МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Серветник О. Л.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ

WEB-ТЕХНОЛОГИИ

Направление подготовки	44.03.01 - Педагогическое образование
Профиль	Информатика и информационные технологии в образовании
Квалификация выпускника	бакалавр
Форма обучения	очная/ заочная
Учебный план	2015 г
Изучается	в 8 семестре

Ставрополь, 2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ

WEB-ТЕХНОЛОГИИ

Направление подготовки	44.03.01 - Педагогическое образование
Профиль	Информатика и информационные технологии в образовании
Квалификация выпускника	бакалавр

Объем занятий: Итого	108 ч.	3 з.е.
В т.ч. аудиторных	50 ч.	
Из них:		
Лекций	20 ч.	
Лабораторных работ	30 ч.	
Практических занятий	0 ч.	
Самостоятельной работы	22 ч.	
Экзамен 8 семестр	36 ч.	

Ставрополь, 2016

Содержание

ВВЕДЕНИЕ
Язык разметки гипертекста HTML. Назначение HTML
World Wide Web

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

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

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

URI обычно состоят из трех частей:

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

Рассмотрим URI этой спецификации HTML на сервере w3:

```
http://www.w3.org/TR/PR-html4/cover.html
```

Этот URI может читаться следующим образом: этот документ можно получить по протоколу HTTP, он располагается на машине www.w3.org, путь к этому документу - "/TR/PR-html4/cover.html". Кроме того, в документах в формате HTML Вы можете увидеть схемы "mailto" для электронной почты и "ftp" для протокола FTP.

Язык разметки гипертекста HTML

Всемирная паутина (World Wide Web — WWW) состоит из множества связанных между собой электронных документов, представляющих кладезь информационных данных, описанных с помощью специальных технологических правил. Эти правила составляются на языке гипертекстовой разметки HTML (Hypertext Markup Language).

Можно с уверенностью сказать, что сегодня язык разметки HTML является основой всех размещенных в Интернете электронных документов. Он выступает в роли некоего фундамента, на базе которого реализуются прочие сетевые программные технологии,

призванные в конечном итоге повысить общую привлекательность, эффективность и интерактивность носителей информационных данных в Сети.

HTML постоянно набирает популярность, причем не только в сфере интернет-технологий, но и в области предоставления презентационных услуг, рекламно-выставочной деятельности, внедряется в состав программного обеспечения и пр. Корпоративные клиенты все чаще разрабатывают CD-презентации и демонстрационные ролики, заказывают электронные визитки и рекламные обращения. Частный пользователь все больше склоняется к мысли о составлении интерактивных портфелей и резюме, позволяющих, в отличие от листа бумаги, ярче и привлекательнее преподнести свои знания и умения потенциальному работодателю. Разработчики программного обеспечения делают информационные и рекламные вставки справочного характера внутри создаваемых ими программных пакетов. И все это так или иначе реализуется с помощью простого и доступного, но вместе с тем эффективного языка разметки HTML.

Термин *HyperText Markup Language (HTML)* - означает «язык маркировки гипертекстов» и включает в себя различные способы оформления гипертекстовых документов, дизайн, гипертекстовые редакторы, браузеры и многое другое. Человек, изучивший этот язык, обретает возможность делать сложные вещи простыми способами и, главное, быстро. HTML как основа создания Web-страниц имеет прямое отношение к такому направлению изобразительного искусства, как Web-дизайн. Художнику в Интернете недостаточно просто нарисовать красивые картинки, оригинальный логотип, создать новый фирменный стиль. Он должен разместить все это в Сети, продумать связь между Web-страницами, чтобы все двигалось, откликалось на действия пользователя, поражало воображение неискушенных клиентов, а у приверженцев Сети вызывало желание создать что-нибудь свое, оригинальное в этой области.

HTML дает авторам средства для:

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

Наиболее распространенными браузерами (программами) для просмотра HTML-файлов являются Microsoft Internet Explorer и Netscape Navigator. Открывая в браузере любую Web-страницу, Вы видите текст, картинки, кнопки,таблицы и многое другое. Как создается Web-страница? Для того чтобы создать Web-страницу, можно использовать текстовый редактор **Notepad (Блокнот).**

Откройте любую Web-страничку. То, что вы видите в браузере, это ее "лицо". Чтобы увидеть "изнанку" Web-страницы, выполните команду **View | HTML Code (Вид | Просмотр HTML кода).** Откроется текстовый редактор Блокнот, в котором вы увидите "устройство" этой страницы.

На лабораторных занятиях мы будем использовать текстовый редактор Блокнот для подготовки HTML-файлов, а браузер Microsoft Internet Explorer – как инструмент контроля за сделанным.

Краткая история HTML

Язык HTML был разработан Тимом Бернерс-Ли сотрудником Европейской лаборатории физики элементарных частиц и распространен браузером Mosaic, разработанным в NCSA. В 1990-х годах он добился особенных успехов благодаря быстрому росту Web. В это время HTML был расширен и дополнен. В Web очень важно использование

одних и тех же соглашений HTML авторами Web-страниц и производителями. Это явилось причиной совместной работы над спецификациями языка HTML.

Многие называют HTML языком программирования. Это не совсем верно, так как в традиционном понимании HTML является языком *разметки* электронных документов, лишь указывающим программам просмотра HTML-страниц (браузерам) форму представления описанной в документе информации. Начиная с середины 90-х годов XX века, HTML претерпел некоторые изменения в своей спецификации – варианты используемых инструкций, применяемых тегов и модулей горячо обсуждались и изменялись. На сегодня последней версией языка HTML является версия 4.01.

Со времени создания первой версии HTML претерпел некоторые изменения, но если сравнить исходные тексты различных Web-страниц, можно легко увидеть сходство их структур. Это объясняется тем, что документы создаются по определенным правилам. В основу синтаксиса языка HTML лег стандарт ISO 8879:1986 «Information processing. Text and office systems. Standard Generalized Markup Language (SGML)».

Начиная с момента своего возникновения, разработкой спецификации языка HTML стала заниматься организация под названием "Консорциум W3C" (World Wide Web Consortium). Ее основной задачей являлось составление и принятие технических рекомендаций единого стандарта разметки гипертекстовых документов. Практическая необходимость работы над стандартом была обусловлена постоянным ростом популярности Интернета, в рамках которого производители программ (браузеров) для просмотра Web-документов выдвигали свои предложения по улучшению правил описания гипертекстовых данных.

HTML 2.0 (ноябрь 1995) был разработан под эгидой Internet Engineering Task Force (IETF) для упорядочения общепринятых положений в конце 1994 года. HTML+ (1993) и HTML 3.0 (1995) - это более богатые версии языка HTML. Несмотря на то, что в обычных дискуссиях согласие никогда не было достигнуто, эти черновики привели к принятию ряда новых свойств. Усилия Рабочей группы World Wide Web Consortium по HTML в упорядочении общепринятых положений в 1996 привели к версии HTML 3.2 (январь 1997). На сегодняшний день наиболее распространенной является к версии HTML 4.0.

Версия HTML 4.0, выпущенная Консорциумом в середине 1997 года и ставшая признанной спецификацией в конце того же года, является на сегодня последней номерной версией языка разметки HTML. Некоторые несущественные дополнения, внесенные в спецификацию в декабре 1999 года, мало повлияли на структуру самого языка, а версия получила небольшое добавление в виде цифры 1, т. е. стала называться 4.01. Хочется добавить, что деятельность Консорциума по сути призвана регулировать и контролировать развитие и совершенствование языка гипертекстовой разметки HTML, учитывая потребности сферы интернет-технологий и компаний-разработчиков, работающих на рынке браузеров. Однако в действительности ситуация не такая однозначная — производители программ для просмотра HTML-документов постоянно предлагают технологические нововведения в спецификацию языка, часть которых получает одобрение Консорциума. Остальная часть инноваций, не вошедшая в состав принятой W3C рекомендации, может, тем не менее, внедряться в программную платформу выпускаемых браузеров, что на практике вызывает проблемы несовместимости электронных документов при их просмотре браузерами разных моделей и версий.

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

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

Язык HTML разрабатывался с той точки зрения, что все типы устройств должны иметь возможность использовать информацию в Web: персональные компьютеры с графическими дисплеями с различным разрешением и числом цветов, сотовые телефоны, переносные устройства, устройства для вывода и ввода речи, компьютеры с высокой и низкой частотой и т.д.

Существует большое различие между стандартом официальным и стандартом фактическим. HTML постоянно развивается, дополняется новыми элементами, и изучать его надо не по официальным первоисточникам, а на практике, обращаясь к последним разработкам ведущих фирм и специалистов.

Программы для просмотра электронных документов

Специальные программы для просмотра электронных документов, созданных по правилам языка разметки HTML, называются браузерами. Основная функция браузера заключается в интерпретации кода HTML и выводе визуального результата на экран монитора пользователя. Сегодня существует большое количество самых разнообразных браузеров, однако наибольшей популярностью пользуются лишь три программы: Internet Explorer, Netscape Navigator и Opera. Рассмотрим немного подробнее каждый из этих браузеров.

Internet Explorer

В августе 1995 года компания Microsoft выпустила очередное обновление своей операционной системы Windows 95, в состав которой впервые был включен интернет-браузер Internet Explorer 1.0. Однако первоначальный программный код браузера принадлежал не Microsoft, а компании Spyglass, которая впоследствии продала лицензию на изменение и коммерческое распространение кода разработчикам Microsoft. Последние поставили Internet Explorer на более высокий уровень развития, что позволило браузеру-дебютанту составить достойную конкуренцию Netscape Navigator — браузеру, которым к 1995 году пользовались свыше 50% посетителей Интернета.

В конце того же года Microsoft выпускает окончательный и доработанный вариант Internet Explorer 2.0, а весной 1996 года появляется версия 3.0, содержавшая для того времени целый ряд нововведений, таких как поддержка фреймов, подключение программных надстроек (plug-in) и пр.

Выход четвертой версии Internet Explorer (вторая половина 1997 года) положил начало стремительному увеличению доли рынка браузеров корпорации Microsoft (к концу года по самым разным данным она охватывала 60% рынка, в то время как ее основной соперник, компания Netscape, — всего около 30%).

Однако Microsoft не останавливается на достигнутом превосходстве. В 1999 году выходит пятая версия популярного во всем мире браузера, в котором были реализованы. новые и усовершенствованы старые технологии. Примерно через год появляется версия 5.5, в октябре 2001 года свет увидела версия 6.0.

Можно назвать несколько основных возможностей браузера, благодаря которым Internet Explorer получил такую популярность:

- быстрый запуск программы;
- поддержка многих технологий, не реализованных или реализованных недостаточно в других браузерах (ActiveX, CSS1/CSS2, "плавающие фреймы" и др.);
- демократичность интерпретации HTML-кода. При загрузке документа, код которого содержит незнакомые конструкции и/или ошибки, Internet Explorer в большинстве

случаев просто не выводит часть, вызывающую затруднения, на экран, в то время как, например, браузер Netscape может отобразить структуру такого документа нарушенной или вообще ничего не вывести на экран монитора;

- полная интеграция с другими приложениями Microsoft, работающими под управлением ОС Windows;
- возможность масштабирования графических изображений, открытых в отдельном окне.

Среди недостатков можно выделить следующие:

- нестабильность работы;
- средняя скорость загрузки информации.

Netscape

Компания Netscape Communications Corporation практически с самого начала считалась основным конкурентом корпорации Microsoft в продвижении своего браузера.

Современный браузер Netscape берет свое начало в марте 1993 года, когда один из основателей будущей компании – Марк Андриссен (Маге Andreessen) – анонсирует выход программы Mosaic (прототипа будущего браузера Netscape). На следующий год Марк Андриссен и его коллега Джим Кларк (Jim Clark) основывают компанию Mosaic Communications (будущая Netscape Communications Corporation), а буквально через несколько месяцев на свет появляется первая версия интернет-браузера Netscape 0.9.

Компания расширяется, Netscape получает распространение, и в итоге к лету 1995 года большинство посетителей Всемирной паутины (около 80%) используют для путешествия по Интернету именно эту программу. Однако вскоре появляется Internet Explorer, который начинает всерьез конкурировать с Netscape, и основатели компании выпускают новую версию своего браузера (2.0), снабдив его не только новым именем Netscape Navigator, но и некоторыми техническими возможностями, тем самым, начав знаменитую "войну браузеров".

В конце 1998 года крупнейший интернет-провайдер Соединенных Штатов Америки America-On-Line (AOL) покупает компанию Netscape и все права на развитие одноименного браузера. В 2000 году выходит Netscape 6.0, параллельно с выходом которого анонсируется проект Mozilla O.6. Оба приложения на тот момент использовали единое программное ядро Gecko, но Netscape как торговая марка принадлежал AOL, а Mozilla разрабатывался в качестве независимого проекта.

В августе 2002 года выходит версия Netscape 7.0, а следом за ней появляется Mozilla 1.0.

Основными преимуществами браузеров семейства Netscape являются:

- сравнительно небольшой размер программы;
- предоставление пользователю расширенного управления содержанием электронных документов;
- улучшенная организация внутренних данных;
- поддержка технологии смены skin'oe (изменение внешнего вида программы в соответствии с выбранной оформительской схемой).

