны тем, что они не накладывают

никакой семантики представления (кроме block-level vs. inline). Вместе с таблицами стилей эти элементы позволяют пользователям неограниченно расширять язык HTML, особенно при использовании атрибутов **class** и **id**.

В следующем примере элемент **SPAN** используется для установки малых прописных букв для стиля шрифта первых нескольких слов абзаца.

```
<HEAD>
  <STYLE type="text/css">
    SPAN.sc-ex { font-variant: small-caps }
  </STYLE>
</HEAD>
<BODY>
<P><SPAN class="sc-ex">Первые
несколько</SPAN> слов абзаца выделены
малыми прописными буквами.
</BODY>
```

В следующем примере мы используем элемент **DIV** и атрибут **class** для установки выравнивания текста для ряда абзацев, составляющих введение в научную статью. Информация о стиле может повторно использоваться для других разделов введения путем установки атрибута **class** в любом месте документа.


```
<HEAD>
  <STYLE type="text/css">
     DIV.Abstract { text-align: justify }
  </STYLE>
</HEAD>
<BODY>
  <DIV class="Abstract">
     <P>The Chieftain product range is our
market winner for the coming year. This
report sets out how to position
Chieftain against competing products.
     <P>Chieftain replaces the Commander
range, which will remain on the price
list until further notice.
  </DIV>
</BODY>
```

Типы устройств

HTML позволяет авторам создавать документы, использующие характеристики устройства, на котором будет представляться документ (например, графические дисплеи, телевизионные экраны, переносные устройства, речевые браузеры, тактильные устройства на базе азбуки Бройля и т.д.). С помощью атрибута **media** авторы могут позволить браузерам загружать и применять таблицы стилей выборочно.

Объявления в следующем примере применяются к элементам **H1**. При показе на проекторе во время встречи все экземпляры будут отображаться синим цветом. При печати они будут отцентрированы.

```
<HEAD>
 <STYLE type="text/css" media="projection">
      H1 { color: blue}
 </STYLE>
 <STYLE type="text/css" media="print">
    H1 { text-align: center }
 </STYLE>
```

Этот пример добавляет звуковые эффекты для устройства речевого вывода:

```
<STYLE type="text/css" media="aural">
A { cue-before: uri(bell.aiff); cue-after:
uri(dong.wav)}
</STYLE>
</HEAD>
```

Управление устройствами особенно интересно при использовании с внешними таблицами стилей, поскольку браузеры могут сэкономить время, загружая из сети только таблицы стилей, применяющиеся к текущему устройству. Например, речевые браузеры могут не загружать таблицы стилей, разработанные для визуального представления.

Внешние таблицы стилей

Авторы могут отделять таблицы стилей от документов HTML. Это дает следующие преимущества:

- Авторы и менеджеры Web-сайтов могут совместно использовать таблицы стилей в ряде документов (и сайтов).
- Авторы могут изменять таблицы стилей без изменения документа.
- Браузеры могут загружать таблицы стилей выборочно (в зависимости от описаний устройств).

Предпочитаемые и альтернативные таблицы стилей

HTML позволяет авторам связывать с документом любое число внешних таблиц стилей. Язык таблиц стилей определяет взаимодействие нескольких внешних таблиц стилей (например, правила «каскадов» CSS).

Авторы могут указать ряд взаимоисключающих таблиц стилей, называемых альтернативными. Пользователи могут выбирать таблицы, которые им больше нравятся. Например, автор может указать один стиль для небольших экранов, другой — для слабовидящих пользователей

(например, с использованием крупного шрифта). Браузеры должны предоставлять пользователям возможности выбора одной из альтернативных таблиц.

Автор может указать, что одна из альтернатив является предпочтительной. Браузеры должны применять предпочитаемые автором таблицы стилей, если пользователь не выбрал другую альтернативу.

Авторы могут сгруппировать несколько альтернативных таблиц стилей (включая предпочитаемые автором) под одним именем стиля. Если пользователь выбирает именованный стиль, браузер должен применять все таблицы стилей с этим именем. Браузеры не должны применять альтернативные таблицы стилей с другим именем стиля.

Авторы также могут указать постоянные таблицы стилей, которые браузеры должны применять в дополнение к альтернативным таблицам стилей.

При применении таблицы стилей браузеры должны учитывать дескрипторы устройств.

Браузеры также должны позволять пользователям полностью отключать таблицы стилей автора; в этом случае браузер не должен применять ни одну из таблиц стилей.

Указание внешних таблиц стилей

Авторы указывают внешние таблицы стилей с помощью атрибутов элемента **LINK**:

- Установите в атрибуте **href** местоположение файла таблицы стилей. Значением атрибута **href** должен быть URI.
- Установите для атрибута **type** значение, указывающее язык связанного ресурса (таблицы стилей). Это позволяет браузерам не загружать таблицы стилей, использующие неподдерживаемые языки.
- Укажите, является ли таблица стилей постоянной, предпочитаемой или альтернативной:
  - Чтобы таблицы была постоянной, установите для атрибута **rel** значение «**stylesheet**», и не устанавливайте атрибут **title**.
  - Чтобы таблица была предпочитаемой, установите для атрибута **rel** значение «**stylesheet**», и дайте таблице имя с помощью атрибута **title**.
  - Чтобы указать альтернативную таблицу, установите для атрибута

rel значение «**alternate stylesheet**» а дайте таблице имя с помощью атрибута **title**.

Браузеры должны обеспечивать пользователям средства просмотра и выбора таблицы стилей из списка альтернатив. Для атрибута **title** рекомендуется устанавливать значение, которое будет представлять эту таблицу в списке.

В этом примере мы сначала определяем постоянную таблицу стилей, находящуюся в файле mystyle.css:

```
<LINK href="mystyle.css" rel="stylesheet"
type="text/css">
```

Установка атрибута **title** назначает ее предпочитаемой автором таблицей:

```
<LINK href="mystyle.css" title="Compact"
rel="stylesheet" type="text/css">
```

Добавление ключевого слова «**alternate**» а атрибут **rel** сделает ее альтернативной таблицей стилей:

```
<LINK href="mystyle.css" title="Medium"
rel="alternate stylesheet" type="text/css">
```

Авторы также могут использовать для установки предпочитаемой таблицы стилей элемент **META**. Например, чтобы установить предпочитаемую таблицу стилей «**compact**» (см. предыдущий пример), авторы могут

включить в элемент **HEAD** следующую строку:

```
<META http-equiv="Default-Style"
content="compact">
```

Предпочитаемую таблицу стилей можно также указать с помощью заголовков HTTP. Объявление **META** выше эквивалентно заголовку HTTP:

```
Default-Style: "compact"
```

Если предпочитаемая таблица стилей указывается двумя или более элементами **META** или заголовками HTTP, преимущество имеет последнее объявление. Считается, что заголовки HTTP обрабатываются раньше, чем объявления **HEAD**.

Если предпочитаемая таблица стилей задается двумя или более элементами **LINK**, преимущество имеет первая.

Предпочитаемые таблицы стилей, задаваемые с помощью **META** или заголовков HTTP имеют преимущество над таблицами, задаваемыми элементом **LINK**.

Каскады таблиц стилей

Каскадные языки таблиц стилей, такие как **CSS**, позволяют использовать информацию о стиле из нескольких

источников. Однако не все языки таблиц стилей поддерживают каскады. Чтобы определить каскад, авторы указывают последовательность элементов **LINK** и/или **STYLE**. Каскад информации таблиц стилей производится в порядке указания элементов в разделе **HEAD**.

В следующем примере мы определяем две альтернативные таблицы стилей с именем «**compact**». Если пользователь выбирает стиль «**compact**», браузер должен применять обе внешние таблицы, а также постоянную таблицу «common.css». Если пользователь выбирает стиль «**big print**», применяться будут только альтернативная таблица «bigprint.css» и постоянная таблица «common.css».

```
<LINK rel="alternate stylesheet"
title="compact" href="small-base.css"
type="text/css">
<LINK rel="alternate stylesheet"
title="compact" href="small-extras.css"
type="text/css">
<LINK rel="alternate stylesheet" title="big
print" href="bigprint.css" type="text/css">
<LINK rel="stylesheet" href="common.css"
type="text/css">
```

Вот пример каскада, в котором задействованы оба элемента — **LINK** и **STYLE**.


```
<LINK rel="stylesheet" href="corporate.css"
type="text/css">
<LINK rel="stylesheet"
href="techreport.css" type="text/css">
<STYLE type="text/css">
      p.special { color: rgb(230, 100,
180) }
</STYLE>
```

Каскады, зависящие от устройств

Каскад может включать таблицы стилей, применяемые к различным устройствам. Элементы **LINK** и **STYLE** могут использоваться с атрибутом media. Браузер несет ответственность за отфильтровывание таблиц стилей, не применяющихся к текущему устройству.

В следующем примере мы определяем каскад, в котором таблица стилей «**corporate**» представляется в нескольких версиях: одна для печати, другая для экранного представления, третья для речевых браузеров (полезная, например, при чтении электронной почты в машине). Таблица «**techreport**» применяется ко всем устройствам. Цветная **rule**, определяемая элементом **STYLE**, используется для печати и для экрана, но не для звукового представления.

```
<LINK rel="stylesheet" media="aural"
href="corporate-aural.css" type="text/css">
<LINK rel="stylesheet" media="screen"
href="corporate-screen.css" type="text/css">
<LINK rel="stylesheet" media="print"
href="corporate-print.css" type="text/css">
<LINK rel="stylesheet"
href="techreport.css" type="text/css">
<STYLE type="text/css">
      p.special { color: rgb(230, 100,
180) }
</STYLE>
```

Наследование и каскады

Если браузер собирается представлять документ, ему необходимо найти значения для свойств стиля, например, семейство шрифтов, начертание, размер шрифта, длину строки, цвет текста и т.д.

Точный механизм зависит от языка таблиц стилей, но в общем применяется следующее: механизм раскодирования используется, если к элементу применяется ряд правил стиля. Этот механизм позволяет браузеру сортировать правила по специфичности и определять, какое правило нужно применить. Если правило невозможно найти, следующий шаг зависит от наследования свойства. Не все свойства могут наследоваться. Для этих свойств язык таблиц стилей обеспечивает

значения по умолчанию для использования в случае отсутствия явных правил для конкретного элемента.

Если свойство может наследоваться, браузер проверяет непосредственно элемент, в который вложен текущий элемент, и ищет правило, применяющееся к нему. Этот процесс продолжается до тех пора, пока применимое правило не будет обнаружено. Этот механизм обеспечивает компактное задание таблиц стилей. Например, авторы могут указать семейство шрифтов для всех элементов в разделе **BODY** с помощью одного правила для элемента **BODY**.

Как скрыть информацию о стиле от браузеров

Некоторые языки таблиц стилей поддерживают синтаксис, позволяющий авторам скрывать содержимое элементов **STYLE** от несоответствующих браузеров.

В данном примере для **CSS** показано, как можно скрыть содержимое элементов **STYLE**, чтобы гарантировать, что более старые несоответствующие данной версии браузеры не будут представлять их в виде текста.

```
<STYLE type="text/css">
<!--
     H1 { color: red }
     P  { color: blue}
     -->
</STYLE>
```

Привязка таблиц стилей с помощью заголовков HTTP

Менеджеры Web-серверов могут сконфигурировать сервер таким образом, чтобы таблица стилей применялась к группе страниц. Заголовок HTTP **Link** действует так же, как элемент **LINK**, с теми же атрибутами и значениями. Несколько заголовков **Link** соответствуют нескольким элементам **LINK** в том же порядке. Например,

```
Link: <http://www.acme.com/corporate.css>;
REL=stylesheet
```

соответствует:

```
<LINK rel="stylesheet"
href="http://www.acme.com/corporate.css">
```

Можно задать несколько альтернативных стилей с помощью нескольких заголовков **Link**, а затем использовать атрибут **rel** для определения стиля по умолчанию.

В следующем примере стиль «**compact**» применяется по умолчанию, поскольку в нем отсутствует ключевое слово «**alternate**» для атрибута **rel**.

```
Link: <compact.css>; rel="stylesheet";
title="compact"
Link: <bigprint.css>; rel="alternate
stylesheet"; title="big print"
```

Это работает и при отправке документов HTML по электронной почте. Некоторые браузеры электронной почты могут изменять порядок заголовков. Чтобы защитить стиль от изменения порядка каскадов для таблиц, задаваемых заголовками **Link**, авторы могут использовать объединение заголовков для объединения нескольких экземпляров одного и того же поля заголовка. Кавычки необходимы только в случае, если значения атрибутов включают пробелы. Используйте **SGML entities** для ссылок на символы, недопустимые в заголовках HTTP или электронной почты или символов, которые могут быть изменены при передаче через шлюзы.

Элементы **LINK** и **META** определяются как встреченные раньше явного элемента **LINK** и **META** в разделе **HEAD** документа.

Приложения

Самоучитель для начинающих WEB-дизайнеров

Создание документа в HTML

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

Пример документа в HTML

Любой гипертекст похож на книгу и может быть разбит на отдельные структурные элементы:

- собственно документ
- главы
- параграфы
- пункты
- подпункты
- абзацы.

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

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

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

Внимание! HTML не различает, какими буквами набраны символы форматирования: **<title>** равносильно **<TITLE>** или **<TiTlE>**. Не все стили поддержаны всеми

WWW-браузерами. Если браузер не поддерживает стиль, то он его игнорирует. (Поэтому не страшно, если вы уже сейчас начнете пользоваться при форматировании абзацев символом.

Основные элементы

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

Заголовки документов

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

Для выделения заголовка служат символы:

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

Заголовки разделов документов

HTML имеет шесть уровней заголовков, имеющих номера с 1 по 6 (заголовок первого уровня является заголовком высшего уровня). По сравнению с нормальным текстом, заголовки выделяются шрифтом — размером

и толщиной букв. Первый заголовок в каждом документе должен быть выделен **<H1>**. Синтаксис заголовков:

```
<Hx> Текст заголовка </Hx>
```

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

Абзацы

В отличие от документов в большинстве текстовых процессоров, прерывания строк и слов в HTML-файлах не существенны. Обрыв слова или строки может происходить в любом пункте в вашем исходном файле, при просмотре это прерывание будет проигнорировано. Напомним, что в нашем примере, первый параграф был представлен как

```
<H2>Параграф 1.</H2> Добро пожаловать в
HTML! Здесь мы расскажем, как надо и как
не надо писать гипертексты.
```

В исходном файле два предложения находятся на двух разных строках. Web браузер игнорирует это прерывание строки и начинает новый абзац только, после знака **<P>**. Однако, чтобы сохранить удобочитаемость в исходных HTML-файлах, мы рекомендуем заголовки размещать на отдельных строках, а абзацы отделять пустыми строками (в дополнение к **<P>**).

Также заметим, что хотя в действующих версиях HTML нет признака форматирования **</P>**, мы вам рекомендуем его употреблять, поскольку он может быть введен в последующих версиях. К ошибке это не приведет, поскольку все незнакомые символы браузер просто игнорирует. В противном случае, вам в последствии, может быть, придется переделывать ваши HTML-документы. В версии HTML+ предлагается, по аналогии с описанием заголовков, использовать как открывающий, так и закрывающий знаки абзаца.

Соединение с другими документами

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

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

Приведем пример такой гипертекстовой ссылки:

```
Пример HTML-текста
```


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

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

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

URL

URL это аббревиатура от Uniform Resource Locator. В него входит, кроме названия файла и директории: сетевой адрес машины и метод доступа к файлу. С помощью URL можно запускать удаленные программы, и передавать им значения. На этом принципе построены шлюзы в другие интернетовские сервиса: finger, archie, ... Здесь представлены несколько наиболее часто используемых типов URL. Допустим файл с именем

«online15.zip» лежит на ftp сервере ftp.simtel.ru в директории /pub/doc/services/, тогда URL этого файла будет выглядеть так:

```
file:// ftp. simtel. ru/ pub/ doc/
services/ online15.zip
```

URL директории в которой лежит файл:

```
file://ftp.simtel.ru/pub/doc/services/
```

а URL корневой директории ftp сервера ftp.simtel.ru выглядит вот так:

```
file://ftp.simtel.ru/
```

Gopher URL's не так разнообразны, как файловые. Это связано с ограниченностью этого сервиса. Для того что бы описать, например, gopher сервер узла gopher.kiae.su необходим URL:

```
gopher://gopher.kiae.su/
```

Некоторые gopher-сервера могут находиться на нестандартном номере порта (по умолчанию обычно используется 70 порт) тогда он должен указываться:

```
gopher://gopher.banzai.edu:1234/
```

где 1234 — номер порта.

