

Дмитрий Кудрец
Основы языка HTML. Часть первая

Дмитрий Кудрец

Основы языка HTML

часть первая



ISBN 9785449621658

Аннотация

В книге рассказывается о назначении и возможностях языка HTML. Рекомендуется учащимся школ, гимназий, а также всем желающим освоить основы языка.

Основы языка HTML Часть первая

Дмитрий Кудрец

© Дмитрий Кудрец, 2019

ISBN 978-5-4496-2165-8 (т. 1)

ISBN 978-5-4496-2166-5

Создано в интеллектуальной издательской системе Ridero

История развития языка HTML

Начало истории HTML относится к 1969 году, когда Чарльз Гольдфарб, работающий в компании IBM, создал прототип языка для разметки технической документации, впоследствии названного GML, а с приданием ему в 1986 году статуса международного стандарта – SGML (**Standard Generalized Markup Language**). Этот обобщенный метаязык предназначен для построения систем логической, структурной разметки любых разновидностей текстов. Слово «структурная» означает, что управляющие коды, вносимые в текст при такой разметке, не несут никакой информации о форматировании документа, а лишь указывает границы и соподчинения его составных частей, т.е. задают его структуру. Однако сам по себе SGML не получил сколько-нибудь заметного распространения до тех пор, пока в 1991 г. сотрудники европейского института физики частиц (CERN), занятые созданием системы передачи гипертекстовой информации через Интернет, не выбрали SGML в качестве основы для нового языка разметки гипертекстовых документов. Этот язык – самое известное из приложений SGML – был назван **HTML (Hyper Text Markup Language)** – язык разметки гипертекста).

Гипертекст – информационная структура, позволяющая устанавливать смысловые связи между элементами текста на экране компьютера таким образом, чтобы можно было легко осуществлять переходы от одного элемента к другому. На практике в гипертексте некоторые слова выделяют путем подчеркивания или окрашивания в другой цвет (гиперссылки). Выделение слова говорит о наличии связи этого слова с некоторым документом, в котором тема, связанная с выделенным словом рассматривается более подробно.

В 1991 – 1992 годах в стенах Европейского совета по ядерным исследованиям в Женеве (Швейцария) разработку языка возглавил Тим Бернерс-Ли. Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). Текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). С течением времени, основная идея платформонезависимости языка HTML была отдана в жертву современным потребностям в мультимедийном и графическом оформлении.

Текстовые документы, содержащие код на языке HTML, обрабатываются специальными приложениями, которые отображают документ в его форматированном виде. Такие приложения, называемые «браузерами», предоставляют пользователю удобный интерфейс для запроса веб-страниц, их просмотра и, при необходимости, отправки введенных пользователем данных на сервер. Наиболее популярными на сегодняшний день браузерами являются Internet Explorer, Mozilla Firefox и Opera.

Версии языка HTML

Версия **HTML 1.0** была предельно проста – кроме тегов для записи гипертекстовых переходов, она предусматривала лишь несколько тегов для логической разметки текста и один тег **IMG** для записи ссылок на файлы с картинками-иллюстрациями. Официальной спецификации HTML 1.0 не существует. До 1995 года существовало множество неофициальных стандартов HTML. Чтобы стандартная версия отличалась от них, ей сразу присвоили второй номер.

Развитие графических средств вывода информации провоцировало появление в следующих версиях HTML многочисленных тегов, позволяющих непосредственно управлять видом элемента на экране. Описанием стандартов HTML, начиная с **версии 2.0** (1994 года), стала заниматься организация W3C (World Wide Web Consortium – консорциум Всемирной паутины). 22 сентября 1995 года HTML 2.0 был одобрен как стандарт языка.

В 1995 году Консорциум ввел в **HTML версии 3.0** поддержку иерархических стилевых спецификаций CSS (Cascading Style Sheets – каскадные таблицы стилей). CSS – это разрешение противоречий между идеологией структурной разметки и потребностями разработчиков в гибких и богатых средствах визуального представления элементов на экране. Язык CSS имеет свой собственный синтаксис и позволяет задавать визуальные параметры представления элементов на экране. Таким образом, CSS берет на себя задачу объяснить браузеру, как отображать элементы на экран, и позволяет тем самым отделить структурную разметку документа (в HTML-коде) от описаний визуальных свойств объектов (в CSS-коде).

Версия 3.1 официально никогда не предлагалась, и следующей версией стандарта HTML стала **3.2** (14 января 1997) в которой были опущены многие нововведения версии 3.0, но добавлены много новых возможностей, таких как создание таблиц, «обтекание» изображений текстом и отображение сложных математических формул, нестандартные элементы, поддерживаемые браузерами «Netscape» и «Mosaic».

В 1999 году W3C опубликовала спецификации версии **HTML версии 4.0** (18 декабря 1997). Она содержала много элементов, специфичных для отдельных браузеров, но в то же время произошла некоторая «очистка» стандарта. Многие элементы были отмечены как устаревшие и nereкомендованные. В частности, элемент **FONT**, используемый для изменения свойств шрифта, был помечен как устаревший (вместо него рекомендуется использовать таблицы стилей CSS). Затем был опубликован стандарт **HTML 4.01** (24 декабря 1999). На этом развитие языка HTML закончилось окончательно.

Черновой вариант **HTML 5** появился в Интернете 20 ноября 2007 года. Параллельно велась работа по дальнейшему развитию HTML под названием XHTML (Extensible Hypertext Markup Language – «расширяемый язык разметки гипертекста»). XHTML по своим возможностям сопоставим с HTML, однако предъявляет более строгие требования к синтаксису.

XHTML создан для хранения структурированной информации. Названия тегам задает разработчик и выбирает их так, чтобы они соответствовали содержанию элемента. Теги в могут содержать атрибуты. Имена атрибутов набор их значений также задаются разработчиком. Язык XHTML является универсальным средством для создания структур данных, которые можно использовать для самых разных надобностей, в том числе, хранить

в XML – файле настроенные данные для компьютерной программы или использовать для создания гипертекстовой страницы. На базе XML можно строить другие языки разметки.

Вариант **XHTML 1.0** был одобрен в качестве рекомендации Консорциума всемирной паутины 26 января 2000 года.

Спецификация **XHTML 2.0** разрывает совместимость со старыми версиями HTML и XHTML. Группой WHATWG (Web Hypertext Application Technology Working Group) разрабатывалась спецификация Web Applications 1.0, часто неофициально называемая HTML 5, которая расширяет HTML для лучшего представления семантики различных типичных страниц, которые не очень удачно вписываются в модель XHTML 2.

На уровне первой версии языка XHTML не дает практически никаких преимуществ по сравнению с последней версией языка HTML, но синтаксические требования в языке HTML существенно выше, что, с одной стороны, усложняет использование этого языка, а с другой – дисциплинирует разработчиков, которые привыкли относиться к гипертекстовому коду небрежно, считая, что браузер додумает за них (что он и делает, исправляя ошибки в HTML – коде по своему усмотрению).

Завершение эпохи совершенствования HTML не означает отказ разработчиков от использования этого языка. HTML продолжает свою жизнь и в качестве подмножества XHTML, и в «частном» виде – как простой инструмент разработки сайтов.

Основные понятия языка HTML

Документ, выполненный в формате HTML, называется **HTML-документом**, **web-документом** или **web-страницей**. Такие страницы, как правило, имеют расширение **htm** или **html**.

Группа Web-страниц, взаимосвязанных общими гиперссылками, образует структуру, которая называется **Web-узлом** или **Web-сайтом**.

HTML-документы можно создавать в любом текстовом редакторе. HTML-документ является простым текстовым файлом, который содержит текст и HTML-теги.

Теги HTML

Тег – это команда HTML, указывающая браузеру, каким образом он должен обрабатывать соответствующее значение. Это значение называется **атрибутом тега**.

При отображении документа в браузере сами теги не отображаются, но влияют на способ отображения документа. Когда HTML-документ открывается в браузере, он просматривает HTML-код, находит в нем теги, и использует их для вставки изображений, изменения вида текста, создания ссылок на другие страницы и др.

Тег HTML состоит из следующих друг за другом в определенном порядке элементов:

- левой угловой скобки < ;
- слэша (/), который означает, что тег является конечным, закрывающим некоторую структуру;

- имени тега;

- необязательных атрибутов;

- правой угловой скобки > .

Например: **<H1>**, **<H1 ALIGN=LEFT>**;

В общем виде тег можно записать так:

<ТЕГ АТРИБУТ_1=ЗНАЧЕНИЕ_1 АТРИБУТ_2=ЗНАЧЕНИЕ_2 ... АТРИБУТ_N=ЗНАЧЕНИЕ_N> Обработываемое значение **</ТЕГ>**;

HTML-теги могут быть условно разделены на две категории:

- теги, определяющие, как будет отображаться браузером тело документа в целом;

- теги, описывающие общие свойства документа, такие как заголовок или автор

документа.

Большинство тегов являются парными. Это означает, что за открывающим тегом следует соответствующий закрывающий тег, а между ними содержится текст или другие теги.

Например: **<H1> Foreword </H1>**;

В таких случаях два тега и часть документа, отделенная ими, образуют блок, называемый **HTML-элементом**.

Открывающий тег создает эффект, а закрывающий – прекращает его действие. Данное свойство HTML позволяет использовать принцип вложения одного тега в другой, когда обрабатываемым значением одной команды может служить другая команда.

Например, **<ТЕГ1 > <ТЕГ2 >** Обрабатываемое значение **</ТЕГ2 >**; **</ТЕГ1 >**;

В зависимости от того, где вы поставили закрывающий тег, отображение данных в браузере может быть различным.

Регистр букв в написании тегов не имеет значения, их можно вводить как большими, так и маленькими буквами. Например, **<body>**, **<BODY>** и **<Body>** будут восприняты браузером одинаково, хотя общепринято использовать прописные буквы, чтобы теги отличались от обычного текста документа.

Если по ошибке в теге указано ключевое слово, отсутствующее в языке HTML, то тег игнорируется целиком.

При работе с кодом HTML необходимо запомнить одно правило: если где-то в тексте программы встречается открывающий тег, обязательно должен присутствовать и закрывающий. Несоблюдение этого правила вызовет ошибку при обработке такого документа интерпретатором браузера.

Дополнительные пробелы, символы табуляции, добавленные в исходный текст HTML-документа для его лучшей читаемости, браузером игнорируются за исключением случаев, когда они помещены внутри тэгов **<PRE>** и **</PRE>**.

Некоторые теги, например **<HR>**, являются непарными. Т.е. у них нет закрывающегося тега.

Атрибуты тегов

Открывающие теги часто могут содержать атрибуты, позволяющие расширить возможности тега. Большинство тегов допускает один или несколько атрибутов, однако атрибутов может и совсем не быть.

В общем случае такое выражение выглядит следующим образом:

<ТЕГ АТРИБУТ=«ЗНАЧЕНИЕ_1; „ЗНАЧЕНИЕ_2“; ... ЗНАЧЕНИЕ_N»>;

Например, **<H1 ALIGN=«LEFT»>**;

Значения атрибутов заключаются в прямые кавычки (») и отделяются друг от друга пробелами или незаполненными строками.

Если внутри атрибута какого-либо тега встречается другое значение, заключенное в кавычки, то есть имеет место вложение одних кавычек в другие, в качестве внутренних кавычек рекомендуется использовать одинарные (') .

Кавычки в записи атрибутов можно и не использовать. В основном это относится для атрибутов, которые состоят из следующих символов:

- символов английского алфавита;
- цифр;
- промежутков времени;
- дефисов.

Тег может и не иметь атрибутов. Например, тег **<HTML>**. Также не имеют атрибутов и закрывающие теги.

Также следует отметить, что некоторые значения атрибутов могут выступать и в роли

тегов. Например, запись **<H1 ALIGN=«LEFT »>** аналогична записи **<H1>**
<LEFT>;

Классификация тегов HTML

Все теги HTML можно классифицировать на три основные категории:

– **заголовочные**, содержащие информацию о документе в целом: HEAD, TITLE, ISINDEX, BASE, META, LINK, SCRIPT, STYLE;

– **блоковые**, организующие структуру документа: H1, H2, H3, H4, H5, H6, ADDRESS, P, UL, OL, DL, PRE, DIV, CENTER, BLOCKQUOTE, FORM, ISINDEX, HR, TABLE;

– **текстовые**, включающие: Escape-последовательности (например, &amp;); выражения разметки: EM, STRONG, DFN, CODE, SAMP, KBD, VAR, CITE; разметку шрифта: I, B, U, STRIKE, BIG, SMALL, SUB, SUP; специальные элементы: A, IMG, APPLET, FONT, BASEFONT, BR, SCRIPT, MAP; элементы формы: INPUT, SELECT, TEXTAREA.

Блоковые теги могут содержать как текстовые, так и другие блочные теги, т.е. блоки могут быть вложенными. Текстовые элементы также могут быть вложенными. Но текстовые элементы не могут включать блочные элементы.

Например, выражение **<CITE>** **<H3>** Текст **</H3>** **</CITE>** неверно (так как **CITE** – текстовый элемент, а **H3** – блочный элемент). Тогда как запись **<H3>** **<CITE>** Текст **</CITE>** **</H3>** правильная, хотя отдельные браузеры воспринимают это с трудом.

Единицы измерения

В языке HTML существует два способа задать линейные размеры элемента: пиксели и проценты. В CSS единиц длины гораздо больше.

Условно единицы измерения можно разделить на три группы.

Первая группа – это величины, которые используются для измерения длин реальных предметов. К ним относятся:

- **in** – дюймы;
- **cm** – сантиметры;
- **mm** – миллиметры.

Ко второй группе можно отнести величины, которые пришли в CSS из типографии. То есть они используются для установки размеров шрифта, межстрочных интервалов и прочих типографских величин. К ним относятся:

- **pt** – типографский пункт;
- **pc** – пика;
- **ex** – высота строчной буквы «x» в шрифте.

Третью группу составляют величины, которые являются относительными, то есть реальный размер элемента вычисляется относительно какой-либо иной величины. К ним относятся:

- **em** – вычисляется относительно размера шрифта элемента;
- **px** – пиксель;
- **%** – процент.

Сохранение и проверка HTML-документа

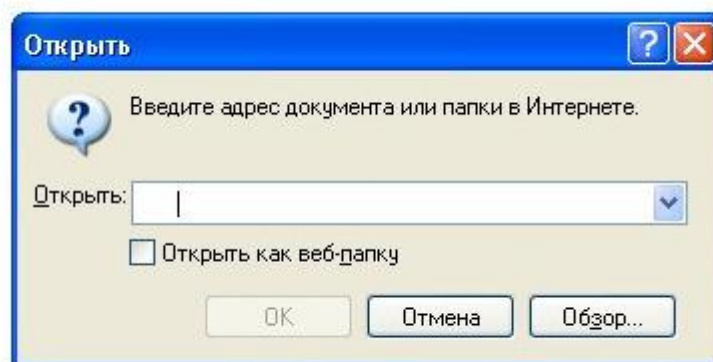
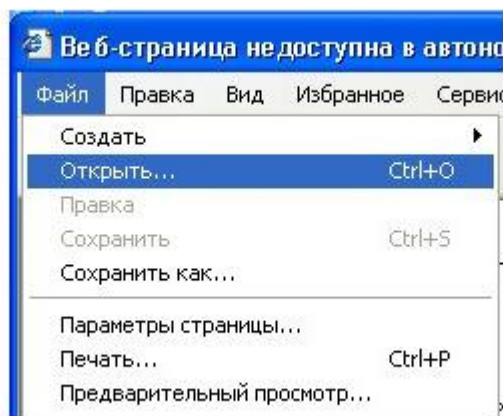
Работа над созданием страницы подразумевает три важных шага: во-первых, нужно ее правильно сохранить; во-вторых, убедиться в том, что в браузере вы видите то, что задумывалось; в-третьих, нужно проверить код на совместимость его со стандартами HTML.

Обычно для создания HTML-документов используется программа Блокнот. Для того

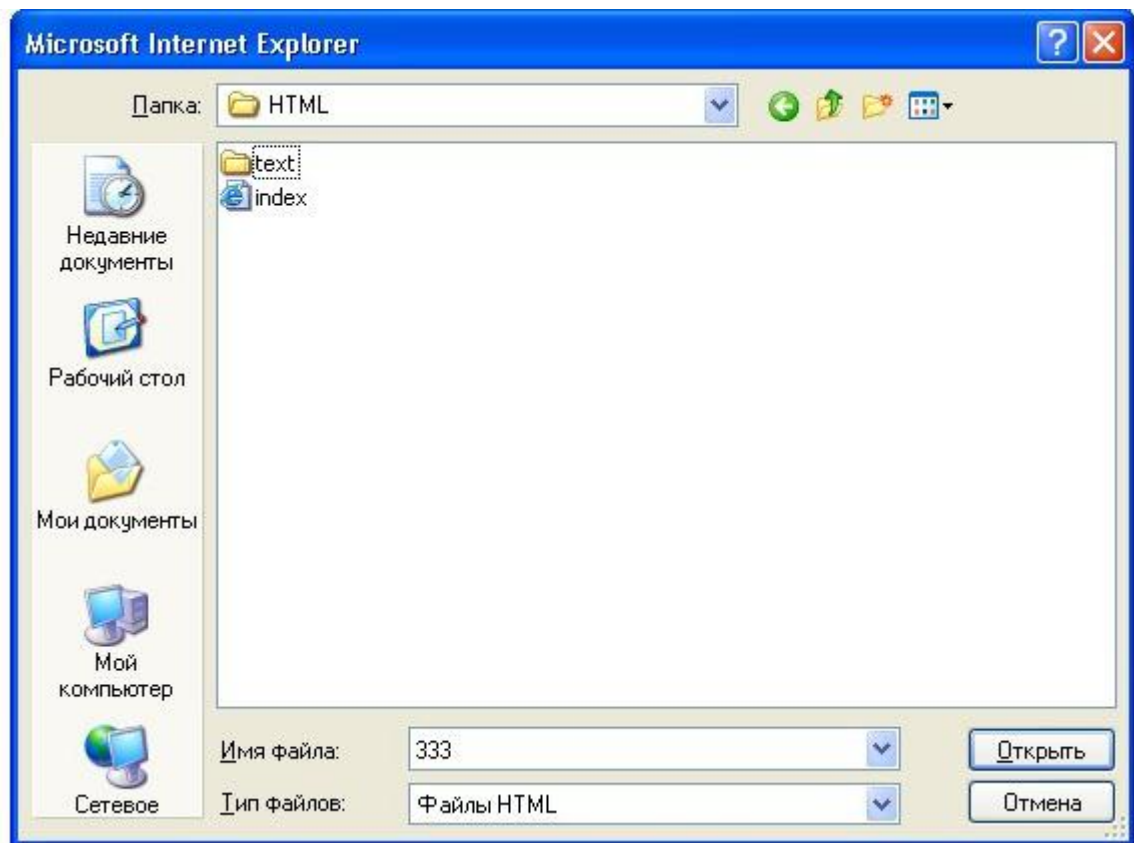
чтобы документ можно было просмотреть в браузере его сохраняют под любым именем, указав расширение. **НТМ** или **HTML** .