Недостатков тоже хватает, учитывая трудный путь развития, который прошли браузеры Netscape:

- отсутствие поддержки некоторых интерактивных технологий, рекомендованных Консорциумом W3C;
- низкая скорость работы;
- слишком долгое время запуска программы.

Opera

Компания Opera Software (г. Осло, Норвегия) разработала одноименный браузер в 1994 году для норвежской телекоммуникационной компании Telenor. Группа разработчиков, включавшая в себя двух основателей Opera Software, Иона Штефенсона фон Тежнера (Jon Stephenson fon Tetzchner) и Гера Иварсоя (Geir Ivarsoy), поставила перед собой задачу создать интернет- и мультимедиа-приложение, которым могли бы пользоваться все желающие, независимо от системных возможностей своих компьютеров.

В первоначальную концепцию браузера были заложены такие критерии, как скорость запуска программы и загрузки информации, небольшой размер приложения, минимальные требования к ресурсам компьютера пользователя.

Программа, изначально задуманная как небольшой по размеру быстрый браузер для компьютеров с незначительными ресурсами, какое-то время использовалась в пределах внутренней информационной сети компании Telenor, а к концу 1995 года авторы Opera покинули стены компании, чтобы продолжить самостоятельное развитие своего детища. Наконец, во второй половине 1996 года браузер Opera 2.1 стал доступен для загрузки в Интернете в качестве 90-дневной условно-бесплатной (Shareware) версии.

К числу основных отличий Opera от других браузеров, которые с полной уверенностью можно считать преимуществами, отнесем следующие:

- небольшой размер;
- минимальные системные требования;
- быструю скорость загрузки HTML-документов;
- расширенные настройки;
- высокую масштабируемость просматриваемого документа.
- Однако и у Opera есть ряд недостатков, которые также следует упомянуть:
- статус коммерческого программного продукта (регистрация стоит 39 долларов для полной версии; Freeware-вариант будет постоянно "радовать" вас показом чужих рекламных баннеров);
- отсутствие поддержки некоторых русских кодировок в английских версиях программы;
- недостаточно высокий уровень надежности выполнения скриптов на стороне пользователя (JavaScript/VBScript).

Существует 7.0 версия браузера Opera.

Как видите, каждый из описанных интернет-браузеров прошел долгую историю развития, имеет свои плюсы и минусы, получает одобрение и порицание, обладает сторонниками и противниками своих функциональных возможностей. В конечном итоге, выбор браузера, с которым вы будете работать в процессе освоения языка гипертекстовой разметки HTML, остается за вами.

Приведем статистику использования браузеров. Согласно подсчетам известной исследовательской группы OneStat, в 2002 году около 94,6% пользователей во всем мире выходило в Интернет с помощью Internet Explorer, в то время как на долю Netscape пришлось всего 3,3%. Остальные браузеры отстают безнадежно – пользователей Mozilla 1.0 и Opera 6.0 насчитывается всего около 0,8%.

Статистика использования браузеров в российской части Интернета почти повторяет мировые показатели: Internet Explorer – 91,5%, Netscape – 3,2%, а прочие браузеры – 5,2.

Лабораторная работа № 1.
Основные понятия языка HTML. Структура документа HTML

Цель работы: знакомство студентов с основными понятиями языка HTML, структурой HTML-документа, обязательными метками, комментариями, способами форматирования текста, физическими и логическими стилями, приобретение навыков создания простейших статических Web-документов.

Основные понятия

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.

Отдельный документ, выполненный в формате HTML, называется:

- HTML-документом;
- Web-документом;
- Web-страницей.

Такие страницы, как правило, имеют расширение **HTM или HTML.**

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

Группа Web-страниц, принадлежащих одному автору или одному издателю и взаимосвязанных общими гиперссылками, образует структуру, которая называется *Web-узлом,* или *Web-сайтом.*

Каждая HTML-страница имеет свой уникальный *URL-адрес* в Интернете.

Фрейм (Frame) - этот термин имеет два значения. Первое – область документа со своими полосами прокрутки. Второе значение – одиночное изображение в анимационном графическом файле (кадр).

Апплет (Applet) - программа, передаваемая на компьютер клиента в виде отдельного файла и запускаемая при просмотре Web-страницы.

Скрипт, или сценарий (Script), - программа, включенная в состав Web-страницы для расширения ее возможностей. Браузер Internet Explorer в определенных ситуациях выводит сообщение: "Разрешить выполнение сценариев на странице?" В этом случае имеются в виду скрипты.

CGI (Common Gateway Interface) - общее название программ, которые, работая на сервере, позволяют расширять возможности Web-страниц. *Например*, без таких программ невозможно создание интерактивных Web-страниц.

Браузер (Browser) - программа для просмотра Web-страниц.

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

Пример

<Начало элемента> Содержание элемента, данные, которые форматирует элемент </**Конец элемента>**

*Тэг (*по-английски — tag-метка, дескриптор, ярлык) - начальный или конечный маркеры элемента. Тэги определяют границы действия элементов и отделяют элементы друг от друга. В тексте Web-страницы тэги заключаются в угловые скобки < >, а конечный тэг всегда снабжается косой чертой. Текст, не находящийся между такими скобками < > – весь виден, при просмотре в браузере.

Пример

<Начальный тэг> Содержание элемента, данные, которые форматирует элемент **</Конечный тэг>**

Пример

```
<P> Этот текст будет расположен в отдельном абзаце </P>
```

Элемент, содержащий некоторый текст, ограничен начальным тэгом (маркером) и конечным тэгом (маркером) , т. е. текст помещен между тэгами, как в контейнер, а тэги и размечают начало и конец абзаца соответственно.

Любая Web-страница представляет собой набор элементов. Один из основных принципов HTML возможность вложения элементов, они могут вкладываться один в другой.

Атрибут – параметр или свойство элемента. Атрибуты располагаются внутри начального тэга и отделяются друг от друга пробелами. Если элемент содержит текст, то атрибуты могут задавать цвет и размер шрифта, выравнивание текстового абзаца и т. п. Если элемент содержит рисунок, то атрибуты могут задавать размер рисунка, наличие и размер рамки вокруг рисунка и пр.

Пример

```
<P align="center"> Этот текст будет выровнен по центру экрана </p>
```

В этом примере опять встречается тэг, определяющий начало и конец абзаца. Однако в начальном тэге находится атрибут align, который задает выравнивание текста по центру экрана.

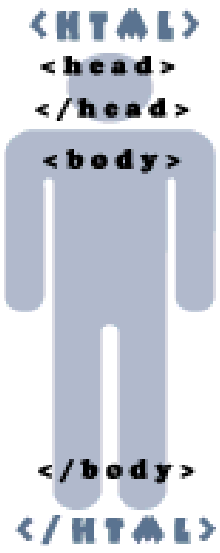
Примечание:

- любая полезная информация должна находится между начальным и конечным тэгами, указывающими ее формат;
- все атрибуты располагаются в начальном тэге;
- для удобства работы начальный тэг вы можете писать с прописной (заглавной) буквы (P), а конечный - со строчной (маленькой) буквы (/p) хотя это и не обязательно;
- не для всех элементов требуется ставить конечный (закрывающий) тэг.
- написание каждого нового элемента начинайте с новой строки. Вложенные элементы выделяйте отступом (табуляцией). Это опять - таки не обязательно, но значительно облегчит вашу работу.

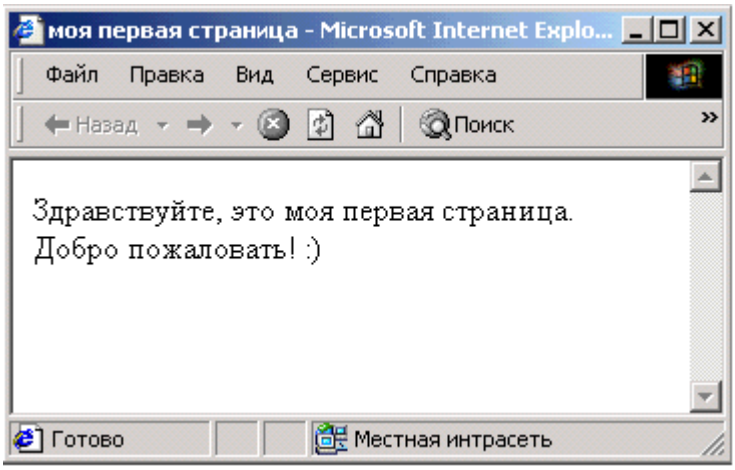
Структура документа HTML

Для того чтобы понять структуру Web-страницы, нам необходимо подробно рассмотреть все элементы, входящие в приведенный листинг.

Пример.




```
<COMMENT> КОММЕНТАРИЙ К ДОКУМЕНТУ </COMMENT>
<HTML>
<HEAD>
<TITLE> моя первая страница </title>
<META NAME="AUTHOR" CONTENT=" IVANOV IVAN ">
<META NAME="KEYWORDS" CONTENT="ЖИВОТНЫЕ, ПРИРОДА,
ФАУНА">
</head>
<BODY>
Здравствуйте, это моя первая страница.
<!--<PRE> КОММЕНТАРИЙ </PRE>-->
<BR>
Добро пожаловать! :)
</body>
</html>
```



<COMMENT> </comment>

Текст комментария. В любом языке программирования есть конструкции, позволяющие создавать произвольные ремарки. HTML в этом смысле — не исключение. Текст, помещенный внутри COMMENT, игнорируется браузером. COMMENT может располагаться в любом месте кода Web-страницы. Комментарий должен быть отделен от основного текста.

Существует, правда, одно ограничение: внутри комментария не должны располагаться другие элементы. Так должно быть, разумеется, только в том случае, когда необходимо, чтобы все содержимое элемента COMMENT не отображалось на экране монитора. Если в комментарии будет присутствовать другой элемент, то его содержимое будет выведено на экран, отформатированное соответствующим образом.

Существует еще один способ обозначения комментария. Он заключается в использовании восклицательного знака и обрамлении текста комментария двойными тире. Например:

<!-- *Строка комментария* -->

< ! -- *Комментарий* -- *Не комментарий* -- *Снова комментарий* -->

<HTML> </html>

Отличительный признак HTML-документа. Одним из принципов языка является многоуровневое вложение элементов. HTML является самым внешним, так как между его стартовым и конечным тегами должна находиться вся Web-страница. В принципе, этот элемент можно рассматривать как формальность. Он имеет атрибуты version, lang и dir, которыми в данном случае мало кто пользуется, и допускает вложение элементов HEAD, BODY и PLAINTEXT, определяющих общую структуру Web-страницы. Естественно, что конечным тегом </html> заканчиваются все гипертекстовые документы.

<HEAD> </head>

Информация о документе, которая не выводится на экран, называется заголовком. Так же как и HTML, HEAD служит только для формирования общей структуры документа. Этот элемент может иметь атрибуты lang и dir и допускает вложение элементов TITLE, ISINDEX, BASE, META, LINK, NEXTID.

<TITLE> </title>

Элемент для размещения заголовка Web-страницы. Строка текста, расположенная внутри, отображается не в документе, а в заголовке окна браузера. Эта особенность часто используется для организации поиска в WWW. Поэтому авторы, создающие Web-страницы, должны позаботиться о том, чтобы строка внутри TITLE, не будучи слишком длинной (не более 64 символов), достаточно точно отражала назначение документа.

<META>

Этот элемент содержит служебную информацию, которая не отображается при просмотре Web-страницы. Внутри него нет текста в обычном понимании, поэтому нет и конечного тега. Каждый элемент META содержит два основных атрибута, первый из которых определяет тип данных, а второй — содержание. Далее приведены несколько примеров meta-данных.

Дата, обозначающая «срок годности» документа:	name ="Expires" content = "*Дата*"
Адрес электронной почты:	name ="Reply-to" content ="*Имя@Адрес*"
Имя автора Web-страницы:	name ="Author" content =*" Имя автора"*
Набор ключевых слов для поиска:	name ="Keywords" content ="*Слово1, слово2, ...*"
Краткое описание содержания Web-страницы:	name ="Description" content ="*Содержание страницы"*
Описание типа и характеристик Web-страницы:	name ="Content-Type" content ="*Описание страницы"*
Указание приложения, в котором была создана Web-страница:	name ="Generator" content =*"Название HTML-редактора"*

Атрибут *name* используется приложением-клиентом для получения дополнительной информации о Web-страницах и их упорядочения. Этот атрибут часто заменяют атрибутом http-equiv. Он используется сервером для создания дополнительных полей при выполнении запроса.

Кроме этого, элемент META может содержать URL. Шаблон соответствующего атрибута таков:

URL = "http: *// адрес"*

<BODY> </body>

Этот элемент заключает в себе гипертекст, который определяет собственно Web-страницу. Эта та часть документа, которую разрабатывает автор страницы и которая отображается браузером. Соответственно, конечный тег этого элемента надо искать в конце HTML-файла. Внутри BODY можно использовать все элементы, предназначенные для дизайна Web-страницы. *Внутри стартового тега* элемента BODY можно расположить ряд атрибутов, обеспечивающих установки для всей страницы в целом.