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

1.

```
<A
Href="http://www.simtel.ru/news/snews.http">
News</A>
```

2.

```
<A  Href="aaa.html">AAA</A>
```

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

Обращение к определенным разделам других документов

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

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

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

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

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

Соединения с разделами текущего документа

Техника соединения аналогична описанной выше, только опускается имя файла. Вот, например, связь с Главой 1 внутри того же самого файла (Документ B) это:

```
<A HREF = "#Глава1">Глава 1</A> текущего
документа.
```

Дополнительные возможности форматирования

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

Списки

HTML поддерживает ненумерованные, нумерованные списки и списки определений.

- Ненумерованные списки

Ненумерованный список:

```
<UL><LI>список пунктов </UL>
```

например:

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

- Нумерованные списки

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

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

Браузер автоматически нумерует элементы такого списка.

- Списки определений

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

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

Вложенные списки

Списки могут быть произвольно вложены, хотя разумнее было бы практически

ограничиться тремя уровнями вложенных списков.

Приведем пример вложенных списков:

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

Авторский стиль редактирования

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

```
#!/bin/csh
cd $SCR
cfs get mysrc.f:mycfsdir/mysrc.f
```



```
cfs get myinfile:mycfsdir/myinfile
fc -O2 -o mya.out mysrc.f
mya.out
cfs save myoutfile:mycfsdir/myoutfile rm *
```

Адреса

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

Например, последняя строка этого документа выглядит:

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

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

Стили

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

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

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

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

Логические стили

<DFN> служит для описания определений.

<EM> служит для выделения слов.

<CITE> служит для выделения заголовков книг, фильмов, цитат и т.п.

<CODE> служит для выделения программных кодов, текстов программ и т.п. Изображается шрифтом фиксированной ширины.

<KBD> используется для ввода с клавиатуры пользователя. Может быть изображено жирным шрифтом (но в большинстве браузеров изображается специальным шрифтом).

<SAMP> используется для машинных сообщений. Изображается шрифтом фиксированной ширины.

<STRONG> служит для особого выделения слов. Обычно выделяется жирным шрифтом.

<VAR> используется для символьных переменных.

Физические стили

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

Вы можете задать:

<B>, **</B>** — жирный шрифт.

<I>, **</I>** — наклонный шрифт.

<TT>, **</TT>** — фиксированный шрифт (шрифт заданной ширины).

Специальные символы

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

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

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

Прерывание строки

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

Например:

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

даст на экране:

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

Горизонтальная линия

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

Внутренние рисунки

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

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

Чтобы включить рисунок, надо описать гиперссылку на него:

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

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

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

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

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

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

Также возможны типы выравнивания:

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

Если браузер не поддерживает рисунки

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

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

В этом случае пользователь увидит только текст «logo.gif».

Внешние рисунки, звуки и мультипликация

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

Пример HTML-документа

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

```
текста</A>.<P>
<H2>Заголовок второго уровня</H2>
Дальнейший текст будет написан шрифтом
фиксированной ширины: <P>
<PRE> On the stiff twig up there Hunches
a wet black rook Arranging and
rearranging its feathers in the rain ...
</PRE>
Это ненумерованный список, состоящий из
двух элементов:
<P>
<UL>
<LI> смородина
<LI> черника
</UL>
Конец документа. <P>
<ADDRESS>Dmitry Tsarik
(DIMON@ts.dpg.msu.au)</ADDRESS> </DD>
</BODY>
```

Cookies

Cookie является решением одной из наследственных проблем HTTP спецификации. Эта проблема заключается в непостоянстве соединения между клиентом и сервером, как при FTP или Telnet сессии, т.е. для каждого документа (или файла) при передаче по HTTP протоколу посылается

отдельный запрос. Включение cookie в HTTP протокол дало частичное решение этой проблемы.

Cookie это небольшая порция информации, которую сервер передает клиенту. Клиент (браузер) будет хранить эту информацию и передавать ее серверу с каждым запросом как часть HTTP заголовка. Некоторые cookie хранятся только в течение одной сессии, они удаляются после закрытия браузера. Другие, установленные на некоторый период времени, записываются в файл. Обычно этот файл называется cookie.txt.

Что можно делать с помощью cookie?

Сами по себе cookies не могут делать ничего, это только лишь некоторая информация. Однако, сервер может на содержащуюся в cookies информацию. Например, в случае авторизованного доступа к чему либо через WWW, в cookies сохраняется **login** и **password** в течение сессии, что позволяет не вводить их при запросе каждого запаролированного документа. Другой пример: cookies могут использоваться для построения персонализированных страниц. Чаще всего встречается такое — на некотором сервере вас просят ввести свое

имя, и каждый раз, когда вы заходите на первую страницу этого сервера, вам пишут что-то типа «Hello, your_name!». На использовании cookies также часто строят функцию оформления заказа в онлайновых магазинах, в частности, в Амазоне, такая своеобразная виртуальная корзина покупателя, как в обычном реальном супермаркете.

Какие браузеры поддерживают механизм cookies?

Не все, конечно, однако самые популярные поддерживают: Netscape (начиная с самой первой версии), Microsoft IE, Mosaic.

Установка cookie

Как выставлять cookies клиенту зависит от того, как они будут использоваться в дальнейшем. Это можно делать как с помощью скриптов, так и с помощью META-тэгов HTML. Можно манипулировать временем жизни выставленных cookies и устанавливать место, в котором установки действительны. Общий формат установки таков:

```
Set-Cookie: NAME=value; EXPIRES=date;
DOMAIN=domain_name; PATH=path; SECURE
```


Установка cookie с помощью HTML

Простейший способ выставить cookie — использовать соответствующий META-тэг в заголовке **<HEAD> </HEAD>** любого статического HTML документа. Это выглядит следующим образом:

```
<META HTTP-EQUIV="Set-Cookie"
CONTENT="NAME=value; EXPIRES=date;
DOMAIN=domain_name; PATH=path; SECURE">
```

Установка cookie с использованием Perl/CGI

Другой способ выставить cookie — с помощью серверного скрипта. На Perl это будет выглядеть примерно следующим образом: перед тем как выдавать серверный ответ генерируется HTTP заголовок

```
print "Content-type: text/html\n";
print "Set-Cookie: username=aaa13;
expires=Friday,31-Dec-99 23:59:59
GMT; path=/win/internet/html/;
domain=citforum.ru;\n\n";
```

Чтобы прочитать скриптом значение cookie, которое было установлено ранее, и соответствующим образом выполнить скрипт, используется переменная окружения HTTP_COOKIE. На Perl это будет выглядеть так:

```
$cookie = $ENV{'HTTP_COOKIE'};
```

При использовании SSI для просмотра значения cookie можно применить директиву:

```
<!--#echo var="HTTP_COOKIE"-->
```

Установка нескольких cookie одновременно

Как с помощью HTML, так и с помощью скриптов можно устанавливать несколько cookie разом:

HTML:

```
<META HTTP-EQUIV="Set-Cookie"
CONTENT="NAME=value; EXPIRES=date;
  DOMAIN=domain_name; PATH=path; SECURE">
  <META HTTP-EQUIV="Set-Cookie"
CONTENT="NAME=value; EXPIRES=date;
DOMAIN=domain_name; PATH=path; SECURE">
```

Perl/CGI:

```
print "Content-type: text/html\n";
print "Set-Cookie: NAME=value;
EXPIRES=date; PATH=path;
DOMAIN=domain_name; SECURE\n";
  print "Set-Cookie: NAME=value;
EXPIRES=date; PATH=path;
DOMAIN=domain_name; SECURE\n\n";
```

Формы в HTML документах

Некоторые WWW браузеры позволяют пользователю, заполнив специальную форму,

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

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

Синтаксис

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

```
<FORM METHOD="get|post" ACTION="URL">
Элементы_формы_и_другие_элементы_HTML
</FORM>
```

METHOD

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

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

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

ACTION

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

Тэги формы

TEXTAREA

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

```
<TEXTAREA NAME="address" ROWS=10
COLS=50>Москва, Дмитровское шоссе, д. 123,
офис 448 </TEXTAREA>
```

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

- **NAME** — имя поля ввода
- **ROWS** — высота поля ввода в символах
- **COLS** — ширина поля ввода в символах

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

INPUT

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

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

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

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

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

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

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

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

- **CHECKBOX**

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

- **HIDDEN**

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

- **IMAGE**

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

- **PASSWORD**

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

- **RADIO**

Данный атрибут позволяет вводить одно значение из нескольких альтернатив. Для

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

- **RESET**

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

- **SUBMIT**

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

- **TEXT**

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

Меню выбора в формах

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

- **Select** — пользователь выбирает одно значение из фиксированного списка значений, представленных тэгами **OPTION**. Данный вид представляется как выпадающий **LISTBOX**.
- **Select single** — то же самое, что и **Select**, но на экране пользователь видит одновременно три элемента выбора. Если их больше, то предоставляется автоматический вертикальный скроллинг.
- **Select multiple** — позволяет выбрать несколько элементов из **LISTBOX**.

SELECT

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

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

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

SELECT SINGLE

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

```
<FORM>
<SELECT  SINGLE  NAME=group  SIZE=4>
```

```
<OPTION> AT 386
<OPTION> AT 486
<OPTION> AT 586
<OPTIONS> Pentium PRO
</SELECT>
</FORM>
```

SELECT MULTIPLE

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

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

Если выбрано одновременно несколько значений, то серверу передаются соответствующее выбранному количество

параметров **NAME=VALUE** с одинаковыми значениями **NAME**, но разными **VALUE**.

Отправление файлов при помощи форм

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

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

Например:

```
<FORM ENCTYPE="multipart/form-data"
ACTION="url" METHOD=POST>
Отправить данный файл: <INPUT
NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Отправить
файл">
</FORM>

<FORM
action="http://pandemonium.cs.nstu.ru/~gun/d
ocs/sites.htm" encType="multipart/form-data"
method="post">
Отправить данный файл: <INPUT
name="userfile" type="file">
<INPUT type="submit" value="Отправить
```

```
файл">
</FORM>
```

Фреймы в HTML документах

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

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

- размещение статической информации, которую автор считает необходимым постоянно показывать пользователю, в одном статическом фрэйме. Это может быть графический логотип фирмы, copyright, набор управляющих кнопок.

- помещение в статическом фрэйме оглавления всех или части WEB-документов, содержащихся на WEB-сервере, что позволяет пользователю быстро находить интересующую его информацию.
- создавать окна результатов запросов, когда в одном фрэйме находится собственно запрос, а в другом результаты запроса.
- создавать формы типа «мастер-деталь» для WEB-приложений, обслуживающих базы данных.

Синтаксис фрэймов

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

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

Однако, фрэйм-документ является специфичным видом HTML-документа, поскольку не содержит элемента **BODY** и

какой-либо информационной нагрузки соответственно. Он описывает только фрэймы, которые будут содержать информацию (кроме случая двойного документа).

Представим общий синтаксис фрэймов:

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

Общий контейнер **FRAMESET** описывает все фрэймы, на которые делится экран. Вы можете разделить экран на несколько вертикальных или несколько горизонтальных фрэймов. Тэг **FRAME** пописывает каждый фрэйм в отдельности. Рассмотрим более детально каждый компонент.

FRAMESET

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

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

поддерживающих фрэймы и не поддерживающих фрэймы.

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

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

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

Синтаксис используемых видов описания величин подокон:

```
value
```

Простое числовое значение определяет фиксированную высоту подокна в пикселах. Это далеко не самый лучший способ описания высоты подокна, поскольку различные браузеры имеют различный размер

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

```
value%
```

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

```
value*
```

Вообще говоря, значение **value** в данном описании является необязательным. Символ «*****» указывает на то, что все оставшееся место будет принадлежать данному фрэйму. Если указывается два или более фрэйма с описанием «*****» (например «***,***»), то оставшееся пространство делится поровну между этими фрэймами. Если перед звездочкой стоит цифра, то она указывает пропорцию для данного фрэйма (во сколько раз он будет больше аналогично описанного чистой звездочкой). Например, описание

«**3*,*,***», говорит, что будет создано три фрэйма с размерами 3/5 свободного пространства для первого фрэйма и по 1/5 для двух других.

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

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

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

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

может привести к ошибочной ситуации.

Примеры:

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

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

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

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

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

аналогично предыдущему примеру.

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

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

FRAME

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

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

```
SRC="url"
```

Описывает URL документа, который будет отображен внутри данного фрэйма. Если он отсутствует, то будет отображен пустой фрэйм.

```
NAME="frame_name"
```

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


```
MARGINWIDTH="value"
```

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

```
MARGINHEIGHT="value"
```

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

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

Этот атрибут позволяет задавать наличие полос прокрутки у фрэйма. Параметр **yes** указывает, что полосы прокрутки будут в любом случае присутствовать у фрэйма, параметр **no** наоборот, что полос прокрутки не будет. **Auto** определяет наличие полос прокрутки только при их необходимости (значение по умолчанию).

```
NORESIZE
```

Данный атрибут позволяет создавать фрэймы без возможности изменения размеров. По умолчанию, размер фрэйма можно изменить при помощи мыши так же просто, как и размер окна Windows.

NORESIZE отменяет данную возможность. Если у одного фрэйма установлен атрибут **NORESIZE**, то у соседних фрэймов тоже не может быть изменен размер со стороны данного.

NOFRAMES

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

Рассмотрим реализацию фрэймов для подобного разбиения окна:

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



```
<FRAME SRC="link3.html">
<FRAME SRC="link4.html">
<FRAME SRC="link5.html">
</FRAMESET>
</FRAMESET>
```

Планирование фрэймов и взаимодействия между фрэймами

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

Ответом на данный вопрос является планирование взаимодействия фрэймов. Каждый фрэйм может иметь собственное имя, определяемое параметром **NAME** при описании данного фрэйма. Существует, также, специальный атрибут — **TARGET**, позволяющий определять, к какому фрэйму относится та или иная операция. Формат данного атрибута следующий:

```
TARGET="windows_name"
```

Данный атрибут может встречаться внутри различных тэгов:

TARGET в тэге A

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

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

TARGET в тэге BASE

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

```
Документ №1.
<FRAMESET ROWS="20,*">
<FRAME SRC="doc2.htm" NAME="Frame1">
<FRAME SRC="doc3.htm" NAME="Frame2">
</FRAMESET>
Документ №2
<HTML>
<HEAD>
<BASE TARGET="Frame2">
</HEAD>
```



```
<BODY>
<A HREF="url1"> Первая часть</A>
<A HREF="url2"> Вторая часть</A>
</BODY>
</HTML>
```

TARGET в тэге AREA

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

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

TARGET в тэге FORM

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

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

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

Зарезервированные имена фрэймов

Зарезервированные имена фрэймов служат для разрешения специальных ситуаций. Все

они начинаются со знака подчеркивания. Любые другие имена фрэймов, начинающиеся с подчеркивания будут игнорироваться браузером.

`TARGET="_blank"`

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

`TARGET="_self"`

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

`TARGET="_parent"`

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

`TARGET="_top"`

Данное значение определяет, что документ, полученный по ссылке будет отображаться на всей поверхности окна, вне зависимости от наличия фрэймов.

Использование данного параметра удобно в случае вложенных фрэймов.

CGI

Большое количество World Wide Web приложений основано на использовании внешних программ, управляемых Web сервером. Использование данных программ позволяет строить Web приложения с динамически обновляемой информацией, хранящейся в базах данных или генерирующейся в зависимости от бизнес-правил решаемых задач. Для связи между Web сервером и вызываемыми программами широко используется Common Gateway Interface (CGI), имеющий реализации как для Windows-ориентированных программ, так и для приложений, функционирующих в среде Unix. Ниже будет описана Windows-модификация интерфейса CGI, иначе называемая Windows CGI интерфейсом.

Разбор данных HTML-форм

Windows CGI требует, чтобы Web сервер декодировал данные из HTML форм, если они переданы при помощи **POST** метода запроса. Он не требует от сервера декодирования параметров, если они

переданы в качестве строки запроса («query string»), являющейся частью URL.

Существует два способа, которыми данные из форм могут быть переданы серверу браузером:

URL-Encoded

Это наиболее используемый формат данных, передаваемых из форм. Содержимое полей формы выделяются из формы и передаются согласно спецификации HTML 4.0, а затем собираются в одну строку, где отделяются друг от друга символом амперсанда. Тип содержания сообщения устанавливается браузером в application/ x — www — form — urlencoded.

Multipart Form Data

Данный формат разработан для эффективной загрузки файлов на сервер с использованием форм. Содержимое полей формы передается как многостраничное **MIME** сообщение. Каждое поле содержится в одной странице. Тип содержания, устанавливается браузером в multipart/ form — data.

«Грамотные» серверы должны уметь обрабатывать оба типа данных из форм.

Вызов CGI программ

Сервер использует функцию **CreateProcess()** для вызова CGI программ. Сервер синхронизируется с CGI программой, поскольку он должен определить момент завершения CGI программы. Это достигается использованием функции Win32 **WaitForSingleObject()**, ожидающей получения сигнала завершения CGI программы.