Чтобы просмотреть созданный HTML-документ в окне браузера, необходимо выполнить следующие действия:

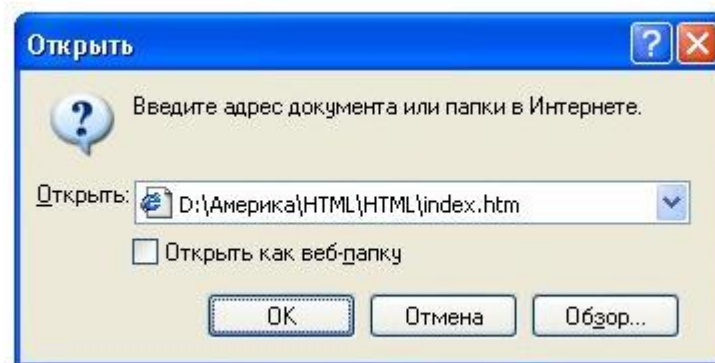
1. Открыть любой браузер (например, Internet Explorer);
2. В меню **Файл** браузера выбрать команду **Открыть** ;



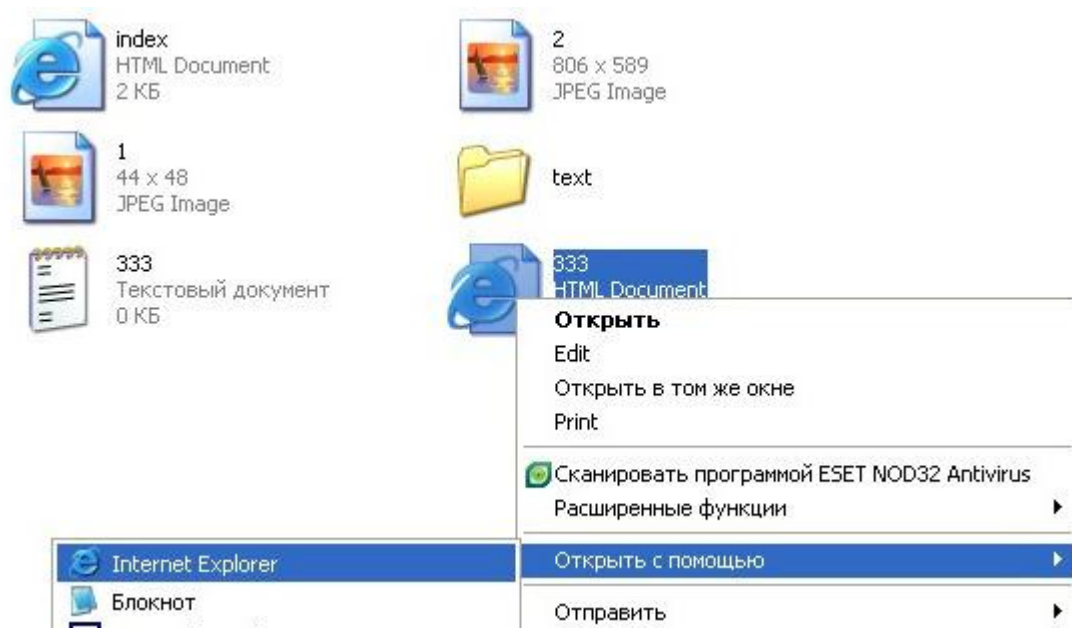
3. Нажать кнопку **Обзор** и выбрать нужный файл;



4. Щелкнуть мышью на кнопке **ОК** .

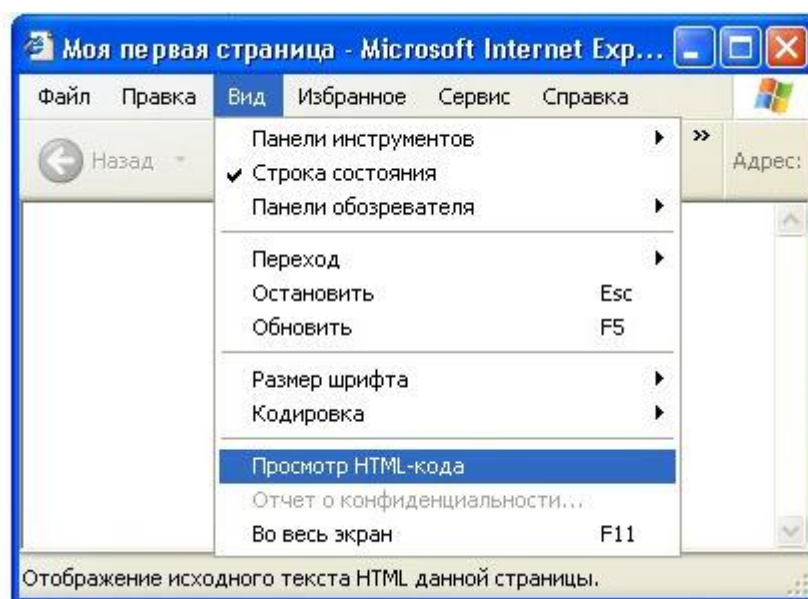


Также для открытия HTML-документа в браузере можно использовать контекстное меню – команду **Открыть с помощью...** , указав приложение Internet Explorer или любой другой браузер.

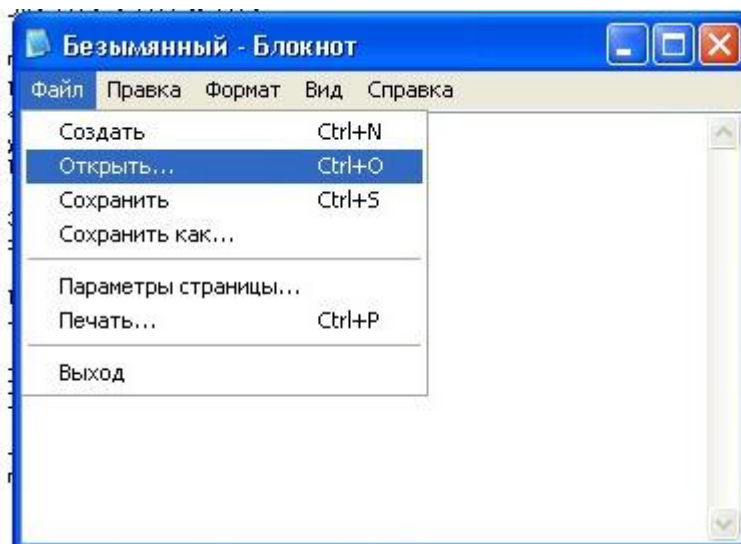


В принципе, документы, сохраненные в HTML-формате открываются в браузере автоматически.

Для просмотра HTML-кода созданного документа используется команда **Вид – Просмотр HTML-кода**.



Использование данной команды гарантирует, что вы просмотрите код именно открытого документа. Также можно просмотреть HTML-код документа и через Блокнот, с помощью команды меню **Файл – Открыть**.



После внесения изменений файл необходимо сохранить. Для этого используется команды меню **Файл – Сохранить** или комбинация клавиш **CTRL+S**.

Чтобы в браузере просмотреть сохраненные изменения, необходимо нажать кнопку **Обновить** или клавишу **F5**.

Для создания документов также широко используются различные HTML-редакторы, например FrontPage.

Сохраненную и протестированную страницу необходимо проверить на ее соответствие стандартам. Это не обязательно, но желательно. Существуют приложения, автоматически определяющие, насколько точно соблюдаются рекомендации на совместимость с HTML. Если страница выдерживает проверку, значит, она полностью совместима со стандартом и не содержит, по крайней мере, синтаксических ошибок, влияющих на ее внешний вид. Если же проверка не проходит, значит, нужно искать где-то опечатку, либо какой-то из элементов не полностью удовлетворяет стандартам. Большинство программ автоматической проверки генерируют такой отчет об ошибках, что вы сможете без труда их отыскать.

Цветовое оформление HTML-документа

Цвет и фон HTML-документа, шрифта, ячеек таблицы и других элементов выбираются по желанию дизайнера, но нужно учитывать и тот факт, что от них зависит визуальное восприятие страницы. При оформлении стараются не использовать более трех различных цветов. Исключение здесь можно сделать для полутонов одного и того же цвета.

При подборе цветового сочетания, например текста с фоном, рекомендуется исходить из того, что текст должен без труда читаться.

Ниже приведен перечень цветовых сочетаний в порядке ухудшения зрительного восприятия:

- синее на белом;
- черное на желтом;
- зеленое на белом;
- черное на белом;
- зеленое на красном;
- красное на желтом;
- красное на белом;
- оранжевое на черном;
- черное на пурпурном;
- оранжевое на белом;

— красное на зеленом.

Корректность сочетания друг с другом остальных цветов и оттенков проверяется с помощью одного простого правила: переведите ваше изображение в формат GRAYSCALE (256 оттенков серого) и посмотрите, читается ли в таком виде ваш текст, контрастно ли выглядят нарисованные элементы. Если нет — принятое цветовое решение лучше пересмотреть. В любом случае для текста рекомендуется выбирать традиционный черный цвет. В качестве фона лучше всего использовать тусклую, едва различимую заливку произвольного оттенка.

Цвета фона и основного текста должны быть контрастными. Чем контрастнее, тем удобнее будет читать текст. Удобочитаем чёрный текст на белом, белый на чёрном, жёлтый на синем. На тёмном фоне лучше всего использовать белый или жёлтый цвета, можно и зелёный использовать. Голубой читается сложнее на белом фоне.

Цвет элементов HTML-документа в основном задается с помощью строчных литералов или в виде шестнадцатеричного кода.



Использование строчных литералов

Строчные литералы представляют собой названия цветов, которые распознаются всеми браузерами.

Цвет	Название	Цвет	Название	Цвет	Название	Цвет	Название
	White		Red		Orange		Yellow
	Green		Blue		Purple		Black
	Aliceblue		Antiquewhite		Aqua		Aquamarine
	Blueviolet		Brown		Burlywood		Cadetblue
	Chartreuse		Chocolate		Coral		Cornflowerblue
	Cornsilk		Crimson		Cyan		Darkblue
	Darkcyan		Darkgoldenrod		Darkgray		Darkgreen
	Darkorchid		Darkred		Darksalmon		Darkseagreen
	Darkslateblue		Darkslategray		Darkturquoise		Darkviolet
	Deeppink		Deepskyblue		Dimgray		Dodgerblue
	Gainsboro		Ghostwhite		Gold		Goldenrod
	Gray		Greenyellow		Honeydew		Hotpink
	Indianred		Indigo		Ivory		Khaki
	Lightgray		Lightpink		Lightsalmon		Lightseagreen
	Lime		Limegreen		Linen		Magenta
	Maroon		Mediumaquamarine		Mediumblue		Mediumorchid
	Olive		Olivedrab		Orangered		Orchid
	Papayawhip		Peachpuff		Peru		Pink
	Silver		Skyblue		Slateblue		Slategray
	Snow		Springgreen		Steelblue		Tan
	Violet		Wheat		Whitesmoke		Yellowgreen

Шестнадцатеричные значения цвета

Перечень строчных литералов с названиями цветов довольно обширен, но при оформлении HTML-документов чаще используют шестнадцатеричные значения цвета.

Цвет	Код	Цвет	Код	Цвет	Код	Цвет	Код	Цвет	Код	Цвет	Код
	#010000		#800000		#000100		#008000		#000001		#000080
	#100000		#900000		#001000		#009000		#000010		#000090
	#200000		#A00000		#002000		#00A000		#000020		#0000A0
	#300000		#B00000		#003000		#00B000		#000030		#0000B0
	#400000		#C00000		#004000		#00C000		#000040		#0000C0
	#500000		#D00000		#005000		#00D000		#000050		#0000D0
	#600000		#E00000		#006000		#00E000		#000060		#0000E0
	#700000		#FF0000		#007000		#00FF00		#000070		#0000FF

Безопасная палитра цветов

Не все браузеры правильно передают цвет. Дело в том, что браузер всегда старается подстроить цветовую палитру документа под системные настройки и возможности монитора, путем самостоятельного их замещения. Избежать этой ситуации помогает применение безопасной палитры, каждый цвет которой передается одинаково всеми браузерами. В эту палитру входят цвета, принимающие значения: **00, 33, 66, 99, CC, FF**, во всех их сочетаниях.

Цвет	Код	Цвет	Код	Цвет	Код	Цвет	Код	Цвет	Код	Цвет	Код
	FFFFFF		CCCCCC		999999		666666		333333		000000
	CCCC66		CCCC33		999966		999933		999900		666600
	CCFF66		CCFF00		CCFF33		CCCC99		666633		333300
	99FF00		99FF33		99CC66		99CC00		99CC33		669900
	CCFF99		99FF99		66CC00		66CC33		669933		336600
	66FF00		66FF33		33FF00		33CC00		339900		009900
	CCFFCC		99CC99		66CC66		669966		336633		003300
	99FF99		66FF66		33FF66		00FF66		339933		006600
	66FF99		33FF99		00FF99		33CC66		00CC66		009933
	66CC99		33CC99		00CC99		339966		009966		006633
	99FFCC		66FFCC		33FFCC		00FFCC		33CCCC		009999
	CCFFFF		99FFFF		66FFFF		33FFFF		00FFFF		00CCCC
	99CCCC		66CCCC		339999		669999		006666		336666
	0066FF		3366CC		0066CC		0033FF		003399		003366
	6699FF		3366FF		0000FF		0000CC		0033CC		000033
	3333FF		3300FF		3300CC		3333CC		000099		000066
	9999CC		6666FF		6666CC		666699		333399		333366
	CCCCFF		9999FF		6666FF		6600FF		330099		330066
	9966CC		9966FF		6600CC		6633CC		663399		330033
	CC99FF		CC66FF		9933FF		9900FF		660099		663366
	CC66FF		CC33FF		CC00FF		9900CC		996699		660066
	CC99CC		CC66CC		CC33CC		CC00CC		990099		993399
	FFCCFF		FF99FF		FF66FF		FF33FF		FF00FF		CC3399
	FF66CC		FF00CC		FF33CC		CC6699		CC0099		990066
	FF99CC		FF3399		FF0099		CC0066		993366		660033
	FF6699		FF3399		FF0066		CC3366		996666		663333
	CC9999		CC6666		CC3333		CC0000		990033		330000
	FFCCCC		FF9999		FF6666		FF3333		FF0000		CC0033
	FF6633		CC3300		FF3300		FF0000		CC0000		990000
	FF9900		FF9933		CC9966		CC6600		996633		663300
	FFCC66		FFCC00		FFCC33		CC9900		CC9933		996600
	FFFFCC		FFFF99		FFFF66		FFFF33		FFFF00		CCCC00

Структура документа

Язык HTML подразумевает стандартизированную структуру построения HTML-документа. Такая структура описывает очередность следования ряда обязательных блоков, которые содержат непосредственно программный код.

HTML-документ всегда должен начинаться с открывающего тега **<HTML>** и заканчиваться соответствующим закрывающим тегом **</HTML>**.

Внутри документа при этом выделяются два основных раздела: раздел заголовков и тело документа.

Раздел заголовков содержит информацию, описывающую документ в целом, и ограничивается тегами **<HEAD>** и **</HEAD>**. В частности, раздел заголовков должен содержать контейнер общего заголовка документа **<TITLE> ... </TITLE>**.

Собственно же содержимое Web-страницы размещается в теле документа, которое ограничивается парными тегами **<BODY>** и **</BODY>**.

```
<HTML>
<HEAD>
<TITLE> Заголовок Web-документа </TITLE>
</HEAD>
<BODY>
Содержимое web-страницы (тело документа)
</BODY>
</HTML>
```

На практике если теги `<HTML>`, `<HEAD>` и `<BODY>`, а также соответствующие им закрывающие теги опущены, то браузер может сама определить то место, где они должны были находиться. Тег `<TITLE>`, определяющий заголовок документа, считается обязательным, но и его пропуск не вызовет катастрофических последствий в современных браузерах. Но все-таки при создании HTML-документов опускать эти теги не рекомендуется, ведь заранее неизвестно, как поведет себя конкретный браузер, установленный на компьютере.

Комментарии

Как любой язык, HTML позволяет вставлять в документ комментарии, которые сохраняются при передаче документа по сети, но не отображаются браузером. Любой текст, помещенный между `<!--` и `-->` будет игнорирован браузером. Исключения составляют сами теги. Их нельзя записывать в комментарии.

Заголовок HTML-документа

Заголовок HTML-документа содержит информацию о самом документе, а иногда специальные директивы и правила, по которым следует обрабатывать составляющий страницу код. Необходимо отметить, что содержимое заголовка не отображается в браузере и не влияет на внешний вид документа: это служебная информация, которая необходима, прежде всего, самому браузеру.

Синтаксис тега заголовка в общем виде выглядит так: `<HEAD> Содержимое </HEAD>`

Раздел **HEAD** следует в HTML-документе непосредственно за тегом `<HTML>` и является второй обязательной командой, которую необходимо включать в код страницы.

Внутри `<HEAD>` могут располагаться различные элементы, среди которых следующие:

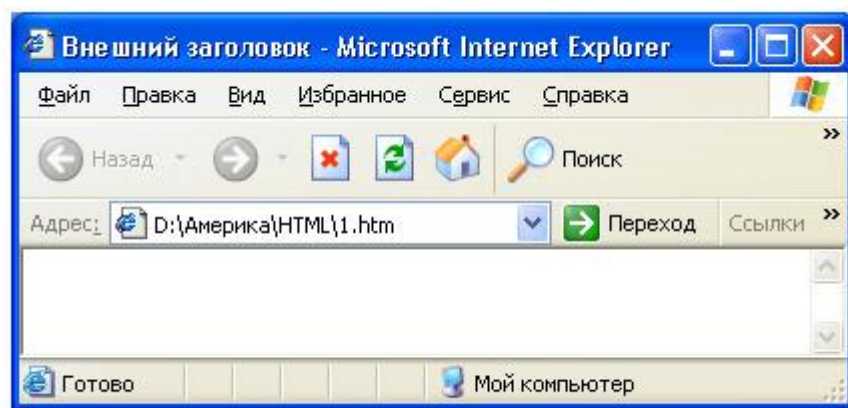
- `<TITLE>` — название документа;
- `<BASE>` — исходный URL документа;
- `<META>` — дополнительная информация о странице

Из них обязательным является только `<TITLE>`. Остальные часто отсутствуют в HTML-документах.

Следующий объект HTML-документа — внешний заголовок, который является вложенной командой тега `<HEAD>`, который записывается в виде:

```
<TITLE> Внешний заголовок </TITLE>
```

Внешний заголовок служит для отображения в верхнем поле браузера в качестве названия страницы при ее открытии, и именно значение тега `<TITLE>` подставляется по умолчанию в соответствующее диалоговое окно, когда пользователь заносит документ в папку «Избранное».