Это могут быть атрибуты, задающие:

- цвет фона Web-страницы;
- "обои" или рисунок фона страницы;
- цвет текста на всей странице;
- цвет гиперссылок (активных, выбранных, посещенных).

Параметр	**Функция**
bgcolor = "#RRGGBB"	Определение цвета фона. Цвет фона задается тремя двухразрядными шестнадцатеричными числами, которые определяют интенсивность красного, зеленого и синего цветов соответственно. Более подробно об определении цветов будет рассказано ниже.
background ="Путь к файлу фона"	Указание фонового рисунка. Оба вышеприведенных атрибута не являются альтернативными и часто используются совместно.
text = "#RRGGBB"	Определение цвета основного текста.
link = "#RRGGBB"	Определение цвета текста гиперссылок.
vlink = "#RRGGBB"	Определение цвета для использованных гиперссылок.
alink = "#RRGGBB"	Определение цвета для последней выбранной пользователем гиперссылки.

Параметр	Функция
bgproperties	Изменение свойств фона (например, фиксирование фонового рисунка)
topmargin=”10”, bottommargin=”10”, leftmargin, rightmargin	Определение размера отступа от верхнего, нижнего, левого и правого краев документа. Значение задается в пикселях
marginwidth=”10”, marginheight=”10”	Netscape объединяет упомянутые параметры в две группы: горизонтальные и вертикальные отступы. Для учета особенностей всех браузеров, надо поставить и те и другие параметры

Пример.

```
<body topmargin=”5”, bottommargin=”5”, leftmargin=”10”, rightmargin=”10”, marginwidth=”10”, marginheight=”5”
```

Задания к лабораторной работе № 1

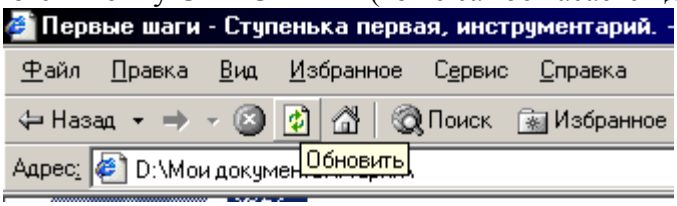
Задание 1. Создание простейшей Web-страницы.

Цель: научиться создавать Web-страницы в текстовом редакторе Блокнот.

Указания к выполнению

1. Создайте на диске отдельную директорию (папку) для будущей страницы.
   `D:\первые шаги\`
2. Откроите текстовый редактор Блокнот (notepad) и наберите в нем структуру HTML-документа, которая приведена в примере 1:
3. Сохраните этот документ, присвоив ему имя *.html
   Например: `D:\первые шаги\index.html`
4. Откройте с помощью Internet Explorer ваш документ (не закрывайте блокнот, он нам еще пригодится).

   Если вы что-то изменили в вашем *.html документе (в Блокноте), то, не забудьте сохранить изменения, а чтобы посмотреть как это выглядит в Internet Explorer, надо нажать в Internet Explorer кнопку **ОБНОВИТЬ** (тоже самое касается других браузеров).



   Если изменений не видно, то это значит, что вы где-то что-то неправильно написали, или забыли сохранить документ.

5. Вернитесь к сохраненному в Блокноте файлу.
6. Внесите в него следующие изменения: пусть это будет ваша первая страничка. Укажите в ней ваши фамилию, имя, ФИО родителей, братьев и сестер, свои увлечения. Используйте для этого форматирование абзацев.
7. В строке <TITLE> укажите: "Домашняя страничка (ваше имя и фамилия)".
8. Сохраните файл как page2.htm.
9. Просмотрите результат в браузере, при необходимости отредактируйте файл при помощи Блокнота.

Контрольные вопросы

1. Охарактеризуйте язык тегов HTML.

2. Какова структура HTML-документов.
3. Перечислите обязательные метки и охарактеризуйте их.
4. Дайте определения следующих понятий: гипертекст, гиперссылка, Web-сайт, URL-адрес в Интернете, фрейм апплет скрипт, браузер элемент, тэг, атрибут.

Лабораторная работа № 2.
Форматирование текста.

Цель: познакомиться с основами HTML, обязательными метками, &-последовательностями, комментариями, способами форматирования текста (шрифта и абзаца), физическими и логическими стилями, горизонтальными линиями.

Текст – единственный объект Web-страницы, который не требует специального определения. Иными словами, произвольные символы интерпретируются по умолчанию как текстовые данные.

Для форматирования текста существует большое количество элементов. Форматировать текст можно с помощью традиционных элементов: выделять фрагменты курсивом, полужирным шрифтом, выбирать шрифт, размер и цвет шрифта, выравнивать текстовые фрагменты. Если автору не хватает простых вариантов форматирования текста, он может прибегнуть к таблицам стилей, которые существенно расширят возможности языка HTML по форматированию.

Рассмотрим стандартные элементы языка HTML, позволяющие форматировать текст.

Форматирование абзаца

<P> … </p>

Элемент абзаца разделяет фрагменты текста вертикальным отступом. Он позволяет использовать только начальный тэг, так как следующий элемент </p> обозначает не только начало следующего абзаца, но и конец предыдущего. В тех случаях, когда по смыслу необходимо обозначить завершение абзаца, можно использовать и конечный тэг.

Вместе с элементом абзаца можно использовать и атрибут выравнивания align!

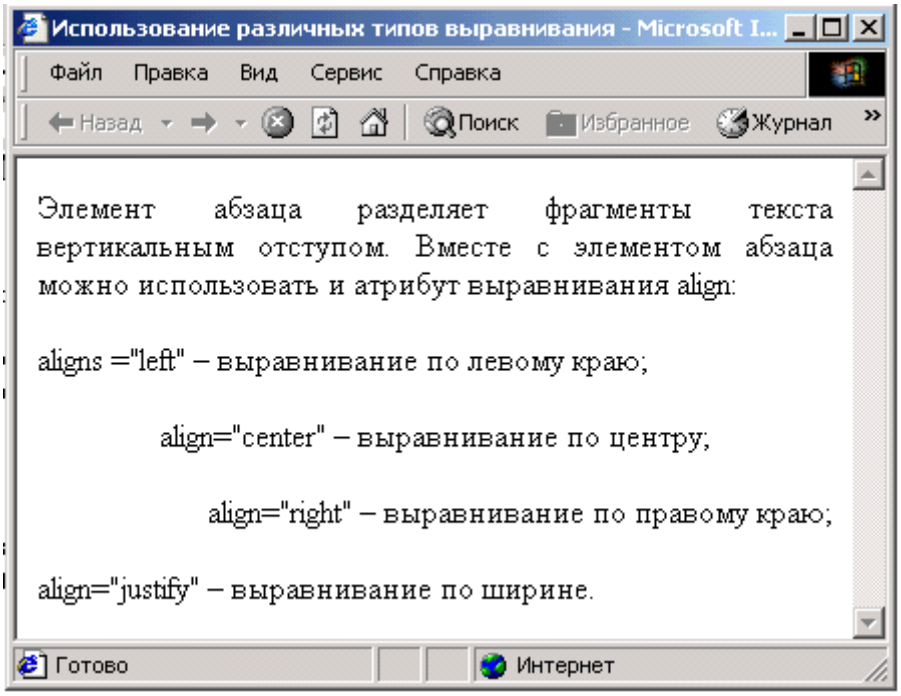
- aligns ="left" – выравнивание по левому краю;
- align="right" – выравнивание по правому краю;
- align="center" – выравнивание по центру;
- align="justify" – выравнивание по ширине.

Выравнивание по ширине – опасный параграф. Он не работает в старых версиях браузеров (т.е. текст выровняется в таком случае по левому краю документа).

Пример

```
<HTML>
<HEAD>
<TITLE> Использование различных типов выравнивания </title>
</head>
<BODY>
<P  align="justify"> Элемент абзаца  разделяет фрагменты текста вертикальным отступом. Вместе с элементом абзаца можно использовать и атрибут выравнивания align: </p>
<P align="left"> aligns ="left"-выравнивание по левому краю;  </p>
<P align="center">  align="center" - выравнивание по центру; </p>
<P align="right"> align="right"-выравнивание по правому краю; </p>
<P align="justify"> align="justify" - выравнивание по ширине.</p>
</body>
```

```
</html>
```

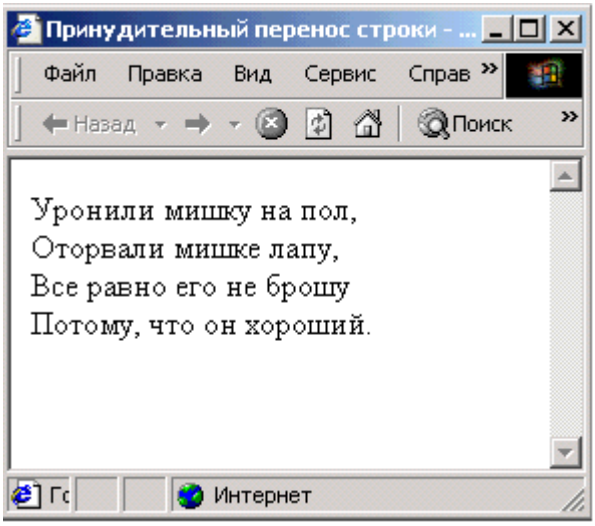


**
**

Элемент, обеспечивающий принудительный переход на новую строку. Он имеет только начальный тэг. В месте его размещения строка заканчивается, а оставшийся текст печатается с новой строки.

Пример

```
<HTML>
<HEAD>
<TITLE> Принудительный перенос строки </title>
</head>
<BODY>
Уронили мишку на пол,
<BR> Оторвали мишке лапу,
<BR> Все равно его не брошу
<BR> Потому, что он хороший.
</body>
</html>
```



Если таким образом расставить элемент
 в этом стихотворении, то в экране браузера мы увидим стандартным образом написанное четверостишие на четырех строках.

<NOBR> </nobr>

Этот элемент по своему действию является прямой противоположностью предыдущему. Текст, заключенный между его тегами, будет выведен в одну строку. Длинная строка не уместится на экране, и для ее просмотра придется использовать горизонтальную полосу прокрутки.

<PRE> </pre>

Элемент для обозначения текста, отформатированного заранее (preformatted). Подразумевается, что текст будет выведен в том виде, в каком он был подготовлен пользователем. Текст, заключенный между метками <PRE> и </pre> (от английского preformatted — предварительно форматированный), выводится браузером на экран как есть — со всеми пробелами, символами табуляции и конца строки. Во всех других случаях браузер игнорирует эти символы. Возможен и обратный эффект: если пользователь введет текст как одну длинную строку, то она не будет разорвана браузером, а уйдет за границы окна программы. В этом смысле элемент PRE работает так же, как элемент NOBR. По умолчанию для отформатированного заранее текста выбирается моноширинный шрифт. Этот элемент удобно использовать для демонстрации листингов программ или для вывода текстовых документов, переформатирование которых может привести к искажению их смысла.

<BLOCKQUOTE> ... </blockquote>

Текст, заключенный между метками <BLOCKQUOTE> и </blockquote>, выводится браузером на экран с увеличенным левым полем (с отступом).

**<CENTER> **

Элемент для центрирования текста, а точнее, любого содержимого. Не является общеупотребительным. В тех случаях, когда это возможно, вместо него используют атрибут align =" center".

**<DIV> **

Элемент, похожий на предыдущий. Он позволяет выравнивать содержимое по левому краю, по центру или по правому краю. Для этого стартовый тег должен содержать соответствующий атрибут:

align ="left"

align ="center"

align ="right"

Форматирование шрифта

HTML допускает два подхода к шрифтовому выделению фрагментов текста – логическое и физическое форматирование.

В группу тегов логического форматирования входят теги, отображающие на экране монитора элементы документа, таким образом, как установлено по умолчанию в спецификации языка разметки HTML. Переопределить их параметры или свойства нельзя, за исключением ситуаций использования стилевых шаблонов CSS и обособления тегами физического форматирования. Результат действия разных тегов логического форматирования визуально может совпадать, ибо основное их предназначение заключается в логическом выделении отдельных элементов HTML.

Теги физического форматирования позволяют разработчику визуально изменять вид текста, делая его **жирным** или *наклонным*, варьируя его параметры и значения.

Физические стили

Под физическом стилем принято понимать прямое указание браузеру на модификацию текущего шрифта.

**<B> **

Выделение текста **полужирным шрифтом** (от слова **Bold**),. Очень популярный элемент. Использование полужирного шрифта – прием, позаимствованный из текстовых редакторов.

Пример

```
Этот текст имеет обычное начертание - <B>, а этот будет выделен полужирным шрифтом </Ь>.
```

<I> </i>

Выделение текста *курсивом* (от слова *Italic*).

Пример

```
Этот текст имеет обычное начертание <I>, а этот выделен курсивом </i>.
```

<TT> </tt>

Элемент, обозначающий текст телетайпа (teletype). Его особенность заключается в использовании моноширинного шрифта, имитирующего пишущую машинку, то есть имеющего фиксированную ширину символа.