Командная строка

Сервер должен вызывать CGI программу, выполняя функцию **CreateProcess()** с командной строкой следующего формата:

```
WinCGI-exe  cgi-data-file
```

Где **WinCGI-exe** — полный путь к исполняемой CGI программе. Сервер не зависит от «текущего каталога» или переменной окружения **PATH**. Примите к сведению, что «исполняемая» не обязательно означает .EXE файл. Это может быть документ, ассоциирующийся с реально исполняемой программой, описанной в WIN.INI или System Registry.

cgi-data-file — полный путь к CGI файлу данных.

Метод вызова

Сервер использует **CreateProcess()** для запуска процесса, не имеющего главного окна. Вызванный процесс не будет отображаться каким либо образом на мониторе сервера.

Некоторые сервера поддерживают режим отладки CGI программ и скриптов, что позволяет серверу запускать CGI программу как обычный процесс с созданием главного окна и отображением информации на мониторе сервера. Данный способ весьма удобен на стадии отладки CGI программ.

CGI файл данных

Сервер передает данные CGI программам через Windows «private profile» afqk, в формате «параметр-значение» (windows INI файл). CGI программа может прочитать данный файл и получит все данные, передаваемые ей из формы, а также автоматически генерируемые браузером данные.

CGI файл данных состоит из следующих секций:

- CGI
- Accept
- System
- Extra Headers

- Form Literal
- Form External
- Form Huge
- Form File

Секция [CGI]

Данная секция содержит большинство специфических CGI параметров (тип доступа, тип запроса, дополнительные заголовки, определенные в других секциях и т.п.). Каждое значение представлено в виде символьной строки. Если значение является пустой строкой, значит данный параметр был опущен. Список параметров данной секции представлен ниже:

- **Request Protocol**

Название и модификация информационного протокола, использованного для передачи данного запроса. Формат:

```
протокол/модификация.
```

Пример:

```
"HTTP/4.0".
```

- **Request Method**

Метод, который использовался для данного запроса. Для HTTP это «**GET**», «**HEAD**», «**POST**» и т.д.

■ **Executable Path**

Логический путь к исполняемой CGI программе, необходимый для ссылки CGI программе на саму себя.

■ **Logical Path**

Запрос также может указывать к ресурсам, необходимым для выполнения данного запроса. Данный параметр содержит путь в том виде, который был получен сервером без мэпирования его на физический путь на диске.

■ **Physical Path**

Если запрос содержит информацию о логическом пути, сервер преобразует его к физическому пути (например, к пути к файлу на диске) доступа согласно синтаксическим правилам операционной системы.

■ **Query String**

Информация, размещающаяся после ? в URL вызываемой CGI программы. Сервер оставляет эту информацию без изменений в том виде, в котором она была помещена в URL.

■ **Request Range**

Byte-range спецификация, получаемая вместе с запросом (если есть). Сервер должен

поддерживать работу CGI программ в **byte-ranging**.

■ **Referer**

URL документа, содержащего ссылку на данную CGI программу. Надо заметить, что некоторые браузеры закрывают данную возможность и не дают ее использовать.

■ **From**

E-mail адрес пользователя браузера. Надо заметить, что данный параметр присутствует в спецификации HTTP, но не используется большинством браузеров из соображений секретности.

■ **User Agent**

Строка, описывающая программное обеспечение браузера. Не генерируется большинством браузеров.

■ **Content Type**

Данный параметр содержит MIME-тип данных, посланных клиентом вместе с полями из формы, если эти данные были посланы. Формат:

```
type/subtype.
```

■ **Content Length**

Для запросов, с которыми посланы дополнительные данные в это поле заносится длина посланных данных в байтах.

■ **Content File**

Для запросов, содержащих дополнительные данные, посланные пользователем, этот параметр содержит имя файла, в которое WEB-сервер записывает эти данные. В дальнейшем, пользовательская программа может считать эти данные. Параметр содержит полный путь к файлу данных.

■ **Server Software**

Название и версия серверного программного обеспечения, обработавшего запрос и вызвавшего CGI-программу. Формат:

```
name/version
```

■ **Server Name**

Сетевое имя сервера или псевдоним, необходимый для ссылающихся на себя URL Этот параметр (в комбинации с параметром **ServerPort**) может быть использован для вычисления полного URL к серверу.

■ **Server Port**

Номер порта, по которому работает сервер.

■ **Server Admin**

E-mail адрес администратора сервера. Данный параметр необходим для генерации сообщений об ошибках и отправки данных сообщений администратору сервера или для генерации форм с URL «mailto:».

■ **CGI Version**

Версия CGI. Формат:

```
CGI/версия.
```

Пример:

```
CGI/1.2 (Win).
```

■ **Remote Host**

Сетевое имя хоста клиента, если доступно. Данный параметр может быть использован для опознавание клиента.

■ **Remote Address**

Сетевой (IP) адрес клиента. Данный параметр может быть использован для проверки пользователя если отсутствует сетевое имя.

- **Authentication Method**

Если используется защищенный вызов CGI программы, это протокол-зависимый метод аутентификации, используемый для аутентификации пользователя.

- **Authentication Realm**

Если используется защищенный вызов CGI программы, это протокол-зависимый сервис, используемый для аутентификации пользователя. Список пользователей для полученного вида сервиса проверяется для аутентификации пользователя.

- **Authenticated Username**

Если используется защищенный вызов CGI программы, это имя пользователя, которое клиент использует для аутентификации при доступе к CGI-программе.

Секция [Accept]

Данная секция содержит типы данных, посылаемых клиентом, найденные в заголовке запроса в виде

```
Accept: type/subtype {parameters}
```

Если данные параметры присутствуют (например, «q=0.100») , они передаются как значения параметра **Accept**. Для каждого типа

передаваемых данных заводится свой параметр **Accept**.

Секция [System]

Данная секция содержит параметры, специфические для Windows реализации CGI:

- **GMT Offset**

Количество секунд, которое необходимо добавить к времени по Гринвичу для вычисления локального времени клиента.

- **Debug Mode**

Данный параметр имеет значение «Yes» если включен режим «CGI/script tracing» на сервере.

- **Output File**

Полный путь к файлу, в который необходимо поместить данные, отсылаемые сервером клиенту после завершения работы программы.

- **Content File**

Полный путь к файлу, в котором содержится дополнительная информация, поступающая вместе с запросом.

Секция [Extra Headers]

Данная секция содержит «дополнительные» заголовки, которые включены в запрос в виде

«параметр=значение». Сервер должен раскодировать как параметр, так и его значение прежде чем они будут помещены в файл данных CGI.

Секция [Form Literal]

Если запрос от клиента пришел в виде HTTP POST из HTML формы (с типом содержимого application/x-www-form-urlencoded или multipart/form-data), то сервер раскодирует данные из формы и поместит их в секцию **[Form Literal]**.

Для URL-кодированных данных формы, строка передаваемых параметров выглядит как

```
«параметр=значение&параметр=значение&...»
```

где значения находятся в url-кодированном формате. Сервер разделяет «параметр=значение» по символу &, затем разделяет собственно «параметр» и «значение», декодирует «значение» и помещает результат в виде

```
«параметр=раскодированное_значение»
```

в секцию **[Form Literal]**.

Для многостраничных данных строка данных представляется в многостраничном **MIME** формате, где каждое поле представлено как отдельная часть (файл). Сервер декодирует имена и значение каждой

части и размещает их в формате «параметр=значение» в секции **[Form Literal]**.

Если форма содержит какие-либо элементы **SELECT MULTIPLE**, то будет создано несколько строк с вида «параметр=значение» с одинаковым именем «параметра». В этом случае генерирует нормальную строку «параметр=значение» для первого встречающегося элемента, а каждый следующий представляет в виде

```
«параметр_X=значение»
```

где «X» — увеличивающийся счетчик.

Секция [Form External]

Если размер декодированной строки превышает 254 символа или декодированная строка содержит управляющие символы, такие, как перевод строки, возврат каретки, двойные кавычки и т.д., то сервер помещает данное значение в отдельный временный файл, а в секцию **[Form External]** помещает строку в виде:

```
параметр=путь  длина
```

где путь — это полный путь и имя временного файла, содержащего декодированное значение параметра, а длина — длина в байтах этого файла.

Секция [Form Huge]

Если общая длина строки с кодированными параметрами превышает 65,535 байт, то сервер не выполняет декодирование, а оставляет данные в **Content File**, а в секцию **[Form Huge]** помещает строки в виде:

```
параметр=смещение длина
```

где смещение — это смещение от начала **Content File** по которому находится требуемый параметр, а длина — длина в байтах значения выбранного параметра. Вы можете использовать смещение для выполнения поиска начала значения выбранного вами параметра и использовать длину для чтения значения выбранного параметра. Не забывайте, что если параметр закодирован, то вам необходимо раскодировать его перед использованием.

Секция [Form File]

Если запрос пришел в виде multipart/form-data, то он может содержать один или несколько загруженных с клиента файлов. В этом случае каждый загруженный файл размещается в специальном временном файле, а в секции **[Form File]** строки имеют тот же формат, что и в секции **[Form External]**. Каждая строка параметра в этом случае выглядит так:


```
параметр=[полный_путь_к_файлу] длина тип
ссылка [имя_файла]
```

где полный_путь_к_файлу — это путь к временному файлу, содержащему загруженный файл, длина — длина в байтах загруженного файла, тип — тип MIME загруженного файла, ссылка — способ кодировки загруженного файла и имя_файла — исходное название загруженного файла. Использование квадратных скобок обязательно, поскольку имя файла и путь могут содержать символы пробела.

Пример декодированных значений формы

В данном примере форма содержит небольшое поле, **SELECT MULTIPLE** с 2-мя небольшими секциями, поле длиной 300 символов, поле, содержащее специальные символы и поле длиной 230KB.

```
[Form Literal]
smallfield=123 Main St. #122
multiple=first selection
multiple_1=second selection
[Form External]
field300chars=C:\TEMP\HS19AF6C.000 300
fieldwithlinebreaks=C:\TEMP\HS19AF6C.001 43
```

```
[Form Huge]
field230K=C:\TEMP\HS19AF6C.002 276920
```

Обработка результата

CGI программа возвращает результат работы, отвечающий (явно или неявно) целям запроса. Сервер кодирует результат работы в соответствии со стандартом HTTP и использует HTTP для отправки результата клиенту. Это означает, что сервер добавляет необходимый HTTP заголовки в сообщение, формируемое CGI программой.

Результат работы CGI программы состоит из двух частей: заголовка и тела сообщения. Заголовок состоит из одной или более строк текста, отделенных от тела пустой строкой. Тело сообщения содержит данные, представленные в MIME формате, указанном в заголовке.

Сервер не изменяет тело документа, что означает, что сервер передает сформированный CGI программой ответ «как он есть».

Специальные строки заголовка

Сервер распознает следующие строки заголовка в выходном потоке:

Content-Type:

Указывает на MIME тип тела сообщения. Значение этого параметра должно быть в формате **type/subtype**.

URI: <value> (value enclosed in angle brackets)

Данное значение указывает на полный URL или ссылку на локальный файл, сообщение из которого будет возвращено клиенту в теле сообщения. Если значение является локальным файлом, сервер отсылает его как результат запроса, как будто клиент воспользовался методом **GET** при генерации запроса. Если значение является полным URL, то сервер возвращает сообщение «**401 redirect**» для обеспечения прямой загрузки указанного объекта.

Location:

То же самое, что и **URI**, но данная форма сейчас не используется. Параметр **value** не должен быть взят в угловые скобки.

Другие заголовки

Другие заголовки передаются клиенту в том виде, в котором они представлены.

Прямой возврат

Сервер позволяет конечному приложению осуществлять прямой возврат результата запроса клиенту. Это осуществляется

посредством включение в заголовок возвращаемого сообщения его информационного протокола. Это позволяет CGI программам формировать непосредственный ответ клиенту с указанием HTTP заголовка без предварительной обработки его сервером.

Сервер анализирует результат запроса, помещаемый CGI программой в выходной файл (Output File), и, если первая строка «**HTTP/4.0**», он предполагает, что сообщение содержит полный HTTP ответ и отсылает его клиенту без упаковки.

VRML

Язык VRML (Virtual Realty Modelling Languagy) предназначен для описания трехмерных изображений и оперирует объектами, описывающими геометрические фигуры и их расположение в пространстве.

Vrml-файл представляет собой обычный текстовый файл, интерпретируемый браузером. Поскольку большинство браузеров не имеет встроенных средств поддержки vrml, для просмотра Vrml-документов необходимо подключить вспомогательную программу — Vrml-браузер, например, Live3D или CosmoPlayer.

Как и в случае с HTML, один и тот же vrml-документ может выглядеть по-разному в разных VRML-браузерах. Кроме того, многие разработчики VRML-браузеров добавляют нестандартные расширения VRML в свой браузер.

Существует немало VRML-редакторов, делающих удобней и быстрее процесс создания Vrml-документов, однако несложные модели, рассматриваемые в данной главе, можно создать при помощи самого простого текстового редактора.

Единицы измерения

В VRML приняты следующие единицы измерения:

- Расстояние и размер: метры
- Углы: радианы
- Остальные значения: выражаются, как часть от 1.

Координаты берутся в трехмерной декартовой системе координат.

Заголовок VRML-файла

Vrml-документ представляет собой обычный тестовый файл. Для того, чтобы VRML-браузер распознал файл с

VRML-кодом, в начале файла ставится специальный заголовок — **file header**:

```
#VRML V1.0 ascii
```

Такой заголовок обязательно должен находиться в первой строке файла, кроме того, перед знаком диеза не должно быть пробелов.

Примитивы VRML

В VRML определены четыре базовые фигуры: куб (скорее не куб, а прямоугольный параллелепипед), сфера, цилиндр и конус.

Эти фигуры называются примитивами (primitives). Набор примитивов невелик, однако комбинируя их, можно строить достаточно сложные трехмерные изображения.

Рассмотрим поподробней каждый из примитивов.

Куб

Возможные параметры: **width** — ширина, **height** — высота, **depth** — глубина.

```
Cube {
        width     2       # ширина
        height    3       # высота
        depth     1       # глубина
      }
```


Сфера

Параметр у сферы только один, это **radius**.

```
Sphere {
radius   1      # радиус
       }
```

Конус

Возможные параметры: **bottomRadius** — радиус основания, **height** — высота, **parts** — определяет, какие части конуса будут видны. Параметр **parts** может принимать значения **ALL**, **SIDES** или **BOTTOM**.

```
Cone {
parts                ALL
#отображаемые  поверхности
bottomRadius      1            #радиус
основания
height            2            #высота
      }
```

Цилиндр

Для цилиндра можно задать параметры **radius** и **height**. Кроме того, с помощью параметра **parts** для цилиндра можно определить будут ли отображаться основания цилиндра и его боковая поверхность. Параметр **parts** может принимать значения **ALL**, **SIDES**, **BOTTOM** или **TOP**.

```
Cylinder {
parts     ALL   #видны  все  части  цилиндра
```

```
radius   1            #радиус  основания
height   2            #высота  цилиндра
      }
```

Цвет и текстура

Цвет фигуры, определяется с помощью объекта **Material**.

```
Material  {
ambientColor      0.2  0.2  0.2
diffuseColor      0.8  0.8  0.8
specularColor     0  0  0
emissiveColor     0  0  0
transparency      0
          }
```

Параметры **ambientColor**, **diffuseColor**, **specularColor** и **emissiveColor** управляют цветами и указываются в палитре RGB (красный, зеленый и голубой), причем первая цифра определяет интенсивность красного цвета, вторая — зеленого, а третья — синего.

К примеру, синий кубик, может быть описан следующим образом:

```
#VRML  V1.0  ascii
Material  {
diffuseColor  0  0  1
         }
Cube  {}
```

Параметр **transparency** может принимать значения от 0 до 1 и определяет степень

прозрачности, причем максимальная прозрачность достигается при **transparency** равном единице. В приведенном примере описано два цилиндра разных размеров, меньший из которых просвечивает сквозь другой.

```
#VRML  V1.0  ascii
       Material  {
       diffuseColor  0  0  1
       transparency      0.7
                 }
       Cylinder  {
         height   1
         radius   1
                 }
       Material  {
       emissiveColor      1  0  0
       transparency      0
                  }
       Cylinder  {
         height   0.8
         radius   0.1
                  }
```

Для имитирования различных поверхностей в VRML существует объект **Texture2**.

В качестве текстуры легче всего использовать обычный графический файл, например, в GIF-формате. В таком случае

для «натягивания» текстуры на трехмерное изображение нужно только указать путь к файлу в параметре **filename** объекта **Texture2**.

```
.#VRML V1.0 ascii
Texture2 {
filename      "krp.gif"
image           0 0 0
   wrapS REPEAT
   wrapT REPEAT
         }
   Cube {
     width 1
     height 1
     depth 1
         }
```

Параметры **wrapS** и **wrapT** могут принимать значения **REPEAT** или **CLAMP**, и управляют натягиванием текстуры по соответственно горизонтальной и вертикальной осям.

