**Форматирование текста**

Как вы могли заметить, чтобы поместить простой текст на страницу, достаточно ввести его в теле документа. При этом браузер отобразит текст, используя шрифт и цвет, заданные по умолчанию для текста тела страницы.

Чтобы чтение информации, содержащейся в HTML‑документе, стало приятным занятием, применяется форматирование и разбиение документа на логически цельные части (структурирование) с визуальным отделением этих частей друг от друга. Далее будут рассмотрены основные возможности HTML, позволяющие сделать содержимое документа легко читаемым и воспринимаемым. Однако перед описанием форматирования и структурирования текста нужно рассмотреть особенности, которые необходимо учитывать при добавлении непечатных и зарезервированных для языка HTML символов в текст документа.

**3.1. Особенности ввода текста**

При вводе текста в документ часто возникает вопрос: как заставить браузер отобразить зарезервированные символы языка HTML (например, > или &) или символы, которые невозможно ввести с клавиатуры. Для ввода в документ таких символов в HTML предусмотрен механизм ссылок на символы. Таким образом, когда необходимо ввести в документ, например, символ &, то в текст на его место подставляется специальная последовательность – ссылка на данный символ.

Ссылки на символы могут быть представлены любым из указанных ниже способов:

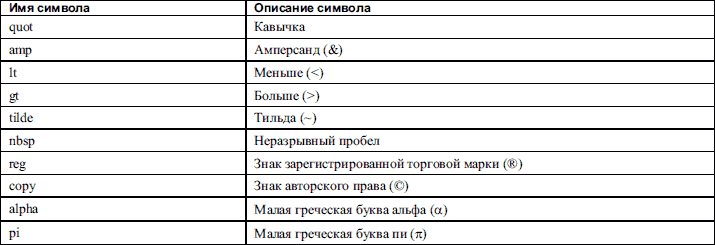
• &#D; – позволяет задать символ, код которого имеет значение D (в десятичной системе счисления);

• &#xH; – позволяет задать символ, код которого имеет значение H (в шестнадцатеричной системе счисления);

• &имя\_символа; – позволяет использовать именованную ссылку на символ.

Как можно увидеть, ссылка на символ в тексте HTML‑документа начинается символом & и заканчивается точкой с запятой (;). Особо стоит рассмотреть использование именованных ссылок на символы. Дело в том, что использование первых двух вариантов предполагает, что автору известны численные коды символов, которые он собирается добавить в документ. Однако согласитесь, что постоянно искать в таблицах численные коды нужных символов по меньшей мере неудобно. По этой причине в HTML предусмотрены имена для наиболее часто используемых символов. Эти имена и записываются вместо кода в тексте ссылок на символы. Имена некоторых часто используемых символов приведены в табл. 3.1. Полный список именованных ссылок на символы приведен в приложении 1.

**Таблица 3.1. Имена символов**



В качестве примера использования ссылок на символы в HTML‑документе можно рассмотреть следующий HTML‑код (пример 3.1).

***Пример 3.1. Использование ссылок на символы***

*<TITLE>Именованные ссылки на символы</TITLE>*

*<BODY text = "yellow" bgcolor = "blue">*

*<TITLE> Именованные ссылки на символы </*

*TITLE><BR>*

*<BODY color = " yellow "*

*bgcolor = " yellow " > <BR>*

*Текст HTML-документа <BR>*

*</BODY>*

*</BODY>*

При обработке данного кода браузером Internet Explorer получится документ, который показан на рис. 3.1.

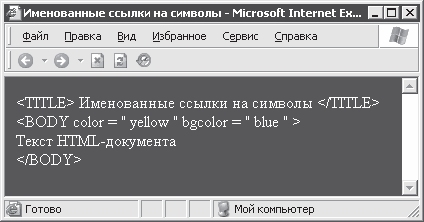


Рис. 3.1. Использование ссылок на символы

Конечно, возможности использования ссылок на символы приведенным примером не ограничены. Ссылки на символы очень часто применяются для записи математических формул. Кроме того, их можно использовать для записи текста на иностранном языке, но с использованием символов другого языка (например, для ввода символа в при использовании символов только английского языка).