<STRIKE> </strike>

Элемент, создающий перечеркнутое начертание текста. В настоящее время его заменяют тегом: **<DEL > </del>.**

<U> </u>

Подчеркнутое начертание текста.

****

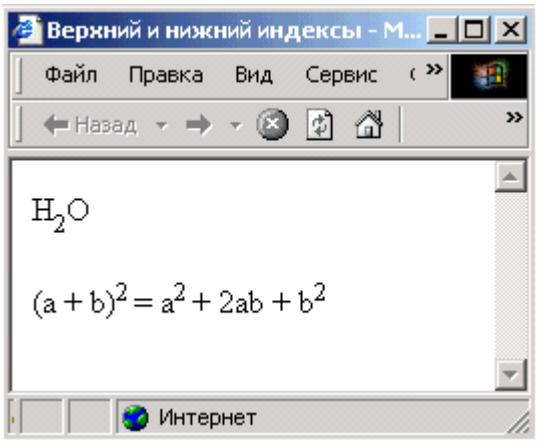
Элемент, создающий эффект нижнего индекса (subscript).

****

Элемент, создающий эффект верхнего индекса (superscript).

Пример.

```
<HTML>
<HEAD>
<TITLE> Верхний и нижний индексы </title>
</head>
<BODY>
H<SUB>2</sub>O
<BR><BR>
(a + b)<SUP>2 </sup> = a<SUP>2 </sup> + 2ab + b<SUP>2 </sup>
</body>
</html>
```



Размер шрифта

<BASEFONT>

Элемент, определяющий базовый (основной) размер шрифта. Внутри элемента необходимо указать атрибут:

```
size = Базовый размер шрифта
```

Величина атрибута может лежать в пределах от 1 до 7. По умолчанию используется величина 3. Установка, выполняемая этим элементом, имеет значение для элемента FONT (см. ниже), который позволяет задавать относительный размер шрифта.

**<FONT> **

Определение типа, размера и цвета шрифта. Все эти характеристики определяются при помощи соответствующих атрибутов.

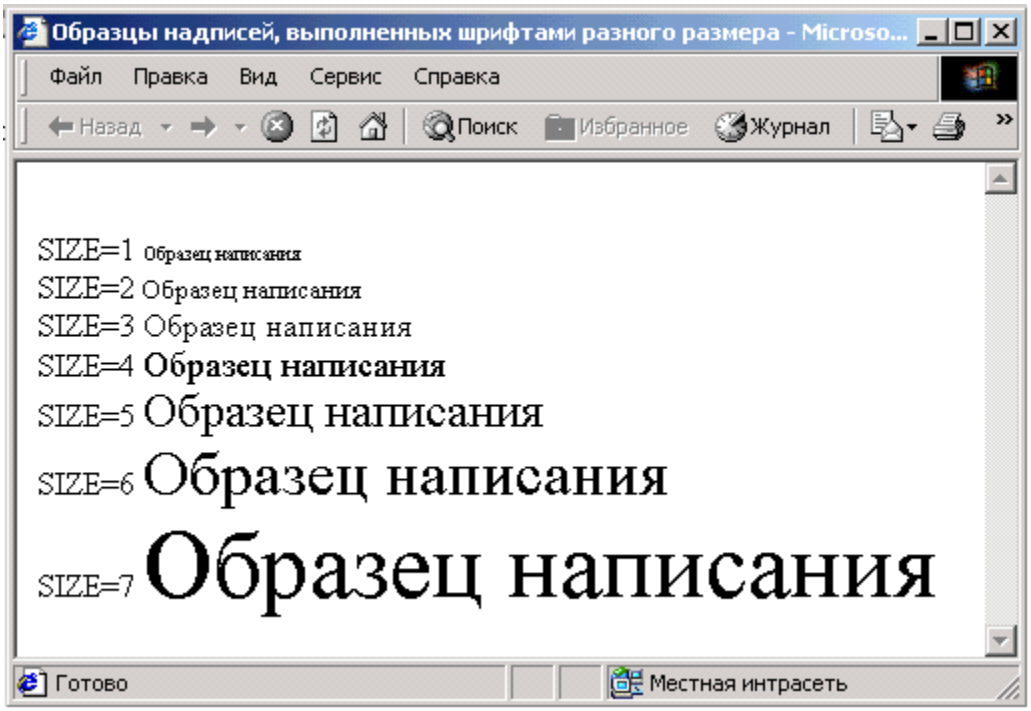
Абсолютный размер шрифта задается при помощи атрибута size (размер):

```
size=Абсолютный размер шрифта
```

Этот атрибут может принимать значения от 1 до 7. В таблице показано несколько образцов надписей, выполненных шрифтами разного размера.

Пример.

```
<HTML>
<HEAD>
<TITLE> Образцы надписей, выполненных шрифтами разного размера
</title>
</head>
<BODY>
<BR> SIZE=1 <FONT size=1>  Образец написания  </font>
<BR> SIZE=2 <FONT size=2>  Образец написания  </font>
<BR> SIZE=3 <FONT size=3>  Образец написания  </font>
<BR> SIZE=4 <FONT size=4>  Образец написания  </font>
<BR> SIZE=5 <FONT size=5>  Образец написания  </font>
<BR> SIZE=6 <FONT size=6>  Образец написания  </font>
<BR> SIZE=7 <FONT size=7>  Образец написания  </font>
</body>
</html>
```



Пример

```
<FONT size=3> Тише едешь - дальше будешь </font>
```

Размер шрифта может задаваться относительного базового:

size=+Число

size=-Чuсло

При назначении величины для size необходимо учитывать величину базового размера. Обе они в сумме должны соответствовать одному из абсолютных размеров. Так, для базового размера, равного 3, относительный размер может находиться в пределах от – 2 до + 4. Если величина выходит за допустимый предел, то используется или шрифт размера 7, или шрифт размера 1. На рис. 2. показаны надписи, выполненные шрифтами с заданным относительным размером.

Шрифт +4 Шрифт +3

Шрифт+2 Шрифт+1 Шрифт 0 Шрифт –1 Шрифт-2

При помощи относительных величин тоже можно получить семь градаций размера шрифта

Для элемента FONT можно использовать атрибут цвета:

```
color="Цвет"
```

Название цвета задается либо числом, написанным в шестнадцатеричной системе, например, color="#ffoooo", либо просто его названием, написанным на английском языке, например, color="red".

Пример

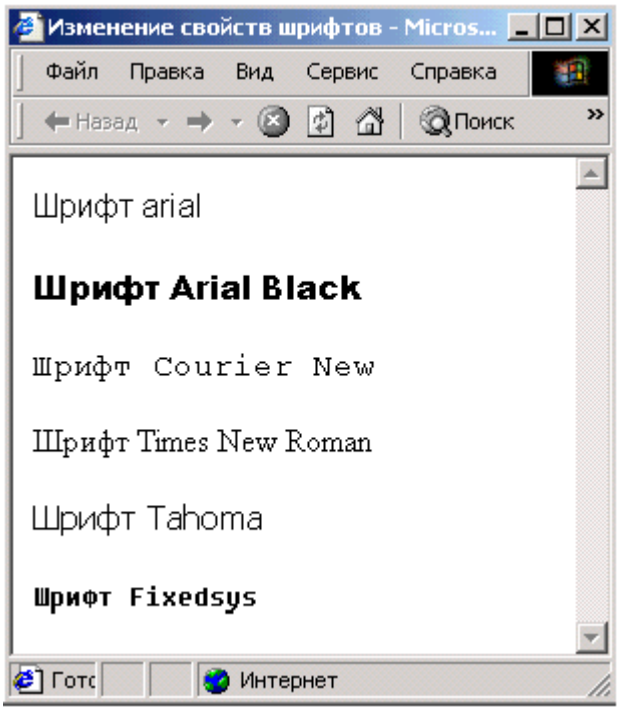
```
<FONT color="blue"> Это шрифт синего цвета </font>
```

Атрибут face (вид) позволяет задавать тип шрифта:

face="Название шрифта"

Пример

```
<HTML>
<HEAD>
<TITLE> Изменение свойств шрифтов </title>
</head>
<BODY>
<FONT face="arial">Шрифт arial</font>
<BR><BR>
<FONT face="Arial Black"> Шрифт Arial Black</font>
<BR><BR>
<FONT face="Courier New"> Шрифт Courier New </font>
<BR><BR>
<FONT face="Times New Roman"> Шрифт Times New Roman</font>
<BR><BR>
<FONT face="Tahoma"> Шрифт Tahoma</font>
<BR><BR>
<FONT face="Fixedsys"> Шрифт Fixedsys</font>
</body>
</html>
```



Правда, есть одна проблема. Web-страницы просматривает множество людей, и нет гарантии, что у каждого из них окажется нужный шрифт. Если в системе не установлен шрифт точно с таким же названием, то браузер использует свой стандартный. Он имеет два назначенных по умолчанию шрифта: один пропорциональный, другой моноширинный.

Все эти атрибуты могут быть использованы совместно внутри тэга <font>.

Пример

```
<FONT face="Arial" size=3 color="blue" > Это шрифт arial размером 3, цвет синий </font>
```

Элемент FONT может с успехом заменять элементы заголовка H1...H6. Для последних, например, не предусмотрена возможность указания цвета букв.

Чтобы заголовок, созданный на основе элемента FONT, хорошо смотрелся, этот элемент необходимо комбинировать с другими: CENTER, B, I, P и т. д.

<BIG> </big>

Этот тег используется, если необходимо выделить часть текста небольшим увеличением размера шрифта относительно остальных слов.

<SMALL> </small>

Этот тег используется, если необходимо выделить часть текста небольшим уменьшением размера шрифта относительно остальных слов.

Логические стили

При использовании логических стилей автор документа не может знать заранее, что увидит на экране читатель. Разные браузера толкуют одни и те же метки логических стилей по-разному. Некоторые браузеры игнорируют некоторые метки вообще и показывают нормальный текст вместо выделенного логическим стилем. Вот самые распространенные логические стили.

<H1> </hl>

Элемент заголовка. Существует шесть уровней заголовков, которые обозначаются H1. . .H6. Заголовок уровня 1 самый крупный, а уровень 6 обеспечивает самый маленький заголовок. Представление об относительных размерах букв в них дает рис.1. Для заголовков можно использовать атрибут, задающий выравнивание влево, по центру или вправо:

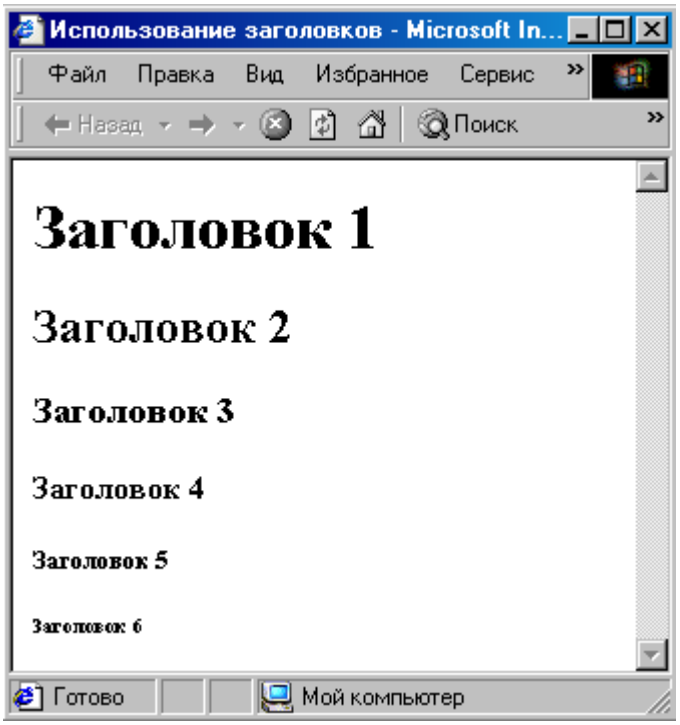
align = "left"

align = "center"

align = "right"

Пример

```
<HTML>
<HEAD>
<TITLE>Использование заголовков</title>
</head>
<BODY>
 <H1>Заголовок 1</H1>
 <H2>Заголовок 2</H2>
 <H3>Заголовок 3</H3>
 <H4>Заголовок 4</H4>
 <H5>Заголовок 5</H5>
 <H6>Заголовок 6</H6>
 </body>
</html>
```

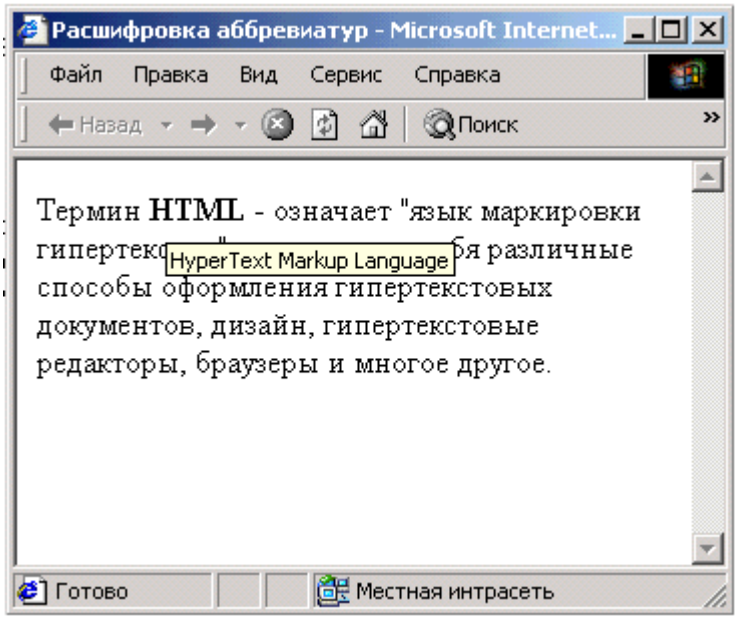


<ACRONYM> </acronym >

Данный элемент реализуется через параметр TITLE и отображается в браузере при наведении курсора на слово-аббревиатуру

Пример.

```
<HTML>
<HEAD>
<TITLE> Расшифровка аббревиатур  </title>
</head>
<BODY>
Термин
<B><ACRONYM TITLE="HyperText Markup Language"> HTML </acronym >
</b>- означает "язык маркировки гипертекстов" и включает в себя
различные способы оформления гипертекстовых документов, дизайн,
гипертекстовые редакторы, браузеры и многое другое.
</body>
</html>
```



<EM> и <DFN> </dfn>

Элементы, означающие выразительность (от английского emphasis – акцент) фрагмента текста и определение чего-либо (definition). Оба элемента аналогичны по своему действию элементу <I>, то есть, в большинстве случаев, позволяют выделить текст курсивом.