Положение объектов в пространстве

Изменение координат

По умолчанию любой описанный нами объект будет располагаться точно по центру окна браузера. По этой причине, если мы опишем к примеру два одинаковых цилиндра, они сольются друг с другом. Для

того, чтобы изменить положение второго цилиндра, применим узел **Translation**.

Узел **Translation** определяет координаты объекта:

```
Translation {
translation 1 2 3
#т.е. соответственно x=1 y=2 z=3
            }
```

Вообще говоря, координаты указываемые в **Translation** не являются абсолютными. Фактически это координаты относительно предыдущего узла **Translation**. Чтобы прояснить это вопрос, рассмотрим пример:

```
#VRML V1.0 ascii
   Cube {
     width 1
     height 1
     depth 1
         }
# Этот куб по умолчанию располагается в
центре
  Translation {
    translation 2 0 0
              }
#Второй куб сдвинут вправо на 2
  Cube {
    width 1
    height 1
    depth 1
```

```
      }
  Translation {
  translation 2 0 0
              }
#Третий куб сдвинут вправо на два
относительно 2-го !!!!
  Cube {
    width 1
    height 1
    depth 1
        }
```

Как видите, третий кубик вовсе не совпадает с первым, хотя в в узле **Translation** указаны те же координаты.

В VRML 1.0 принято следующее правило: узлы, модифицирующие свойства фигур (**Translation**, **Material** и т.п.), действуют на все далее описанные фигуры.

Чтобы ограничить область действия модифицирующих узлов, фигуры необходимо сгруппировать с помощью узла **Separator**.

```
Separator
{
   другие узлы
}
```

Узел **Separator** работает как контейнер, он может содержать любые другие узлы, и основным его предназначением является

именно ограничение области действия узлов типа **Translation** и **Material**.

Сравните следующий пример с предыдущим:

```
#VRML V1.0 ascii
    Separator {
              Cube {
                      width 1
                      height 1
                      depth 1
                      }
  }# конец области действия узла Separator
        Separator {
                Translation {
                        translation 2 0 0
                                    }
      #Второй куб сдвинут вправо на 2
                 Cube {
                        width 1
                        height 1
                        depth 1
                        }
  }# конец области действия узла Separator
          Separator {
                   Translation {
                       translation 2 0 0
                                   }
#Третий куб сдвинут вправо на два
относительно 1-го.
```

```
                    Cube {
                          width 1
                          height 1
                          depth 1
                             }
}# конец области действия узла Separator
```

Хотя в примере описано три кубика, мы видим только два, так как второй и третий совпадают.

Вообще говоря рекомендуется всегда и везде использовать узел **Separator**. Он не только избавит от ошибок, связанных с относительностью координат, но и сделает VRML-код более простым и понятным.

Вращение

Для вращения фигур вокруг осей координат применяется узел **Rotation**.

```
Rotation {
 rotation 0 1 0   1.57
          }
```

Первые три цифры определяет будет ли осуществлен поворот вокруг соответственно осей x, y и z, а четвертая задает угол вращения в радианах. В приведенном выше листинге поворот осуществляется вокруг оси y на 90 градусов.

Углы в градусах — радианах

30° — 0.52

45° — 0.78

60° — 1.04

90° — 1.57

180° — 3.14

270° — 4.71

Составим букву T из двух цилиндров. По умолчанию цилиндр ориентирован вертикально. Поэтому для успешного выполнения задачи повернем его вокруг оси z на 90 градусов.

```
#VRML V1.0 ascii
 Separator { #Красный цилиндр
 Material { emissiveColor    1 0.6 0.6 }
       Cylinder {
          height      1
          radius      0.3
                }
          }
Separator { # Синий цилиндр, повернутый
на 90 градусов вокруг оси z
 Translation {
  translation 0 0.5 0
             }
 Rotation {
    rotation      0 0 1   1.57
          }
```

```
Material { emissiveColor     0.5 0.5 1 }
  Cylinder {
          height       1
          radius       0.3
            }
      }
```

Масштабирование

Узел **Scale** масштабирует фигуры по одному или нескольким измерениям. Три цифры, стоящие после параметра **scaleFactor** определяют коэффициенты масштабирования относительно осей x, y и z.

```
Scale {
  scaleFactor 1 1 1
        }
```

В следующем примере узел **Scale** сжимает сферу по оси x, и из сферы получается эллипсоид.

```
#VRML V1.0 ascii
  Material { emissiveColor     1 1 0 }
    Scale {
scaleFactor 0.7 1 1 #сжимаем сферу по
оси x
          }
Sphere { radius 1}
           }
```


Определение собственных объектов

VRML предоставляет прекрасную возможность сократить и сделать более понятным исходный код VRML-файла путем описания собственных объектов. Это значит, что если в изображении несколько раз повторяется одна и та же фигура, то ее можно описать всего лишь один раз и в дальнейшем только ссылаться на нее.

Объект описывается одним из способов:

DEF name

```
Cube {}
```

или

DEF name

```
Material {}
```

или

DEF name

```
Separator {
Сгруппированные узлы, описывающие фигуру и
свойства материала
              }
```

Для того, чтобы вставить в VRML-файл ранее определенную фигуру, используется команда **USE**

```
Separator {
            USE name
          }
```

Создадим VRML-файл, описывающий стул, при этом ножку стула опишем как объект LEG:

```
#VRML V1.0 ascii
 Material { emissiveColor 1 0.5 0.5 }
 Separator {
 Translation { translation 1 1 1 }
DEF LEG   #Определяем объект - ножку
стула
Separator { # leg
   Cylinder {
      height   0.8
      radius   0.1
            }
          } # определили ножку
          }
 Separator {
 Translation { translation 0 1 1 }
 USE LEG # используем определенный объект
           }
 Separator { # еще одна ножка
 Translation { translation 1 1 0 }
 USE LEG
           }
 Separator { # последняя ножка
 Translation { translation 0 1 0 }
 USE LEG
           }
```



```
Separator { # сиденье
Translation { translation 0.49 1.5 0.5 }
   Cube {
     height 0.2
     width 1.2
     depth 1.2
          }
           }
Separator { # спинка
Translation { translation 0.49 2 0 }
   Cube {
     height 0.8
     width 1.2
     depth 0.2
          }
           }
Separator { # закругление спинки
Translation { translation 0.49 2.1 0 }
Rotation {
  rotation      1 0 0 1.57
          }
  Cylinder {
     radius 0.6
     height 0.2
           }
          }
```

Как видите, нам не понадобилось описывать каждую ножку в отдельности — в

результате объем VRML-кода стал меньше, а сам код более читабельным.

Еще один способ уменьшить размер VRML-файла — вставлять фигуры из другого файла.

Это позволяет делать узел WWWInline:

```
#VRML V1.0 ascii
      Separator {
            WWWInline {
               name              ""
               bboxSize       0  0  0
               bboxCenter    0  0  0
                      }
            }
```

Параметр **name** — это путь к файлу, параметры **bboxSize** и **bboxCenter** не обязательны и показывают пользователю размеры и положение вставляемого объекта, пока объект подгружается.

Хочется обратить ваше внимание на две особенности VRML, незнание которых сильно затруднит создание VRML-документов вручную.

1. Все описания узлов и параметров в VRML регистрозависимы. Если вы используете буквы неправильного регистра — то VRML-браузер просто проигнорирует такое описание.

2. В VRML имеет огромное значение порядок описания узлов.

Конвертирование HTML в удобочитаемый текст

Любой более или менее продвинутый обозреватель Internet способен сохранять страницу в виде текстового файла.

Оценка длительности воспроизведения гипертекстовой страницы

Длительность воспроизведения гипертекстовой страницы зависит от скорости вашего модема. Обычно модем со скоростью 14,4 Кбит/с загружает килобайт данных около секунды.

Хороший и несложный редактор для домашних страничек

Редактор страниц Web — программа Microsoft FrontPage Express позволяет использовать все возможности, которыми располагает HTML.

Скорость перерисовки картинки

Любое изображение, которое вы видите на экране отображается с использованием цветовой палитры Windows. Скорость перерисовки картинки зависит от точности

изображения на экране. Цвет или градации серого при печати передаются в соответствии с информацией графического файла. При уменьшении растра картинки — тонкие линии, узоры, линии сетки, некоторые фрагменты могут исчезнуть на экране монитора, но появиться на распечатке с высоким разрешением.

Зачем вам нужен DirectX

Стандарт DirectX является программным интерфейсом, прежде всего позволяющим вашей звуковой плате более полно общаться с системой. Имея DirectX, вы сможете заставить Windows 98 более ускоренно воспроизводить звуковые файлы и использовать большее количество аудиоканалов в играх и профессиональных системах обработки звука.

Зачем вам нужен ActiveX

Стандарт ActiveX является программным интерфейсом, который связывает тот или иной web-ресурс с операционной системой Windows.

KOI8 и как с ней бороться

Стандарт KOI8 является кириллической кодовой таблицей, позволяющей вам обмениваться некоторыми сообщениями

электронной почты и просматривать web-страницы. Для того, чтобы Winindows 98 понимала KOI8, необходимо установить многоязыковую поддержку и шрифты с кодировкой KOI8.

Кодировки российских серверов

Помните, что российские серверы Web поддерживают русский язык либо через кодовую страницу 1251 (Windows 95), либо через таблицу KOI8 (Unix).

Автоматическая подстановка адресов Internet

Откройте обозреватель Microsoft Internet Explorer. Поставив флажок **Использовать Автозавершение** (доступ: **Вид ➪ Свойства обозревателя ➪ Дополнительно**), вы можете использовать автоматическую подстановку адресов по мере их введения в поле **Адрес**.

Отображение адресов страниц URL в полном формате

Вы можете отображать в строке состояния адреса Internet или адреса страниц URL в полном формате. Откройте обозреватель Microsoft Internet Explorer, обратитесь к диалоговому окну **Свойства обозревателя**, перейдите к вкладке **Дополнительно** и поставьте флажок

Показывать краткие ссылки (доступ: **Вид ⇨ Свойства обозревателя ⇨ Дополнительно**).

Уменьшение вероятности сбоев других приложений

Для того, чтобы уменьшить вероятность сбоев других приложений, открывайте новую копию Internet Explorer при каждом его запуске. Для этого откройте обозреватель Microsoft Internet Explorer, обратитесь к диалоговому окну **Свойства обозревателя**, перейдите к вкладке **Дополнительно** и поставьте флажок **Использовать для просмотра новый процесс**.

Способы подчеркивания ссылок

Вы можете выбрать способ подчеркивания ссылок (подчеркивать все ссылки, не подчеркивать ссылки или подчеркивать, если указатель мыши находится на ссылке). Для этого откройте обозреватель Microsoft Internet Explorer, обратитесь к диалоговому окну **Свойства обозревателя**, перейдите к вкладке **Дополнительно** и активизируйте одну из опций раздела **Подчеркивание ссылок**.

Выключение графических изображений при отображении страниц

Если вы хотите ускорить вывод страниц Internet, откройте обозреватель Microsoft Internet Explorer, обратитесь к диалоговому окну **Свойства обозревателя**, перейдите к вкладке **Дополнительно** и снимите все флажки в разделе **Мультимедиа**.

Отправка и получение секретных сведений

Вы можете отправлять и получать секретные сведения по частному протоколу SSL3 (Secured Sockets Layer Level 3). Для этого откройте обозреватель Microsoft Internet Explorer, обратитесь к диалоговому окну **Свойства обозревателя**, перейдите к вкладке **Дополнительно** и активизируйте опцию **SSL 3.0** в разделе **Безопасность**.

Прямая обработка Cookies-файлов

Вы можете сообщить обозревателю, чтобы он обрабатывал все файлы с вашими личными данными без подтверждения. Для этого откройте обозреватель Microsoft Internet Explorer, обратитесь к диалоговому окну **Свойства обозревателя**, перейдите к вкладке **Дополнительно** и активизируйте опцию **Всегда принимать файлы cookie** в разделе **Безопасность**.

Включение фонового цвета и картинок в распечатываемую web-страницу

Вы можете включить фоновые цвета и картинки в распечатываемую web-страницу только одним способом. Для этого откройте обозреватель Microsoft Internet Explorer, обратитесь к диалоговому окну **Свойства обозревателя**, перейдите к вкладке **Дополнительно** и активизируйте опцию **Печатать цвета и рисунки фона** в разделе **Печать**.

Поиск адресов Internet с использованием других суффиксов

Вы можете вести поиск адресов Internet, используя заданные суффиксы. Для этого откройте обозреватель Microsoft Internet Explorer, обратитесь к диалоговому окну **Свойства обозревателя**, перейдите к вкладке **Дополнительно** и активизируйте опцию **Автопроверка общих корневых доменов** в разделе **Поиск**.

Различия между HTML 3.2 и HTML 4.0

Различия в элементах

Новые элементы

В HTML 4.0 введены следующие новые элементы:

- ABBR
- ACRONYM
- BDO
- BUTTON
- COLGROUP
- DEL
- FIELDSET
- FRAME
- FRAMESET
- IFRAME
- INS
- LABEL
- LEGEND
- NOFRAMES
- NOSCRIPT
- OBJECT
- OPTGROUP

- PARAM
- SPAN
- TBODY
- TFOOT
- THEAD
- Q.

Нежелательные элементы

Следующие элементы являются нежелательными:

- APPLET
- BASEFONT
- CENTER
- DIR
- FONT
- ISINDEX
- MENU
- S
- STRIKE
- U.

Устаревшие элементы

Следующие элементы являются устаревшими:

- LISTING

- PLAINTEXT
- XMP.

Вместо них авторам следует использовать элемент PRE.

Различия в атрибутах

Почти все атрибуты, определяющие представление документа HTML (цвета, выравнивание, шрифты, графика и т.д.) являются нежелательными, взамен рекомендуется использовать таблицы стилей. В списке атрибутов в приложении указано, какие атрибут являются нежелательными.

Атрибуты **id** и **class** позволяют авторам назначать элементам информацию об имени и классе для таблиц стилей, якорей, скриптов, объявления объектов, общей обработки документа и т.д.

Различия в доступности

В HTML 4.0 имеется множество изменений касательно доступности, в том числе:

- Атрибут **title** теперь можно устанавливать практически для каждого элемента.
- Авторы могут указывать длинные описания таблиц, изображений и фреймов (атрибут longdesc).

Различия в метаданных

Теперь авторы могут определять профили с описаниями о метаданных, заданных в элементах **META** или **LINK**.

Различия в тексте

Новые функции интернационализации позволяют авторам определять направление и язык текста.

Элементы **INS** и **DEL** позволяют размечать изменения в документах.

Элементы **ABBR** и **ACRONYM** позволяют размечать в документах сокращения и акронимы.

Различия в ссылках

Атрибут **id** позволяет сделать любой элемент целевым якорем ссылки.

Различия в таблицах

Модель таблиц HTML 4.0 происходит из работы над HTML+ и начального черновика HTML3.0. Прошлая по просьбам провайдеров информации модель расширена следующим образом:

- Авторы могут определять таблицы, которые будут отображаться последовательно по мере получения данных агентом пользователя.

- Авторы могут определять таблицы, более доступные пользователям с невизуальными агентами.
- Авторы могут определять таблицы с фиксированными заголовками и сносками. Агенты пользователей могут использовать это преимущество при прокрутке больших таблиц или при представлении на устройствах со страничной организацией.

Модель таблиц HTML 4.0 также обеспечивает необязательные настройки по умолчанию в зависимости от столбцов для выравнивания, большую гибкость при определении границ и обрамления таблицы и возможность выравнивания по определенным символам. Однако ожидается, что задачу представления таблиц в ближайшем будущем возьмут на себя таблицы стилей.

Кроме того, основной целью было обеспечение совместимости с широко используемой компанией Netscape реализацией таблиц. Другой целью было упрощение импортирования таблиц в соответствии с моделью SGML CALS. В последнем черновике атрибут **align** совместим с последними версиями наиболее популярных браузеров. Внесены некоторые разъяснения в

роль атрибута **dir** и рекомендуемое поведение в случае, если используются смешанные абсолютные и относительные ширины столбцов.

Введен новый элемент, **COLGROUP**, позволяющий группировать наборы столбцов с различными свойствами ширины и выравнивания, определяемыми одним или несколькими элементами **COL**. По сравнению с предыдущими черновиками, разъяснена семантика элемента **COLGROUP**, а **rules="basic"** заменено **rules="groups"**.

Атрибут **style** используется как средство расширения свойств, связанных с группами ячеек. Например, стиль линии: точечная, двойная, тонкая/толстая и т.д.; цвет/заливка для внутреннего содержимого; поля ячеек и информация о шрифтах. Это является темой спецификации таблиц стилей.

Атрибуты **frame** и **rules** изменены во избежание **SGML name clashes** друг с другом и во избежание **clashes** с атрибутами **align** и **valign**. Мотивом для этих изменений также послужило желание избежать проблем в будущем, если эта спецификация будет расширена и будет допускать атрибуты **frame** и **rules** в других элементах таблиц.