Название документа должно быть информативным и коротким. Длинное название может выглядеть странно в строке заголовка окна браузера и может не поместиться в строку списка закладок или таблицы «Избранное». Им также частенько пользуются поисковые системы Интернета.

Задавая название, следует придерживаться следующих правил:

- избегайте общих слов. Старайтесь максимально точно сообщить, чему посвящена ваша страница. Помните, что любое название может быть использовано в качестве записи в поисковых системах, например Google или AlltheWeb.com;

- по возможности избегайте затасканных слоганов. Просто написать имя фирмы не всегда помогает, особенно если использовать одно и то же название на множестве страниц сайта;

- старайтесь использовать в названии не более 60 символов. Спецификация HTML не вводит ограничений на длину текста в элементе **<TITLE>**. И все же, прежде чем давать своей страничке имя, помните о том, что строка заголовка окна ограничена.

Элемент **<BASE>** предназначен для установки «базового» пути, с его помощью определяются все относительные URL.

Элемент **<META>**

Элемент **<META>** — это способ определить некоторую переменную путем указания ее имени (атрибут **NAME**) и значения (атрибут **CONTENT**). Большинство META-тегов являются не обязательными.

Вот некоторые наиболее типичные META-инструкции:

<META NAME=«DESCRIPTION» CONTENT=«Руководство по работе в глобальной компьютерной сети Интернет»> — эта META-инструкция определяет переменную **DESCRIPTION**, содержащую краткое описание документа. Многие поисковые механизмы постоянно сканируют Интернет в поисках HTML-файлов, отыскивают в них эту переменную, сохраняют ее в своих базах данных и демонстрируют ее в ответ на запросы пользователей.

<META NAME=«KEYWORDS» CONTENT=«Интернет, HTML, WWW, руководство, публикация, гипертекст»> — такая META-инструкция определяет переменную **KEYWORDS**, содержащую набор ключевых слов, описывающих содержание документа. На практике поиск по ключевым словам очень удобно использовать при строительстве поискового механизма, работающего в пределах одного сервера.

<META NAME=«AUTHOR» CONTENT=«ИМЯ АВТОРА»> — указание имени автора.

<META NAME=«DATE» CONTENT=«МЕСЯЦ, ЧИСЛО, ГОД И ВРЕМЯ ЧЕРЕЗ ПРОБЕЛ»> — дата создания сайта. Например: **<META NAME=«DATE» CONTENT=«MAY 28 1999 15:34 AM»>**

Другая группа META-инструкций определяет эквиваленты команд протокола передачи гипертекстов.

Например, META-инструкция **<META HTTP-EQUIV=«CONTENT-TYPE» CONTENT=«text/html; CHARSET=koi8-r»>** дает браузеру указание интерпретировать загружаемый документ как содержащий HTML-текст в кодировке КОИ-8.

Использование элемента **<META>** и его атрибутов позволяет загружать новую страницу через указанный промежуток времени. Тем же способом можно перезагружать или обновлять один и тот же документ снова и снова. Эта технология называется **CLIENT-PULL**. При этом пользователь ни на чем не щелкает и ни на что не нажимает.

Для реализации обновления требуется наличие двух атрибутов: **HTTP-EQUIV** и **CONTENT**.

Атрибут **HTTP-EQUIV** всегда должен иметь значение **REFRESH**, он загружает новую страницу, если что-нибудь указано в атрибуте **URL**. В противном случае он обновляет текущую страницу.

Атрибут **CONTENT** задает время (в секундах), по истечении которого браузер загружает новую страницу или осуществляет обновление. После указанного в этом атрибуте числа ставится точка с запятой, после чего может быть еще один атрибут **URL**, в котором задается адрес автоматически загружаемой страницы.

Вот пример кода, который обновляет текущую страницу каждые 10 секунд:

<META HTTP-EQUIV=«refresh» CONTENT=«10»>

Загрузка новой страницы:

<META HTTP-EQUIV=«refresh» CONTENT=«2; URL=2.htm»>

Также с помощью тега **META** можно задавать различные эффекты при заходе или уходе пользователя со странички или сайта.

Например, эффект при заходе на страницу:

<META HTTP-EQUIV=«Page-Enter» CONTENT=«revealTrans (Duration=4.0, Transition=12)»>

Эффект при уходе со страницы: **<META HTTP-EQUIV=«Page-Exit» CONTENT=«revealTrans (Duration=4.0, Transition=12)»>**

DURATION задает время в секундах, **TRANSITION** — номер эффекта.

0 — Box in (в поле); **1** — Box out (из поля); **2** — Circle in (в круг); **3** — Circle out (из круга); **4** — Wipe up (стирание вверх); **5** — Wipe down (вниз); **6** — Wipe right (вправо); **7** — Wipe left (влево); **8** — Vertical blinds (вертикальные жалюзи); **9** — Horizontal blinds (горизонтальные жалюзи); **10** — Checkboard across (в шахматном порядке поперечно); **11** — Checkboard down (вниз); **12** — Random dissolve (случайный наплыв); **13** — Split vertical in; **14** — Split vertical out; **15** — Split horizontal in; **16** — Split horizontal out; **17** — Strips left down; **18** — Strips left up; **19** — Strips right down; **20** — Strips right up; **21** — Random bars horizontal (случайные горизонтальные линии); **22** — Random bars vertical; **23** — Random (случайный эффект)

Содержимое HTML-документа

Содержимое HTML-документа размещается в теге **<BODY> ... </BODY>**.

Тег **<BODY>** может также иметь параметры отступов в документе (определяются числовым значением).

LEFTMARGIN — отступ слева;

RIGHTMARGIN — отступ справа;

TOPMARGIN — отступ сверху;

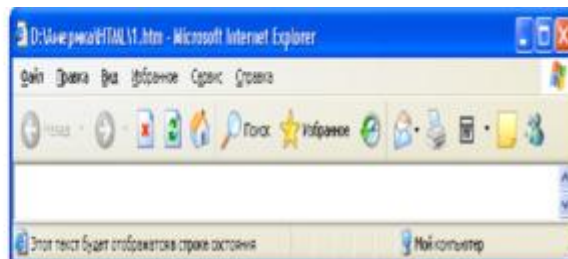
BOTTOMMARGIN — отступ снизу.

Например, **<BODY**

LEFTMARGIN=«0», TOPMARGIN=«0», MARGINWIDTH=«0» MARGINHEIGHT=«0»>.

Следующим параметром можно задать текст, который будет отображен в статусной строке.

```
<BODY ONLOAD="window.defaultStatus='Этот  
текст будет отображаться в строке состояния'">
```



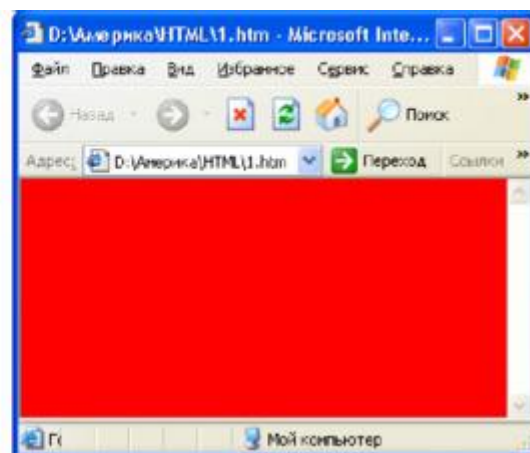
Цветовая гамма HTML-документа определяется атрибутами, размещенными внутри метки **<BODY>**.

BGCOLOR — определяет цвет фона документа.

Например, **<BODY BGCOLOR=«red»>** — заливает страницу красным цветом;

<BODY BGCOLOR=«#ff0000»> — то же самое, но с использованием шестнадцатеричного кода.

```
<BODY BGCOLOR="red">
```



Кроме того, **<BODY>** может включать атрибут **BACKGROUND=» [имя файла]>**, который задает изображение, служащее фоном для текста и других изображений в формате GIF или JPEG.

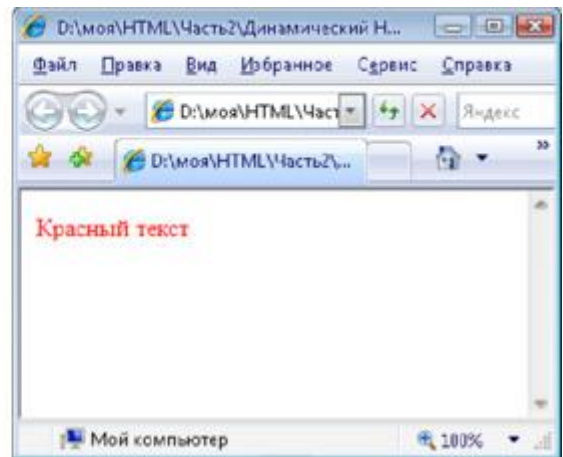
```
<BODY BACKGROUND="images.gif">
```



Предполагается, что графический файл расположен в одной папке с HTML-документом, иначе нужно указать к нему путь. Путь к файлу прописывается относительно HTML-документа в виде FOLDER_1/FOLDER_2/IMAGES.GIF. Если файл расположен на несколько уровней вверх, то имена папок заменяются двумя точками (..), например, ../../IMAGES.GIF. Подобное указание путей применяется для разных элементов, например, ссылок, рисунков, файлов. Следует учесть, что использование графических изображений в качестве фона замедляет загрузку HTML-документа.

TEXT — определяет цвет текста документа.

<BODY TEXT="red">

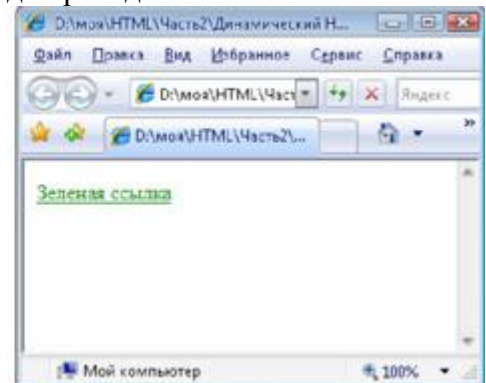


LINK — определяет цвет выделенного элемента текста, при нажатии на который происходит переход по гипертекстовой ссылке.

VLINK — определяет цвет ссылки на документ, который уже был просмотрен ранее.

ALINK — определяет цвет ссылки в момент, когда на нее указывает курсор мыши и нажата ее правая кнопка, то есть непосредственно перед переходом по ссылке.

<BODY LINK="green" VLINK="red"
ALINK="blue">



Оформление текстов

Большая часть текста заключается в контейнерные элементы **<P>** и **</P>**. Они используются для разделения текста на абзацы. Элемент **<P>** сообщает браузеру о том, что весь текст, заключенный между открывающим и закрывающим тегами, представляет собой единый абзац. Когда вы ставите открывающий тег, браузер вставляет пустую строку, обозначая тем самым начало нового абзаца.

Лишние пробелы между словами не влияют на то, как HTML-документ выглядит в браузере. Внутри контейнера, задающего абзац, любые лишние пропуски между буквами будут игнорироваться.

Переход на новую строку внутри абзаца задается тегом **
**. Эта метка используется, если необходимо перейти на новую строку, не прерывая абзаца. Очень удобно при публикации стихов:

<P> Однажды в студеную зимнюю пору **
**

Сижу за решеткой в темнице сырой. **
**

Гляжу — поднимается медленно в гору **
**

Вскормленный в неволе орел молодой. **</P>**

<P> И шествуя важно, в спокойствии чинном, **
**

Мой грустный товарищ, махая крылом, **
**

В больших сапогах, в полушубке овчинном, **
**

Кровавую пищу клюет под окном. **</P>**

Теги **<P>** и **
** закрывать не обязательно. Конечно, если Вы не используете в теге **<P>** элемент **ALIGN**, которым задается выравнивание абзаца.

Если вы не хотите, чтобы браузер автоматически переносил строку, то вы можете обозначить ее тэгами **<NOBR>** и **</NOBR>**. В этом случае браузер не будет переносить строку, даже если она выходит за границы экрана; вместо этого браузер позволит горизонтально прокручивать окно.

Если же вы хотите все же позволить разбиение данной строки на две, используйте тег **<WBR>**.

Тег **<DIV> ... </DIV>** позволяет определить в документе раздел. Его функции похожи на функции тега **<P>**.

**** — позволяет выделить некоторое количество текста для последующего его форматирования, но в отличие от **<DIV>** не начинает новый абзац.

Например, ** Текст **

Выравнивание текстов

В HTML-документах применяется четыре способа выравнивания текстов: по левому краю; по правому краю; по центру; по ширине.

Задаются они с помощью атрибута **ALIGN**:

<P ALIGN=«LEFT»> ... </P> — выравнивание по левому краю.

<P ALIGN=«CENTER»> ... </P> — выравнивание по центру.

<P ALIGN=«RIGHT»> ... </P> — выравнивание по правому краю.

<P ALIGN=«JUSTIFY»> ... </P> — текст, находящийся между этими элементами выравнивается по ширине.

Элементы **LEFT**, **CENTER**, **RIGHT** и **JUSTIFY** можно использовать, не включая их в тег абзаца.

Например, запись **<P ALIGN=«LEFT»> ... </P>** аналогична записи **<P> <LEFT> ... </LEFT> </P>**

Использование заголовков

Для создания заголовков используют теги **<Hn> ... </Hn>**, где n — число от 1 до 6.

ЗаклЮчив текст между этими тэгами, вы получите заголовок определенного размера.

<H1> Заголовок </H1>

<H2> Заголовок </H2>

<H3> Заголовок </H3>

<H4> Заголовок </H4>

<H5> Заголовок </H5>

<H6> Заголовок </H6>

В идеале, пропускать уровни нельзя. То есть **<H3>** не может следовать после **<H1>**, если между ними нет **<H2>**.

Нельзя в одной и той же строке использовать и заголовок, и обычный текст. Элемент заголовка работает так же, как знак абзаца **<P>**, то есть он вставляет новую строку после своего закрывающего тега.

Например, **<H1> Это заголовок </H1> А это — обычный текст** будет выглядеть в браузере примерно так же, как **<H1> Это заголовок </H1> <P> А вот еще текст абзаца </P>**

Для расположения заголовков на странице используют теги выравнивания.

Например, **<CENTER> <H1> Заголовок </H1> </CENTER>** — размещает заголовок по центру.

Шрифтовые контейнеры

В HTML для оформления шрифтов существуют также специальные контейнеры, предназначенные для изменения начертания фрагмента текста на курсивное, полужирное и другие, преобразование его в верхние либо нижние индексы и др.

** ... ** — полужирный текст.

<I> ... </I> — курсивный текст.

<U> ... </U> — подчеркнутый текст. Подчеркнутый текст лучше использовать только для гиперссылок; любое другое его использование обманывает посетителей сайта.

^{...} — верхний индекс.

_{...} — нижний индекс.

<STRIKE> ... </STRIKE> — зачеркнутый текст.

<PRE> ... </PRE> — текст пишется, как есть, включая все пробелы. Злоупотреблять применением данной команды не рекомендуется, поскольку в силу несхожести алгоритмов обработки кода HTML различными браузерами возможно искажение отформатированного таким образом текста в зависимости от экранных настроек пользователей. Внутри тега **PRE** запрещено использовать другие теги форматирования текста.

** ... ** — курсивный текст.

** ... ** — полужирный текст. Следует заметить, что контейнеры ** ... ** и ** ... ** выполняют те же функции, что и контейнеры **<I> ... </I>** и ** ... **.

<BLOCKQUOTE> ... </BLOCKQUOTE> — вывод текста на экран с увеличенным левым полем.

<TT> ... </TT> — «телетайпный» текст, т.е. текст одного размера.

<BIG> ... </BIG> — большой шрифт.

<SMALL> ... </SMALL> — малый шрифт.

<BLINK> ... </BLINK> — мигающий текст. Этот тег может не работать в браузере Internet Explorer версий 5 и ниже без применения JavaScript

<ADDRESS> ... </ADDRESS> — используется для создания текста, в котором содержится информация об авторе страницы. В большинстве браузеров текст, содержащийся в этом элементе, отображается курсивом. Традиционно этот элемент размещается в конце Web-документа и сообщает следующую информацию:

- когда страница была обновлена в последний раз;
- с кем можно обсудить вопросы, связанные с этой страницей (обычно e-mail);
- URL данной страницы;
- телефонные номера или физические адреса компании или организации

Большинство этих данных не являются важными для страницы, но являются прекрасным дополнением информации, представленной на странице.

Пример использования элемента **<ADDRESS>**:

<ADDRESS>Эта страница была последний раз
обновлена 6/12 в 9:08 pm.**
**
В случае проблем пишите по адресу:
kin@mailpro.ru.**
**
Адрес: 12345 Москва**
**
ул. Почтовая, д. 1.**</ADDRESS>**

*Эта страница была последний раз
обновлена 6/12 в 9:08 pm.
В случае проблем пишите по адресу:
kin@mailpro.ru.
Адрес: 12345 Москва
ул. Почтовая, д. 1.*

<COMMENT> — позволяет скрыть от пользователя комментарии к исходному коду, а так же для сокрытия сценариев Java Script от браузеров, которые не поддерживают их. Полностью аналогичен варианту задания комментариев **<!--** — Текст комментария **-->**

<INS> ... </INS> и ** ... ** — элементы, представляющие особый интерес для создателей официальных проектов, посвященных бизнесу или образованию. Элемент **** используется для обозначения фрагмента текста, который, скорее всего, будет удален, а **<INS>** — чтобы обозначить фрагмент, вставляемый на место удаляемого. Они могут использоваться и независимо, если вы просто хотите вставить или удалить текст. Это полезно, когда над страницей работают несколько человек, и вы хотите отслеживать изменения.

У обоих элементов могут быть необязательные атрибуты **CITE**, **DATETIME** или **TITLE**. Они используются для того, чтобы пояснить свои

действия. При этом с помощью **DATETIME** можно указать время, когда соответствующий элемент был вставлен.

Например, 2001-12-05T09:00:00—05:00 — означает: 9:00 am EST, 12/05/2001. Что касается -05:00, то это часовой пояс.

Атрибут **CITE** позволяет задавать ссылку на URL, на котором содержится объяснение данного добавления или удаления. А атрибут **TITLE** позволяет делать то же самое с помощью тегов.

```
<P>Это важная информация для <INS  
DATETIME="2001-12-05T09:00:00-  
05:00" TITLE="Изменить по результатам  
голосования">Совета директоров,</INS>  
<DEL>президента и его  
помощников.</DEL> Именно им  
необходимо иметь на руках более 50%  
акций предприятия.</P>
```

Это важная информация для Совета директоров президента и его помощников.
Именно им необходимо иметь на руках более Изменить по результатам голосования

Любые теги шрифтового выделения текста можно использовать совместно друг с другом. Например, чтобы сделать текст одновременно жирным и курсивным, используется сочетание контейнеров ** <I> ... </I> **. Порядок их вложения друг в друга не важен, но нужно учитывать правила вложенности.

```
<B><I>Текст можно</I> оформить</B>  
различными<SUB><U> способами.</U></SUB>
```

*Текст можно оформить
различными способами*

Оформление шрифтов

Для оформления шрифтов используется тег ** ... **.