Казалось бы, два последних элемента — лишние. С точки зрения дизайна документа это так. Они могут пригодиться только для того, чтобы *единообразно* выделить одинаковые по назначению (или смыслу) фрагменты текста, находящиеся в разных частях документа или даже на разных страницах. Разработчик в этом случае не может точно знать, какой именно шрифт будет использован: это определяется каждым браузером по-своему. Но он может быть уверен, что все фрагменты текста будут отформатированы одинаково. В языке можно найти еще несколько элементов, которым можно дать такую же характеристику.

<CITE> </cite>

Предполагается, что этот элемент может быть использован для форматирования цитат, высказываний и ссылок в обычном понимании этого слова. Текст, расположенный внутри него, выводится по умолчанию курсивом. Визуально аналогичен тегам <EM> и <I>

<ADDRESS> </address>

Подобно элементу CITE, данный элемент отличается только предусмотренным содержанием. Он также обеспечивает форматирование курсивом.

<STRONG> ... – от английского strong emphasis — сильный акцент.

Элемент, отвечающий за выделение текста. Обычно его применение равносильно использованию элемента для выделения полужирным: **<B>.**

<KBD> ... </kbd> – от английского keyboard — клавиатура.

Этот элемент предназначен для указания текста, который пользователь должен ввести с клавиатуры (keyboard). Можно рассчитывать, что текст, выделенный с помощью этого элемента, будет выведен моноширинным шрифтом в полужирном начертании.

<CODE> </code>, <SAMP> </samp> и <VAR> </var>

Элементы, предназначенные для вывода фрагментов программ.

<CODE> ... </code>

Этот элемент предназначен для визуального выделения небольших фрагментов

программного кода. Код отображается моноширным шрифтом.

<SAMP> ... </samp> – от английского sample — образец.

Рекомендуется использовать для демонстрации образцов сообщений, выводимых на экран программами. SAMP предполагается задействовать при иллюстрации примеров (sample) вывода данных на экран.

<VAR> ... </var> – от английского variable — переменная.

Этот элемент был создан для выделения переменных (variables). Как правило, все эти элементы обеспечивают вывод информации с использованием моноширинного шрифта.

<PLAINTEXT> </plaintext>

Этот элемент предназначен для создания текста с конструкциями HTML, которые должны восприниматься именно как текст. Все теги, заключенные в PLAINTEXT, воспринимаются браузером только как произвольные символы. Элемент удобно использовать для обсуждения вопросов, связанных с использованием HTML.

Горизонтальные линии

Горизонтальная линия (horizontal rule) или, на типографский манер, горизонтальные линейки – очень часто используемый элемент. Во-первых, потому, что с его помощью очень удобно делить страницу на части, использовать для отделения одного блока текста от другого. Небольшой по размеру текст, сверху и снизу которого располагаются горизонтальные линии, привлекает больше, чем обычно, внимание читателя. Во-вторых, выбор таких элементов оформления у автора страницы очень небольшой. Действительно, в HTML практически отсутствуют подобные конструкции, а вот для горизонтальной линии почему-то было сделано исключение. Правда, несмотря на некоторую скупость языка в этой области, можно придумать немало стандартных графических образов, которые разнообразили бы вид страниц.

Если нужна простая, без затей линия, можно воспользоваться тегом **<HR>.**

Пример.

```
<HR>
Пример создания горизонтальных линий с помощью тега HR
<HR>
```

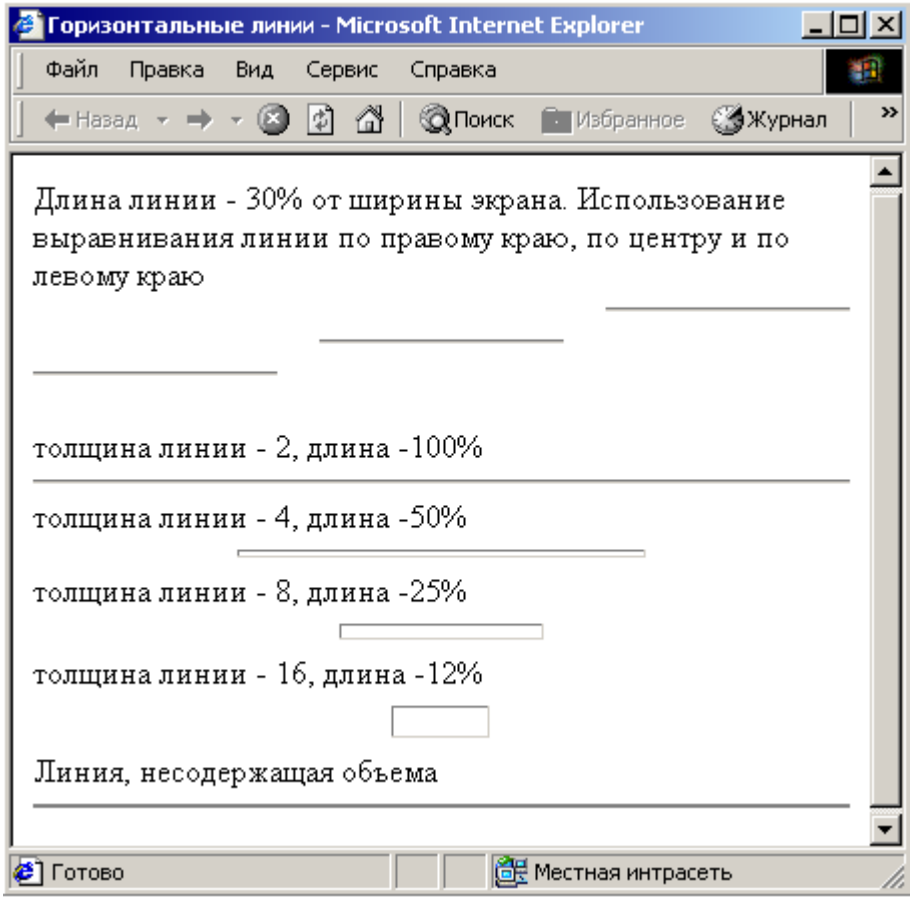
Пример создания горизонтальных линий с помощью тега HR

Элемент не имеет конечного тега, но допускает ряд атрибутов:

1.	`<HR align="right">`	выравнивание влево, по центру, вправо, по ширине: align = "left" align = "center" align = "right" align = "justify"
2.	`<HR width="30%">`	длина линии в процентах и пикселях width = Длина в пикселях width = Длина в процентах % от ширины экрана
3.	`<HR size="6">`	Толщина линии
4.	`<HR NoShade>`	Отмена объемности
5.	`<HR color="red">`	цвет линии, только в IE

Пример.

```
<HTML>
<HEAD>
<TITLE> Горизонтальные линии </title>
</head>
<BODY>
Длина линии - 30% от ширины экрана. Использование выравнивания
линии по правому краю, по центру и по левому краю
<HR width="30%" align="right">
<HR width="30%" align = "center">
<HR width="30%" align = "left">
<BR>
толщина линии - 2, длина -100% <HR SIZE=2 WIDTH=100%>
толщина линии - 4, длина -50% <HR SIZE=4 WIDTH=50%>
толщина линии - 8, длина -25% <HR SIZE=8 WIDTH=25%>
толщина линии - 16, длина -12% <HR SIZE=16 WIDTH=12%>
Линия, несодержащая объема <HR NoShade>
</body>
</html>
```



Escape последовательности (символьные объекты)

Некоторые символы, такие как "<" и ">" воспринимаются браузерами как начало и конец HTML-меток, поэтому должен существовать способ их выражения, как символьных данных внутри самого документа или в URL. В HTML это делается с помощью &-последовательностей (их еще называют символьными объектами или escape (эскейп) –

последовательностями). Браузер показывает на экране символ "<", когда встречает в тексте последовательность `<` (по первым буквам английских слов less than — меньше, чем). Знак ">" кодируется последовательностью `>` (по первым буквам английских слов greater than — больше, чем).

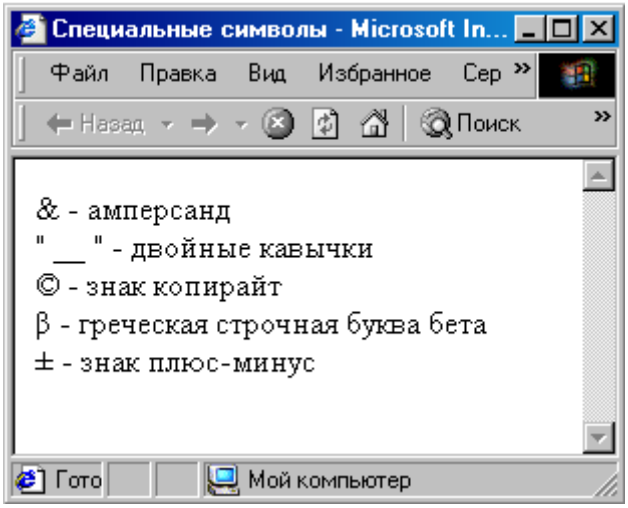
Принято использовать следующие нотации (соответствия):

символ	обычное имя символа	HTML запись символа (escape последовательность)
<	символ "меньше чем", левая угловая скобка	<
>	символ "больше чем", правая угловая скобка	>
&	амперсанд	&
"	двойные кавычки	"
©	знак копирайт	&copy
®	знак зарегистрированной торговой марки	&reg
	непрерывный пробел	

Точка с запятой – обязательный элемент &-последовательности. Кроме того, все буквы, составляющие последовательность, должны быть в нижнем регистре (т.е., маленькие). Использование меток типа " или &AMP не допускается. Более полный набор специальных символов представлен в приложении 1.

Пример.

```
<HTML>
<HEAD>
<TITLE> Специальные символы </title>
</head>
<BODY>
&amp - амперсанд
<BR>
&quot __ &quot - двойные кавычки
<BR>
&copy - знак копирайт
<BR>
&beta; - греческая строчная буква бета
<BR>
&plusmn; - знак плюс-минус
</body>
</html>
```



Вообще говоря, &-последовательности определены для всех символов из второй половины ASCII-таблицы (куда, естественно, входят и русские буквы). Дело в том, что некоторые серверы не поддерживают восьмибитную передачу данных, и поэтому могут передавать символы с ASCII-кодами выше 127 только в виде &-последовательностей.

Задания к лабораторной работе № 2
Форматирование текста на Web-странице по образцу

Задание 1.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе <TITLE> укажите название странички «Задание 1. *Форматирование текста на Web-странице по образцу*».
3. Отформатируйте следующий текст согласно указаниям, которые даны в скобках курсивом. Используйте линий разного вида и размера.

Домашние животные
(Заголовок HI)

Собаки
(Заголовок H2)
Сторожевые
(выравнивание слева, полужирный шрифт, цвет шрифта зеленый)

Охотничьи
(выравнивания справа, полужирный шрифт, цвет шрифта оранжевый)

Дрессировка
(выравнивание по центру, полужирный курсив)

Клубы Выставки Площадки
*(Выравнивание по центру, размер **шрифта 3, цвет** шрифта **красный)***
Стихотворение *(цвет шрифта синий, выравнивание **по центру)***

По жизни я скромен,
Оваций не надо!
Но как же я классно
Смотрюсь у снаряда!
(Выравнивание по левому краю, размер шрифта 4)

Задание 2.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе <TITLE> укажите название странички «Задание 2. *Форматирование текста на Web-странице по образцу*».
3. Напишите HTML-файл, чтобы при его выполнении получился текст следующего формата:

Элементы форматирования текста

Здесь находиться цитата

Фрагменты текста можно выделять

жирным
или
наклонным
шрифтом. Кроме того, можно включать в текст фрагменты с фиксированной шириной символа

(имитация пишущей машинки).