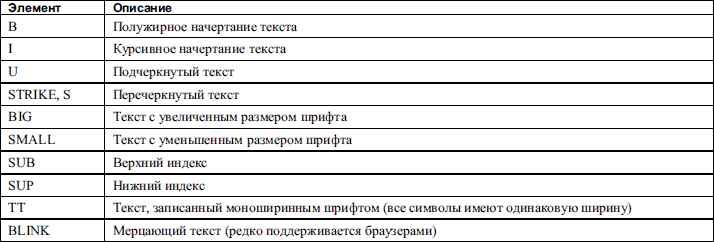
**3.2. Форматирование текста**

Для изменения вида текста, отображаемого браузером, применяется форматирование с использованием специальных HTML‑тегов. Форматирование текста HTML‑документа сходно с форматированием в любом текстовом редакторе (например, Microsoft Word). Оно позволяет выделить цветом, начертанием, изменением шрифта некоторый текст, подчеркнуть его значимость или просто украсить.

**Задание начертания текста**

Задание начертания текста является, возможно, самым простым средством форматирования содержимого документа, которое доступно в HTML. Для изменения начертания текста в HTML‑код добавляются элементы, приведенные в табл. 3.2.

**Таблица 3.2. Элементы задания начертания текста**



Для наглядности можно рассмотреть пример HTML‑документа, в котором используются различные элементы задания начертания текста (пример 3.2).

***Пример 3.2. Задание начертания текста***

*<TITLE>Задание начертания текста</TITLE>*

*<BODY>*

*<B>Полужирный текст</B><BR>*

*<I>Курсив</I><BR>*

*<U>Подчеркнутый текст</U><BR>*

*<S>Зачеркнутый текст</S><BR>*

*<BIG>Текст увеличенного размера</BIG><BR>*

*<SMALL>Текст уменьшенного размера</SMALL><BR>*

*<SUP>Верхний индекс</SUP>Текст<SUB>Нижний индекс</SUB><BR>*

*<TT>Текст, записанный моноширинным шрифтом</TT><BR>*

*</BODY>*

В приведенном коде задействованы все элементы задания начертания текста из табл. 3.2, кроме BLINK.

При обработке HTML‑кода браузером получится документ, показанный на рис. 3.2.

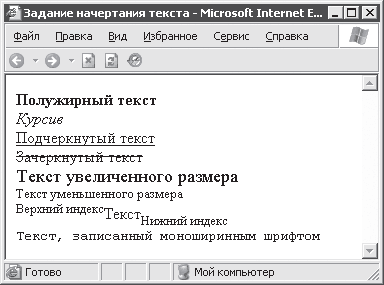


Рис. 3.2. Задание начертания текста

Рассматриваемые HTML‑элементы могут быть вложены друг в друга. При этом на начертание текста влияют все элементы, внутри которых находится текст. Например, чтобы одновременно зачеркнуть и подчеркнуть курсивный полужирный текст, можно использовать следующий код:

*<I><B><U><S>Текст непонятного начертания</S></U></B></I>*

**Задание выравнивания текста**

Сразу подчеркну, что пока рассматривается работа с неструктурированным текстом. Для задания горизонтального выравнивания текста, отличного от выравнивания, используемого браузером по умолчанию, предусмотрен элемент CENTER. Этот элемент задается парными тегами <CENTER> и </CENTER>. Текст, записанный между этими тегами, отображается в середине окна браузера.

**Сохранение авторского форматирования**

При обработке кода документа браузер заменяет все символы перехода на новую строку пробелами, а при выводе текста на экран вставляет только по одному пробелу между словами. При этом выполняется автоматический перенос непомещающихся слов на новую строку. Кроме того, при отсутствии специальных элементов текст выводится одним большим абзацем.

Что же делать, если нужно вставить в HTML‑документ текст, в котором важно сохранить хотя бы разрывы строк в определенных автором местах, например текст какой‑то программы? Рассмотрим пример 3.4.

***Пример 3.4. Страница с текстом программы (оригинал)***

*<TITLE>Текст программы (оригинал)</TITLE>*

*<BODY>*

*function fact(int num):int*

*if (num>0)*

*return fact(num–1)\*num;*

*else*

*return 1;*

*end if*

*end function*

*</BODY>*

При обработке приведенного кода браузер покажет весь текст, расположенный в элементе BODY, в одну строку так, как показано на рис. 3.4.