Различия в изображениях, объектах и изображениях-картах

Элемент **OBJECT** используется для общего включения объектов.

Элементы **IFRAME** и **OBJECT** позволяют авторам создавать внедренные документы.

Атрибут **alt** обязателен для элементов **IMG** и **AREA**.

Механизм создания изображений-карт теперь позволяет авторам создавать более доступные изображения-карты. Модель содержимого элемента **MAP** по этой причине изменилась.

Различия в формах

В этой спецификации вводится несколько новых атрибутов и элементов, относящихся к формам:

- Атрибут **accesskey** позволяет авторам определить прямой доступ с клавиатуры к управляющим элементам формы.
- Атрибут **disabled** позволяет авторам отключить управляющие элементы формы.
- Атрибут **readonly** позволяет авторам запретить изменение управляющих элементов формы.

- Элемент **LABEL** связывает метку с определенным управляющим элементом формы.
- Элемент **FIELDSET** группирует связанные поля вместе и, вместе с элементом **LEGEND**, может использоваться для присвоения имени группе. Оба эти новых элемента обеспечивают лучшее представление и интерактивность. Речевые браузеры могут лучше описывать формы, а графические браузеры могут сделать действующими метки.

Новый набор атрибутов вместе со скриптами позволяет провайдерам проверять ввод данных пользователем.

- Элементы **BUTTON** и **INPUT**, у которых для атрибута **type** установлено значение «button», могут использоваться вместе со href="../interact/scripts.html">скриптами для создания более разнообразных форм.
- Элемент **OPTGROUP** позволяет авторам группировать пункты меню в элементе **SELECT**, что особенно важно для доступности форм.

Советы по Web-дизайну

Путешествуя по Internet, вы, наверное, обратили внимание на различный вид web-страниц. Некоторые страницы выглядят весьма красиво и до отказа заполнены нужной вам информацией. Другие — безобразны, отвратительны и заполнены, как правило, всякой ерундой. Разработкой и тех и других страниц занимаются пользователи, называющие себя web-дизайнерами или web-мастерами. Понятно, что хороший web-дизайн напрямую зависит от того, как пользователь владеет интеллектуальной дисциплиной, называемой web-дизайном.

Если вы программист, никогда не пускайтесь в авантюры типа «Разработка web-сервера». Поверьте, это не для вас. Web-дизайн — это далеко не коды HTML! Именно на этих самых кодах зацикливаются многие программисты, забывая про главное — красивый дизайн, функциональность и содержание. Профессиональный дизайн web-страничек и серверов WWW не может быть выполнен программистом.

Web-сервер

Web-сервер начинается с создания и разметки гипертекстовых документов. Далее создается концепция развития web-сервера. До этого необходимо найти различные публикации, скомпоновать их, выписать ключевые слова и правильно оформить заголовки.

Web-сервер должен иметь удобный навигационный интерфейс.

Прописка web-сервера

Первое, что нужно сделать — правильно сформировать название вашего web-сервера.

Далее, необходимо найти 10 ключевых слов и записать их, разделяя пробелом.

После этого идет описание web-сервера.

Теперь укажите информацию, с помощью которой с вами можно связаться.

Если вы хотите прописать свой web-сервер в русскоязычных поисковых системах, то просто добавьте свой ресурс через ссылки Add URL или Добавить URL, обратившись к следующим системам и каталогам:

- Altavista
- Rambler
- Апорт!

- Yandex
- Search
- Weblist

Web-дизайн и графика

Считается, что графическое наполнение web-страницы должно быть выполнено в едином стиле. Это обстоятельство, в частности, и характеризует профессиональное исполнение web-страницы. Далее уже идут средства того, как это воплотить в Internet.

Прежде всего это цвет, шапка, меню, таблицы и ссылки.

В частности, цвет в web-дизайне подчиняется тем же законам, что и в препресс-дизайне. Например, пастельные цвета и яркие цвета считаются едиными стилевыми элементами, а холодные серые пролонгируют фиолетовую цветовую гамму и ближайший к ней спектр.

Концепция web-сервера

Концепция (удобная навигация, легкость модификации) web-сервера начинается с организации его структуры.

Самая простая структура web-сервера называется полносвязной структурой. Web-сервер с такой структурой состоит из главной страницы, на которой размещены

все имеющиеся ссылки. Древняя и давно забытая последовательная организация web-сервера основывается на последовательно (вперед и назад) переходящих ссылках. Весьма сложная древовидная структура web-сервера используется многими корпоративными узлами. Комбинированная структура сочетает в себе все вышеупомянутые.

Помните, что любая структура должна иметь возможность развиваться.

Как быстро и правильно построить web-сервер

Для начала создайте так называемую домашнюю страницу, т.е. документ HTML, с которого, собственно, и начинается ваш web http://members.rotfl.com/bobleon). Не забивайте эту страницу графикой (стандарт де-факто — от 65 до 95 килобайт графики)! Просто сообщите посетителю о том, что за информацию он может найти здесь, а также предложите ему выбрать нужную кодировку.

Например, если ваш web является информационным, обязательно оптимизируйте все ваши странички на самое оптимальное (640x480) разрешение экрана.

Удобная навигация

Схема, по которой посетитель вашего web-сервера просматривает гипертекстовые документы, называется навигацией. Понятно, что навигация должна быть удобной, понятной и продуманной.

При построении навигационной схемы используйте такую цветовую гамму, которая не сможет отвлечь посетителя от основной темы вашего web-сервера.

Основной навигационный элемент, например шапка, располагается, как правило, наверху, а вторичные элементы находятся в любых других местах, но являются едиными дизайнерскими решениями.

Поисковые роботы

Поисковыми роботами называются специальные программы, которые производят индексацию гипертекстовых страниц того или иного web-сервера.

Под индексацию может попадать различная информация: файлы, текст и даже скрипты CGI.

Для того, чтобы помочь ориентироваться роботу в структуре web-сервера используют так называемые мета-тэги.

Происходит все это дело по весьма сложной схеме, ибо идентичных роботов в Internet не существует. Робот насквозь просматривает ваш гипертекстовый документ и прежде всего старается обнаружить ссылки на другие узлы Internet. Если подобная ссылка обнаружена, то робот осуществляет переход по найденному URL и покидает ваш web. Если же никаких чужаков не найдено, то робот просто путешествует по внутренней структуре вашего web-сервера.

Помните, что роботы просматривают текст и никогда не загружают графику вашего web-сервера, поэтому всегда подписывайте ваши картинки.

Нужно сказать, что в Internet кроме простых роботов существуют так называемые роботы-шпионы. Роботы-шпионы не индексируют web-страницы. Подобные монстры загружают статистическую информацию о вашем web-сервере: подсчитывают ссылки, ведут учет всех web-страниц и т.д.

Мета-тэги

Любой мета-тэг находится в заголовке гипертекстового документа между тэгами <HEAD> и </HEAD> и состоит из следующих атрибутов:


```
<META  HTTP-EQUIV="имя"
CONTENT="содержимое">
<META  NAME="имя"  CONTENT="содержимое">
```

HTTP-EQUIV

Этот атрибут сообщает обозревателю:

- Заголовок документа HTML.
- Дату создания документа HTML. Для этого используется атрибут Expires, например
  <META HTTP-EQUIV="expires"
  CONTENT="Fri,
  12 Dec 1998 08:12:44 GMT">.
- Тип документа HTML. Для этого используется атрибут **Content-Type**. Здесь вы можете указать кодировки (charset) ваших HTML-документов, например
  <META HTTP-EQUIV="Content-type"
  CONTENT="text/html;
  charset=KOI8-R>
- Язык документа HTML. Для этого используется атрибут **Content-language**. Именно эту информацию загружают роботы, когда индексируют ваш web.
- Время, по истечении которого обозреватель произведет обновление документа HTML. Для этого

используется атрибут **Refresh**, например <META HTTP-EQUIV="Refresh" Content="5; URL= http://members.rotfl.com/bobleon">.

- Окно для текущей гипертекстовой страницы. Для этого используется атрибут **Window-target**, например <META HTTP-EQUIV="Window-target" CONTENT="_top">.
- Расширенный кэш. Для этого используется атрибут **Ext-cache**, например <META HTTP-EQUIV= "Ext-cache" CONTENT="name=/bob/leon/index.db; istructions=User Instructions">
- Информацию, которую сервер передаст текущему обозревателю. Для этого используется атрибут **Set-Cookie**, например <META HTTP-EQUIV= "Set-Cookie" CONTENT="NAME=value; EXPIRES=date; DOMAIN=domain_name; PATH=path; SECURE">
- «На этот сайт дети до 16 лет не допускаются!» Используется атрибут **PICS-Label**.

- Как будет работать механизм кэширования в данном документе HTML. Основной атрибут: **Cache-Control**. Используются атрибуты **Public** (документ кэшируется во всех доступных кэшах), **Private** (документ кэшируется только в частном кэше), **no-cache** (документ вообще не кэшируется) и **no-store** (документ кэшируется без сохранения)

NAME

Этот атрибут сообщает роботу:

- Индексацию web-страницы. Используется атрибут **Robots** с допустимыми параметрами **ALL**, **NONE**, **INDEX**, **NOINDEX**, **FOLLOW** и **NOFOLLOW**, например <META NAME="Robots" CONTENT= "NOINDEX,FOLLOW">
- Аннотацию (текст до 100 символов) документа HTML. Используется атрибут **Description**, например <META NAME="Description" CONTENT="Профессионально: дизайн, разработка вашей имиджевой и продуктовой рекламы">
- Список (до 1000 символов) ключевых слов документа HTML. Используется

атрибут **Keywords**, например
<META NAME="Keywords"
CONTENT="реклама, дизайн, полиграфия">

- Как часто необходимо индексировать документ HTML. Используется атрибут **Document-state**, например
<META NAME="Document-state"
CONTENT="Static">.
Допустимые параметры: **Static** и **Dynamic**.
- Каким образом необходимо индексировать web-страницы. Используется атрибут **URL**, например
<META NAME="URL"
CONTENT="absolute_url">.
- Имя автора. Используется атрибут **Author**.
- Название HTML-редактора. Используется атрибут **Generator**.
- Авторские права на документ HTML. Используется атрибут **Copyright**.
- Описание данного файла. Используется **Resource-type**. Помните, что робот начнет индексацию лишь в том случае, если присутствует параметр **document**.

Счетчики посещений

Счетчик посещений позволяет подсчитывать количество пользователей, посетивших ваш web-сервер. Как правило, для этого используется рисунок в формате GIF и простейший CGI-сценарий.

Понятно, что счетчик посещения в основном интересен лишь web-мастеру для того, чтобы оценить популярность своего творения.

Тестируйте свой дизайн различными обозревателями!

Помните, что не все пользователи Internet путешествуют через обозреватель Microsoft Internet Explorer. В мире существует масса других обозревателей, например, Netscape Communicator.

Функциональность и содержание

Дизайн дизайном, но web-страница должна иметь хорошее информационное наполнение.

Как работать с графикой

Стандартный пользователь Internet — это пользователь, имеющий монитор с разрешением 640 на 480 пикселов. Поэтому любая графика вашего web-дизайна должна укладываться в этот размер. Кроме этого,

графика в документе HTML должна иметь небольшой размер, иначе, ваше творение останется незамеченным.

Подписывайтесь под своим дизайном!

Это правила хорошего тона. Подпись должна иметь ссылку на ваш адрес электронной почты.

Отмечайте время!

Всегда ставьте дату последнего редактирования web-страницы.

Оставляйте в документе белые места!

Они помогают отдыхать пользователю от обилия графики и текстовой информации.

Если вы что-то рекламируете...

...то используйте в тексте повторения.

Чтобы создать иллюзию безразмерности...

...выносите ваши рисунки на самую малость за край страницы.

Служебные объявления

Служебные объявления всегда черно-белые.

Жирный шрифт и фотография

Жирный шрифт и фотография — это элементы одного дизайна.

Текстовые и графические объекты

Выравнивайте текстовые и графические объекты только одним способом, например, по центру.

Симметричность оформления

Соблюдайте симметричность оформления графики и текста.

Не трогайте основной стиль!

Изменяйте дизайн, не трогая основной стиль оформления вашего документа.

Текст над заголовками

Текст над заголовками всегда оформляется уплотненным шрифтом, при этом заголовки оформляются одним шрифтом, а основной текст — другим.

Дизайн заголовка, текста над заголовком и строки автора — всегда единый и гармоничный.

Строка автора набирается маленькими прописными буквами. Шрифт строки автора идентичен шрифту заголовка.

Огромная оформительская работа

Вы должны так спроектировать свой документ так, чтобы у пользователя сложилось впечатление, что над документом

была проделана огромная оформительская работа.

Принцип контрастности

Применяйте принцип контрастности: на фоне текста, набранного светлым шрифтом располагаются жирные заголовки.

Цитата

Цитата — внедренный текст. Шрифт цитаты идентичен шрифту строки автора.

Прочно привязывайте смысловую графику

Прочно привязывайте смысловую графику к соответствующему ей основному тексту.

Комбинируйте!

Комбинируйте различные элементы дизайна.

Дизайн первой страницы

Дизайн первой страницы вашего документа должен отличаться от дизайна всех остальных страниц.

Прежде чем начать

Прежде чем начать создание вашего документа, проведите предпроектное исследование, то есть выясните, для кого будет предназначена публикуемая

информация: профессионалы, художники, бизнесмены и т.д.

Симпатичные картинки

Если вы хотите вставить в документ HTML несколько симпатичных картинок, но у вас их нет, то рекомендуем вам воспользоваться абсолютно бесплатными миниатюрами, листинги которых можно найти по адресу: hhtp:/www.yahoo.com/Computers/Multimedia/Pictures/Clip_Art.

Только JPEG и GIF

Помните, что в Web-страницах используются только файлы графических форматов JPEG и GIF.

Создавайте зеркала вашего сервера

Помните о конечных пользователях. Гипертекстовые зеркала ощутимо не бьют по их карманам.

Всегда используйте название документа

Если ваш документ не имеет названия, то такой документ считается непрофессионально подготовленным. Все web-страницы должны иметь заголовок. Это помогает пользователям понимать тему вашего документа.

Могут быть другие шрифты...

Помните, что пользователи, посетившие вашу страницу, могут использовать для ее просмотра шрифты, весьма сильно отличающиеся от тех, которыми вы оформляли исходный документ HTML.

Забудьте о фреймах!

Без комментариев. Так сказать, примите на веру!

Список тэгов, пригодных только для обозревателя Microsoft Internet Explorer

- <BGSOUND>...
- <BODY>... BGPROPERTIES=значение
- <BODY>... LEFTMARGIN=n
- <BODY>... TOPMARGIN=n
- <CAPTION>... VALIGN=позиция
- <COL>...
- <COLGROUP>...
- <COMMENT>...
- <FORM>... TARGET=имя
- <FRAMESET>... FRAMESPACING=n
- <IFRAME>...
- <ISINDEX>... ACTION=url

- <LINK>... SRC=url REL=связь TYPE=TEXT/CSS
- <MARQUEE>...
- <OBJECT>...
- <SPAN>...
- <STYLE>...
- <TABLE>... BORDERCOLOR=цвет
- <TR>...
- <TABLE>... FRAME=значение
- <TABLE>... RULES=значение
- <TBODY>...
- <TFOOT>...
- <THEAD>...

Список тэгов, пригодных только для обозревателя Netscape Navigator

- <FRAME>... BORDERCOLOR=цвет
- <FRAMESET>... BORDER=n
- <IMG>... LOWSRC=url
- <MULTICOL>...
- <NOSCRIPT>...
- <SCRIPT>... SRC=url
- <SPACER>...
- <TEXTAREA>... WRAP=стиль

- <UL>... ... TYPE=метка

Страница должна уместиться на экране пользователя

Не заставляйте пользователей прокручивать вашу web-страничку! Пусть они увидят ее целиком сразу после загрузки!

Оптимальное разрешение экрана пользователя Internet

Большинство пользователей видят ваш web-дизайн на экране с разрешением 640 на 480 и лишь 10% используют экран с разрешением 1024 на 768.

Итоговая статистика обозревателей Internet

Более 60% пользователей путешествуют по Internet с помощью обозревателя Microsoft Internet Explorer и лишь 30% используют для этих целей Netscape Navigator.

Плохие хорошие продвинутые тэги

Помните, что тэги новых (например, великолепный тэг <MULTICOL> — «многоколоночный» текст) языка HTML распознаются пока еще не всеми обозревателями Internet.

Кодировки и ваш web-дизайн

Ваш web должен поддерживать несколько кодировок.

Не раздражайте посетителей!

Продвинутые апплеты Java иногда не распознаются даже самыми продвинутыми обозревателями Internet, а потому просто раздражают посетителей.

Не играйте в детские игрушки!