Тег **** может иметь несколько атрибутов:

SIZE — задает размер текста (по умолчанию размер текста равен 3). Поместив текст между тегами ** ... **, где n — числовое значение, Вы придадите ему нужный вам размер:

```
<FONT SIZE="1"> Пример 1 </FONT>  
<FONT SIZE="2"> Пример 2 </FONT>  
<FONT SIZE="3"> Пример 3 </FONT>  
<FONT SIZE="4"> Пример 4 </FONT>  
<FONT SIZE="5"> Пример 5 </FONT>  
<FONT SIZE="6"> Пример 6 </FONT>
```

Пример 1
Пример 2
Пример 3
Пример 4
Пример 5
Пример 6

Вместо знака = можно использовать двоеточие (:). Также размер можно задавать в пунктах. Например, **FONT-SIZE: 16pt**.

FACE — задает стандартное имя шрифта. Используйте шрифты, которые установлены на компьютере пользователя, в противном случае Обозреватель будет использовать шрифт, определенный по умолчанию (обычно Times New Roman). К стандартным шрифтам можно отнести шрифты, поставляемые с Windows 98, Ms Plus, Ms Office. В самой нижней строке данной таблицы представлено использование семейства шрифта — имя каждого шрифта пишется через запятую. Если у пользователя на компьютере нет шрифта Comic Sans MS, Обозреватель подставит следующий в этом списке — Tahoma.


```

<FONT FACE="Times New Roman"> Times New Roman </FONT>
<FONT FACE="System"> System </FONT>
<FONT FACE="Arial"> Arial </FONT>
<FONT FACE="Courier"> Courier </FONT>
<FONT FACE="Verdana"> Verdana </FONT>
<FONT FACE="Comic Sans MS, Tahoma"> Comic Sans MS </FONT>

```

Times New Roman
System
Arial
Courier
Verdana
Comic Sans MS

COLOR — задает цвет текста (по умолчанию черный — #000000). Цвет текста может определяться как самим названием, например, red, blue и т.д, так и представлен в шестнадцатеричном виде — #FF0000 (красный) #0000FF (синий).

```

<FONT COLOR="red">Красный </FONT>
<FONT COLOR="#FF0000">Красный </FONT>
<FONT COLOR=green>Зеленый </FONT>
<FONT COLOR=880000>Коричневый</FONT>

```

Красный
Красный
Зеленый
Коричневый

Тег **<BASEFONT>** задает базовый шрифт для оформления всего документа. Данный тег непарный.

Например, **<BASEFONT SIZE=«4» FACE=«Arial»>** задает базовый шрифт Arial, размером 4.

Escape-последовательности

Иногда возникает необходимость использовать в тексте символы, зарезервированные для обозначения элементов кода HTML. Это могут быть угловые скобки, прямые кавычки и т. д. Для того чтобы избежать проблем при отображении подобных элементов, были придуманы так называемые Escape-последовательности.

Все Escape-последовательности начинаются с символа амперсанд «&» и заканчиваются точкой с запятой, а между ними размещается сама команда, записываемая в строчном регистре.

Например, неразрывный пробел (словосочетания, разделенные таким пробелом, не разрываются при переносе) обозначается как **&NBSP;**

Escape-последовательности записываются в тексте по тем же правилам, по которым употребляются обозначаемые ими символы, иными словами, объект просто вставляется в тот участок текста, где должен следовать соответствующий служебный символ. Возможно, некоторые символы не будут правильно отображаться. Зависит это от используемого шрифта.

В таблице представлены наиболее распространенные символы и их коды.

Вид	Код	Описание
Различные знаки		
§	§	параграф
«	«	левая двойная угловая скобка
	­	место возможного переноса
°	°	градус
±	±	плюс-минус
»	»	правая двойная угловая скобка
·	∙	умножить
Греческий алфавит		
Α	Α	греческая заглавная буква альфа
Β	Β	греческая заглавная буква бета
Γ	Γ	греческая заглавная буква гамма
Δ	Δ	греческая заглавная буква дельта
Ε	Ε	греческая заглавная буква эпсилон
Ζ	Ζ	греческая заглавная буква дзета
Η	Η	греческая заглавная буква эта
Θ	Θ	греческая заглавная буква тета
Ι	Ι	греческая заглавная буква иота
Κ	Κ	греческая заглавная буква каппа
Λ	Λ	греческая заглавная буква лямбда
Μ	Μ	греческая заглавная буква мю
Ν	Ν	греческая заглавная буква ню
Ξ	Ξ	греческая заглавная буква кси
Ο	Ο	греческая заглавная буква омикрон
Π	Π	греческая заглавная буква пи
Ρ	Ρ	греческая заглавная буква ро
Σ	Σ	греческая заглавная буква сигма
Τ	Τ	греческая заглавная буква тау
Υ	Υ	греческая заглавная буква ипсилон
Φ	Φ	греческая заглавная буква фи
Χ	Χ	греческая заглавная буква хи

Ψ	Ψ	греческая заглавная буква пси
Ω	Ω	греческая заглавная буква омега
α	α	греческая строчная буква альфа
β	β	греческая строчная буква бета
γ	γ	греческая строчная буква гамма
δ	δ	греческая строчная буква дельта
ε	ε	греческая строчная буква эpsilon
ζ	ζ	греческая строчная буква дзета
η	η	греческая строчная буква эта
θ	θ	греческая строчная буква тета
ι	ι	греческая строчная буква иота
κ	κ	греческая строчная буква каппа
λ	λ	греческая строчная буква лямбда
μ	μ	греческая строчная буква мю
ν	ν	греческая строчная буква ню
ξ	ξ	греческая строчная буква кси
ο	ο	греческая строчная буква омикрон
π	π	греческая строчная буква пи
ρ	ρ	греческая строчная буква ро
ς	ς	греческая строчная буква сигма (final)
σ	σ	греческая строчная буква сигма
τ	τ	греческая строчная буква тау
υ	υ	греческая строчная буква ипсилон
φ	φ	греческая строчная буква фи
χ	χ	греческая строчная буква хи
ψ	ψ	греческая строчная буква пси
ω	ω	греческая строчная буква омега
Стрелки		
←	←	стрелка влево
↑	↑	стрелка вверх
→	→	стрелка вправо

↓	↓	стрелка вниз
↔	↔	стрелка влево-вправо
Математические операторы		
Π	∏	произведение последовательности - знак произведения
Σ	∑	сумма последовательности
√	√	квадратный корень
∞	∞	бесконечность
∩	∩	пересечение
∪	∫	объединение
∫	∫	интеграл
≈	≈	почти равно - асимптотически стремится
≠	≠	не равно
≡	≡	тождественно
≤	≤	меньше либо равно
≥	≥	больше либо равно
∠	∠	угол
⊥	⊥	перпендикулярность
∈	∈	принадлежит
∉	∉	не принадлежит
∥	‖	параллельность
⊥	∦	не параллельны
Прочие символы		
&	&	ampersand
<	<	знак "меньше"
>	>	знак "больше"

Escape-последовательности чувствительны к регистру: Нельзя использовать **<** вместо **<**..

Бегущая строка

Для задания бегущей строки используется тег **<MARQUEE> ... </MARQUEE>**. Для тега **<MARQUEE>** используются следующие атрибуты:

BGCOLOR — задает цвет бегущей строки.

Например, **<MARQUEE BGCOLOR="880000">** Бегущая строка **</MARQUEE>**.

HEIGHT — задает высоту бегущей строки в пикселях или процентах от окна.

ALIGN — задает выравнивание по верхнему краю (**TOP**), по центру (**MIDDLE**) ил по нижнему краю (**BOTTOM**)

BEHAVIOR — задает тип отображения. Параметры: **SLIDE** — прокрутка с остановкой, **SCROLL** — прокрутка, **ALTERNATE** — движение от края к краю.

DIRECTION — определяет направление движения: слева направо (**RIGHT**), справа налево (**LEFT**), снизу вверх (**UP**), сверху вниз (**DOWN**).

Например, `<MARQUEE DIRECTION=«RIGHT»>` Бегущая строка `</MARQUEE>`.

HSPACE — задает смещение бегущей строки вправо в пикселях.

Например, `<MARQUEE HSPACE=100>`

VSPACE — задает пустое пространство выше и ниже бегущей строки.

Например, `<MARQUEE VSPACE=10>`

LOOP — атрибут задает количество проходов бегущей строки. Например, `<MARQUEE LOOP=5>`

SCROLLAMOUNT — задает скорость движения бегущей строки.

Например, `<MARQUEE SCROLLAMOUNT=5>`

SCROLLDELAY — задает временной интервал между шагами бегущей строки.

С помощью него можно заставить строку двигаться рывками. Например, `<MARQUEE SCROLLDELAY=5>`.

Списки

Язык HTML позволяет создавать пять разных видов списков. Из этих пяти видов два (списки меню и списки каталога) считаются устаревшими и практически не употребляются. Таким образом, в настоящее время используют три вида списков: **нумерованные** (упорядоченные), **маркированные** (неупорядоченные) и **списки определений**.

Элементы списков, как абзацы и форматированный текст могут включать в себя другие элементы разметки текста. В списке всегда должно быть два элемента, один из которых задает тип списка, а другой отвечает за один конкретный пункт. Этими пунктами могут быть слова, предложения, абзацы и другие элементы, например изображения. В большинстве списков используется следующий формат:

`<ТИП СПИСКА>`

`` Первый пункт

`` Второй пункт

`` Третий пункт

...

`</ТИП СПИСКА>`

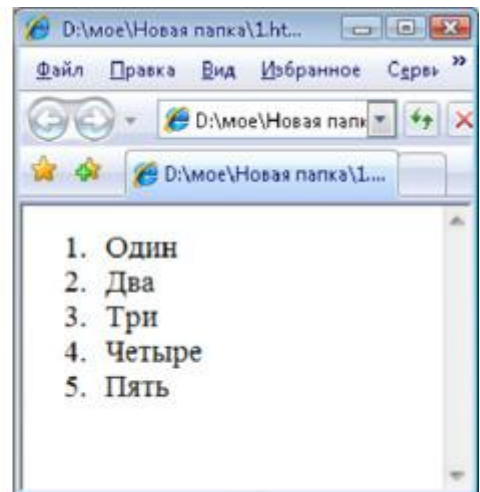
Каждый элемент `` — это пункт списка, который всегда пишется с новой строки.

Нумерованные списки

Нумерованные списки служат для автоматической расстановки чисел перед каждым элементом списка.

Начинается нумерованный список с тега `` и завершается тегом ``. Для задания элементов списка используется тег ``.

```
<OL>
<LI> Один
<LI> Два
<LI> Три
<LI> Четыре
<LI> Пять
</OL>
```



Тег **** может использовать различные атрибуты.

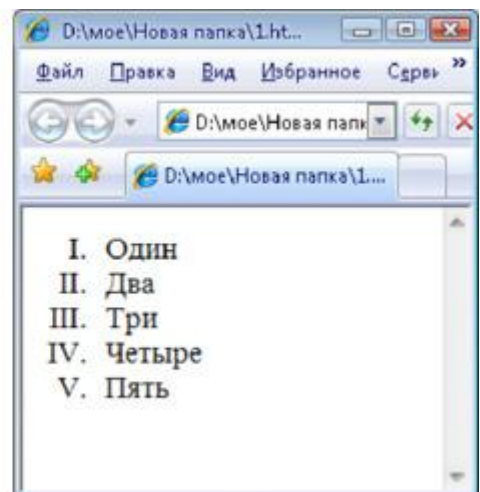
Атрибут **TYPE** позволяет определять вид маркеров, которыми будут обозначаться составляющие список значения.

В составе атрибута **TYPE** команды **** можно использовать один из следующих параметров:

- «1» — обычные арабские числа 1, 2, 3 и т. д.
- «I» — римские цифры в заглавном регистре I, II, III, IV и т. д.
- «i» — римские цифры в строчном регистре i, ii, iii и т. д.
- «A» — символьная маркировка в заглавном регистре A, B, C и т. д.
- «a» — символьная маркировка в строчном регистре a, b, c и т. д.

По умолчанию значение атрибута **TYPE** принимается как «1», нумерация начинается с первого элемента в каждом из типов маркировки.

```
<OL TYPE="I">
<LI> Один
<LI> Два
<LI> Три
<LI> Четыре
<LI> Пять
</OL>
```



Атрибут **START** позволяет задавать позицию, с которой следует начать маркировку (по умолчанию 1).

Например, список, пронумерованный большими латинскими буквами и начинающийся с восьмой цифры, запишется в следующем виде:

<OL TYPE="I" START="8">

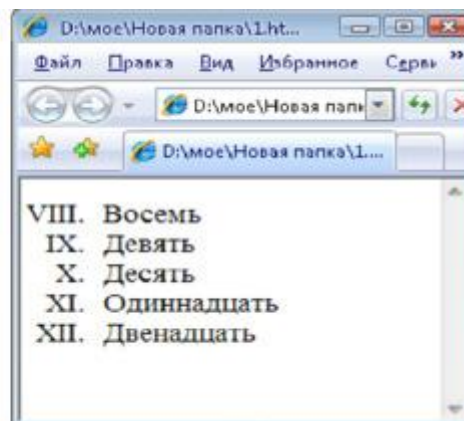
 Восемь

 Девять

 Десять

 Одиннадцать

 Двенадцать



При использовании атрибута **START** необходимо внимательно следить, чтобы его значение соответствовало типу маркировки, указанному в атрибуте **TYPE**. Запись <OL TYPE="I" START="A"> не допускается. Наоборот, запись <OL TYPE="A" START="I"> вполне допустима.

Атрибут **VALUE** используется для изменения значения конкретного элемента списка. Например, можно изменить нумерацию прямо внутри списка:

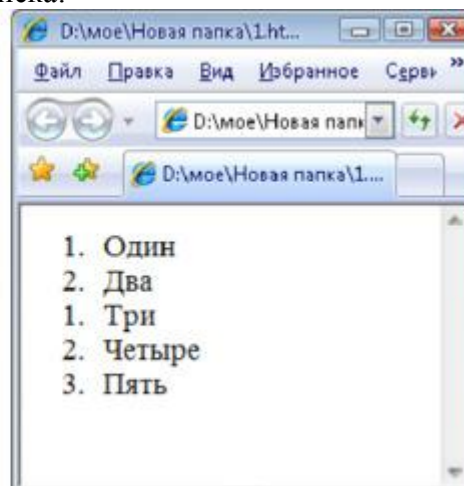
 Один

 Два

<LI VALUE="1"> Три

 Четыре

 Пять



Для сокращенного расстояния между элементами используется атрибут **COMPACT**.

Маркированные списки

Для маркированных списков используются маркеры для пометки элементов списка.

Маркированный список начинается открывающимся тэгом и завершается тэгом . Каждый элемент списка начинается с тэга .

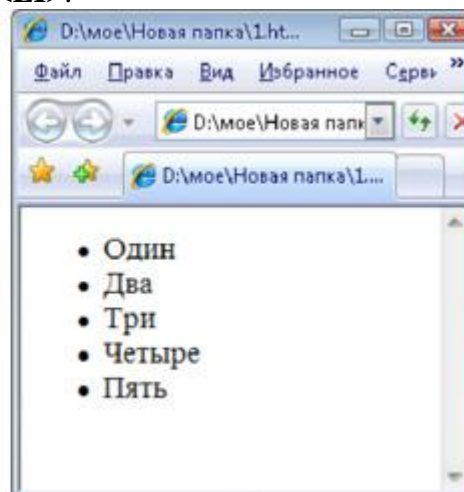
 Один

 Два

 Три

 Четыре

 Пять

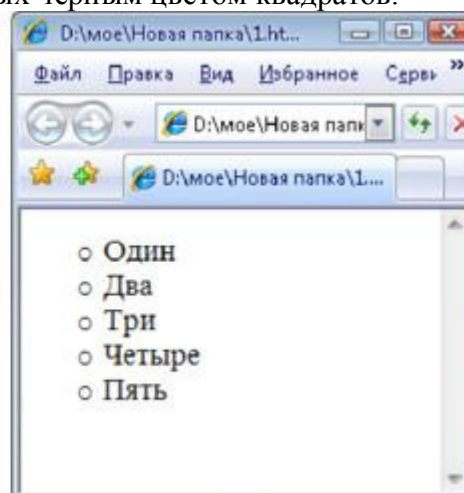


Вид маркера определяется атрибутом **TYPE**. Параметр этого атрибута может принимать одно из трех значений:

DISC — метки отображаются в виде заполненных черным цветом окружностей;
CIRCLE — метки отображаются в виде полых окружностей;
SQUARE — метки отображаются в виде заполненных черным цветом квадратов.

<UL TYPE="CIRCLE">

**** Один
**** Два
**** Три
**** Четыре
**** Пять



По умолчанию, то есть в случае, когда в маркированном списке тег **** записывается без атрибутов, **TYPE=«DISC»**.

Далеко не все браузеры воспринимают применение различных маркеров. Вместо заданного, они могут использовать, к примеру, кружок.

Списки определений

Помимо нумерованных и маркированных списков язык разметки гипертекста позволяет создавать списки определений.

Списки определений используются в основном при создании словарей, глоссариев и прочих перечней, включающих в себя термин и его определение.

Список определений несколько отличается от других видов списков. Вместо меток **** в списках определений используются метки **<DT>** (**Definition Term** — определяемый термин) и **<DD>** (**Definition Definition** — определение определения).

Списки определений имеют следующий вид:

<DL>

<DT> Термин

<DD> Определение термина

<DD> Определение термина

<DD> Определение термина

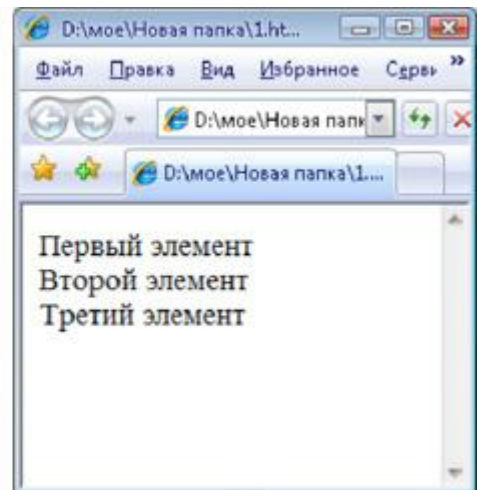
<DD> Определение термина

</DL>

Если определяемые термины достаточно коротки, с элементом **<DL>** можно использовать атрибут **COMPACT**.

Если использовать только тег **<DT>**, то список будет чем-то напоминать маркированный, за исключением того, что почти все браузеры будут отображать его без маркеров.