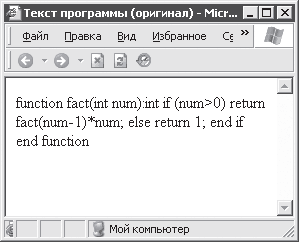


Рис. 3.4. Страница с текстом программы (оригинал)

Как видно, браузер проигнорировал все символы перевода строки, которые были в первоначальном тексте HTML‑документа, и выполнил автоматический перенос непомещающихся строк.

Для принудительного перехода на следующую строку в нужных местах текста можно использовать элемент BR, который задается одиночным тегом <BR>. Именно этот тег и использовался для повышения наглядности предыдущих примеров. Тег <BR> можно вставлять в любом месте текста, в данном случае (пример 3.5) этот тег логично вставить в конце каждой строки программы.

***Пример 3.5. Использование разрывов строк***

*<TITLE>Текст программы (с разрывами строк)</TITLE>*

*<BODY>*

*function fact(int num):int<BR>*

*if (num>0)<BR>*

*return fact(num–1)\*num;<BR>*

*else <BR>*

*return 1;<BR>*

*end if <BR>*

*end function<BR>*

*</BODY>*

Результат обработки этого варианта HTML‑кода приведен на рис. 3.5.

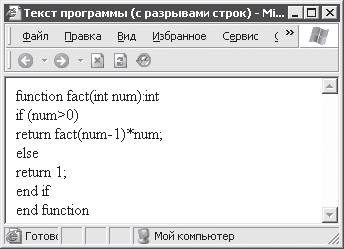


Рис. 3.5. Использование разрывов строк

Как видно, браузер разорвал строки в указанных в тексте HTML‑документа местах. Как и ранее, браузером проигнорированы все отступы (пробелы слева, показывающие уровни вложенности инструкций программы). Кроме того, если уменьшить ширину окна, то непомещающийся текст будет снова автоматически перенесен на следующие строки.

Для обычного текста использования <BR> вполне хватает. Однако в данном примере рассматривается программа, а не обычный текст. Чтобы полностью сохранить исходное форматирование текста (с учетом всех отступов), можно применить элемент PRE. Этот элемент задается парными тегами <PRE> и </PRE>. Браузер учитывает все символы, которые встречаются в тексте HTML‑документа, и выводит их на экран. Пример 3.6 иллюстрирует использование элемента PRE для сохранения оригинального форматирования текста.

***Пример 3.6. Страница с текстом программы (использование PRE)***

*<TITLE>Текст программы (PRE)</TITLE>*

*<BODY>*

*<PRE>*

*function fact(int num):int*

*if (num>0)*

*return fact(num–1)\*num;*

*else*

*return 1;*

*end if*

*end function*

*</PRE>*

*</BODY>*

Результат обработки кода примера 3.6 приведен на рис. 3.6.

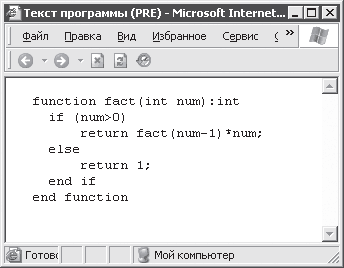


Рис. 3.6. Текст с полным сохранением форматирования

Если внимательно посмотреть на рисунок, то можно заметить, что текст отображается моноширинным шрифтом. Таков побочный эффект использования PRE.

**Запрет разрыва строки**

Иногда бывает нужно не разорвать строки в тексте документа, а, наоборот, не допустить разделения некоторых слов в строках. Для этого HTML позволяет использовать неразрывные пробелы и элементы NOBR.

Неразрывный пробел можно ввести в текст HTML‑документа с использованием соответствующей ссылки на символ:  . Например:

*Текст, отображаемый без переноса.*

Если текст, слова которого разделены неразрывными пробелами, не помещается в окне браузера, то появится горизонтальная полоса прокрутки.