Длинными бегущими строчками на полосе состояния обозревателя сейчас уже никого не удивишь!

Альтернативный текст

Подписывайтесь под картинкой. Используйте для этого альтернативный текст тэга <IMG>. Профессионалы встраивают под этот тэг даже большие текстовые абзацы.

Избавляйтесь от горизонтальных прокруток

Помните, что горизонтальные прокрутки снижают восприятие информации.

Разбивайте большие документы!

Если ваш документ слишком громоздкий, разбейте его на несколько частей, каждой из которых присвойте отдельную ссылку.

Цвет ссылки, которую посетили

Не забывайте о цвете фона посещенных ссылок, ибо может случиться так, что ваша ссылка станет невидимой.

Помните о регистре!

Ссылки должны быть написаны только маленькими буквами. Не все пользователи Internet работают в Windows!

Помните о мета-тэгах!

Роботы загружают 200 строк гипертекста исходя из мета-тэговой информации.

Безопасные цвета

Используйте только безопасную палитру с 256 цветами.

Обновляйте свой сайт

Дизайн должен обновляться! Информация быстро устаревает, поэтому и она должна обновляться!

Время загрузки вашего сайта должно быть не более 10 секунд!

Не испытывайте терпение пользователей! Ограничьтесь 100 килобайтами графики и распределите их на все ваши страницы.

Схема типичного web-сервера

- Техническая часть.
- Маркетинговая часть.
- Коммерческая часть.
- Двуязычность коммерческой и маркетинговой информации.
- «Горячие новости».
- Действующие программы.
- Изменения цен.
- Ассортимент складской продукции.
- При упоминании продукта должна быть ссылка на его описание.
- Ссылки извне и изнутри, т.е. в каждой из рубрик должны быть ссылки на все остальные рубрики.
- Автоматическая регистрация обращающихся к коммерческой информации и передача ее на обработку в отдел продаж.
- Специальные предложения.
- Описания программ, условий, сроков действия. Упорядочиваются по производителям.
- Распродажи.
- Продукция со склада.

- Информация для партнеров.

Дополнительные изменения в области интернационализации

Различия в таблицах стилей

HTML 4.0 поддерживает более обширный набор дескрипторов устройств, так что авторы могут писать таблицы стилей в зависимости от устройств.

Различия во фреймах

HTML 4.0 поддерживает документы с фреймами и встроенные фреймы.

Различия в скриптах

Многие элементы теперь имеют атрибуты для событий, которые могут объединяться со скриптами; при наступлении события выполняется скрипт (например, при загрузке документа, при щелчке мыши и т.д.).

Различия в интернационализации

В HTML 4.0 объединяются рекомендации по интернационализации HTML.

Однако эта спецификация отличается следующим образом:

- Атрибут **accept-charset** теперь определяется для элемента **FORM**, а не для **TEXTAREA** и **INPUT**.

В спецификации HTML 4.0 даются дополнительные пояснения относительно двунаправленного алгоритма.

- Использование элемента **CDATA** для определения элементов **SCRIPT** и **STYLE** не сохраняет способности транскодирования документов.

Microsoft FrontPage Express

Microsoft FrontPage Express один из тех редакторов Web-страниц, которые не требуют от вас знаний основ HTML. Забудьте о программировании! Просто размечайте и оформляйте!

В законченной среде разработки Web в режиме непосредственного отображения вы можете профессионально создавать web-документы и даже организовывать небольшие web-сервера.

В Microsoft FrontPage Express вы можете подготавливать к публикации в Internet web-страницы, используя JavaScript, VB Script, ActiveX и даже выборки к вашим базам данных. Кроме этого, в Microsoft FrontPage Express встроены весьма неплохие графические эффекты, которые оживят ваш дизайн.

Вы можете управлять созданным в среде Microsoft FrontPage Express web-сервером из локальной сети или используя удаленный доступ к основному компьютеру.

Забудьте о контроле за ложными ссылками! В Microsoft FrontPage Express встроено средство, позволяющее автоматически фиксировать подобные ссылки.

Вы можете:

- создавать и сохранять web-страницы
- сохранять web-страницы непосредственно в Web
- загружать из Internet и редактировать web-страницы
- использовать в оформлении вашей web-страницы подложки
- просматривать и администрировать web-страницы
- создавать сложный дизайн web-страниц
- использовать готовые тэги HTML
- использовать готовые изображения из комплекта поставки программы
- использовать компоненты WebBot для придания вашей web-страничке динамики

- использовать в оформлении web-страниц элементы управления ActiveX

Если вы хотите стать профессионалом в области дизайна web-страниц, вам необходимо освоить некоторые специальные HTML-редакторы с полной поддержкой тэгов HTML.

Интерфейс пользователя

Microsoft FrontPage Express считается одним из лучших средств подготовки документов для их публикации Web. Вы просто создаете web-страницу в том виде, в каком она будет выглядеть в окне обозревателя.

Ниже приведено краткое описание набора средств, с помощью которого вы можете взаимодействовать с редактором web-страниц Microsoft FrontPage Express.

С помощью стрелок **полос прокрутки** или посредством клавиш **PgUp** и **PgDn** вы можете перемещать текущую страницу в вертикальном или горизонтальном направлении.

Вверху экрана находятся **панели инструментов**, с помощью которых вы можете быстро выполнить то или иное действие.

Доступ к командам редактора осуществляется через щелчок мышью на соответствующем имени меню.

Команды меню

В редакторе гипертекстовых документов Microsoft FrontPage Express все наборы команд и опций, использующиеся в процессе работы с Internet содержатся в:

- меню **Файл**
- меню **Правка**
- меню **Вид**
- меню **Переход**
- меню **Вставка**
- меню **Формат**
- меню **Сервис**
- меню **Таблица**
- меню **Окно**
- меню **Справка**

Панели инструментов

Редактор Microsoft FrontPage Express содержит три панели инструментов. Каждая кнопка той или иной панели соответствует команде меню.

Панель форматирования

С помощью **Панели форматирования** вы можете форматировать абзацы, определять параграфу тот или иной шрифтовой стиль, раскрашивать текст и выравнивать параграфы.

Панель редактирования

С помощью **Панели редактирования** вы можете:

- создавать, открывать и сохранять web-страницы
- выводить на печать и предварительно просматривать web-публикации перед выводом на печать
- выполнять операции копирования и вставки
- создавать или изменять ссылки
- вставлять компоненты WebBot, изображения, горизонтальные строки и таблицы
- показывать или скрывать метки параграфа
- возвратиться на одну страницу назад
- перейти на одну страницу вперед
- вызвать справочную систему редактора Microsoft FrontPage Express

- прервать загрузку текущей страницы

Панель форм

С помощь **Панели форм** вы можете создать поле формы из:

- однострочного текстового поля
- прокручиваемого текстового поля
- флажка
- кнопки-переключателя
- простой кнопки

Всплывающее меню

Всплывающее меню позволяет более удобно обратиться к той или иной команде. Просто поместите курсор на интересующий элемент вашего документа и нажмите правую кнопку мыши.

С помощью элементов всплывающего меню вы можете выполнять следующие команды и функции:

- **Вырезать** (выполнить операцию удаления элемента и помещения его в буфер промежуточного хранения)
- **Копировать** (поместить элемент в буфер промежуточного хранения)
- **Вставить** (вставить элемент из буфера промежуточного хранения)

- **Свойства страницы** (обратиться к диалоговому окну свойств **Свойства страницы**)
- **Свойства таблицы** (обратиться к диалоговому окну свойств **Свойства таблицы**)
- **Свойства ячейки** (обратиться к диалоговому окну **Свойства ячейки**)
- **Свойства абзаца** (обратиться к диалоговому окну **Свойства абзаца**)
- **Свойства шрифта** (обратиться к диалоговому окну **Шрифт**)
- **Свойства изображения** (обратиться к диалоговому окну свойств **Свойства изображения**)
- **Проверка достоверности поля формы** (обратиться к диалоговому окну **Проверка достоверности текстового поля**)
- **Свойства поля формы** (обратиться к диалоговому окну **Свойства прокручиваемого текстового поля**)
- **Свойства компонента WebBot** (обратиться к диалоговому окну **Свойства компонента WebBot**)

Буксировка

Вы можете перемещать один или несколько элементов вашего документа из одного окна программы в другое, или между элементами основного окна программы, или из Проводника. Просто выделите тот или иной элемент и отбуксируйте его в нужное место редактируемого документа. При буксировке файлов изображений Microsoft FrontPage Express автоматически преобразует файлы формата BMP в формат JPEG.

Создание новой web-страницы

Создание новой web-страницы осуществляется через выбор команды **Создать** в меню **Файл** (вы также можете выбрать эту команду через комбинацию клавиш **Ctrl+N**). В появившемся диалоговом окне **Новая страница** просто выберите из списка **Шаблон или мастер** параметр **Нормальная страница**.

Вы можете также создать web-страницу с помощью **Мастера личной основной страницы** или **Мастера страниц форм**. Для этого выберите из списка **Шаблон или мастер** необходимый параметр и далее следуйте указаниям **Мастера страниц**.

Сохранение web-страницы

Сохранение web-страницы осуществляется через выбор команды **Сохранить** в меню **Файл**

(вы также можете выбрать эту команду через комбинацию клавиш **Ctrl+S**). В появившемся диалоговом окне введите название страницы. Если же вы хотите сохранить страницу в виде файла, нажмите кнопку **Как файл** и в появившемся диалоговом окне укажите имя файла вашего документа HTML.

Если вы хотите сохранить страницу прямо в Web, определите ее местоположение.

Открытие файла

Выбранная команда **Файл ⇨ Открыть** открывает доступ к диалоговому окну **Открыть файл**, с помощью которого можно загрузить в программу:

- файлы HTML
- файлы HTML после предварительной обработки
- текстовые файлы
- шаблоны гипертекста
- документы Microsoft Word 6.0, Microsoft Word 95 и Microsoft Word 97
- документы текстового процессора Windows Write
- документы WordPerfect 5.x и WordPerfect 6.x
- книги Microsoft Excel

Чтобы открыть файл нужно выбрать его имя из списка и нажать кнопку **Открыть**.

Выход из программы

Выйти из редактора Microsoft FrontPage Express можно, выбрав команду меню **Файл ⇨ Выход**. Если документ, находящийся на экране был изменен, то Microsoft FrontPage Express выведет запрос о том, сохранять ли внесенные изменения. В качестве ответа можно воспользоваться кнопкой **Да**, **Нет** или **Отмена**.

Редактирование текста web-страницы

Microsoft FrontPage Express позволяет редактировать также, как вы это делаете в текстовых процессорах Windows.

Ввод текста осуществляется через щелчок левой клавишей мыши в том месте вашего документа, где вы хотите ввести текст.

Помните, что четыре символа — знак меньше (<), знак больше (>), амперсанд (&) и двойные кавычки (“) имеют специальное значение внутри HTML и следовательно не могут быть использованы в тексте вашего документа в своем обычном значении.

Отмена и восстановление выполненной команды

Отмена выполненной команды осуществляется через выбор команды **Отменить** в меню **Правка**.

Microsoft FrontPage Express позволяет отменить только последнее выполненное действие. Например, если удален текст, а затем набран новый, то нельзя отменить удаление текста.

Выбранная команда **Правка ⇨ Восстановить** восстанавливает последнее отмененное действие.

Выделение слова

Если вы хотите выбрать слово, дважды щелкните по нему левой клавишей мыши.

Выделение строки

Выделение строки осуществляется простым щелчком левой клавишей мыши при нажатой клавише **Alt** в любом месте строки.

Выделение параграфа

Выделение параграфа осуществляется простым щелчком левой клавишей мыши при нажатых клавишах **Alt** и **Ctrl** в любом месте параграфа.

Выделение всех элементов вашего документа

Команда **Правка ⇨ Выделить все** используется для выделения текстовых и графических элементов на текущей странице.

Отображение абзацных меток

Отображение абзацных меток осуществляется через выбор команды **Маркеры форматов** в меню **Вид**.

Операции вырезания, копирования и вставки

Выбранная команда **Правка ⇨ Вырезать** удаляет активный элемент из документа Microsoft FrontPage Express и помещает его в буфер промежуточного хранения. Вы можете удалить из документа и поместить в буфер обмена любой выбранный текстовой или графический элемент как Microsoft FrontPage Express, так и приложения Windows.

Команда **Файл ⇨ Скопировать** используется для вставки активного элемента в буфер промежуточного хранения. Вы можете поместить в буфер обмена любой выбранный текстовой или графический элемент как Microsoft FrontPage Express так и приложения Windows.

Выбранная команда **Правка ⇨ Вставить** помещает копию элемента из буфера промежуточного хранения в документ Microsoft FrontPage Express.

Вы можете поместить в документ Microsoft FrontPage Express текстовой или графический элемент из любого приложения Windows.

Поиск и замена текста

Выбранная команда **Правка ⇨ Заменить** открывает доступ к встроенному в Microsoft FrontPage Express инструменту редактирования, который используется для поиска и замены символов (включая прописные и строчные буквы), целых слов или частей слов.

Поиск и замена спецсимволов осуществляется так же, как поиск и замена простого текста — достаточно ввести в поля данных **Образец** или **Заменить** на ключевые комбинации для специальных символов или ANSI (ASCII) коды символов.

Вывод на печать

Выбранная команда **Файл ⇨ Напечатать** открывает доступ к диалоговому окну **Печать**, с помощью которого можно выбрать принтер для печати текущего документа, установить

параметры работы этого принтера и вывести на печать текущий документ.

Оперирование открытыми окнами

С помощью команд меню **Окно** вы можете:

Рядом

Расположить все окна без взаимного перекрытия по горизонтали.

Каскадом

Упорядочить все окна с перекрытием так, чтобы название каждого окна было видимо.

Упорядочить значки

Упорядочить пиктограммы минимизированных окон.

Установив курсор мыши на кнопке со стрелкой и буксируя мышью один из указателей прокрутки, можно перемещать страницу до тех пор, пока в документе не покажется нужное место. Вертикальный указатель предназначен для передвижения страницы по вертикали, горизонтальный — по горизонтали.

Доступ к системе оперативной подсказки

С помощью команд меню **Справка** можно получить доступ к системе оперативной

подсказки редактора web-страниц Microsoft FrontPage Express.

Стили форматирования

В языке описания гипертекстовых документов слова и строки кодируются логическими и физическими стилями. Поэтому в редакторе web-страниц Microsoft FrontPage Express доступны только два стиля оформления параграфов:

Стили оформления символов (физические стили) влияют на выравнивание параграфов, тип, размер, гарнитуру и цвет шрифта.

Стили оформления параграфов (логические стили) влияют на все параграфы выделенного текста. Эти виды стиля позволяют вести форматирование через указания заголовков того или иного уровня и игнорировать информацию о размере шрифта и гарнитуре. Поэтому, чтобы изменить форматирование заголовка, вы должны модифицировать заголовок первого уровня. Через стили оформления вы можете сформировать согласованный гипертекстовый документ, то есть определить заголовок первого уровня в качестве только **<H1>** (без информации о гарнитуре шрифта и его кегле).

Физические стили

Физические стили или стили оформления параграфов доступны через **Панель форматирования** или через команду **Абзац** меню **Формат**.

Вы можете оформить доступным физическим стилем не только выбранный абзац, но и несколько выделенных абзацев.

К каждому из этих стилей вы можете применить выравнивание по центру, по левому или по правому краю. Для этого в меню **Формат** выберите команду **Абзац** и в появившемся диалоговом окне **Свойства абзаца** обратитесь к списку **Выравнивание абзаца**.

Просто определите точку вставки и выберите необходимый стиль абзаца из списка следующих стилей:

Адрес

Подпись автора web-страницы. Обычно располагается в конце документа.

Заголовки с 1 по 6 уровень

Разделение текста документа на разделы, каждый из которых оформлен текстом определенного размера.

Маркированный список

Список, оформленный маркерами.

Обычный

Стиль абзаца по умолчанию.

Список определений

Оформление глоссариев или других списков.

Нумерованный список

Иерархия с использованием цифр.

Список каталогов

Специальный стиль для оформления списков файлов или каталогов.

Список меню

Специальный стиль для оформления небольших параграфов.

Форматированный

Специальный стиль для оформления листингов программ и другой технической документации.

Логические стили

Microsoft FrontPage Express поддерживает следующие типы логических стилей:

- Полужирный
- Курсивный
- Полужирный курсив

- Машинописный шрифт
- Верхний индекс
- Нижний индекс
- Подчеркнутый
- Зачеркнутый

Вы можете оформить доступным логическим стилем не только выбранный символ, но и несколько выделенных символов. Просто выделите текст, обратитесь к диалоговому окну **Шрифт** (доступ: **Формат ⇨ Шрифт**) и выберите необходимый атрибут шрифта.