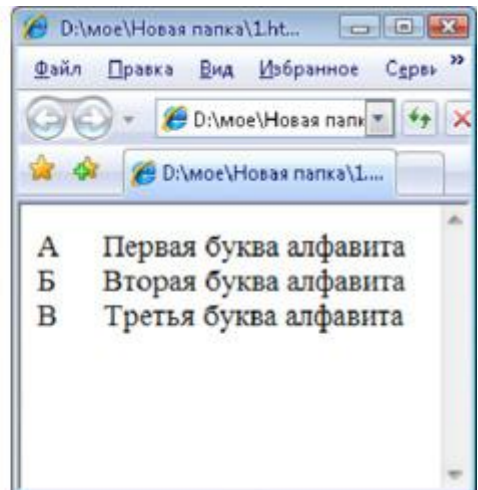
<DL>
<DT> Первый элемент
<DT> Второй элемент
<DT> Третий элемент
</DL>



Элемент <DD> можно применять для того, чтобы сделать несколько абзацев с одинаковыми отступами.

Элементы <DL> должны быть спарены с соответствующими элементами <DD>. Элемент <DD> может иметь множество элементов <DT>, то есть несколько терминов могут иметь одно и тоже определение. Но при этом документ не должен содержать несколько последовательных элементов <DD>.

<DL COMPACT>
<DT>А
<DD>Первая буква алфавита
<DT>Б
<DD>Вторая буква алфавита
<DT>В
<DD>Третья буква алфавита
</DL>



Списки меню

Списки меню используется с целью создания списков по логическому определению, такие списки могут связываться с другими элементами документа. Для создания таких списков используется тег <MENU>. Но на практике большинство браузеров представляют элемент <MENU> таким же образом, как элемент .

Списки меню записываются в виде:

<MENU>
 Первый элемент
 Второй элемент
 Третий элемент
</MENU>

Теоретически элемент <MENU> отображается, как одностолбцовый список элементов.

Списки каталогов

Для создания списков каталогов используется тег <DIR>.

Списки каталогов записываются в виде:

<DIR>
 Первый элемент
 Второй элемент

**** Третий элемент

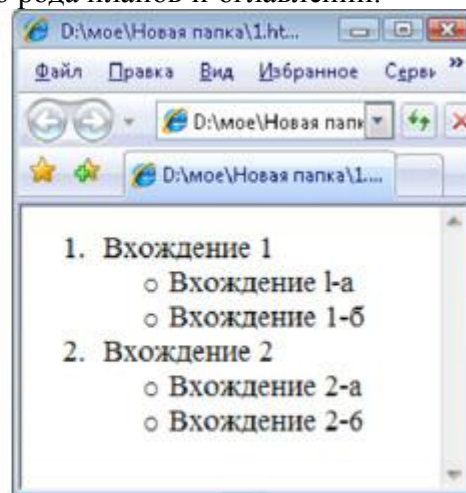
</DIR>

Элемент **<DIR>** должен отображаться, как многостолбцовый список каталогов, но с ним возникают те же проблемы, что и при использовании списков меню.

Вложенные списки

При создании HTML-документов возможно создание так называемых вложенных списков, реализуемых посредством размещения одной пары тегов в другой, например, команд создания маркированного списка внутри нумерованного. Число уровней вложенности в принципе не ограничено, однако злоупотреблять вложенными списками все же не следует. Вложенные списки очень удобны при подготовке разного рода планов и оглавлений.

```
<OL>
<LI>Вхождение 1
  <UL>
    <LI>Вхождение 1-а
    <LI>Вхождение 1-б
  </UL>
<LI>Вхождение 2
  <UL>
    <LI>Вхождение 2-а
    <LI>Вхождение 2-б
  </UL>
</OL>
```



Теги вложенных списков могут включать те же атрибуты, что и теги самих списков **** и ****. Для создания вложенных списков следует запомнить одно правило: там, где нужен вложенный список элемент **** основного списка просто заменяется на другой список.

Изображения в HTML-документах

HTML-документы используют два вида изображений: горизонтальные линии и различного рода рисунки (фотографии, картинки, кнопки и т.д.)

Горизонтальная линия

Горизонтальные линии в HTML-документах используются в основном в качестве разделителей и задаются тегом **<HR>** (Horizontal Rule).

Длину, ширину, цвет и расположение горизонтальной линии можно задавать с помощью дополнительных атрибутов.

Атрибут выравнивания **ALIGN**.

ALIGN=«LEFT» — выравнивание по левому краю;

<HR ALIGN=LEFT >



ALIGN= — выравнивание по центру;

<HR ALIGN= CENTER>



ALIGN= — выравнивание по правому краю.

<HR ALIGN= RIGHT>



Атрибут размера **SIZE**.

SIZE =число — задает высоту линии в пикселях (от 1 до 100, дробные числа игнорируются);

<HR SIZE= 20>



WIDTH =«число» — задает длину линии в пикселях;

<HR WIDTH= 100>



WIDTH= «число %» — задает длину линии в процентах от ширины окна браузера.

<HR WIDTH= 80%>



Атрибут цвета **COLOR**.

COLOR=«цвет», где в качестве значения после знака равенства пишется английское название цвета или числовой код оттенка.

<HR COLOR= RED>



Атрибут **NOSHADE** — определяет способ закраски линии как сплошной. Параметр является флагом и не требует указания значения. Без данного параметра линия отображается объемной.

<HR NOSHADE>



Горизонтальную линию можно превратить в вертикальную, квадрат или прямоугольник. Для этого необходимо соответствующим образом задавать значения ширины и высоты линии. Если значение **SIZE** намного больше значения **WIDTH**, то получается вертикальная линия или вертикальный прямоугольник.

<HR SIZE=100 WIDTH=10>



Если значение **SIZE** равно значению **WIDTH**, получается квадрат.

<HR SIZE=100 WIDTH=100>



А если значение **SIZE** меньше значения **WIDTH**, то получается горизонтальный прямоугольник.

<HR SIZE=10 WIDTH=100>



Следует помнить, что тег **<HR>** браузерами воспринимается как абзац, поэтому расположить две линии в одну обычными способами строку нельзя.

Вставка изображений

Вставка изображений в HTML-документ осуществляется с помощью тега ****, где [«рисунок»] — путь к файлу. При этом картинка вставляется непосредственно в то место, где написан этот тег.

Адрес файла может представлять собой как полный путь (например, **HTTP://WWW.FAKE CORPCOM/IMAGES/PRODUCTLJPG**), так и относительный (если, например, файл расположен на том же сервере, что и HTML-страница).

Полный адрес имеет смысл указывать только тогда, когда он содержит ссылку на изображение, расположенное где-то в другом месте в Интернете. Если файл находится в подкаталогах (относительно HTML-файла), подойдет такой вариант:

Следует помнить, что нельзя помещать ссылки на файлы, расположенные на вашем локальном жестком диске. Поэтому если вы создадите ссылку вида

****, то пользователи не смогут увидеть указанное изображение.

Для размещения изображений на странице используется атрибут **ALIGN**. С помощью **ALIGN** определяется, каким образом текст, расположенный на одной строке с картинкой, будет выравниваться относительно боковых сторон последней.

Синтаксис атрибута **ALIGN** следующий: ****
VALUE принимает следующие значения:

TOP — текст располагается рядом с верхней границей изображения.

<P> текст располагается
рядом с верхней границей
изображения **<IMG**
ALIGN="TOP"
SRC=1.JPG></P> текст располагается рядом с верхней границей изображения



MIDDLE -текст располагается рядом с серединой изображения.

<P> текст
располагается рядом с
серединой изображения **<IMG**
ALIGN="MIDDLE"
SRC=1.JPG></P> текст располагается рядом с серединой изображения



BOTTOM — текст располагается рядом с нижней границей изображения.

<P> текст
располагается
рядом с нижней
границей
изображения **<IMG**
ALIGN="BOTTOM" текст располагается рядом с нижней границей изображения
SRC=1.JPG></P>



По умолчанию **ALIGN=BOTTOM**. Поэтому в том случае, если именно этот вариант вас устраивает, атрибут **ALIGN** можно опустить.

BASELINE — изображение выравнивается по условной «базовой линии». Применение этого параметра рекомендуется в случае, когда web-мастер размещает несколько рисунков или несколько фрагментов одного рисунка в разных ячейках строки таблицы. Именно с использованием значения **ALIGN=«BASELINE»** удастся добиться оптимального выравнивания иллюстраций для браузеров всех типов.

Атрибут **ALIGN** может принимать значения **RIGHT** и **LEFT**, которые переводят изображения из разряда обычных в разряд плавающих. Они не появятся именно в том месте, где расположен элемент ****, вместо этого вы обнаружите их рядом с указанным полем (левым или правым) HTML-документа. Кроме того, текст будет «огигать» картинку.

`<P><IMG SRC="1.JPG"
ALIGN="LEFT"> Больше
всего туристов привлекают на
Инн и Спа великолепные
виды. Куда ни глянешь, всюду
такая красота, что
невозможно удержаться,
чтобы не воскликнуть что-
нибудь
жизнеутверждающее.</P>`



Больше всего туристов привлекают на Инн и Спа великолепные виды. Куда ни глянешь, всюду такая красота, что невозможно удержаться, чтобы не воскликнуть что-нибудь жизнеутверждающее.

`<P><IMG SRC="1.JPG"
ALIGN="RIGHT"> Больше
всего туристов привлекают на
Инн и Спа великолепные
виды. Куда ни глянешь, всюду
такая красота, что
невозможно удержаться,
чтобы не воскликнуть что-
нибудь
жизнеутверждающее.</P>`

Больше всего туристов привлекают на Инн и Спа великолепные виды. Куда ни глянешь, всюду такая красота, что невозможно удержаться, чтобы не воскликнуть что-нибудь жизнеутверждающее.



Старые браузеры воспринимают еще два атрибута: **VSPACE** и **HSPACE**. Данные атрибуты задают горизонтальные и вертикальные отступы от изображения в пикселях, которые необходимы, например, в случае, когда картинка помещается рядом с текстом. **HSPACE** определяет величину незаполненного пространства справа и слева от картинки, **VSPACE** соответственно сверху и снизу.

`<IMG SRC="1.JPG" ALIGN=LEFT
HSPACE="50" VSPACE="20">
<P>Поздравляем с Рождеством!</P>`



Поздравляем с
Рождеством!

В HTML-документах изображения отображаются в натуральную величину. Для изменения размеров используются атрибуты **WIDTH** (ширина) и **HEIGHT** (высота). Ширина и высота задается либо в пикселях, либо в процентах от исходного изображения.

`<IMG SRC="1.JPG" WIDTH="250"
HEIGHT="100">`



Будьте внимательны с заданием размеров, они могут привести к искажению изображения.

В HTML-документах можно размещать изображения с прозрачным фоном. Прозрачность никак не влияет на размер графического файла и на скорость скачивания. Данный параметр поддерживается форматами GIF и PNG, он позволяет выбрать один какой-нибудь цвет и сделать его прозрачным для любых фоновых изображений и цветов.

Цвет, который вы выбираете для придания картинке прозрачности, не обязательно должен быть цветом фона. Можно сделать прозрачными и какие-то другие части изображения, если есть на то причины. Тем не менее, нужно помнить, что прозрачным может быть только один цвет, об этом следует подумать заранее при создании изображения. Если, например, фон имеет два цвета, вам не удастся сделать его прозрачным.



Формат **GIF** также имеет поддержку определенного вида анимации. Вставка анимированных изображений в HTML-документы значительно оживляет Web-страницы. Единственный недостаток анимированных изображений — их низкое качество.

Для задания обрамления изображению используется атрибут **BORDER**. Значение размера рамки задается числом. По умолчанию **BORDER=0** (изображение без рамки).

<IMG SRC="1.GIF"
BORDER=5>



Для вывода подсказки при наведении на рисунок используется атрибут **ALT**.

Например,

Альтернативный текст должен быть коротким, но действительно информативным. Если вы желаете, как можно полнее описать отсутствующее изображение, это можно сделать следующим образом. Поместите в элемент атрибут **LONGDESC**. Этот атрибут позволяет разместить ссылку на URL, где может располагаться хоть целая статья, посвященная данному рисунку.

Например,

Гиперссылки

В отличие от обыкновенного текста, гипертекст позволяет осуществлять мгновенный переход от одного фрагмента текста к другому. При нажатии левой кнопкой мыши на некоторый выделенный фрагмент текущего документа происходит переход к некоторому заранее назначенному документу или фрагменту документа.

URL-адрес

В HTML-документах широко используются **URL-адреса** — унифицированный адрес, который используется для однозначной идентификации любых веб-компонентов. URL — это то, что написано в строке **Адрес браузера**. Он же иногда появляется в нижней строке некоторых браузеров при наведении курсора на гиперссылку. URL состоит всегда из двух частей: имени протокола и пути назначения. Имя протокола говорит о том, с каким типом

Интернет-ресурса вы собираетесь иметь дело. Самым распространенным в Интернете протоколом является **http://**, с помощью которого запрашиваются HTML-документы.

Путь назначения может представлять собой имя файла, каталога или компьютера. **http://www.fakecorp.com/products/index.html** указывает на конкретный документ, а **ftp://ftpnetscape.com** ftp.netscape.com.

Абсолютные и относительные URL- адреса

URL-адреса можно условно разделить на две группы: абсолютные и относительные.

Абсолютные адреса записываются в виде: **протокол/имя_сервера/имя_документа**.

Например, **http://www.london.net/drwatson/index.html**.

С помощью абсолютных URL-адресов можно попасть на указанную страницу из любого места Интернета.

Также в HTML-документах используются и относительные URL-адреса. Это чем-то напоминает работу с файлами в папках. В этом случае указывается расположение нового документа относительно загруженного.

Например, **/service/contact.html**. Здесь файл contact.html размещен в папке service. Если файлы располагаются в одном и том же каталоге достаточно указать его имя. Если файлы расположены в каталоге относительно текущего, можно воспользоваться URL вида: **/каталог1/каталог2/.../имя_файла**. Таким же образом можно переходить и к элементам, расположенным в каталогах верхних уровней, только вместо имени каталога необходимо указать.. (две точки в адресе — это обозначение перехода на один каталог вверх).

Например, **../index.html**.

Относительные ссылки следует использовать аккуратно. При каких-то изменениях в файловой структуре сайта они могут перестать работать.

Текстовые ссылки

В HTML переход от одного фрагмента текста к другому задается с помощью тега **> Текст **

В качестве параметра [адрес перехода] может использоваться несколько типов аргументов. Самое простое — это задать имя другого HTML-документа, к которому нужно перейти.

Например: **> Перейти к оглавлению **

Такой фрагмент HTML-текста приведет к появлению в документе выделенного фрагмента Перейти к оглавлению, при нажатии на который в текущее окно будет загружен документ index.htm.

Чтобы снять подчеркивание с гиперссылки, необходимо добавить атрибут **STYLE=TEXT-DECORATION: NONE**.

Например, ** Перейти к оглавлению ** выведет гиперссылку без подчеркивания.

Если в адресе перехода не указан каталог, переход будет выполнен внутри текущего каталога. Если в адресе перехода не указан сервер, переход будет выполнен на текущем сервере.

На практике часто бывает необходимо дать ссылку на документ, находящийся на другом сервере. Для этого необходимо ввести в свой HTML-документ примерно такой фрагмент:

**Руководство по HTML **

Текст, как правило, выделяется цветом или оформляется подчеркиванием.

Элемент **<A>** используется со следующими параметрами:

HREF — URL (унифицированный локатор ресурсов) — адрес любого файла в Интернете. Может быть абсолютным, то есть указывается полный адрес странички

(например, <http://lenininc.narod.ru/index.html>) и относительным, как видно из названия указывается файл относительно текущего (например, [index.htm](#)).

TARGET — определяет, в каком окне загрузить гиперссылку. Может иметь значения:

TOP — загружает гиперссылку на всем пространстве окна браузера (если до этого существовало разбиение на фреймы, то оно исчезнет).

BLANK — загружает гиперссылку в новом окне браузера.

SELF — загружает содержимое страницы, в окно, которое содержит эту ссылку (и так используется по умолчанию, так что если вам надо именно так загрузить ссылку, то параметр **TARGET** вообще можно не использовать).

PARENT — загружает содержимое страницы, заданной ссылкой, в окно, являющееся непосредственным владельцем набора фреймов.

Например, `` Новое окно `` — открытие документа в новом окне.

TITLE — текст подсказки, который будет появляться при наведении мышки на гиперссылку. Параметр не обязательный.

Цвет ссылки при наведении на нее курсором задается в теге `<BODY>`.

LINK — цвет ссылки.

VLINK — цвет пройденной ссылки.

ALINK — цвет активной ссылки, когда подводится к ней курсор.

Например, `<BODY LINK=«BLUE» VLINK=«PURPLE» ALINK=«RED»>` задает синий цвет ссылки, после подведения курсора цвет изменяется на красный, после щелчка — на пурпурный.

Данные атрибуты определяют свойства для всего документа. Поместив такой код в HTML страницу, уже не надо будет назначать каждый раз цвет ссылке.

Ссылки изображения

Если вы хотите использовать в качестве ссылки изображение, его вставляют вместо текста ссылки.

Например, ` ` в качестве ссылки на документ 1.HTM будет использован рисунок 1.GIF, при наведении на который будет появляться текст «Переход на новую страницу».



Переход на новую страницу

В окне браузера изображение-ссылка отличается от обычного наличием рамки, которая появляется при создании графической ссылки. Если вы хотите задать толщину рамки или убрать ее, используйте атрибут **BORDER**.

Например, ` `

Анкера

При необходимости можно задать к определенному месту внутри HTML-документа. Для этого необходимо создать в документе, к которому будет задан переход, некоторую опорную точку, или анкер.

Например, необходимо осуществить переход в файле 1.htm. Для этого в месте, куда будет осуществлен переход, размещается тег ** Переход **

Слово «Переход» при этом никак не будут выделены в тексте документа.

Затем можно определить переход на этот анкер: ** Переход к анкеру AAA **

Символы «#AAA» сообщает браузеру, что необходимо найти в данном HTML-документе маркер с именем «AAA». Аналогично можно осуществить переход к указанному анкеру и из другого HTML-документа.

Следите за написанием имен анкеров: большинство браузеров отличают большие буквы от маленьких. Если имя анкера определено как AAA, ссылка на анкер aaa или AaA не выведет Вас на анкер AAA, хотя документ, скорее всего, будет загружен корректно.

Карты изображений

Карты изображений — это способ сделать различные части одного графического изображения гиперссылками. Они позволяют выделить отдельные области изображений и определить для каждой из них свое действие.

Такие карты можно создать с помощью тега **<MAP>** и в общем виде описываются следующим образом:

<MAP NAME=«имя карты»>

<AREA HREF=«URL» SHAPE=«параметр» COORDS=«x1. y1. x2. y2»

ALT=«альтернативный текст»>

</MAP>

Атрибут **NAME** определяет имя карты, записываемое латинскими символами. Следует учитывать, что при задании имени карты важно соблюдение регистра, поскольку интерпретатор различает в данной команде строчное и заглавное написание.