Текст может быть ~~зачеркнутым~~ или подчеркнутым.

Можно также включать элементы, создающие эффект $_{\text{нижнего индекса}}$ или $^{\text{верхнего индекса}}$.

Использование увеличения шрифта – BIG
Использование уменьшения шрифта – SMALL

Кроме физических существуют также логические стили

Использование акцента – EM
Использование сильного акцента – STRONG
Фрагмент исходного текста CODE
Использование образца – SAMP
Текст, вводимый с клавиатуры – KBD
Значение переменной – VAR
Использование элемента CITE

Так выглядит формат адреса (элемент ADDRESS)

Использование элемента CENTER

Использование элемента DIV

Элемент BR обеспечивает разрыв
строки

Задание 3. Использование линий.

- Создайте новую Web-страницу в редакторе Блокнот.
- В элементе <TITLE> укажите название странички "Задание 3. Использование линий".
- Изобразите радугу, используя горизонтальные линии различного цвета (толщина 10 пикселов, длина 50% от ширины экрана, и выровняйте ее по центру).

Задание 4.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе <TITLE> укажите название странички "Задание 4. *Домашняя страничка (ваше имя и фамилия)*".
3. Напишите HTML-файл, чтобы при его выполнении получился текст следующего содержания: укажите ваши фамилию, имя, отчество, год рождения, название вашего учебного заведения, название специальности, группу. Отразите в нем также следующие сведения: мой город и что мне нравится в нем, мои друзья. Для оформления используйте различные шрифты, стили и способы выравнивания, а также горизонтальные линии.

Контрольные вопросы

1. Дайте характеристику форматированию абзаца.
2. Дайте характеристику форматированию шрифта.
3. Логические стили в HTML - документах.
4. Комментарии в HTML-документах.
5. Горизонтальные линии.
6. Escape последовательности.

Лабораторная работа № 3.
Управление цветом

Цель: научиться создавать цветной фон Web-страницы, использовать шрифт различного цвета.

С помощью цвета вы можете оживить любую Web-страницу, сделать ее более выразительной и удобной для просмотра. Кодирование цвета используется для раскрашивания шрифтов, горизонтальных линий и фона, других элементов странички. Цвета обозначаются английскими названиями или числовыми шестнадцатеричными кодами.

Самый простой способ определить цвет написать его название на английском языке. Так, например, задается красный цвет шрифта в элементе **<FONT>:**

Пример.

```
<FONT color="red"> Текст красного цвета </font>
```

В таблице представлены все допустимые названия цветов. На самом же деле цвет определяется не названием, а так называемым RGB-кодом. Любой цвет представляется в этом случае как комбинация красного (R), зеленого (G) и синего (B) цветов, взятых в определенной пропорции. Цвет задается в шестнадцатеричной системе счисления шестью цифрами (от 0 до F) первые две цифры - красный цвет следующие две цифры - зеленый цвет последние две цифры - синий цвет. Доля каждой цветовой составляющей определяется интенсивностью цвета.

Русское название	**Английское название**	**RGB-код**
Белый	white	#FFFFFF
Желтый	yellow	#FFFF00
Золотой	Gold	#FFD700
Оранжевый	orange	#FFA500
Красный	red	#FF0000
Пурпурный	purple	#800080
Каштановый	maroon	#800000
Светло-зеленый	lime	#00FF00
Зеленый	green	#008000
Оливковый	olive	#808000
Аквамарин	aqua	#00FFFF
Синий	blue	#0000FF

Русское название	Английское название	RGB-код
Сизый	teal	#008080
Ультрамарин	navy	#000080
Индиго	Indigo	#4B0080
Фуксиновый	fuchsia	#FF00FF
Фиолетовый	violet	#EE80EE
Серебристый	silver	#C0C0C0
Серый	gray	#808080
Черный	black	#000000

Если вы внимательно посмотрите на коды, приведенные в таблице, то обнаружите, что для формирования стандартных цветов используются или крайние значения интенсивности базового цвета 00 и FF, или среднее значение 80. Многие современные приложения имеют средства для работы с цветом, предоставляя пользователю возможность, выбрав в палитре цвет, увидеть его численные характеристики. И наоборот, задав численные значения, получить новый оттенок. Нельзя, правда, сказать, что все подобные программы совместимы между собой в смысле генерации цвета. Создав некоторый оттенок в одной программе, а потом, задав его RGB-код в другой, вы не обязательно получите тот же результат. Цветовые нюансы для Web-страниц лучше всего проверять на самих страницах.

Еще один аспект применения цвета. Выше упоминалось, что элемент HR, создающий горизонтальную линию, допускает использование ряда атрибутов. С их помощью линию можно превратить в цветной прямоугольник. Вот, например, прямоугольник светло-зеленого цвета, выровненный влево, высотой 20 и шириной 18 пикселов:

Пример.

```
<HR color="lime" size=20 width=18 align="left">
```

Подобные изображения можно применять для разделения частей страницы или в качестве маркеров для списков. Теоретически их можно использовать и в гиперссылках (как деталь, на которой надо щелкать мышью), но это не совсем удобно: элемент HR всегда размещается на отдельной строке.

Цветовая гамма HTML-документа

Цветовая гамма HTML-документа определяется атрибутами, размещенными внутри метки <BODY>. Вот список этих атрибутов:

bgcolor – определяет цвет фона документа;

Пример.

```
<BODY bqcolor=“#FFFFFF"> ...</body>
```

Насыщенность красным, зеленым и синим – одинакова – FF (это шестнадцатеричное представление числа 255). Результат — белый цвет.

text – определяет цвет текста документа;

Пример.

```
<BODY text="#0000FF ">
```

Это значит, что весь текст страницы будет синим, кроме текста, для которого мы специально прописали <FONT> (если цвет в <BODY> не задавать, то по умолчанию он будет черным).

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.

link = “#FF0000” - цвет гипертекстовой ссылки. Насыщенность красным — FF (255), зеленым и синим — 00 (ноль). Результат — красный цвет.

Кроме того, метка **<BODY>** может включать атрибут **background = "[имя файла]",** который задает изображение, служащее фоном для текста и других изображений. Как и любое другое изображение, фон должен быть представлен в формате GIF (файл с расширением *.gif) или JPEG (файл с расширением *.jpg или *.jpeg).

Браузеры заполняют множественными копиями изображения-фона всё пространство окна, в котором открыт документ, подобно тому, как при строительстве большие пространства стен покрывают маленькими (и одинаковыми) плитками.

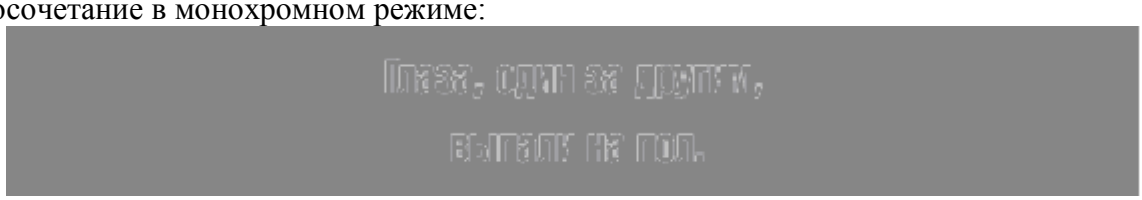
Важно отметить, что цвет фона и изображение-фон никак не отображаются на бумаге при выводе HTML-документа на печать. Из этого есть одно важное практическое следствие: *старайтесь не использовать текст белого цвета.*

Сочетание цветов

Сочетать можно любой цвет с любым. Ну, кроме тех, которые нельзя сочетать. Для большинства людей всегда очевидно – удачен ли подбор оттенков, однако многие страдают страшной, почти неизлечимой болезнью – они рисуют ярко-желтым по ярко-бирюзовому:

Глаза, один за другим,
выпали на пол.

Можно прописать несложное лекарство – больному рекомендуется проверять цветосочетание в монохромном режиме:



Не читается. Цвета одинаковой интенсивности вообще плохо читаются друг на друге.

Подбор цветовой гаммы всегда должен начинаться с вопроса – «для чего?». Если вы оформляете сайт, то, вероятно, у вас есть некий объем текста, который на сайте нужно разместить. Если есть текст, то нужно позаботиться о его читаемости.

Исследования показывают, что человеческий глаз лучше всего воспринимает черным по белому. Настолько хорошо воспринимает, что в случае, если кто-то все-таки не увидел, ему незамедлительно сообщают: «Написано же черным по белому!!!».

Белый на черном тоже имеет право на существование, но это худший вариант. Для больших текстов такое сочетание точно не подходит.

С каждым из этих прекрасных цветосочетаний можно сделать отличный дизайн. Точно с таким же успехом можно сделать и отвратительный. Оказывается, не все зависит от цветов.

Задания к лабораторной работе № 3
Управление цветом

Задание 1. Оформление цветом Web-страницы

Указания к выполнению

- Создайте новую Web-страницу в редакторе Блокнот.
- В элементе <TITLE> укажите название странички «Задание 1. Оформление цветом Web-страницы.»
- В начальном элементе <BODY> используйте соответствующий атрибут, чтобы сделать цвет фона странички черным.
- Сделайте надпись белым цветом «Спокойной ночи». Выровняйте ее по середине страницы. Размер шрифта 4.
- Вставьте горизонтальную линию красного цвета.
- Сделайте надпись желтого цвета «Приятных сновидений». Выровняйте ее по середине страницы. Размер шрифта 7.
- Вставьте горизонтальную линию синего цвета толщиной 10 пикселов, длиной 50% от ширины экрана, и выровняйте ее по центру.

Контрольные вопросы

1. Цветовая гамма HTML-документа.
2. Сочетание цветов.

Лабораторная работа № 4.
Гипертекстовые ссылки

Цель работы: познакомиться с основными принципами построения гипертекстовых ссылок: научиться связывать несколько HTML-документов с помощью гиперссылок, определять цвет гиперссылок, использовать рисунок в качестве гиперссылки.

HTML предлагает множество условных оборотов для текстовых и структурированных документов, но что отличает его от большинства других языков разметки – его возможности разметки гипертекста и интерактивных документов. Ссылки связывают один ресурс Web с другим. Несмотря на простоту, ссылки стали основным залогом успеха Web.

Гипертекст (Hyper Text) – это многомерный текст, т.е. такая организация документов, при которой один документ или текст может включать в себя разнонаправленные ссылки. Эти ссылки, называемые гипертекстовыми ссылками или гиперссылками (Hyper Text links, hyperlinks), позволяют просматривать документ в любом необходимом порядке. Объединение одномерных текстов с включенными в них гиперссылками называется гипертекстом.

Самым простым примером гипертекста является система контекстной помощи Microsoft Windows.

Любая гиперссылка состоит из двух частей: указателя ссылки («якоря» – от англ. *anchors)* и адреса ресурса, которым может быть любым ресурсом Web (например, изображением, видеоклипом, звуковым файлом, программой, документом HTML, элементом в документе HTML и т.д.), на который необходимо осуществить переход.

Внешне отличить гиперссылку от обычного текста очень просто: при наведении курсора мыши на ссылку указатель принимает вид руки с указательным пальцем, как бы показывающим, что этот текст содержит гиперссылку. Сама ссылка подчеркивается (в случае если указателем является текст).

В качестве указателя может выступать текст (отдельное слово, фразы и даже целые страницы текста) и графические изображения. В ряде случаев возможно объединение графики и текста в рамках единого указателя ссылки.

Структура гипрессылки

Указатель ссылки описывается тегом <A>, а адрес перехода реализован с помощью параметра HREF, значением которого является путь к тому или иному интернет-ресурсу.

**<A> **

Если файл находится в том же каталоге, что и документ, на который сделана ссылка, то полный путь к документу указывать необязательно. Достаточно использовать сокращенные версии адресов, называемые *относительными адресами.*

Чаще всего используются следующие шаблоны:

```
текст  <A href ="Адрес ссылки"> текст для щелчка </a>
<A href="Адрес ссылки"> <IMG src="Ссылка на рисунок"> </a>
```

Первый шаблон применяется в том случае, когда гиперссылка встречается в тексте. Атрибут **href** может указывать на ресурс Internet, файл на локальном диске или метку внутри текущей страницы. Текст, расположенный внутри элемента <A>, представляет собой видимую часть гиперссылки. Именно на нем должен щелкнуть пользователь, чтобы осуществить переход. Браузер выделяет этот фрагмент цветом, а после использования гиперссылки меняет цвет, чтобы обеспечить подсказку.