Раскрашивание текста

Вы можете оформить цветом не только выбранный символ, но и несколько выделенных символов. Просто выделите текст, обратитесь к диалоговому окну **Шрифт** (доступ: **Формат ⇨ Шрифт**) и выберите из списка **Цвет** понравившийся вам цвет шрифта.

Многоплатформенные шрифты

В принципе с помощью Microsoft FrontPage Express вы можете оформлять свои web-страницы всеми доступными шрифтами вашей системы. С другой стороны, если вы хотите, чтобы вашу страницу могли просматривать пользователи других

операционных систем, вы должны использовать в оформлении так называемые многоплатформенные шрифты. К ним относятся следующие шрифты:

- Arial
- Courier
- Times

Горизонтальные линии

Вы можете украсить дизайн вашей web-страницы горизонтальной линией. Для этого определите место в вашем документе, в котором должна быть линия, и выберите из меню **Вставка** команду **Горизонтальная линия**.

Просмотр и правка исходного кода HTML

Microsoft FrontPage Express позволяет вам просмотреть текст исходного кода HTML и, в случае необходимости, произвести редактирование. Для этого выберите из меню **Вид** команду **HTML**.

Вставка в документ тэгов HTML

Вы можете вставить в ваш документ любой тэг HTML, включая Java Script. Для этого выберите из меню **Вид** команду **Разметка HTML**, обратитесь к диалоговому окну **Разметка HTML** и поместите ваш тэг в поле данных **Вставляемая метка HTML**.

Одновременно вы можете вставить только один тэг.

Дизайн web-страниц с помощью таблиц

Мощнейшее средство HTML — таблицы, позволяют создавать достаточно сложный дизайн web-страницы, например, вы можете расположить текст вашего документа в нескольких колонках.

Для того, чтобы разместить в вашем документе таблицу, определите точку вставки и обратитесь к диалоговому окну **Добавить таблицу** через команду **Вставить таблицу** меню **Таблица**.

Опции диалогового окна **Добавить таблицу** позволяют:

Размер строки

Указать количество строк таблицы.

Размер колонки

Указать количество колонок таблицы.

Выравнивание

Определить тип выравнивания элементов таблицы.

Размер рамки

Определить размер линий границ ячеек в пикселах (до 100 пикселей). Если размер

рамки будет иметь нулевое значение, то линия границы таблицы отобразится пунктирами, а при просмотре документа в обозревателе станет невидимой.

Промежуток между столбцами

Определить расстояние между столбцами в пикселах.

Ширина

Указать ширину таблицы в пикселах или в процентах от ширины основного окна соответствующего обозревателя.

Независимо от того или иного обозревателя, таблица, ширина которой определена в пикселах, будет иметь постоянный размер.

Таблица, ширина которой определена в процентах, будет подстраиваться к тому или иному обозревателю.

Дополнительно

Обратиться к диалоговому окну **Дополнительные атрибуты**, с помощью которого можно вставить в таблицу тэг HTML или сценарий Java.

Как добавить в таблицу строку или столбец

В созданную таблицу вы можете добавить строку или столбец. Для этого выберите в таблице место вставки и обратитесь к

диалоговому окну **Вставить строки или столбцы** (доступ: **Таблица ⇨ Вставить строки или столбцы**).

Совет: Ничто вам не мешает вставить таблицу в другую таблицу. Этим часто пользуются опытные дизайнеры web-страниц.

Как добавить в таблицу ячейку

В созданную таблицу вы можете добавить ячейку. Для этого выберите в таблице место вставки и дайте команду **Вставить ячейку** (доступ: **Таблица ⇨ Вставить ячейку**).

Как выделить таблицу

С помощью меню **Таблица** вы можете выделить всю таблицу или только одну ячейку, строку или столбец, просто дав одну из следующих команд: **Выделить таблицу**, **Выделить ячейку**, **Выделить строку**, **Выделить столбец**.

К выделенной таблице вы можете применить стандартные операции копирования и вставки.

Как разбить ячейку на столбцы или строки

Вы можете разбить ячейку на столбцы или строки с помощью диалогового окна **Разбить ячейки** (доступ: **Таблица ⇨ Разбить ячейки**). В поле данных **Число колонок** определите количество столбцов вашей новой ячейки, а в

поле данных **Число строк** укажите количество строк.

Свойства таблицы

Созданной таблице вы можете назначить те или иные свойства (ширина линии границы, поля и отступы ячеек, ширина, цвет). Для этого в меню **Таблица** выберите команду **Свойства таблицы** и обратитесь к диалоговому окну **Свойства таблицы**.

Опции диалогового окна **Свойства таблицы** позволяют:

Выравнивание

Определить тип выравнивания элементов таблицы.

Размер рамки

Определить размер линий границ ячеек в пикселах (до 100 пикселей). Если размер рамки будет иметь нулевое значение, то линия границы таблицы отобразится пунктирами, а при просмотре документа в обозревателе станет невидимой.

Заполнение ячеек

Определить максимальный размер заполнения ячейки в пикселах (до 100 пикселей).

Промежуток между столбцами

Определить расстояние между столбцами в пикселах.

Ширина

Указать ширину таблицы в пикселах или в процентах от ширины основного окна соответствующего обозревателя. Независимо от того или иного обозревателя, таблица, ширина которой определена в пикселах будет иметь постоянный размер. Таблица, ширина которой определена в процентах, будет подстраиваться к тому или иному обозревателю.

Специальный фон

Использовать в качестве фона таблицы картинку. Просто выберите флажок **Использовать фоновое изображение** и нажмите кнопку **Обзор** для поиска файла картинки в форматах GIF, JPG, BMP, TIF, WMF, RAS, EPS, PCX или TGA.

Цвет фона

Раскрасить подложку таблицы.

Дополнительные цвета

В этом разделе вы можете определить цвет для рамки вашей таблицы. Если вы выберите только параметр **Граница**, т.е. параметры **Светлая** и **Темная рамка** останутся

используемыми по умолчанию, то через ниспадающий список цветов можно присвоить цвет всей рамке вашей таблице.

Имеющие приоритет параметры **Светлая** и **Темная рамка** позволяют присвоить цвет верхней и нижней рамкам таблицы.

Манипулирование ячейками таблицы

Вы можете назначить те или иные свойства не всей таблице, а только ее ячейкам. Для этого в меню **Таблица** выберите команду **Свойства ячейки** и обратитесь к диалоговому окну **Свойства ячейки**.

Опции диалогового окна **Свойства ячейки** позволяют:

Выравнивание по горизонтали

Определить горизонтальную выключку элементов таблицы (по горизонтали, влево и вправо).

Вертикальное выравнивание

Определить вертикальную выключку элементов таблицы (сверху, снизу и посередине).

Ячейка с заголовком

Присвоить выбранной ячейке таблицы заголовок.

Без разбиения

Не допускать переноса текста ячейки на следующую строку.

Минимальная ширина

Указать минимальную ширину ячейки в пикселах или в процентах от ширины таблицы. Ячейка, ширина которой определена в процентах, будет подстраиваться к тому или иному обозревателю.

Специальный фон

Использовать в качестве фона ячейки картинку. Просто выберите флажок **Использовать фоновое изображение** и нажмите кнопку **Обзор** для поиска файла картинки в форматах GIF, JPG, BMP, TIF, WMF, RAS, EPS, PCX или TGA.

Цвет фона

Раскрасить подложку ячейки.

Дополнительные цвета

В этом разделе вы можете определить цвет для рамки ячейки. Если вы выберите только параметр **Граница**, т.е. параметры **Светлая** и **Темная рамка** останутся используемыми по умолчанию, то через ниспадающий список цветов можно присвоить цвет всей рамке ячейки.

Имеющие приоритет параметры **Светлая** и **Темная рамка** позволяют присвоить цвет верхней и нижней рамкам ячейки.

Дизайн web-страниц с использованием графики

Редактор web-страниц Microsoft FrontPage Express поддерживает графические изображения двумя форматами GIF (CompuServe Graphics Interchange Format Расширение .GIF) и JPEG (Joint Photographic Experts Group Расширение .JPG). Это означает, что любой файл формата BMP, TIF, WMF, RAS, EPS, PCX и TGA при вставке в рабочую среду Microsoft FrontPage Express будет преобразован в GIF-картинку.

Несколько слов о форматах GIF и JPEG. GIF-формат хорош тем, что он маленький и имеет всего 256 цветов. Формат JPEG используется только для картинок, имеющих более 256 цветов. С другой стороны, вы можете создать JPEG-файл с потерей качества, но без потери количества цветов, это может значительно уменьшить размер файла. Поэтому, если вы хотите иметь хорошую картинку, просто преобразуйте ее в формат JPG.

Вы можете поместить в Microsoft FrontPage Express изображение посредством

выбора из меню **Вставка** команды **Изображение**.

Изображение Microsoft FrontPage Express может быть помещено в любое место вашего документа, определенное точкой вставки. Просто поместите точку вставки в нужное место, выберите через диалоговое окно **Изображение** (доступ: **Вставка ➪ Изображение**) имя графического файла и нажмите кнопку **OK**.

Вы можете помещать в документ Microsoft FrontPage Express только растровые (двоичные отображения) картинки. Растровые картинки строятся из отдельных точек и хранятся в виде групп точек, а экран представляется в виде прямоугольного массива точек, компьютеру указывается цвет каждого элемента изображения.

Если документ с помещенной картинкой был сохранен, а затем картинка подверглась редактированию в какой-либо программе, то Microsoft FrontPage Express не внесет эти изменения в ваш документ.

Если вы переместите файл изображения в другую папку, то само изображение не отобразится в вашем документе. Помните, что ваше изображение должно храниться в

том же каталоге, в каком находится файл исходного документа.

Если формат графического файла не поддерживается программой, то стоит попытаться вставить картинку в Microsoft FrontPage Express из буфера обмена, или открыть картинку в исходном приложении, или сохранить ее в другом формате. Если же файл загружается в исходное приложение, но неправильно отображает картинку, то необходимо записать файл в другом графическом формате или распрощаться с этим файлом раз и навсегда.

При импортировании картинки в Microsoft FrontPage Express размер файла документа практически не увеличивается.

Библиотека рисунков Microsoft FrontPage Express

С помощью вкладки **Картинки** диалогового окна **Изображение** вы можете воспользоваться библиотекой рисунков Microsoft FrontPage Express.

Альтернативное отображение картинок

Microsoft FrontPage Express отображает изображения посредством заменяющего текста и так называемой превью-картинки. Превью-картинка имеет низкое разрешение (при загрузке исходного изображения сначала

проявится изображение с низким разрешением), а заменяющий текст используется при просмотре web-страниц в текстовом режиме.

Чтобы отобразить картинку посредством заменяющего текста, выберите ее, обратитесь к вкладке **Общие** диалогового окна **Свойства изображения** (доступ: **Правка ➪ Свойства изображения**) и в поле данных **Текст** введите информацию, тем или иным образом описывающую вашу картинку.

Чтобы отобразить картинку посредством превью-картинки, выберите ее, обратитесь к вкладке **Общие** диалогового окна **Свойства изображения** (доступ: **Правка ➪ Свойства изображения**) и с помощью кнопки **Обзор** раздела **Другие представления** выберите картинку с низким разрешением.

Как присвоить изображению ссылку

Вы можете присвоить помещенной в рабочую среду Microsoft FrontPage Express картинке любую ссылку. Для этого выберите картинку, обратитесь к вкладке **Общие** диалогового окна **Свойства изображения** (доступ: **Правка ➪ Свойства изображения**) и с помощью кнопки **Обзор** раздела **Ссылка по умолчанию** выберите документ HTML, на который должен быть сделан переход.

Позиционирование изображений

Вы можете позиционировать изображениями, помещенными в рабочую среду Microsoft FrontPage Express. Для этого выберите изображение и обратитесь к вкладке **Внешний вид** диалогового окна **Свойства изображения** (доступ: **Правка ➪ Свойства изображения ➪ Внешний вид**).

С помощью ниспадающего списка **Выравнивание** укажите тип позиционирования вашего изображения относительно страницы.

С помощью опции **Интервал по горизонтали** вы можете сместить выбранное изображение относительно страницы по горизонтали. Параметр смещения одновременно воздействует на правую и левую части выбранного изображения.

С помощью опции **Интервал по вертикали** вы можете сместить выбранное изображение относительно страницы по вертикали. Параметр смещения одновременно воздействует на верхнюю и нижнюю части выбранного изображения.

Основные свойства web-страницы

Свойства созданной страницы представлены диалоговым окном **Свойства страницы** (доступ: **Файл ➪ Свойства страницы**).

Папка

Укажите путь к папке, в которой хранятся все файлы вашей web-страницы.

Заголовок

Текст, который вы впишите в это поле данных отобразится в заголовке окна обозревателя. Практически все поисковые машины Internet начинают сканирование web-страниц с заголовка.

Как озвучить web-страницу

Вы можете озвучить вашу web-страницу, просто поместив в нее файлы звуков WAV, AIFF, AU MIDI. Для этого определите точку вставки и обратитесь к разделу **Фоновый звук** вкладки **Общие** диалогового окна **Свойства страницы** (доступ: **Файл ⇨ Свойства страницы**).

С помощью кнопки **Обзор** выберите звуковой файл на вашем диске. Если вы хотите, чтобы ваш звук все время играл в окне обозревателя, поставьте флажок **Непрерывно**. Если же вы хотите, чтобы ваш звук воспроизводился в течение определенного цикла, снимите флажок **Непрерывно** и укажите количество циклов воспроизведения с помощью счетчика **Цикл**.

Кодирование web-страниц

С помощью Microsoft FrontPage Express вы можете изменить кодировку символов web-страницы. Для этого обратитесь к разделу **Кодировка HTML** вкладки **Общие** диалогового окна **Свойства страницы** (доступ: **Файл ⇨ Свойства страницы**).

Отображение русских букв зависит от вашей кодировки. Поэтому для того, чтобы в обозревателе нормально отображались русские буквы, вам необходимо установить в систему многоязыковую поддержку и шрифты с соответствующей кодировкой, например KOI8. Как правило, при работе в Windows 98 для кириллизации используется кодировка CP1251, а на непонятных компьютерах с UNIX — кодировка KOI-8.

Стандарт KOI8 является кириллической кодовой таблицей, позволяющей вам обмениваться некоторыми сообщениями электронной почты и просматривать web-страницы. Для того, чтобы Windows 98 понимала KOI8, необходимо установить многоязыковую поддержку и шрифты с кодировкой KOI8.

Помните, что российские серверы Web поддерживают русский язык либо через

кодовую страницу 1251 (Windows 95), либо через таблицу КОИ-8 (Unix).

Как присвоить цвет или фон вашей web-странице

Вы можете указать цвет фона для вашего документа HTML. Для этого обратитесь к вкладке **Фон** диалогового окна **Свойства страницы** (доступ: **Файл ➪ Свойства страницы**), поставьте флажок **Фоновое изображение** и с помощью кнопки **Обзор** выберите файл изображения, который будет использоваться в качестве фона.

Вы можете указать дополнительный цвет фона для текущего документа. Для этого обратитесь к вкладке **Фон** диалогового окна **Свойства страницы** (доступ: **Файл ➪ Свойства страницы**) и выберите из ниспадающего списка **Фон** понравившийся вам цвет.

Как присвоить цвет тексту и ссылкам

Вы можете присвоить цвет любому выделенному фрагменту текста, а также гиперссылке, просмотренной и активной ссылкам. Для этого обратитесь к вкладке **Фон** диалогового окна **Свойства страницы** (доступ: **Файл ➪ Свойства страницы**) и выберите из ниспадающего списка **Текст** цвет для выделенного фрагмента текста.

Цвет ссылки вы можете изменить с помощью ниспадающих списков **Гиперссылк**а, **Просмотренная ссылка** и **Активная ссылка**.

Мета-тэги

С помощью вкладки **Специальный** диалогового окна **Свойства страницы** (доступ: **Файл ➪ Свойства страницы**) вы можете включить в ваш документ HTML дополнительную информацию, которая не отобразится в окне обозревателя, но помогает той или иной поисковой машине Internet более легко обнаружить вашу web-страницу.

Речь идет о так называемых мета-тэгах. Используя мета-тэги вы можете включить в свой документ, например, ключевые слова и короткое описание вашей web-странички:

```
<META name="description"
content="Хороший сайт">
<META name="keywords"
content="автомобиль, стоп">
```

Использование гипертекстовых ссылок

Гипертекстовая ссылка считается активной частью документа HTML. Это означает, что любому пользователю Internet достаточно щелкнуть по ссылке для перехода к другой web-странице или другому узлу Internet. Для того, чтобы создать ссылку, выделите текст

или изображение и обратитесь к диалоговому окну **Создать гиперссылку** (доступ: **Правка ⇨ Гиперссылка**). С помощью ниспадающего списка **Тип гиперссылки** укажите необходимый тип гиперссылки, а в поле данных **Адрес (URL)** введите необходимый адрес URL.