Тег **<AREA>** определяет непосредственно активную область изображения.

Атрибут **HREF** указывает на адрес документа, вызываемого при нажатии кнопки мыши над данным участком изображения.

Атрибут **ALT** определяет вывод альтернативного текста при наведении на область изображения.

Атрибут **SHAPE** управляет формой активной области и может принимать одно из трех значений: **RECT** — прямоугольник, **CIRCLE** — круг или **POLY** — многоугольник.

Атрибут **COORDS** позволяет определить относительные координаты вершин активной области.

Значение SHAPE	Синтаксис записи COORDS	Значения параметров COORDS
RECT	COORDS="x1, y1, x2, y2"	x1 и y1 задают координаты левого верхнего угла фигуры, x2и y2 - правого нижнего
CIRCLE	COORDS="X, Y, R"	X и Y - координаты центра окружности, R - ее радиус в пикселях
POLY	COORDS="x1, y1, x2, y2, x3, y3, ... xN, yN"	x1, y1 ... xN, yN -координаты вершин многоугольника

Значения координат активной области отсчитываются в пикселях по длине и ширине картинки от левого верхнего угла изображения, принимаемого за точку с координатами 0,0. Координаты x и y могут быть также заданы в процентах от реального размера изображения, например: **SHAPE=«RECT» COORDS=«0.0.50%.50%»**. Необходимо помнить, что если

несколько активных областей одного изображения перекрывают друг друга, область, описанная тегом **<AREA>** первой, имеет приоритет.

Вот несколько примеров создания активных областей:

SHAPE=«RECT» COORDS=«0, 0, 20, 20» — создает прямоугольник размером 20x20 пикселей в левом верхнем углу изображения;

SHAPE=«CIRCLE» COORDS=«30, 30, 10» — создает круг с центром, расположенном в точке (30, 30) и радиусом 10 пикселей;

SHAPE=«POLY» COORDS=«10, 60, 15, 30, 30, 60» — создает треугольник с координатами вершин (10, 60), (15, 30) и (30, 60).

Теперь, когда графическая карта описана при помощи тега **<MAP>**, можно поместить ее Web-страницу, подготовив соответствующее изображение и вызвав его тегом **** с атрибутом **USEMAP**:

Обратите внимание на то, что имя карты, заданное атрибутом **NAME** тега **<MAP>**, предваряется в атрибуте **USEMAP** тега **** символом «#». Все остальные атрибуты данной директивы записываются так же, как обычно.

Специальные ссылки

Кроме обычных в HTML-документах можно использовать специальные ссылки.

К ним относятся:

- ссылки на электронный адрес;
- ссылка на сайт FTP;
- ссылки на группы новостей;
- ссылки на серверы Telnet.

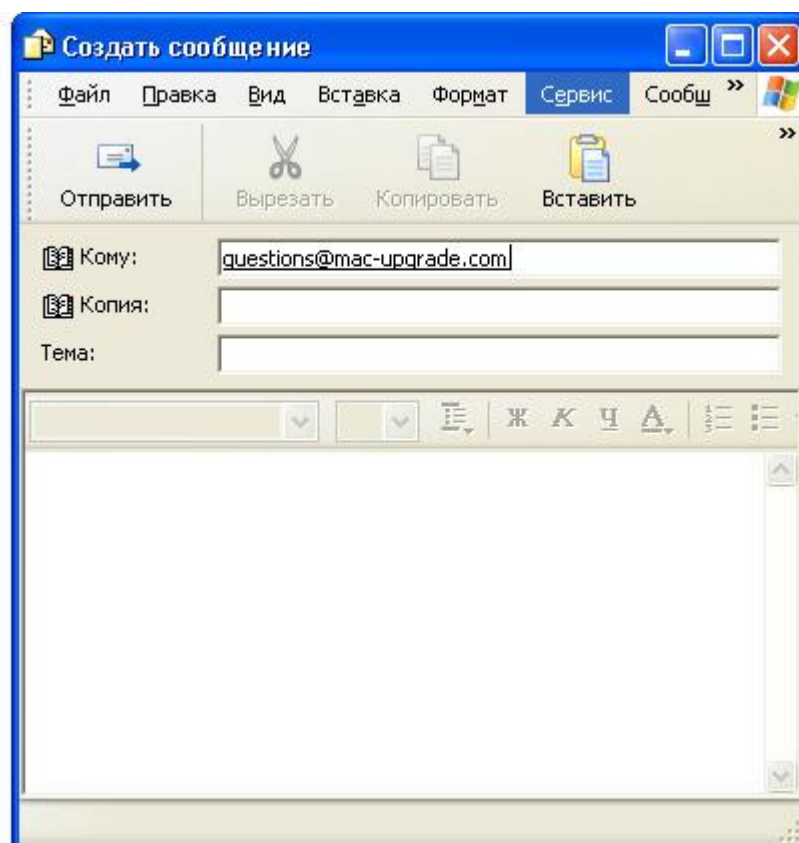
Создание ссылки на e-mail осуществляется вводом электронного адреса, состоящий из четырех частей: имени пользователя, символа @, имени компьютера и имени домена.

Например, questions@mac-upgrade.com

Именем пользователя в данном случае является questions, именем домена -mac-upgrade.com, имя компьютера в данном случае отсутствует.

Ссылка на e-mail запишется в виде: ****
Присылайте вопросы на мой e-mail. ****

При щелчке на данной ссылке автоматически подключит вас к электронной почте.



Некоторые браузеры позволяют автоматически заполнять поле **Тема писем**. Делается это с помощью атрибута **TITLE**: ` Заказать книгу `

Ссылка на сайт FTP

Протокол FTP используется для обмена файлами между компьютерами. Пользователи FTP должны установить сессию с удаленной машиной, после чего могут получить нужные файлы.

Все, что нужно для создания ссылки на FTP, — это адрес соответствующего сервера. Например, для создания ссылки на сервер, расположенный по адресу **ftp.microsoft.com**, следует написать: ` FTP-сайт Microsoft `

Если вы предлагаете пользователю скачать какой-нибудь файл, нужно указать путь к нему. Например, у вас есть файл **PROGRAM.ZIP** в каталоге **LOADS**, доступ к которому вы хотите открыть.

Ссылка в этом случае должна выглядеть следующим образом: ` Программа `.

Эта команда приказывает браузеру соединиться с FTP-сервером, перейти в указанный каталог и немедленно начать скачивать файл.

Протокол HTTP тоже способен передавать двоичные данные, причем особенно хорошо он справляется с передачей *exe*, *zip*. Обычная HTTP-ссылка на файл приводит к тому, что браузер начинает прием этого файла и предлагает сохранить его на жестком диске.

Например: `Получить файл `.

При этом у пользователя спросят, где он хочет сохранить файл, и начнется его пересылка.

Ссылки на группы новостей

Группы новостей USENET — это группы Интернет-конференций. Ссылка на группу новостей может понадобиться тогда, когда на вашем сайте обсуждается какая-то смежная проблема.

Типичная ссылка на группу новостей выглядит следующим образом: ** Конференция Usenet **

При этом браузер выполнит одно из двух действий: либо попытается самостоятельно подключиться к серверу новостей и найти группу сообщений, либо передаст команду программе работы с Usenet-конференциями.

Если вы хотите, чтобы ссылка указывала на сервер новостей, а не на конкретную конференцию, следует вместо обычного имени протокола **NEWS** написать **NEWS://**. Тогда ссылка будет выглядеть так: ** Посетите наш сервер новостей **

Ссылки на серверы TELNET

Ссылка на сервер TELNET позволяет пользователю установить сессию непосредственно с тем компьютером, на котором установлено программное обеспечение для удаленного доступа. Синтаксис Telnet-ссылки ничего сложного не представляет. Вы пишете выражение **TELNET://**, а вслед за ним указываете адрес удаленного компьютера: ** Установить связь с Telnet-сервером **

Таблицы в HTML_документах

Таблицы используются для представления информации в табличном виде, а также для разметки Web-страниц.

Таблица начинается открывающимся тегом **<TABLE>** и завершается закрывающимся **</TABLE>**.

Тег **<TABLE>** может включать следующие атрибуты:

WIDTH=«n» — определяет ширину таблицы в пикселях или процентах, по умолчанию ширина таблицы определяется содержимым ячеек.

HEIGHT=«n» — определяет высоту таблицы в пикселях или процентах.

Например, таблица высотой 200 и шириной 200 пикселей задается следующим образом:

<TABLE WIDTH=«200» HEIGHT=«200»> ... </TABLE>

BORDER=«n» — устанавливает толщину рамки. По умолчанию n=0 — таблица рисуется без рамки.

BORDERCOLOR=«#FFFFFF» — устанавливает цвет окантовки.

BGCOLOR=«#FFFFFF» — устанавливает цвет фона для всей таблицы.

Например,

<TABLE BORDER=«3» BORDERCOLOR=«RED» BGCOLOR=«GREEN»> ... </TABLE>

задаст ширину границы в 3 пикселя, цвет границы — красный, цвет фона таблицы — зеленый.

BACKGROUND="image.gif" — заполняет фон таблицы изображением.

CELLSPACING=«n» — определяет расстояние между рамками ячеек таблицы в пикселях.

CELLPADDING=«n» — определяет расстояние в пикселях между рамкой ячейки и текстом.

ALIGN=«LEFT» — определяет расположение таблицы в документе. По умолчанию таблица прижата к левому краю страницы. Допустимые значения атрибута **LEFT** (слева), **CENTER** (по центру страницы) и **RIGHT** (справа).

FRAME=«значение» — управляет внешней окантовкой таблицы, может принимать следующие значения:

VOID — окантовки нет (значение по умолчанию).

ABOVE — только граница сверху.

BELOW — только граница снизу.

HSIDES — границы сверху и снизу.
VSIDES — только границы слева и справа.
LHS — только левая граница.
RHS — только правая граница.
BOX — рисуются все четыре стороны.
BORDER — также все четыре стороны.

Таблица может включать заголовок, который располагается между тегами `<CAPTION> ... </CAPTION>`. Он должен быть непосредственно после тега `<TABLE>`. В таблице не может быть более одного `<CAPTION>`.

К заголовку возможно применение атрибута **ALIGN**, определяющего его положение относительно таблицы:

TOP — значение по умолчанию, заголовок над таблицей по центру.
LEFT — заголовок над таблицей слева.
RIGHT — заголовок над таблицей справа.
BOTTOM — заголовок под таблицей по центру.
По умолчанию `<CAPTION>` отображается над таблицей.

Задание строк и ячеек таблицы

Строки таблицы задаются тегом `<TR> ... </TR>`.

У элементов строк может быть два атрибута **ALIGN** и **VALIGN**. Они используются для определения выравнивания данных в ячейках по горизонтали и вертикали соответственно.

Атрибут **ALIGN** может иметь значения **CENTER**, **LEFT** и **RIGHT** (что означает выравнивание по центру, левому и правому краю). У атрибута **VALIGN** могут быть значения: **TOP**, **BOTTOM** и **CENTER** (верх, низ, середина).

Высота строки задается атрибутом **HEIGHT**. Следует учесть, что общая высота строк должна быть равна высоте таблицы.

Цвет строки задается с помощью атрибутов **BGCOLOR**=«#FFFFFF» (Устанавливает цвет фона) и **BACKGROUND**="image.gif" (Заполняет фон строки изображением).

Каждая новая строка также задается тегами `<TR> ... </TR>` и может иметь свои значения выравнивания и заливки.

Ячейки таблицы задаются тегами `<TD> ... </TD>` и размещаются между тегами строки.

Например, запись `<TR> <TD> </TD> <TD> </TD> <TD> </TD> </TR>` означает, что в строке размещается три ячейки.

Для ячеек таблицы используются следующие атрибуты:

WIDTH — устанавливает ширину ячейки в пикселях (например, **WIDTH**=«200»).

HEIGHT — устанавливает высоту ячейки в пикселях (например, **HEIGHT**=«40»).

Заливка цветом или изображением, а также выравнивание данных в ячейках таблицы осуществляется аналогично строкам.

Кроме этого, любая ячейка таблицы может быть определена не тегами `<TD> ... </TD>`, а тегами `<TH> ... </TH>` — Table Header (заголовок таблицы). В принципе, это обычная ячейка, но текст внутри этих тегов будет выделен полужирным шрифтом и отцентрирован.

Если ячейка пустая, то вокруг нее рамка не рисуется. Если рамка все же нужна вокруг пустой ячейки, то в нее надо ввести символьный объект ` ` (неразрывный пробел).

Теги, устанавливающие шрифт (``, `<I>`, `<FONT SIZE=«n»`, `FONTCOLOR=«#FFFFFF»`), необходимо повторять для каждой ячейки.

Например, необходимо создать следующую таблицу:

ЗНАМЕНИТЫЕ АЛМАЗЫ

Название	Страна	Год	Масса
Куллинан	Южная Африка	1905	3106
Эксцельсиор	Южная Африка	1893	971,5
Звезда Сьерра-Леоне	Западная Африка	1972	968,9

Таблица состоит из заголовка, 4-х строк по 4-е ячейки в каждой строке. Первая строка — строка заголовков. На языке HTML это выглядит так:

```
<TABLE>
<CAPTION>ЗНАМЕНИТЫЕ
АЛМАЗЫ</CAPTION>
<TR><TH>Название</TH> <TH>Страна</TH>
<TH>Год</TH> <TH>МАССА</TH></TR>
<TR><TD>Куллинан</TD> <TD>Южная
Африка</TD> <TD>1905</TD>
<TD>3106</TD></TR>
<TR><TD> Эксцельсиор </TD> <TD>Южная
Африка</TD> <TD>1893</TD>
<TD>971,5</TD></TR>
<TR><TD> Звезда Сьерра-Леоне </TD>
<TD>Западная Африка</TD> <TD>1972</TD>
<TD>968,9</TD></TR>
</TABLE>
```

Название	Страна	Год	МАССА
Куллинан	Южная Африка	1905	3106
Эксцельсиор	Южная Африка	1893	971,5
Звезда Сьерра-Леоне	Западная Африка	1972	968,9

Для задания границ, цвета границы и заливки таблицы в тег **<TABLE>** необходимо добавить атрибуты **BORDER=2**, **BORDERCOLOR=«GREEN»** и **BGCOLOR=«RED»**, которые зададут зеленую границу в 2 пункта и красный цвет таблицы.

```
<TABLE BORDER=2
BORDERCOLOR="GREEN"
BGCOLOR="RED">
<CAPTION>ЗНАМЕНИТЫЕ
АЛМАЗЫ</CAPTION>
<TR><TH>Название</TH>
<TH>Страна</TH> <TH>Год</TH>
<TH>МАССА</TH></TR>
<TR><TD>Куллинан</TD> <TD>Южная
Африка</TD> <TD>1905</TD>
<TD>3106</TD></TR>
<TR><TD> Эксцельсиор </TD>
<TD>Южная Африка</TD>
<TD>1893</TD> <TD>971,5</TD></TR>
<TR><TD> Звезда Сьерра-Леоне </TD>
<TD>Западная Африка</TD>
<TD>1972</TD>
<TD>968,9</TD></TR></TABLE>
```

Название	Страна	Год	МАССА
Куллинан	Южная Африка	1905	3106
Эксцельсиор	Южная Африка	1893	971,5
Звезда Сьерра-Леоне	Западная Африка	1972	968,9

В ячейки таблицы можно вносить не только текстовые и числовые данные, но и изображения. Они вносятся с помощью тега ****. Оформление изображений осуществляется обычными способами.

Вложенные таблицы

Иногда в качестве данных таблицы выступают другие таблицы. Такие таблицы называют **вложенными**.

```
<TABLE BORDER=1
BORDERCOLOR="GREEN">
<TR> <TD>Основная таблица</TD>
<TD> <TABLE BORDER=1
BORDERCOLOR="RED">
<TR> <TD>Вложенная таблица</TD>
</TR> </TABLE>
</TD> </TR> </TABLE>
```

Основная таблица	Вложенная таблица
------------------	-------------------

При создании вложенных таблиц не забывайте о существовании плавающих таблиц, которые можно расположить справа или слева с помощью атрибута **ALIGN** элемента **<TABLE>**.

Плавающие таблицы позволяют лучше организовать информацию. Текст «огихает» таблицу, а специальная разметка позволяет обратить внимание на данные, содержащиеся в ней.

```
<TABLE WIDTH="100%" CELLPADDING="0"
CELLSPACING="0">
<TR> <TD> <TABLE WIDTH="150"
CELLPADDING="3">
<TR> <TD>Разметка для левой части</TD> </TR>
</TABLE>
</TD>
<TD> <TABLE CELLPADDING="5">
<TR> <TD><!-- Плавающая таблица -->
<TABLE ALIGN="RIGHT" CELLPADDING="5"
WIDTH="200" BORDER="1">
<TR BGCOLOR="GREEN">
<TD>ТАБЛИЦА</TD></TR>
<TR><TD>ЭТА Таблица может перемещаться по
странице</TD> </TR> </TABLE>
<H1>Плавающие таблицы</H1>
<P> Плавающие таблицы позволяют лучше
организовать информацию. Текст «огихает»
таблицу, а разметка позволяет обратить внимание
на данные, содержащиеся в ней. /P>
</TD> </TR> </TABLE>
</TD> </TR> </TABLE>
```

Разметка для левой
части

Плавающие таблицы

Плавающие таблицы позволяют лучше организовать информацию. Текст «огихает» таблицу, а специальная разметка позволяет обратить внимание на данные, содержащиеся в ней

Таблица
Эта таблица может перемещаться по странице

Объединение ячеек

Иногда требуется сделать так, чтобы две или несколько ячеек имели одну общую границу (в рамках одной строки или одного столбца) и вели себя как одна ячейка. Причины для этого могут быть разные: одинаковая информация, содержащаяся в них, необходимость сделать единый заголовок для нескольких строк или столбцов и т. д.

Для этих целей используются два атрибута тегов **<TH>** и **<TD>** — **COLSPAN** и **ROWSPAN**. **COLSPAN** объединяет (растягивает) ячейки по горизонтали (по строке), **ROWSPAN** — по вертикали (по столбцу).

Например, **<TD COLSPAN=«2»>** означает, что ячейка будет растянута на 2 колонки, **<TD ROWSPAN=«2»>** — ячейка будет растянута на две строки таблицы.