Когда необходимо выделить большой участок текста, для которого недопустим автоматический перенос слов, то целесообразнее использовать элемент NOBR. Ему соответствуют парные теги <NOBR> и </NOBR>. Весь текст, находящийся между этими тегами, будет отображаться браузером в одной строке, например:

*<NOBR>*

Этот текст отображается в одной строке независимо от ширины окна браузера

*</NOBR>*

Остерегайтесь создания слишком длинных неразрывных строк, поскольку необходимость горизонтальной прокрутки для прочтения таких строк только ухудшает чтение HTML‑документа. Для вставки разрывов строк в тексте, заключенном между <NOBR> и </NOBR>, можно использовать рассмотренный ранее элемент BR.

**Переносы**

При написании текста HTML‑документа автор имеет возможность явно указать места, в которых текст может или должен переноситься на следующую строку (обычно это используется только для указания места возможного разрыва слов, потому что даже с наличием символов переноса браузер автоматически переносит целые слова так, как было рассказано ранее). Указания мест переносов в тексте достигается вставкой специальных символов переноса.

HTML предоставляет два типа переноса: простой и мягкий. Простой перенос в тексте HTML‑документа обозначается символом –. Этот символ указывает, в каком именно месте должно разрываться слово, если оно не помещается в окне. Если же слово помещается в окне, то символ переноса все равно отображается (то есть простой перенос логично использовать только для указания дефисов в тексте, как это обычно и делается). При помощи мягкого переноса можно указать браузеру место, в котором слово можно разрывать при необходимости переноса его части на следующую строку. Мягкий перенос задается ссылкой на символ ­. При использовании мягкого переноса сам символ переноса (–) отображается браузером только в случае разрыва слова в указанном месте.

Ниже приведен небольшой пример использования простого и мягкого переноса в тексте HTML‑документа (пример 3.7).

***Пример 3.7. Использование переносов***

*<TITLE>Использование переносов</TITLE>*

*<BODY>*

*Это длинное-длинное слово отображается браузером с дефисом и переносится в месте дефиса.<BR>*

*Слова этого текста могут раз­рывать­ся в указан­ных местах.<BR>*

*<NOBR>*

*А этот текст не разрывает­ся несмотря на нали­чие в нем символов пере­носа.*

*</NOBR>*

*</BODY>*

Обратите внимание: внутри элементов NOBR (как в примере 3.7) и PRE переносы браузером игнорируются. Однако внутри элемента NOBR можно указать место возможного переноса текста. Делается это при помощи одиночного тега <WBR> (элемента WBR соответственно).

Например:

*<NOBR>*

*Этот текст будет разорван браузером в указанном месте при <WBR>*

*необходимости (когда текст не поместится в окне).*

*</NOBR>*

**Обтекание текстом нетекстовых элементов**

Далее рассмотрена еще одна возможность элемента BR, которая используется при вставке в текст документа различных изображений, таблиц и прочих нетекстовых элементов (сами эти элементы будут рассмотрены позже). Таким объектам можно задать выравнивание по правому или левому краю окна браузера (объекты с таким выравниванием называются плавающими).

Текст может обтекать плавающие объекты справа или слева. При необходимости принудительного разрыва строки с использованием элемента BR можно указать, где должна появиться следующая строка, задавая значения left, right, all или none атрибуту clear этого HTML‑элемента. Расшифровка этих значений следующая:

• none – используется по умолчанию, означает, что новая строка начнется нормально, то есть с минимальным промежутком по вертикали и выравниванием по нужному краю;

• left – следующая строка начнется под плавающим объектом, выровненным по левому краю (если объект выровнен по правому краю, то действует аналогично none);

• right – аналогично значению left, только наоборот;

• all – следующая строка начнется под плавающим объектом независимо от края, по которому выровнен объект.

**3.3. Структурирование текста**

Все, что рассматривалось ранее, – это простое физическое форматирование текста. Теперь пришло время заняться разбиением текста HTML‑документа на логически цельные, но различные между собой части, – сформировать структуру документа. В этом разделе будут рассмотрены основные возможности структурирования документа с использованием соответствующих HTML‑элементов, а также то, как роль текста в структуре документа влияет на его отображение.