Абсолютные ссылки указывают путь к месту вашего диска, где находится документ HTML. Такие ссылки совершенно не пригодны для публикации вашего документа в Web. Относительные ссылки не имеют пути, поэтому, если вы переместите ваши файлы, ссылки все равно будут работать.

Список файлов, которые вы можете повстречать в Internet

.a
Библиотека откомпилированных процедур.

.afm
Adobe Font Metrics — метрики символов шрифта PostScript.

.ai
Файл PostScript.

.aif, .aifc, .aiff
Audio-данные.

.ar
Архив программы ar.

.arc
Архив программы arc, pkarc или arca/arcb.

.arj
Архив программы arj.

.asm
Исходный текст программы на Ассемблере.

.au
Формат хранения звука.

.avi
Формат хранения видеоизображения.

.b
Встроенный редактор.

.bak
Старая версия некоторого файла.

.bas
Исходный текст программы на языке Basic (GWBasic, TurboBasic, QuickBasic).

.bat
Набор команд.

.bgi

Borland Graphics Interface. Динамически подгружаемая библиотека графических программ-драйверов, зависящих от типа видеоадаптера.

.bmp

Растровый графический формат.

.C

В Unix архив программы compact.

.c

Исходный текст программы на языке C.

.cfg

Конфигурационный файл программы.

.cgi

WWW-сервер под Unix. Запускаемая программа, работающая по протоколу Common Gatawey Interface (входные данные — в переменной окружения QUERY_STRING при методе GET или в стандартном вводе stdin при методе POST). Может быть любым запускаемым файлом.

.clo, .cls

Описания классов документов и макроопределений TEX.

.com

Исполняемый двоичный файл.

.cpp

Исходный текст программы на языке C++.

.cpt

Архив программы CompactPro (Macintosh).

.ddi

Disk Dupe Image. Образ дискеты, созданный программой DiskDupe.

.def

Описания шрифтов TEX без начертаний каждого символа.

.dir

Файл-директория редактора MicroMir или NanoMir.

.diz

Файл с кратким описанием продукта или содержимого диска/архива, обычно для самопальных или похаканных пиратами продуктов.

.dll

Dynamic Linked Library. В системе Windows динамически присоединяемая библиотека.

.doc
Документ редактора Microsoft Word для Windows.

.dot
Шаблон документа редактора Microsoft Word для Windows.

.dp
Digital Paper. Аналог .PDF фирмы Common Ground Software.

.evy
Envoy. Аналог .PDF фирмы Novell.

.exe
Исполняемый бинарный файл.

.f
В Unix исходный текст программы на языке ForTran.

.fd
Описания шрифтов TEX без начертаний каждого символа.

.fon
Шрифт.

.for
Исходный текст программы на языке ForTran.

.fot
Шрифт.

.f
Папка с письмами почтовой системы Waffle для MS-DOS.

.gif
Graphics Interchange Format. Растровый графический формат фирмы CompuServe.

.gz
Архив, созданный программой gzip (Unix) и распаковываемый gunzip или gzip.

.h
В языке C header-файл, содержащий описания заголовков процедур.

.hqx
Файл программы BinHex (Macintosh). Предназначен для передачи без искажений бинарных файлов через Internet.

.htm, .html
Текст написанный на Hyper Text Markup Language. Содержит команды (tags) в угловых скобках (<команда>), остальное интерпретируется как текст. Знаки <, > и & заменяются на <, > и &amp.

.ice
Архив программы ice.

.iff
Формат хранения звука, разработанный для компьютеров Amiga.

.ini
Файл с установками какой-либо программы.

.jfif, .jpeg, .jpg
Растровый графический формат JPEG, позволяющий сохранять картинку с потерей информации без существенной потери качества.

.l3
MPEG-1: звук только Layer-3.

.latex, .ltx
Одно из расширений TEX.

.lha, .lzh
Архив программы lzh.

.lib
Библиотека процедур.

.m1s
MPEG-1: системный поток.

.m2a
MPEG-2: только звук.

.m2s
MPEG-2: системный поток.

.m2v
MPEG-2: только видео.

.man
Файл в формате TROFF (Typesetting Run OFF).

.me
Файл редактора MultiEdit.

.me
Расширение файла read.me.

.mf
MetaFont. Программа, представляющая шрифт для TEX в виде матричных и растровых комбинаций.

.mia
MPEG-1: только звук.

.mid
MIDI. Звуковой файл.

.mim
Файл редактора MicroMir.

.miv
MPEG-1: только видео.

.mod
Формат хранения звука.

.mov, .Moov
Формат хранения видео и аудио.

.mpg
MPEG. Формат хранения видео и звука с компрессией и потерей данных.

.mps
MPEG-1.

.nfo
Краткое описание того, что находится в директории или на диске.

.o
В Unix откомпилированный, но не собранный для выполнения код программ.

.obj
Откомпилированный, но не собранный для выполнения код программ.

.ovl
OVerLay. Динамически подгружаемый модуль программы.

.p
В Unix исходный текст программы на языке Pascal.

.pas
Исходный текст программы на языке Pascal.

.pbm
Portable BitMap Простой формат хранения черно-белых картинок.

.pcx
Растровый графический формат, поддерживаемый большинством редакторов.

.pdf
Portable Document Format.

.pfm
PostScript Font Metrics. Метрики символов в шрифте PostScript.

.pgm
Portable GrayMap. Простой формат хранения полутоновых картинок.

.pict
Формат хранения графических изображений в буфере обмена на компьютерах Macintosh.

.pif
Файл, описывающий параметры запуска DOS-задачи под Windows.

.pk
Растровый шрифт для TEX.

.pl
Файл на языке Perl.

.pl
Описания шрифтов TEX без начертаний каждого символа.

.pnm
Простой формат хранения картинок, объединяющий .pbm, .pgm и .ppm.

.pop
В Unix временный файл POP3-сервера в той же директории, что и почтовые ящики пользователей.

.ppd
PostScript Printer Description. Описание принтера для программы, печатающей .ps.

ppm
Portable PixelMap. Простой формат хранения цветных картинок.

.ppt
Презентация, созданная в программе Microsoft PowerPoint.

.ps
Векторный графический формат PostScript.

.rar
Архив программы rar.

В Windows 95/NT файл с описанием ресурсов программы.

.rpm
RedHat Packing Manager. Формат хранения дистрибутивов в RedHat Linux.

.rtf
Rich Text Format.

.sea
Архив программы stuff (Macintosh).

.sfx
Самораспаковывающийся архив. В MS-DOS его необходимо переименовать в .exe.

.sgm, .sgml
Standard Generalized Markup Language. Язык разметки, используемый для управления большими подборками

документов. Частным случаем SGML является HTML.

.sh
Запускаемый пакетный файл на языке shell.

.so
Динамически присоединяемая библиотека.

.snd
Звуковой файл.

.spl
Future Splash Player.

.sty
Описания классов документов и макроопределений TEX.

.swf
ShockWare Flash.

.swp
Файл подкачки.

.sys
Файлы ядра DOS IO.sys и MSDOS.sys.

.tar
Архив программы tar (Unix) без компрессии.

.taz
Аналог .tar.gz в DOS и других системах, использующих три буквы в расширении файла.

.tfm
Описания шрифтов TEX без начертаний каждого символа.

.tgz
Эквивалентно .tar.gz; создается tar с ключом -z.

.tif, .tiff
Tagged Image File Format, растровый графический формат.

.ttf
TrueType Font.

.uc, .uc2
Архив программы uc (UltraCompressir).

end
Файл, созданный программой uuencode и раскрываемый программой uudecode.

.vp, .vpl
Файлы виртуальных шрифтов для TEX.

.vrml
Virtual Reality Modeling Language.

.xx, .xxe
Файл, созданный программой xxencode и раскрываемый программой xxdecode.

.Z
Архив программы compress.

.z
Архив программы pack.

.zip
Архив программы pkzip или WinZip.

.zoo
Архив программы zoo.

Толковый словарь

Active Channel
Узел Web, автоматически поставляемый на рабочий стол пользователя.

Active Desktop
Интерфейс, интегрированный с рабочим столом Windows и обозревателем Microsoft Internet Explorer.

ActiveMovie
Технология цифрового видео, позволяющая через Web просматривать файлы AVI, QuickTime или MPEG.

ActiveX
Термин программного интерфейса технологии Microsoft, позволяющей разработчикам создавать интерактивное содержимое для WWW, а также для компонентов программного обеспечения, написанных на различных языках. Основные элементы технологии ActiveX — COM и DCOM.

Address
Уникальный код с информацией, находящейся в файле конкретной сети.

Anchor
То, что собственно, и образовывает гипертекстовую ссылку.

Anonymous
Один из методов получения доступа к той или иной информации. Вы можете только копировать файлы, передавать свои — нет.

ANSI
American National Standards Institute.

Выделенный канал
Постоянное соединение, всегда позволяющее передать поток информации от одного компьютера к другому.

Authorization

Право, которое дается пользователю на тот или иной ресурс компьютерной системы.

BBS

Bulletin Board System. Тип компьютерного сервиса. Юзеры могут читать и публиковать различные сообщения. Передавать или скачивать файлы.

Big mail

Электронная почта огромных размеров. Передается некоторыми сволочами в качестве мести.

Bookmark

Закладка. Файл Gopher или WWW. Информация в этом файле размещена так, что получить доступ к желаемой странице можно без дополнительного серфинга.

Browser

Нечто, похожее на графический интерфейс. Позволяет блуждать по той или иной сети, или искать себе на определенное место приключений.

Bullet

Маркер. Элемент оформления текста. Например, маленький круг, квадрат, звездочка.

Character Set

Символы, которые объединены в определенную группу. Например, ASCII.

Character

Любой символ, введенный с клавиатуры в компьютер.

Chart

Диаграмма. Рисунок, показывающий взаимодействие тех или иных данных.

Check Box

Флажок в виде крестика. Один из многочисленных элементов графической операционной системы. Позволяет включать или отключать то или иное действие.

CompuServe

Коммерческая компьютерная сеть США.

Cookies

Технология, позволяющая сохранять сугубо индивидуальную информацию о пользователе сети.

Cracker

Хакер, который ломает все и вся.

Cybermall

Электронный магазин.

Cyberpunk

Самоутвержденный хакер.

DejaNews

Ищем новости в конференциях Usenet по ключевому слову, автору статьи или по адресу. Доступ через http://www.dejanews.com.

DNS

Domain Name System. Доменная система адресации. Нечто, похожее на базу данных и конвертирующее буквенное сетевое имя в набор цифр. Этот набор цифр является числовым адресом Internet. Понятен только компьютерам и некоторым фанатам с шизофренической памятью.

Domain Name Server

Имя сервера домена. Каждое подразделение Internet имеет два домена. Основной DNS обычно располагается на сетевой машине. DNS-сервера используют в своих обращениях к удаленным узлам 32-битные адреса IP, мнемонически заключенные в четырехразрядную буквенную комбинацию. Любой хост может получить соответствующий DNS у ближайшего информационного сервера DNS по известной системе Domain Name Server через сетевой протокол DNS. Просто хост посылает запрос на известный IP-адрес DNS-сервера свой

IP-адрес и имя сервера. Сервер DNS штудирует собственную базу данных, находит IP-адрес и отправляет на хост соответствующий ответ DNS. Схема весьма примитивная. Если же сервер DNS не находит искомую буквенную комбинацию, то он отсылает запрос на так называемый корневой сервер, который, в свою очередь, сверяет информацию с файлом настроек root.cache. Так происходит до тех пор, пока имя хоста не будет найдено в Internet.

Domain

Подразделение Internet. У каждого домена есть своя метка. Например, метки .com, .net, .mil, .org означают, что это соответственно домен коммерческой, сетевой, военной и общественной организации.

Download

Закачивание софта с другого компьютера на собственный винчестер.

Ethernet

Один из видов сети с весьма классной пропускной способностью. Довольно часто компьютеры, использующие протоколы TCP/IP подсоединяются к Internet через Ethernet. Да какая там поэзия! Проза, сухая проза.

Font

Шрифт или семейство знаков с индивидуальным стилем.

Freeware

Бесплатное программное обеспечение.

FTP

File Transfer Protocol. Метод пересылки файлов на ваш компьютер. В Internet имеются тысячи мест, поддерживающих этот метод. Иногда единственная возможность заиметь файл — это воспользоваться протоколом FTP. Помните об этом протоколе.

FTP-client

Софт, позволяющий вам приконнектиться к серверу FTP.

FTP-команды

Команды с синтаксисом FTP.

FTP-mail

Крутейшая штучка. Вы можете получать файлы с серверов FTP по электронной почте.

FTP-server

Компьютер, битком набитый всякой всячиной и поддерживающий протокол FTP.

GIF

Graphic Interchange Format. Один из стандартных графических файлов WWW.

Hacker

Очень умный человек. Тот, кто взламывает сети. Этот умник может проникнуть в любую сеть. Делается это разными способами. Например, через порт терминала или порт электронной почты. Выбрав жертву, хакер прежде всего определяет, имеются ли на сервере плохие пароли, плохо настроенный софт или испорченная операционка. Затем выбирает метод собственной безопасности.

HTML

HyperText Markup Language. Язык, лежащий в основе формирования документов World Wide Web.

HTTP

HyperText Transport Protocol. Система, передающая документы HTML по World Wide Web.

Hypermedia

Гиперсреда. Через определенные ссылки связывается текст, графика, звук, видео и картинки.

Hypermedia

Нечто, которое хорошо знают хакеры. Грубо говоря то, что дискретно может отобразить фотографию на выбранном вами web-обозревателе.

Hypertext

Нечто представляющее данные так, что можно легко организовать межстраничные связи различных документов Internet.

Internet

Мировая с англоязычным уклоном свободная конфедерация компьютерных сетей, объединяющая более 12 тысяч локальных сетей, более одного миллиона компьютеров и около 30 миллионов юзеров. Негласный победитель FIDO.

IP Address

Тридцатидвухбитовый (ну и словечко) адрес протокола Internet, включающий в себя номер узла и сети.

IP

Самый важный из всех протоколов, на котором основана Internet. Через этот протокол осуществляется прямое подключение к Internet.

IRC

Internet Relay Chat. Одна из форм WWW-психоза. IRC чем-то напоминает работу в конференциях Usenet. Но если там вы общаетесь не в реальном времени, то здесь может вестись живой разговор. Разве что, — вас никто не слышит. Вас могут прочитать. Вы набиваете текст на клавиатуре. Ваша информация попадает на общий дисплей. Различные группы видят ваш бред. Если интересно — отвечают.

JPEG

Joint Photographic Experts Group. Графический формат, принятый всеми в качестве стандартного формата при оформлении Web-страниц.

Lamer

Юзер-идиот. Термин, придуманный американскими системными администраторами.

Login

Регистрационное имя пользователя.

Lurk

Пассивные наблюдатели. Например, те, кто только читают новости конференций Usenet или сообщения, проходящие по каналам IRC.

Mail server
Почтовый сервер. Компьютер, обрабатывающий электронную почту.

Mail
Обмен частными текстовыми сообщениями в Internet или в другой компьютерной сети.

Mailing list
Список рассылки электронных почтовых сообщений, классифицированных по определенным темам. Своеобразная подписка.

Marquee
Бегущая строка в документе HTML.

Media
Почти то же, что Multimedia.

Microsoft FrontPage Express
Редактор web-страниц. Средство для создания и оформления документов HTML в режиме непосредственного отображения.

MIME
Multipurpose Internet Mail Extensions. Протокол передачи звука, графики и других двоичных данных. Применяется при передаче почтовых сообщений. И как только этот MIME не интерпретируют!

Mosaic
Древнейший графический пользовательский интерфейс, позволяющий просматривать World Wide Web. Имеются версии для X-Window, Windows и Macintosh.

MPEG
Moving Pictures Expert Group. Протокол, через который упаковываются видеозаписи.

Multimedia
Весьма туманный термин, не переводящийся на нормальный язык.

Netscape Communicator
Еще один обозреватель Internet. Графический пользовательский интерфейс, позволяющий просматривать World Wide Web. Имеются версии для X Window, MS Windows и Macintosh. Весьма неплохой.

Newsgroup
Область сообщений в конференциях Usenet.

NNTP
Сетевой протокол, с помощью которого весь мир пользуется прелестями от конференций Usenet.

Offline
Автономный режим работы компьютера.

Online
Интерактивный режим работы сетевого компьютера.

Page
Страница. Любой документ World Wide Web.

Prompt
Поле данных удаленного терминала.

Protocol
Метод, с помощью которого передается информация от хоста к юзеру.

Provider
Поставщик некоторых услуг. В частности, — услуг Internet.

Список использованной литературы

HTML 4.0: зрелость и совершенство

Нейл Рэндалл,СК Пресс 98-06, e-mail: pcmagedt@aha.ru, PC Magazine, November 18, 1997, p. 287.

Web-design

http://www.novgorod.ru/frisbee/codenet/index.php3?doc=webmast/vrml20