Например, необходимо создать следующую таблицу:

Результаты тестирования			
Фамилия	Предметы		
	Математика	Физика	Информатика
Воробьев	85	85	90
Санина	74	72	58
Осипов	90	72	88

На языке HTML это выглядит следующим образом:


```

<TABLE BORDER=1>
<TR><TH COLSPAN=4>Результаты
тестирования</TH> </TR>
<TR> <TH ROWSPAN=2>Фамилия</TH>
<TH COLSPAN=3>Предметы</TH> </TR>
<TR><TH>Математика</TH> <TH>Физика</TH>
<TH>Информатика</TH> </TR>
<TR><TD>Воробьев</TD> <TD>85</TD>
<TD>85</TD> <TD>90</TD></TR>
<TR><TD>Санина</TD> <TD>74</TD>
<TD>72</TD> <TD>58</TD></TR>
<TR><TD>Осипов</TD> <TD>90</TD>
<TD>72</TD> <TD>88</TD></TR> </TABLE>

```

Результаты тестирования			
Фамилия	Предметы		
	Математика	Физика	Информатика
Воробьев	85	85	90
Санина	74	72	58
Осипов	90	72	88

Для объединения в первой строке 4-х ячеек в теге <TH> используется атрибут **COLSPAN**=«4». Для объединения 2-х ячеек в столбце — **COLSPAN**=«3».

Дмитрий Кудрец
Основы языка HTML. Часть вторая

Дмитрий Кудрец

Основы языка HTML

часть вторая



Аннотация

В книге рассказывается об использовании форм, фреймов и мультимедийных объектов в HTML-документах. Рекомендована учащимся школ, гимназий, а также всем желающим изучить основы языка HTML.

Основы языка HTML Часть вторая

Дмитрий Кудрец

© Дмитрий Кудрец, 2019

ISBN 978-5-4496-2167-2 (т. 2)

ISBN 978-5-4496-2166-5

Создано в интеллектуальной издательской системе Ridero

Формы

Форма – это инструмент, с помощью которого HTML-документ может отправить информацию по заданному адресу. Формы применяются для опроса посетителей, покупки чего-либо, отправки электронной почты.

Принцип работы форм следующий: пользователь заполняет форму, а после нажатия определенной кнопки форма берет данные из заполненных полей и отправляет их в назначенное место.

Формы размещаются между тегами **<FORM>** ... **</FORM>** .

HTML-документ может содержать в себе несколько форм, но они не должны находиться одна внутри другой.

Тег **<FORM>** может содержать следующие атрибуты:

ACTION – обязательный атрибут. Определяет, где находится обработчик формы. Он должен содержать URL скрипта, который будет обрабатывать полученные данные. Очень часто скрипты хранятся в директориях под названием **BIN/** или **CGI-BIN/** на сервере.

Примером элемента, задающего границы формы, может быть следующий код:

```
<FORM METHOD =«POST» ACTION =«HTTP://WWW.F.NET/CGI-BIN/S.PL»>  
</FORM >
```

METHOD – определяет, каким образом данные из формы будут переданы обработчику. Допустимые значения: **METHOD=POST** и **METHOD=GET** . По умолчанию предполагается **METHOD=GET** .

Метод **GET** означает, что данные формы будут добавлены в конец URL назначения. В большинстве случаев это накладывает серьезные ограничения на размеры данных (чаще всего не более 100 символов). Впрочем, если форма состоит из одного-двух элементов и важно передавать данные с высокой скоростью, то используется именно **GET** .

Например: **<FORM METHOD =«GET» ACTION =«/CGI-BIN/SEARCH»>**

Если же вы собираетесь поместить большое количество данных в большую форму, метод **GET** использовать не стоит.

Для таких случаев существует **POST** , который посылает данные отдельно и не имеет практического ограничения на их размер. Открывающий тег элемента **<FORM>** при использовании **POST** выглядит так: **<FORM METHOD =«POST» ACTION =«/CGI-BIN/SURVEY.PL »>**

ENCTYPE – определяет, каким образом данные из формы будут закодированы для

передачи обработчику. Его включают в элемент **<FORM>**; только в том случае, если вы просите пользователя прислать на сервер какой-либо файл. В этом случае следует указать следующее значение: **ENCTYPE** =«MULTIPART/FORM-DATA».

Атрибуты **NAME** и **ID** можно применять для идентификации форм при использовании скриптов или таблиц стилей. Атрибут **ID** является совместимым со стандартом XHTML, а **NAME** – это лишь дань старым традициям. Надежнее всего использовать оба атрибута:

NAME =«MYFORM» **ID** =«MYFORM»

Для внесения информации пользователем в форму используется элемент **<INPUT>**.

Каждый элемент **<INPUT>** включает атрибут **NAME=имя**, определяющий имя данного поля (идентификатор поля).

Для ввода данных существуют следующие типы элементов:

TYPE=« TEXT» – определяет окно для ввода строки текста. Может содержать дополнительные атрибуты **SIZE**=« число» (ширина окна ввода в символах) и **MAXLENGTH**=« число» (максимально допустимая длина строки в символах).

```
<INPUT TYPE=text SIZE=20 NAME=User  
VALUE="LENIN INC">
```



Определяет окно шириной 20 символов для ввода текста. По умолчанию в окне находится текст «Введите текст», который пользователь может изменить.

TYPE=« PASSWORD» – определяет окно для ввода пароля. Абсолютно аналогичен типу **TEXT**, только вместо символов вводимого текста показывает на экране звездочки (*):

```
<INPUT TYPE=password NAME=PW SIZE=20  
MAXLENGTH=10>
```



Определяет окно шириной 20 символов для ввода пароля. Максимально допустимая длина пароля – 10 символов.

TYPE=« RADIO» – определяет переключатели. Может содержать дополнительный атрибут **CHECKED** (показывает, что кнопка отмечена). В группе переключателей с одинаковыми именами может быть только один помеченный переключатель.

```
<INPUT TYPE=radio NAME=Question VALUE="Yes"  
CHECKED> Да<BR>  
<INPUT TYPE=radio NAME=Question VALUE="No"> Нет<BR>  
<INPUT TYPE=radio NAME=Question VALUE="Possible">  
Возможно
```



Определяет группу из трех переключателей, подписанных **YES**, **NO** и **POSSIBLE**. Первоначально помечен первый переключатель. Если пользователь не отметит другую кнопку, обработчику будет передана переменная **QUESTION** со значением **YES**. Если пользователь отметит другую кнопку, обработчику будет передана переменная **QUESTION** со значением **NO** или **POSSIBLE**.

TYPE=« CHECKBOX» – определяет флажки, в которых можно сделать пометку. Может содержать дополнительный атрибут **CHECKED** (показывает, что квадрат помечен).

В отличие от переключателей, в группе флажков с одинаковыми именами может быть несколько помеченных.

```
<INPUT TYPE=checkbox NAME=Comp VALUE="CPU">
Процессоры<BR>
<INPUT TYPE=checkbox NAME=Comp VALUE="Video"
CHECKED> Видеоадаптеры<BR>
<INPUT TYPE=checkbox NAME=Comp VALUE="Scan">
Сканеры<BR>
<INPUT TYPE=checkbox NAME=Comp VALUE="Modem"
CHECKED> Модемы
```

☐ Процессоры
☒ Видеоадаптеры
☐ Сканеры
☒ Модемы

Определяет группу из четырех флажков. Первоначально помечен второй и четвертый. Если пользователь не произведет изменений, обработчику будут переданы две переменные: **COMP** =«VIDEO» и **COMP** =«MODEM».

TYPE=« HIDDEN» – определяет скрытый элемент данных, который не виден пользователю при заполнении формы и передается обработчику без изменений. Такой элемент иногда полезно иметь в форме, которая время от времени подвергается переработке, чтобы обработчик мог знать, с какой версией формы он имеет дело.

Например, <input type="hidden" name="VERSION" value="1.1"> определяет скрытую переменную **VERSION**, которая передается обработчику со значением 1.1.

TYPE=« SUBMIT» – определяет кнопку, при нажатии на которую запускается процесс передачи данных из формы обработчику.

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

Отправить

TYPE=«RESET» — определяет кнопку, при нажатии на которую очищаются поля формы. Поскольку при использовании этой кнопки данные обработчику не передаются, кнопка типа **RESET** может и не иметь атрибута **NAME**.

```
<INPUT TYPE="reset" VALUE=" Сброс ">
```

Сброс

Формы могут содержать поля для ввода большого текста <**TEXTAREA**>.

```
<TEXTAREA NAME=address ROWS=5
COLS=50> Наберите здесь
сообщение</TEXTAREA>
```

Наберите здесь сообщение

Атрибут **NAME** определяет имя, под которым содержимое окна будет передано обработчику.

Атрибут **ROWS** устанавливает высоту окна в строках.

Атрибут **COLS** устанавливает ширину окна в символах.

Текст, размещенный между тегами <**TEXTAREA**> ... </TEXTAREA>, представляет собой содержимое окна по умолчанию. Пользователь может его отредактировать или просто стереть.

Кроме всего этого формы могут содержать меню выбора, которое начинается открывающимся тегом **<SELECT>** (содержит обязательный атрибут **NAME**, определяющий имя меню) и завершается закрывающимся **</SELECT>**.

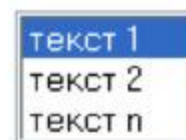
Между ними находятся теги **<OPTION>**, определяющие элемент меню. Обязательный атрибут **VALUE** устанавливает значение, которое будет передано обработчику, если выбран этот элемент меню. Тег **<OPTION>** может включать атрибут **SELECTED**, показывающий, что данный элемент выбран/отмечен по умолчанию.

```
<SELECT NAME="имя"><BR>
<OPTION VALUE="option_1" selected>текст 1<BR>
<OPTION VALUE="option_2">текст 2<BR>
<OPTION VALUE="option_n">текст n<BR>
</SELECT>
```



Тег **<SELECT>** может также содержать атрибут **MULTIPLE**, присутствие которого показывает, что из меню можно выбрать несколько элементов. Большинство Обозревателей показывают меню **<SELECT MULTIPLE>** в виде окна, в котором находятся элементы меню. Высоту окна в строках можно задать атрибутом **SIZE**=«число».

```
<SELECT MULTIPLE SIZE=3 NAME="имя"><BR>
<OPTION VALUE="option_1" SELECTED>текст 1<BR>
<OPTION VALUE="option_2">текст 2<BR>
<OPTION VALUE="option_n">текст n<BR>
</SELECT>
```



Группы элементов формы

Для группирования элементов форм Используется тег **<FIELDSET>**. Все, что находится между ним и его закрывающим тегом **</FIELDSET>**, будет обведено в рамку и озаглавлено так, как указано в теге **<LEGEND>**.

Теги **FIELDSET** и **LEGEND** поддерживаются пока только в Internet Explorer версии 4 и выше, а браузер Netscape начал поддерживать их только в шестой версии. Кроме того, элементы **FIELDSET** пока не очень хорошо позиционируются в HTML-документах.

Тег **<FIELDSET>** можно употребить следующим образом:

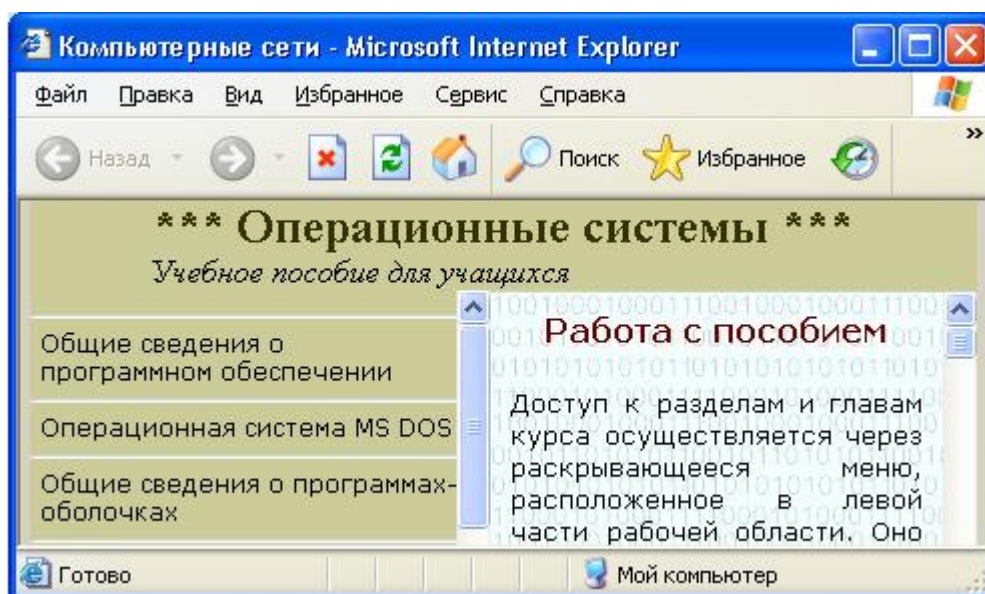
```
<FIELDSET TITLE=«Сведения о клиенте»>
<LEGEND ALIGN=«left»> Расскажите о себе </LEGEND>
```

После этого можно поместить несколько элементов формы и поставить закрывающий тег **/FIELDSET**.

В браузере значение атрибута **TITLE** выводится в качестве всплывающей подсказки при наведении мыши на содержимое элемента **FIELDSET**. Это относится также и к другим тегам, которые могут иметь атрибут **TITLE**. А то, что находится между тегами **LEGEND** и **/LEGEND**, будет вынесено в заголовок рамки.

Фреймы

Фреймы позволяют выводить в одном окне браузера одновременно несколько HTML-документов, имеющих разные URL, различные полосы прокрутки (они могут и отсутствовать) и ведущих себя довольно независимо.



Фреймы кажутся весьма привлекательными с точки зрения удобства обслуживания сайта: например, для того чтобы изменить состав навигационных элементов, нет необходимости вносить изменения во все страницы, достаточно лишь подправить документ, отображающийся в соответствующем фрейме.

Однако на практике оказывается, что объективных недостатков у данной формы представления значительно больше, чем достоинств.

Во-первых, фреймы весьма сложны в разработке и настройке. В некоторых фреймах возможно периодическое появление полос прокрутки, затрудняющих просмотр документа, а жесткий запрет на отображение средств скроллинга может повлечь за собой исчезновение части отображаемых во фреймах страниц за границей видимой области экрана.

Во-вторых, фреймы распознаются далеко не всеми браузерами, в полной мере их поддерживают лишь современные версии Microsoft Internet Explorer и Netscape Navigator.

В-третьих, применение фреймов существенно затрудняет управление навигацией сайта при помощи встроенных в браузер кнопок «Вперед» и «Назад».

И последним, самым существенным недостатком фреймов является то, что использующие их страницы не всегда корректно индексируются поисковыми серверами. Использовать фреймы лучше всего только тогда, когда это действительно необходимо, например:

- для создания единообразной системы навигации по сайту. С их помощью можно организовать стройную и понятную структуру, при этом не повторять ссылочное меню на каждой странице;

- необходимость одновременной загрузки двух разных страниц;

- необходимость размещения информации, постоянно присутствующей в HTML-документе.

Во всех остальных случаях от их применения лучше отказаться.

Создание фреймов

Фреймы задаются элементом `<FRAMESET> ... </FRAMESET>`.

У элемента `<FRAMESET>` может быть два атрибута: **COLS** и **ROWS**.

COLS — задает разбиение страницы на колонки.

ROWS — задает разбиение страницы на строки.

Значение параметров **COLS** и **ROWS** может задаваться как в пикселях, так и в процентах. Например, запись `<FRAMESET COLS=«200, 100, *»>` обозначает, что будет создано три колонки. Первая — шириной 200 пикселей, вторая — 200. Знак «*» означает, что колонка займет все оставшееся место. Задание ширины колонок или высоты строк в пикселях устанавливает жесткие параметры, так что лучше использовать проценты.


```
<FRAMESET COLS="25%, 75%">
</FRAMESET>
```

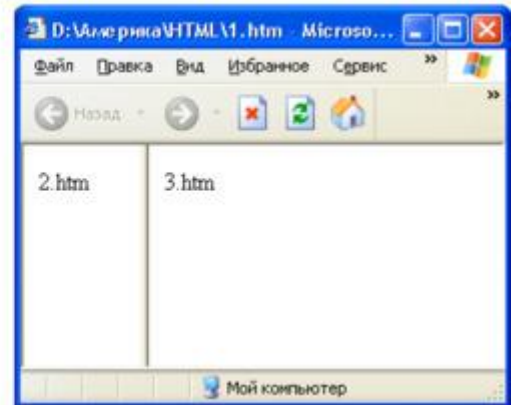


Не удивляйтесь, если у вас ничего не отобразится на экране. Данный пример не будет работать в браузере до тех пор, пока вы не включите в контейнер **<FRAMESET>** хотя бы один элемент **<FRAME>**.

Элемент **<FRAME SRC=«URL»>** определяет содержимое создаваемых вами колонок или строк.

Например, если вы определяете набор фреймов, состоящий из двух колонок, то нужно написать два элемента **<FRAME>**:

```
<FRAMESET COLS="25%, 75%">
<FRAME SRC="2.HTM">
<FRAME SRC="3.HTM">
</FRAMESET>
```



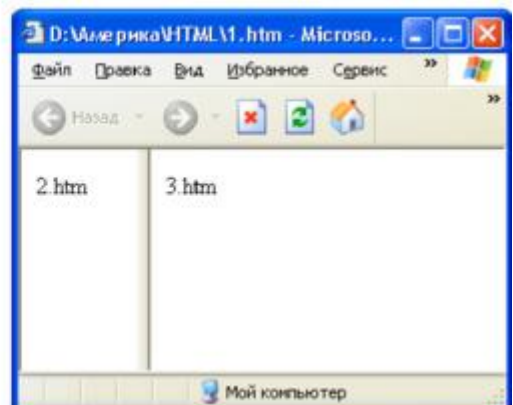
Внутри фреймов отдельные страницы могут содержать гиперссылки, причем указывается, в каком фрейме будут выводиться документы, расположенные по URL, на который они указывают. Например, можно создать фрейм под названием «главное окно просмотра», в нем будут отображаться все страницы, на которые указывают ссылки.

Для задания границ фреймов используются атрибуты **FRAMEBORDER=«значение»** и **FRAMESPACING=«значение»**.

FRAMEBORDER определяет, надо ли отображать на экране пользователя границы фреймов. Параметр этого атрибута может принимать одно из двух значений: 1, если разделители нужны (используется по умолчанию), или 0, если программист желает сделать их невидимыми.

FRAMESPACING указывает на толщину разделителей в пикселях.

```
<FRAMESET cols="25%, 75%"
FRAMEBORDER="1"
FRAMESPACING="5">
<FRAME SRC="2.htm">
<FRAME SRC="3.htm">
</FRAMESET>
```



Тег **<FRAME>** может включать следующие атрибуты:

SCROLLING — определяет наличие («YES»), отсутствие («NO») или отображение по мере необходимости («AUTO») полос прокрутки в текущем фрейме.

NORESIZE — установка запрета на изменение размеров окна фрейма.

MARGINWIDTH и **MARGINHEIGHT** — это атрибуты, которые отвечают за горизонтальные и вертикальные поля. Оба принимают числовые значения в пикселях.

SRC — указывает адрес открываемой в данном фрейме страницы.

NAME — задает уникальное имя для данного конкретного фрейма, набранное с использованием символов латинского алфавита.