**Разбиение на абзацы**

В предыдущих примерах весь текст HTML‑документов даже при наличии в нем элементов BR воспринимался браузером как один абзац. Чтобы действительно отделить абзацы текста друг от друга, используется специальный элемент P.

Элемент P задается при помощи парных тегов <P> и </P>. При этом следует отметить, что использование закрывающего тега </P> считается не просто необязательным, но даже нежелательным. При отсутствии закрывающего тега концом элемента P считается начало следующего абзаца (следующий тег <P>) или тег конца документа, если абзац последний.

Для элемента P можно задать несколько атрибутов. Список наиболее используемых атрибутов:

• align – задает горизонтальное выравнивание содержимого абзаца, может принимать значения: left (используется по умолчанию), right, center, justify;

• title – задает текст подсказки.

К тексту абзаца может применяться любое форматирование, однако оно не должно нарушать восприятие абзаца как единого целого. Обычно сами абзацы браузерами визуально отделяются друг от друга. Далее приведен небольшой пример, в котором используется разбиение текста на абзацы (пример 3.8).

***Пример 3.8. Использование абзацев***

*<TITLE>Разбиение текста на абзацы</TITLE>*

*<BODY>*

*<P title = "Первый абзац">*

*Неформатированный текст*

*<P align = "right" title = "второй абзац">*

*Текст с <B>изменением <I>начертания</I></B>*

*<P align = "center" title = "третий абзац">*

*<FONT size = "+2" face = "arial">Текст с измененным шрифтом</FONT>*

*<P align = "justify" title = "четвертый абзац">*

*Текст этого абзаца автоматически выравнивается по ширине справа и слева при переносе слов*

*</BODY>*

При обработке приведенного HTML‑кода получится документ, показанный на рис. 3.7.

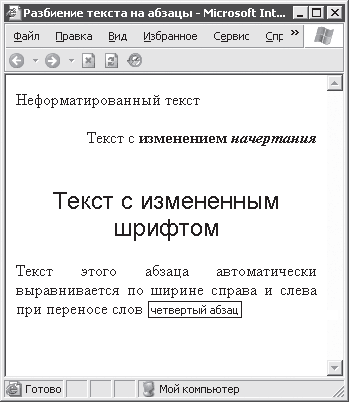


Рис. 3.7. Использование различного оформления абзацев

При наведении указателя мыши на текст абзаца появляется подсказка, заданная атрибутом title.

**Заголовки**

Следующим важным этапом в структурировании HTML‑документа является использование заголовков (в их обычном понимании) для обозначения начала больших фрагментов текста.

В HTML поддерживаются шесть видов заголовков. Им соответствуют элементы H1, H2, H3, H4, H5 и H6. Номера определяют уровни заголовков от наиболее важного (1) до наименее важного (6). Элементы H1–H6 задаются при помощи соответствующих парных тегов. Например, для задания заголовка третьего уровня можно применить следующий код:

*<H3>Текст заголовка третьего уровня</H3>*

Для заголовков можно задать свойства:

• align – выравнивание текста заголовка (по умолчанию используется выравнивание по левому краю);

• title – текст подсказки.

Заголовки различных уровней обычно отображаются браузерами различными шрифтами и различного размера. При этом размер шрифта более важных заголовков обычно больше, чем размер шрифта менее важных.

На рис. 3.8 приведен пример того, как отображаются заголовки различного уровня (HTML‑код не приводится ввиду его чрезвычайной простоты).

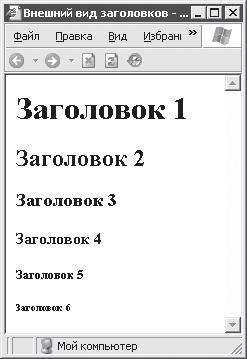


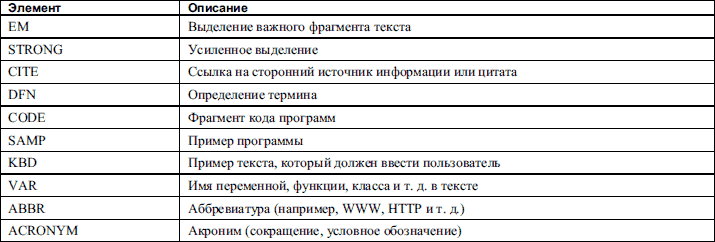
Рис. 3.8. Вид заголовков в Internet Explorer

В тексте, который заключен между тегами начала и конца заголовка, можно использовать рассмотренные ранее элементы форматирования (например, изменить шрифт). Однако это крайне не рекомендуется. Ведь стандартные заголовки придуманы именно для того, чтобы сделать похожими друг на друга HTML‑документы различных авторов, чтобы пользователь при просмотре документа браузером не терялся в догадках относительно роли того или иного текста.