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

link = “#FF0000”

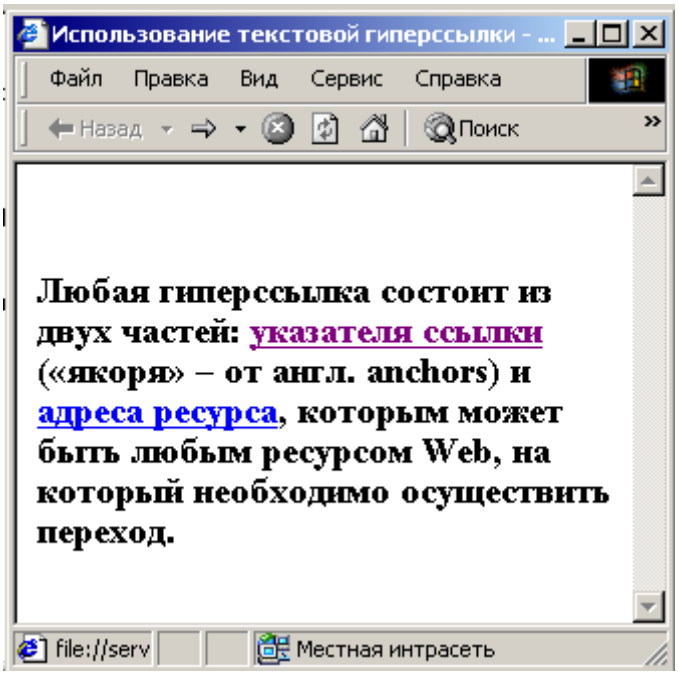
– цвет гипертекстовой ссылки. Насыщенность красным — FF (255), зеленым и синим — 00 (ноль). Результат — красный цвет.

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.

Пример.

```
<HTML>
<HEAD>
<TITLE> Использование текстовой гиперссылки</TITLE>
</HEAD>
<BODY>
<Любая гиперссылка состоит из двух частей:
<A HREF ="1.html"> указателя ссылки </a>
(«якоря» - от англ. anchors) и
<A HREF="2.html"> адреса ресурса</a>,
которым может быть любым ресурсом Web, на который необходимо
осуществить переход.
</BODY>
</HTML>
```



Второй шаблон предназначен для тех случаев, когда видимая часть гиперссылки представляет собой рисунок. Если для последнего определена рамка, то она тоже меняет цвет после использования. Если ссылка указывает на рисунок, который находится на локальном диске, она обязательно должна начинаться со слова file:

file:// *Диск: \ Путь к файлу*

или

file: ///*Диск: /Путь к файлу*

Рамка нужна не только для красоты. Если рисунок используется внутри элемента <A>, то изменение цвета рамки позволяет отличить пройденную гиперссылку от нетронутой.

Справа и слева от рисунка можно создать *пустое пространство*:

hspace = *Число пикселов*

Эта область никак не будет выделяться на экране и примет цвет фона страницы. О ее существовании может говорить наличие промежутка между текстом и рисунком.

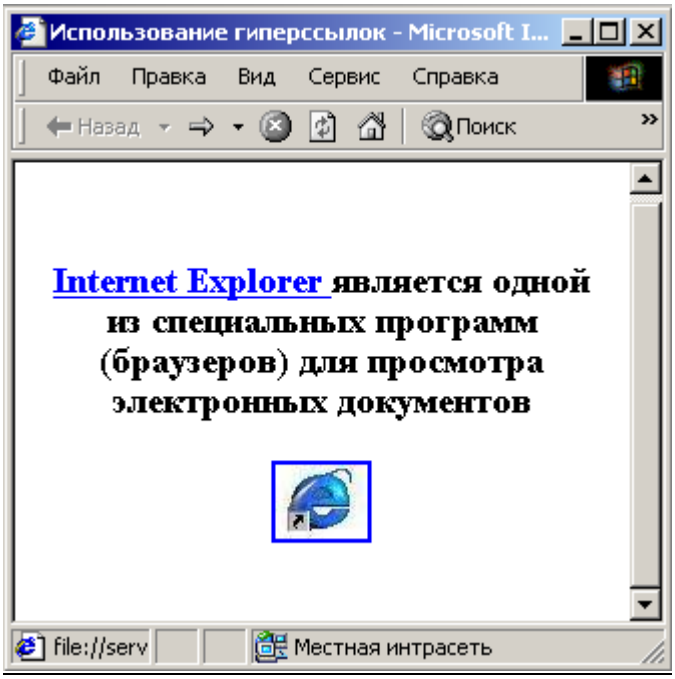
По аналогии, можно создать пустое пространство выше и ниже рисунка:

vspace = *Число пикселов*

Пример.

```
<HTML>
<HEAD>
<TITLE> Использование гиперссылок</TITLE>
</HEAD>
<BODY>
<H3 align = "center">
<A HREF ="7.html"> Internet Explorer </a>
является одной из специальных программ (браузеров) для просмотра электронных документов
<BR><A HREF ="6.html"><IMG SRC = "IE11.jpg" hspace=5 vspace = 20> </A>
</h3>
</BODY>
```

```
</HTML>
```



Правила описания гиперссылок

Гиперссылки можно разделить на два типа: внешние и внутренныие. Внешние ссылки ведут на другие ресурсы глобальной сети или другие документы одного Web-сайта, а внутренние позволяют посетитулю путешествовать в пределах одного HTML-документа.

Внешние гиперссылки

Структура внешней гиперссылки состоит из указателя (якоря) и адреса. Существует два способа записи адреса перехода по внешей ссылке.

1. **Абсолютный.** В значении параметра href указывается полный путь к ресурсу, ссылка на который ставится в документе.

```
<A href="http:/www.site.ru/docs/page1.html"> Ссылка на страницу 1</a>
```

2. **Относительный.** В значении параметра href указывается конечный документ, относительно которого размещена страница, содержащая ссылку.

```
<A href="page2.html"> Ссылка на страницу 2 со страницы 1</a>
```

Такой формати записи внешней ссылки подразумевает расположение файла page2.html (на который указывает гиперссылка) в том же каталоге, что и файл page1.html (с которого он осуществляет переход). В этом случае оба файла расположены в каталоге docs, находящемся на сайте www.site.ru.

Указание протокола перехода по ссылке

Понятие Интернета как глобальной информационной сети подразумевает не только Worid Wide Web. Интернет – это более емкая инфраструктура, включающая в себя различные информационные сервисы, работа которых реализуется с помощью так называемых протоколов – наборов технологических правил взаимодействия документов друг с другом.

Напимер, WWW работает на основе протокола HTTP (HyperText Transfer Protocol, протокол передачи гипертекста). Кроме того, существуют такие технологии, как FTP (File Transfer Protocol, протокол передачи файлов), E-mail (служба электронной почты) и др.

WWW, как самая современная система, должна обеспечивать совместимость с более ранними, поэтому от старых протоколов не отказываются, а стараются приспособить их к современным нуждам (например, FTP).

Вполне возможна ситуация, когда разработчику HTML-документа понадобится поставить ссылку на другие, отличные от HTTP сервисы. В этом случае структура гиперссылки остается прежней – указатель (текстовый и/или графический) и адресная часть.

```
<A href="URL"> текст для щелчка </a>
```

или

```
<A href="Протокол: //Адрес ссылки"> текст для щелчка </a>
```

Пример: <A href = "*http*: / /*www.netscape.com"*> текст для щелчка

Кодовое слово, стоящее в начале URL, обозначает так называемую *схему доступа.* Она определяет тип сервера, доступный при помощи данной ссылки. Для пользователя это представляется как доступ к одной из «разновидностей» Internet. В этом смысле можно сказать, что Internet – это как бы несколько сетей в одной. У каждой из них существуют свои правила доступа, достоинства, недостатки, свои приверженцы и противники. Но все ее клиенты используют одни и те же каналы связи. Похожая ситуация наблюдается и в обычных телефонных сетях. Их можно использовать для связи голосом, передачи факсов, межкомпьютерной связи и т. д.

Существуют следующие схемы доступа:

file	доступ к файлу на локальном диске;
ftp	доступ к архивам файлов при помощи протокола передачи файлов (file transfer protocol);
gopher	доступ к системе Gopher;
http	доступ к WWW;
mailto	отправка сообщения по электронной почте;
news	доступ к новостям USENET;
nntp	доступ к новостям USENET с использованием протокола NNTP;
telnet	подключение по протоколу telnet;
wais	подключение к системе поиска WAIS.

Когда гиперссылка используется для указания адреса электронной почты, ее выбор обеспечивает не переход к новому документу, а запуск усто\ановленной по умолчанию программы чтения и отправки электронной почты (Microsoft Outlook, The Bat и др.). Обычно такую ссылку размещают в конце страницы для обеспечения связи с Web-мастером или автором страницы.

Пример.

```
<A href="mailto:AIvanov@neva.spb.ru"> ссылка на адрес электронной почты </a>
```

Можно несколько усложнить структуру ссылки, добавив в нее готовое поле заголовка электронного письма.

Пример.

```
<A href="mailto:AIvanov@neva.spb.ru?Subject=Здравствуйте!">
ссылка на адрес электронной почты </a>
```

Внутренние ссылки

Если HTML-документ слишком большого размера и нет возможности разбить его на несколько отдельных файлов, можно прибегнуть к помощи внутренних гиперссылок, перемещающих пользователя в пределах одной Web-страницы.

Структура внутренней гиперссылки включает две части – сама ссылка и ее именной идентефикатор (диез плюс имя элемента, аналогичное значению параметра HREF самой гиперссылки), позволяющий переместиться в нужное место гипертекстового документа.

```
Текст подсказки <A href ="#Метка"> Текст для щелчка </a>
```

Для обозначения места, в которое нужно перенести пользователя документа, применяется пустая конструкция

```
<A name ="Метка"> </a>
```

При создании внутренних ссылок визуально выделять расположение именного идентификатора не имеет смысла, его основное назначение – переход в соответствующий раздел текущего документа.

Относительные адреса ссылок

Тип ссылки	Тег
Гипертекстовая ссылка, указывающая на другой HTML-документ или файл	<A href ="имя_файла"> активный текст </A>
Гипертекстовая ссылка, указывающая на анкер (якорь) в другом месте того же документа	<A href = "#имя">активный текст </A>
Ссылка, указывающая на анкер (якорь) в другом HTML-документе	<A href = "имя_файла #имя"> активный текст </A>
Анкеры для двух указанных выше типов ссылок	<A name = "имя"> активный текст </A>
Сочетание ссылки и анкера	<A href = "имя_файла" name = "имя"> активный текст </A>
Ссылка в виде графического изображения	<A href ="имя _ файла><IMG src = " графическое изображение"> </A>

Задания к лабораторной работе № 4
Гипертекстовые ссылки

Задание 1. Создание простейшей гиперссылки

Указания к выполнению

- Создайте две Web-страницы в редакторе Блокнот. Одну назовите pagel.htm, вторую page2.htm. Обе странички сохраните в одной папке под названием site. Обратите внимание, что названия папки и страничек должны быть на английском языке и начинаться со строчной буквы
- В элементе <TITLE> укажите название странички "Задание 1. Создание простейшей гиперссылки".
- Пусть фон первой странички будет зеленого цвета, а второй синего.
- На первой страничке создадим гиперссылку для перехода на вторую страницу. Для этого создайте элемент:

<A href="page2.htm">На страницу 2

- На второй страничке создадим гиперссылку для перехода на первую страницу. Для этого создайте элемент:

<A href="page1.htm">На страницу 1

- Откройте в браузере первую страничку и убедитесь, что обе гиперссылки работают правильно.

Задание 2. Создание гиперссылок

Указания к выполнению

- Выполнить второе задание вы можете, если успешно справились с *Заданием 1*. Создайте третью страничку page3.htm. Пусть ее фон будет черным.
- На страничке page3.htm создайте гиперссылки для перехода на странички pagel.htm и page2.htm.
- На страничках pagel.htm и page2.htm добавьте гиперссылку для переход на страничку page3.htm.
- На страничке page3.htm создайте в конце документа гиперссылку для перехода в начало этого же документа.
- Откройте в браузере первую страничку и убедитесь, что теперь можно перейти с любой странички на любую другую из трех созданных.

Задание 3. Цвет гиперссылок

Указания к выполнению

- Выполнить задание вы можете, если успешно справились с *Заданием 2.*
- На страничках pagel.htm, page2.htm и page3.htm в тэге <BODY> определите цвет гиперссылок:
- все гиперссылки на странице - белые;
- активные гиперссылки - красные;
- посещенные гиперссылки - серые.
- Откройте в браузере первую страничку и, переходя по ссылкам со странички на страничку, убедитесь, что цвет гиперссылок задан верно.

Контрольные вопросы

1. Понятие гипертекста
2. Понятие гиперссылки.
3. Структура гипрессылки
4. Относительный адрес.
5. Абсолтный адрес.
6. Внешние и внутренние гиперссылоки.

Лабораторная работа № 5.
Работа со списками

Цель работы: познакомиться с разными видами списков в HTML-документах, научиться создавать списки на Web-странице.

Очень важный элемент Web-страничек – списки.

Использование списков в качестве средства представления информации объясняется слудующим.

1. Информация в виде списка позволяет большие массивы даннызх на отделные четкие фрагменты, которые человеку гораздо удобнее воспринимать, чем весь поток целиком.

2. Списки позволяют создавать понятную и логичную внутреннюю структуру информационных данных ориентироваться в которой просто и удобно.
3. Использование списков удобно для отображения определенных пошаговых и прочих последовательных процессов.