TARGET=«значение» — содержит информацию о целевом фрейме (рекомендуется использовать значение по умолчанию — «CONTENTS»). Если вместо значения атрибута **TARGET** подставить имя фрейма, заданное вами ранее в качестве параметра атрибута **NAME**, при активизации ссылки целевой документ загружается в окне с указанным именем.

Допустимые значения атрибута **TARGET** при использовании гиперссылок следующие:

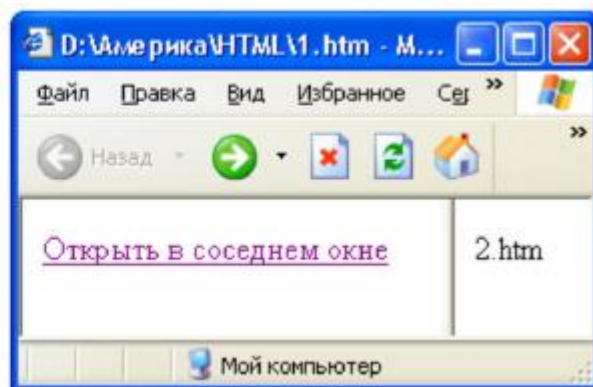
_SELF — документ, с которым установлена гиперсвязь, открывается в текущем фрейме;

_PARENT — отображение осуществляется в родительском окне фреймов, независимо от того, какие параметры указаны в директиве **<FRAMESET>**;

_TOP — при активизации гиперссылки фреймы перестают отображаться, а содержимое целевого документа выводится в отдельном окне.

_BLANK — с помощью этого значения можно открывать документ в новом окне браузера.

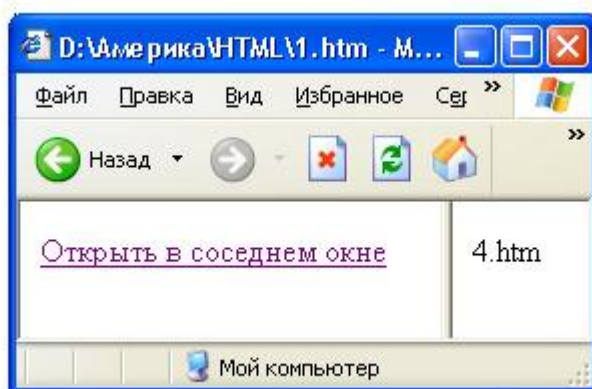
```
<FRAMESET COLS="25%, 75%">  
<FRAME SRC="3.htm" >  
<FRAME SRC="2.htm" NAME="RRR">  
</FRAMESET>
```



Содержимое файла **3.htm**:

** Открыть в соседнем окне **

При нажатии на гиперссылку новый документ открывается в заданном окне.



Вложенные фреймы

Иногда приходится создавать сложные фреймовые структуры. Разделение колонок на строки и строк на колонки осуществляется при помощи вложенных элементов **<FRAMESET>**.

```
<FRAMESET ROWS="100, *">
<FRAME SRC="2.htm"
SCROLLING="NO">
<FRAMESET COLS="25%, 75%">
<FRAME SRC="3.htm">
<FRAME SRC="4.htm">
</FRAMESET>
</FRAMESET>
```

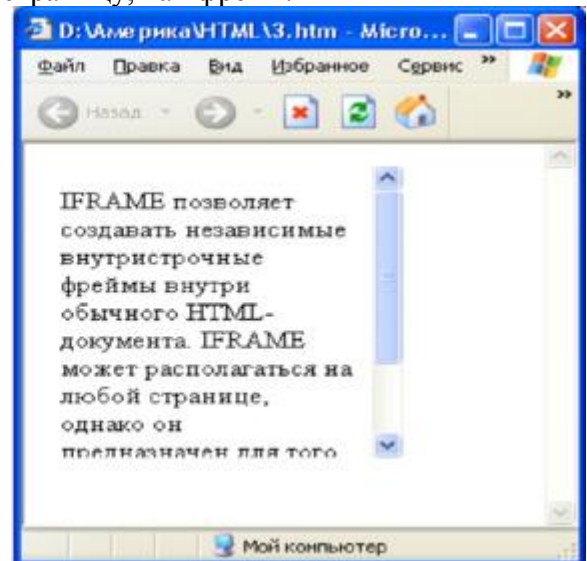


В данном примере второй **<FRAMESET>** вложен в первый. На самом деле он заменяет собой второй элемент **<FRAME>**, требующийся первому набору фреймов. В нем и определяются две колонки, на которые разделяется вторая строка.

Элемент **<IFRAME>**

<IFRAME> позволяет создавать независимые внутрискриптовые фреймы внутри обычного HTML-документа. **<IFRAME>** может располагаться на любой странице, однако он предназначен для того, чтобы содержать другую страницу, как фрейм.

```
<IFRAME SRC="2.htm" WIDTH="200"
HEIGHT="200" FRAMEBORDER="1"
SCROLLING="yes">Если вы видите этот
текст, значит ваш браузер не имеет
поддержку внутрискриптовых фреймов.
Щелкните <A HREF="2.htm"> Здесь</a>,
чтобы увидеть содержимое этого фрейма
</IFRAME>
```



Элемент **<IFRAME>** имеет те же атрибуты, что и **<FRAME>**, включая **FRAMEBORDER**, **MARGINWIDTH**, **MARGINHEIGHT** и **SCROLLING**.

Кроме того, есть и специфические атрибуты именно данного контейнера: **HEIGHT** и **WIDTH**. Они задают, соответственно, высоту и ширину фрейма в пикселях.

Еще **<IFRAME>** может иметь атрибут **ALIGN**, делающий внутрискриптовый фрейм плавающим. Значениями этого атрибута, как обычно, являются **RIGHT** и **LEFT**.

Текст, который содержится в **<IFRAME>**, выводится на экран в том случае, когда браузер пользователя не имеет поддержки внутрискриптовых фреймов.

Элемент **<NOFRAMES>**

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

Например, **<NOFRAMES> <P>** Ваша версия WEB-браузера не поддерживает фреймы!
</P> </NOFRAMES>

Мультимедиа в HTML-документах

Термин **мультимедиа** относится к такому компьютерному представлению информации, которое состоит из более чем одного типа данных. Большинство элементов мультимедиа, основаны на том или ином зависящем от времени материале, будь то видео, звук или анимация. Процесс проигрывания можно запустить, остановить, «перемотать», как обычную видео- или аудиокассету. Некоторые мультимедиа-презентации содержат интерактивные элементы — например, кнопки перехода. Но чаще всего мультимедийные элементы напоминают привычные видео- и аудиоролики. Существует множество типов мультимедийных файлов.

Формат файла	Тип файла	Расширение
Sun Systems sound	Цифровое аудио	.au
Windows sound	Цифровое аудио	.wav
Audio Interchange	Цифровое аудио	.aiff, .aifc
MPEG/MP3 audio	Цифровое аудио	.mpg, .mp3
MIDI audio	Команды управления звуком	.mid, .midi
CompuServe GIF	Графика	.gif
JPEG (метод сжатия)	Графика	.jpg, .jpeg
TIFF	Графика	.tif, .tiff
Windows bitmap	Графика	.bmp
Fractal animations	Анимация	.fli, .flc
MPEG video	Видео	.mpg, .mpeg
QuickTime	Видео	.mov, .qt
Microsoft video	Видео	.avi
Digital video (формат DV)	Видео	.dv
Macromedia Shockwave Flash	Анимация	.swf
Microsoft Excel documents	Данные электронных таблиц	.xl, .xls
Microsoft Word documents	Форматированный текст	.doc

При вставке мультимедийных объектов не следует забывать о довольно больших объемах файлов, содержащих мультимедиа. Самыми большими являются видеофайлы. Минутный фрагмент может занимать 10...20 Мбайт и более. Трехминутная аудиозапись высокого качества занимает до 3,5 Мбайт. Анимация может занимать сотни килобайт, что гораздо больше, чем средний графический файл.

Связывание и внедрение мультимедийных объектов

Большинство браузеров поддерживает весьма ограниченный набор форматов, поэтому использование мультимедиа осуществляется двумя способами — **внедрением** и **связыванием**.

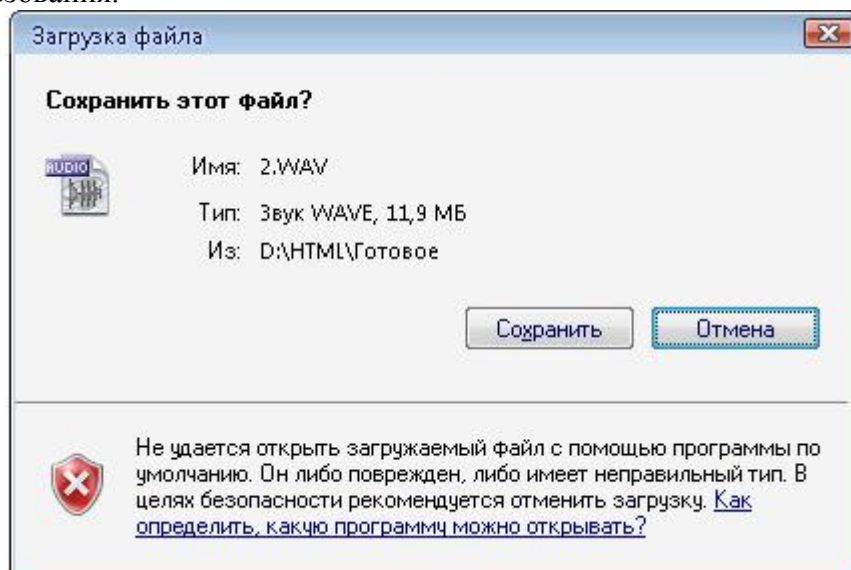
Внедрить непосредственно на страницу можно звуковые файлы типа WAV, MIDI, файлы в формате QuickTime, файлы Windows Media, анимация Macromedia Flash.

Внедрение может осуществляться при помощи обычных гиперссылок.

**** Поздравление (5 Мб)****

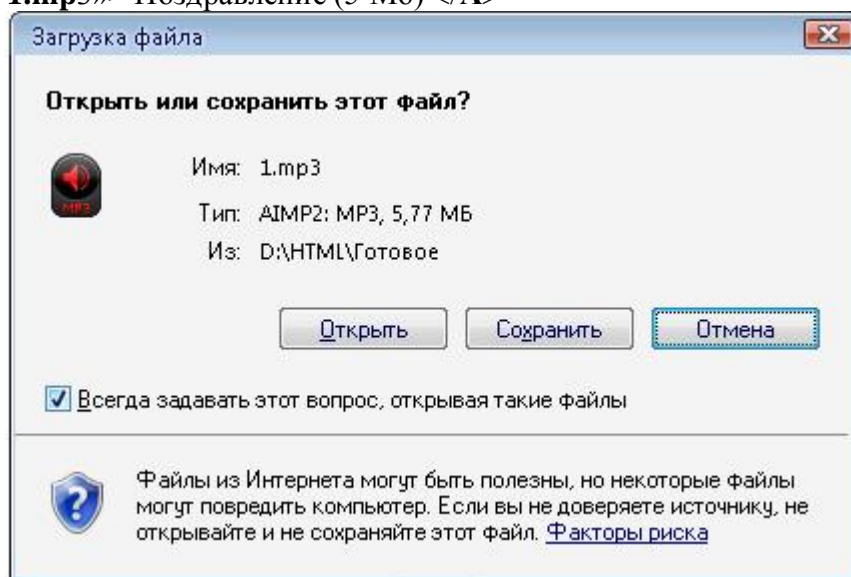
[Поздравление \(5 Мб\)](#)

Когда пользователь щелкает на ссылке, файл загружается на компьютер. После этого браузер должен запустить то приложение, которое ассоциировано с данным форматом. Если такого приложения нет или оно не настроено, можно просто сохранить файл на жестком диске для будущего использования.



В тексте ссылки старайтесь указывать примерный размер файла, тогда посетители смогут представить себе, сколько времени им потребуется на его загрузку. Также зачастую нужно получать разрешение на то, чтобы поставить ссылку на файл с чужого сервера. Дело не только в авторских правах, но и в том, что передача файлов больших размеров сильно замедляет работу сервера. За превышение трафика могут взимать дополнительную плату.

 Поздравление (5 Мб)



Другой способ внедрения мультимедиа — использование тега <EMBED>.

У <EMBED> есть атрибут **NAME**, что очень полезно при использовании JavaScript, а также атрибуты, задающие высоту и ширину (**WIDTH** и **HEIGHT**).

```
<EMBED NAME="MOVI" SRC="1.MOV"  
WIDTH="200" HEIGHT="200">  
</EMBED>
```

Видеозапись

Привет! Мы только что вернулись из отпуска, который провели на море. Две недели подряд мы загорали и записали небольшой фильм об этом.



Включение элемента мультимедиа в страницу делает управление им менее доступным для пользователей с ограниченными возможностями. Еще одно неудобство метода внедрения заключается в том, что при проигрывании видеофайлов прямо в окне браузера могут возникнуть определенные проблемы совместимости. Так что лучше применить метод связывания.

Также для внедрения объектов используется тег **<OBJECT>**. Он чаще всего используется для Java-апплетов, но помогает реализовывать и другие возможности внедрения. Например, можно внедрять один HTML-документ в другой. Internet Explorer 5.5 и последующие версии не имеют поддержки элемента **<EMBED>**. Чтобы внедрить мультимедийный элемент, теперь нужно применять элемент **<OBJECT>**.

Внедрение QuickTime

Если у вас есть желание внедрить в свою страницу видео в формате QuickTime, то это можно сделать при помощи элемента **<EMBED>**.

```
<EMBED NAME="MOVI" SRC="1.MOV"  
WIDTH="200" HEIGHT="200">  
</EMBED>
```

Видеозапись

Привет! Мы только что вернулись из отпуска, который провели на море. Две недели подряд мы загорали и записали небольшой фильм об этом.



При внедрении QuickTime можно использовать дополнительные атрибуты:

AUTOPLAY = «TRUE» — автоматически запускает видео.

CONTROLLER = «FALSE» — прячет панель управления.

LOOP = «TRUE» — зацикливает видео.

PLAYEVERYFRAME = «TRUE» — заставляет отказаться от пропуска части кадров, даже если скорость замедленная (ограничена техническими возможностями машины).

VOLUME = «0—256» — устанавливает уровень громкости. 256 — максимальная громкость.

HIDDEN = «TRUE» — скрывает видеоряд (полезно для звуковых файлов в формате QuickTime).

HREF = «URL» — создает окно, в котором проигрывается видео, как гиперссылку.

Перечисленные дополнительные атрибуты характерны только для внедрения с помощью **<EMBED>** файлов QuickTime!

Элемент **<EMBED>** иногда используется немного по-другому, делая процесс просмотра видео двухступенчатым.

Здесь изначально загружаемый видеофайл (post_movie.mov) подразумевается лишь демонстрационным, то есть он размещается в окне небольшого размера с изображением

низкого качества. Если пользователь желает просмотреть видео целиком с нормальным качеством, он может воспользоваться окном внедренного видео как гиперссылкой и перейти по ней на другой файл (full_movie.mov).

Поскольку в данном примере **CONTROLLER**=«FALSE», то пользователь не имеет возможности контролировать процесс воспроизведения с помощью панели управления — она скрыта. После загрузки клип начнет проигрываться автоматически.

Некоторые браузеры не имеют поддержки элемента **<EMBED>**, зато работает с ActiveX, технологией программного внедрения элементов, разработанной Microsoft. Для таких браузеров следует применять **<OBJECT>**

Формат Windows Media

Для файлов формата Windows Media оптимальным является не внедрение, а связывание. Для этих целей используется элемент **<EMBED>**.

Существуют следующие параметры элемента **<EMBED>**:

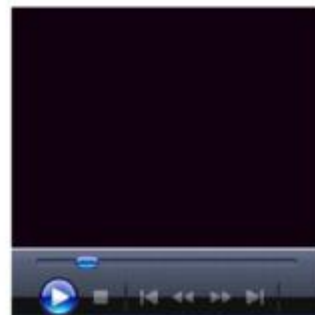
SHOWCONTROLS=«0» — панель управления скрыта; любые другие значения — панель видна.

AUTOSIZE=«0» — отсутствие автоподбора размера в соответствии с кадром.

SHOWSTATUSBAR=«0» — строка состояния скрыта.

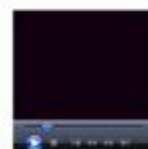
AUTOSTART =«0» — не запускать видео автоматически после загрузки.

```
<EMBED SRC="1.AVI" WIDTH="200"
HEIGHT="200" AUTOSTART=0 </EMBED>
```



Если вы применяете технологию ActiveX, следует снова вспомнить про **<OBJECT>**. Внутри него может быть **<EMBED>** — это для тех браузеров, которые «в танке».

```
<OBJECT ID="Player" TYPE="application/x-oleobject"
CLASSID="CLSID:6BF52A52-394A-11d3-B153-00C04F79FAA6"
STANDBY="Загружаем Media Player..." WIDTH="320" HEIGHT="240"
CODEBASE="http://activex.microsoft.com/activex/controls/mplayer/en/
nsmpt2inf.cab#Version=6.4.7.1112">
<PARAM NAME="autostart" VALUE="TRUE">
<PARAM NAME="URL"
VALUE="http://www.fakecorp.com/movies/rnymovie4avi">
<EMBED SRC="mov.avi" WIDTH="320" HEIGHT="240"
SHOWSTATUSBAR="-1" SHOWCONTROLS="-1" SHOWDISPLAY="1"
PLUGINSPAGE="http://www.microsoft.com/netshow/download/player.htm">
</EMBED>
</OBJECT>
```



Атрибут **CODEBASE** используется для автоматической загрузки и установки необходимых компонентов ActiveX.

Подключение анимации в формате Flash

Подключение анимации в формате Flash осуществляется с помощью двух элементов: **<EMBED>** и **<OBJECT>**.

```
<OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" WIDTH="100" HEIGHT="100"
CODEBASE="http://active.macromedia.com/flash5/
cabs/swflash.cab#version=5.0.0,0">
<PARAM NAME="movie" VALUE="flashmovie.swf">
<PARAM NAME="PLAY" VALUE="TRUE">
<PARAM NAME="LOOP" VALUE="TRUE"
<PARAM NAME="QUALITY" VALUE="HIGH"/>
<EMBED SRC="flashmovie.swf" WIDTH="200" HEIGHT="200"
PLAY="TRUE" LOOP="FALSE" QUALITY="HIGH"
PLUGINSOURCE="http://www.macromedia.com/
Shockwave/download/index.cgi?
pl_prod_version=shockwaveflash">
</EMBED></OBJECT>
```



Флэш-анимации можно сохранять и в формате QuickTime.

Файлы, не попадающие ни в одну из этих категорий, нуждаются во вспомогательных приложениях.

В поздних версиях Microsoft Windows таблица типов файлов и приложений, с ними связанных, входит в набор настроек самой операционной системы.