**Задание типов фраз**

Даже в разбитом на отдельные абзацы тексте смешаны различные по важности и смыслу участки текста. Чтобы их можно было отделить друг от друга, при написании HTML‑документа можно пользоваться специальными элементами, задающими типы фраз в тексте (или, как еще говорят, элементами логического форматирования текста). Описания этих HTML‑элементов приведены в табл. 3.3.

**Таблица 3.3. Элементы задания типов фраз**



Первые два элемента используются для подчеркивания важности какого‑либо отрезка текста. Остальные элементы используются в основном в технических текстах. Все приведенные элементы задаются при помощи соответствующих парных тегов. Текст подсказки к каждому из этих элементов задается при помощи атрибута title. Наиболее часто этот атрибут используют с элементами ABBR и ACRONYM для отображения в подсказке полной расшифровки сокращения или аббревиатуры.

Естественно, что логическое разделение текста при помощи элементов из табл. 3.3 отражается на его представлении браузером (правда, разные браузеры могут отображать одинаковый по значению текст различным образом). В примере 3.9 приведен текст HTML‑документа, использующий все доступные типы фраз.

***Пример 3.9. Использование различных типов фраз***

*<TITLE>Использование различных типов фраз</TITLE>*

*<BODY>*

*Обычный текст <BR>*

*Выделение: <EM>Важный текст</EM><BR>*

*Сильное выделение: <STRONG>Это очень важный текст</STRONG><BR>*

*Цитата, ссылка: <CITE>см. стандарт ISO 3273</CITE><BR>*

*Определение: <DFN>WWW – это...</DFN><BR>*

*Пользователь должен ввести:<KBD>Пример ввода пользователя</KBD><BR>*

*Переменная: <VAR>strText</VAR><BR>*

*Аббревиатуры: <ABBR>HTTP, WWW, FTP</ABBR><BR>*

*Сокращения: <ACRONYM>Внешпосылторг, UNIBEL</ACRONYM><BR>*

*Фрагмент программы в тексте:<CODE>CALL main</CODE><BR>*

*Пример программы:<SAMP>void main(){return;}</SAMP><BR>*

*</BODY>*

На рис. 3.9 приведен внешний вид страницы, сгенерированной при обработке текста примера 3.9.

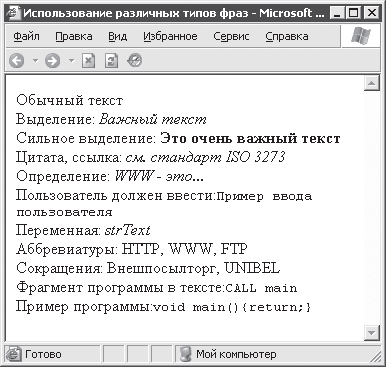


Рис. 3.9. Внешний вид фраз различного типа

Как можно видеть из рисунка, большинство фраз различного типа отображаются браузером Internet Explorer совершенно одинаково. Не стоит полагать, что это отображение нельзя изменить. Можно использовать, например, рассмотренные ранее способы форматирования. Однако при этом польза от элементов, задающих типы фраз, становится весьма сомнительной. Для изменения внешнего текста HTML‑элементов настоятельно рекомендуется использовать таблицы стилей, которые будут рассмотрены позже.

Стоит также отметить, что сохранить оригинальное форматирование примера программы или прочего текста внутри элемента SAMP весьма проблематично, в отличие от того же элемента PRE. Именно поэтому в примере 3.9 текст программы выбран таким образом, чтобы он мог быть помещен в одну строку без потери его читабельности.

**Цитаты**

Кроме элемента CITE, для оформления цитат могут также использоваться элементы BLOCKQUOTE и Q. Для их задания используются соответствующие парные теги.

Элемент BLOCKQUTE используется для задания длинных цитат и представляет собой элемент уровня блока. При этом текст цитаты обычно оформляется браузерами как текст с дополнительным отступом.

Элемент Q является элементом уровня текста и используется для создания небольших цитат. Текст, помещенный между тегами <Q> и </Q>, оформляется различными браузерами по‑разному (например, может быть помещен в кавычки).

Атрибуту cite элементов BLOCKQUOTE и Q можно присвоить URI источника информации для цитаты.

**Указание изменений в документе**

В ряде случаев возникает необходимость изменять важное содержимое HTML‑документа уже после того, как он опубликован (например, если речь идет о разрабатываемом законопроекте). В таких случаях практически незаменимыми являются HTML‑элементы INS и DEL, применяемые для выделения участков текста, которые были добавлены или удалены в новой версии документа.

Элемент INS задается при помощи парных тегов <INS> и </INS>, между которыми помещается добавленный текст. Элемент DEL задается парными тегами <DEL> и </DEL>. В эти теги заключается текст, удаленный из новой версии документа.

Естественно, что содержимое элементов INS и DEL отображается браузерами совершенно поразному. Содержимое элемента DEL может быть, например, зачеркнутым или вообще не показываться браузером.

Элементы INS и DEL могут содержать как небольшие участки текста, так и целые разделы документа. Однако недопустимым является вложенность этих элементов друг в друга.

*<INS><DEL>Никогда так не делайте</DEL></INS>*

Наиболее часто используемыми атрибутами элементов INS и DEL являются следующие:

• cite – задает URI HTML‑документа с пояснениями сделанных изменений;

• datetime – время, когда в документ были внесены изменения.

Здесь вы впервые встретились с заданием атрибуту значения даты и времени. Это значение в HTML задается в следующей форме:

*ГГГГ–ММ–ДДTчч:мм:cc*

Здесь ГГГГ обозначает год, ММ – месяц, ДД – день, чч – значение от 0 до 23 (час), мм и сс – значения от 0 до 59 (минуты и секунды). Кроме того, в конец значения даты и времени добавляется одна из следующих записей:

• Z – обозначает, что используется UTC‑время (общее скоординированное время, или время по Гринвичу);

• +чч:мм или –чч:мм – обозначает, что местное время опережает или отстает от UTC на заданное количество часов и минут.

Ниже приведен пример различных вариантов задания московского времени 14 часов 5 минут 31 декабря 1997 года (с использованием местного времени и времени UTC):

*1997–12–31T14:05+03:00*

*1997–12–31T11:05Z*

**Контактная информация**

В HTML предусмотрен специальный элемент ADDRESS, в который может заключаться различная контактная информация. Он задается при помощи парных тегов <ADDRESS> и </ADDRESS>.

Текст внутри элемента ADDRESS может иметь произвольную структуру, однако чаще всего в него помещаются имена сотрудников организаций, ссылки на другие HTML‑документы, телефон или адрес электронной почты контактного лица.

**3.4. Комментарии в HTML-коде**

При написании достаточно сложных HTML‑документов зачастую бывает полезно иметь возможность вставлять в исходный текст небольшие или развернутые комментарии. Добавление в сложный HTML‑документ грамотных комментариев позволяет сэкономить уйму времени при необходимости, например, повторного редактирования документа месяц спустя.

Текст комментария помещается между символами <!– и –>. При обработке HTML‑документа текст комментария игнорируется браузером. Комментарии могут быть как однострочными, так и многострочными:

*<!– Это пример однострочного комментария –>*

*<!– А это*

*многострочный комментарий–>*

Важным моментом является то, что между <! и – нельзя вставлять пробелы. Кроме того, следует избегать использования в тексте комментария двух и более символов переноса (–).

В завершение еще одно замечание. Перед опубликованием HTML‑документа лучше удалить из исходного текста все комментарии (особенно, если автор этого документа любит оставлять себе большие подробные послания на будущее). Если объем комментариев достаточно большой, то, удалив их, можно уменьшить время загрузки HTML‑документа с сервера.