Список отличается от обычного текста тем, что пользователю не надо думать о нумерации его пунктов: эту задачу программа берет на себя. Если список дополняется новыми пунктами или укорачивается, нумерация корректируется автоматически. В случае ненумерованных списков программа ставит перед каждым пунктом маркеры: кружки, прямоугольники, ромбы или другие изображения. В результате список принимает удобочитаемый вид.

Спецификация HTML предусматривает три основных типа списков:

– маркерованные списки;

– нумерованные списки;

– списки определений.

Маркированный (ненумерованный - Unordered List) список

Маркерованный список представляет собой ненумеронанный или неупорядоченный перечень элементов, для заголовка которых используются специальные маркеры. В качестве маркеров выступают специальные символы, называе6мые буллетами (bullets).

Текст, расположенный между тегами **<UL>** и **
**, воспринимается как ненумерованный список. Каждому элементу списка предшествует символ bullet (пуля) или графическое изображение.

Элемент <UL> является своеобразным обрамлением списка. Он позволяет отделять один список от другого. Каждый новый элемент списка следует начинать с метки <LI> (List Item – пункт списка или элемент списка). У метки <LI> нет парной закрывающей метки. Для элемента <UL> возможными значениями атрибута type являются disc, square и circle, где

● – маркер в виде закрашенного круга – для значения "disc"

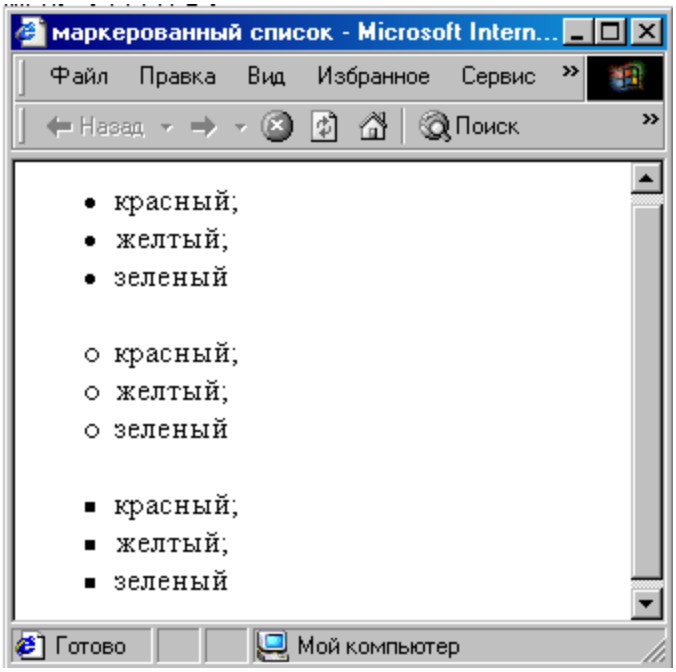
○ – маркер в виде незакрашенного круга – для значения "circle"

□ – маркер в виде закрашенного квадрата – для значения "square"

Пример.

```
<HTML>
<HEAD>
<TITLE> маркерованный список </title>
</head>
<BODY>
<UL>
<LI> красный;
<LI> желтый;
<LI> зеленый
</ul>
<UL type=circle>
<LI> красный;
<LI> желтый;
<LI> зеленый
</ul>
<UL type=square>
<LI> красный;
<LI> желтый;
<LI> зеленый
</ul>
```

```
</body>
</html>
```



После каждого пункта списка осуществляется перенос строки, причем без указания каких-либо структурных тегов.

Если заголовок списка размещается между тегом <UL> и <LI>, то он будет отделенот начала самого списка одним переносом строки и отступом от левого края документа (как и все пункты вписка). При указании заголовка перед тегом <UL> браузер создает двойной перенос и размещает текст заголовка левее пунктов списка.

Отличительной особенностью маркерованных списков вляяется возможность использования вместо стандартных HTML-маркеров свои собственные графические изображения. Для указания браузеру того, что в качестве маркера будет использовано графическое изображение, внутри тега <UL> вместо указателя элемента <LI>размещается обыкновенная HTML-конструкция описания графических объектов.

Пример.

```
<UL>
      <IMG src = “marker.gif> красный
      <IMG src = “marker.gif> желтый
      <IMG src = “marker.gif> зеленый
</UL>
```

Нумерованные списки

Нумерованный список часто называется упорядоченным и представляет собой определенную последовательность элементов.

Нумерованные списки (Ordered List) отмечают тегами **<OL>... ** внутри которых располагаются пункты перечня информационных данных, за указание которых отвечает специальный тег <LI>. Чтобы программа правильно расставила номера вместо символов, выделяющих новый элемент, используются арабские или римские числа или буквы латинского алфавита.

Тег составления нумерованных списков <OL> может включать атрибуты type, start и compact.

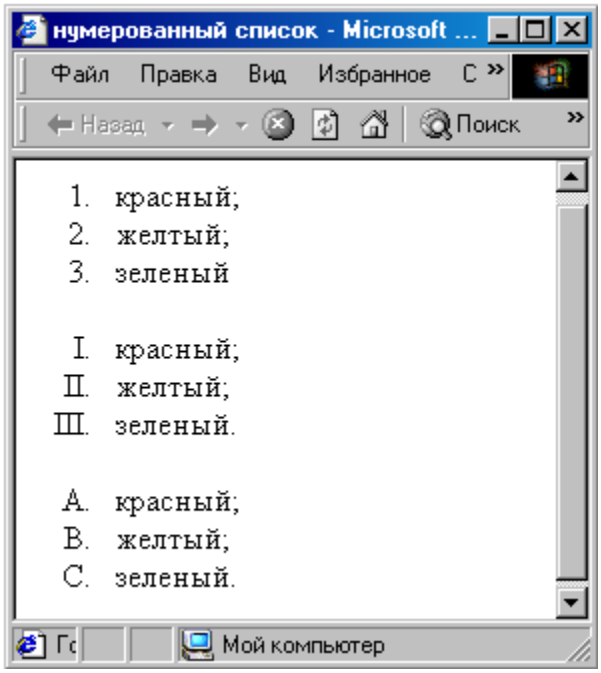
Способ нумерации задается при помощи атрибута type.

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

Типом нумерации, используемым по умолчанию, является type=”1”, поэтому при создании нумеронанных списков такого типа необязательно создавать значения атрибута type (сам перпметр также можно опустить).

Пример.

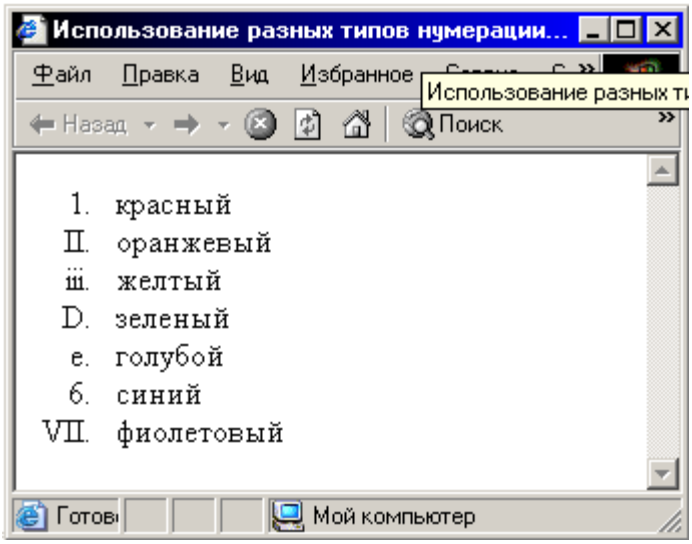
```
<HTML>
<HEAD>
<TITLE> нумерованный список </title>
</head>
<BODY>
<OL>
<LI> красный;
<LI> желтый;
<LI> зеленый.
</ol>
<OL type="I">
<LI> красный;
<LI> желтый;
<LI> зеленый.
</ol>
<OL type="A">
<LI> красный;
<LI> желтый;
<LI> зеленый.
</ol>
</body>
</html>
```



Возможно смешанное использование различных типов нумерации в пределах одного списка. Однако браузеры по-разному реагируют на такое смешение стилей.

Пример.

```
<HTML>
<HEAD>
<TITLE> Использование разных типов нумерации в одном списке</title>
</head>
<BODY>
<OL type="1">
<LI> красный
<LI type="I"> оранжевый
<LI type="i"> желтый
<LI type="A"> зеленый
<LI type="a"> голубой
<LI type="1"> синий
<LI type="I"> фиолетовый
</ol>
</body>
</html>
```



Параметр start позволяет начинать нумерованный список не с первой позиции. При этом значение этого параметра может быть только числовой эквивалент конкретного типа списка. Напимер, чтобы начать список, созданный на основе нумерации большими латинскими буквами, с буквы F , надо указать следующую HTML-конструкцию:

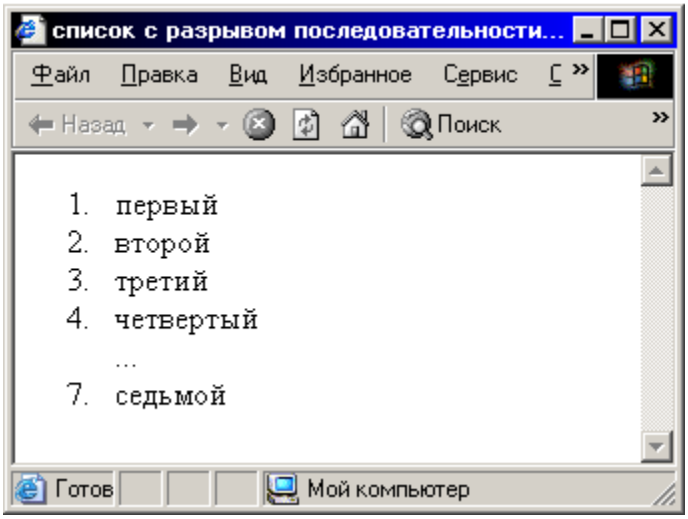
```
<OL type=”A” start=”6”>
```

Параметр compact изначатьно был предусмотрен для списков, отображение которых должно производиться в компактном виде, например, с использованием меньшего размера шрифта, меньшего отступа от края страницы или меньшего расстояния между пунктами списка. Однако на практике этот параметр почти не используется, поскольку наиболее распространенные браузеры никак не реагируют на его применение.

Иногда может возникнуть ситуация, когда список содержит опредленный разрыв в последовательности представления пунктов. Для этой цели используется специальный параметр тега <LI> – value.

Пример.

```
<HTML>
<HEAD>
<TITLE> нумерованный список с разрывом последовательности</title>
</head>
<BODY>
<OL>
<LI> первый
<LI> второй
<BR> ...
<LI value=7> седьмой
</ol>
</body>
</html>
```

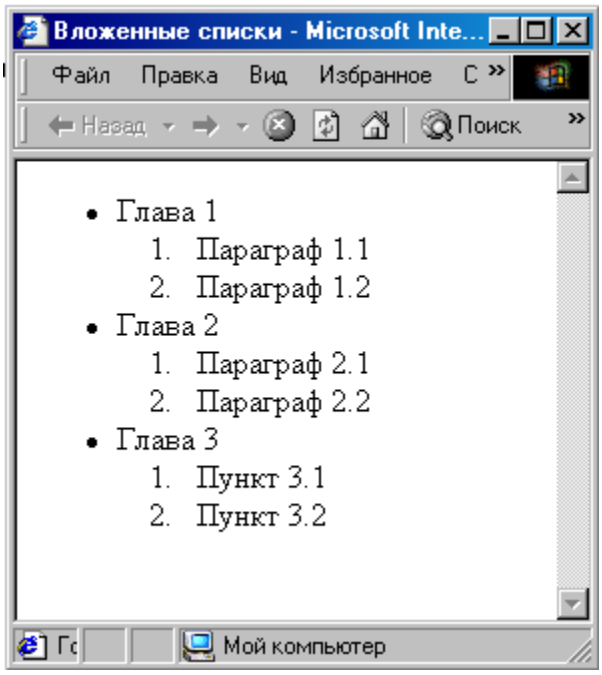


Вложенный список

Вложенный список (Nested Lists) может содержать в себе элемент или целый список любого вида. Вложенные списки очень удобны при подготовке разного рода планов и оглавлений. Число уровней вложенности в принципе не ограничено, однако злоупотреблять вложенными списками все же не следует. Необходимо быть особенно внимательным при расположении флагов начала и конца каждого списка! Чтобы не ошибиться, удобно записывать элементы вложенной конструкции с небольшим отступом.

Пример.

```
<HTML>
<HEAD>
<TITLE>Вложенные списки</title>
</head>
<BODY>
<UL>
<LI> Глава 1
<OL>
<LI> Параграф 1.1
<LI> Параграф 1.2
</ol>
<LI> Глава 2
<OL>
<LI> Параграф 2.1
<LI> Параграф 2.2
</ol>
<LI> Глава 3
<OL>
<LI>Пункт 3.1
<LI>Пункт 3.2
</ol>
</ul>
</body>
</html>
```



Списки определений

Списки определений – это особый тип структуризации информационных данных, идеально подходящий для отображения терминологичесих и толковых словарей, а также различных справочников средствами HTML.

<DL> ... </DL>

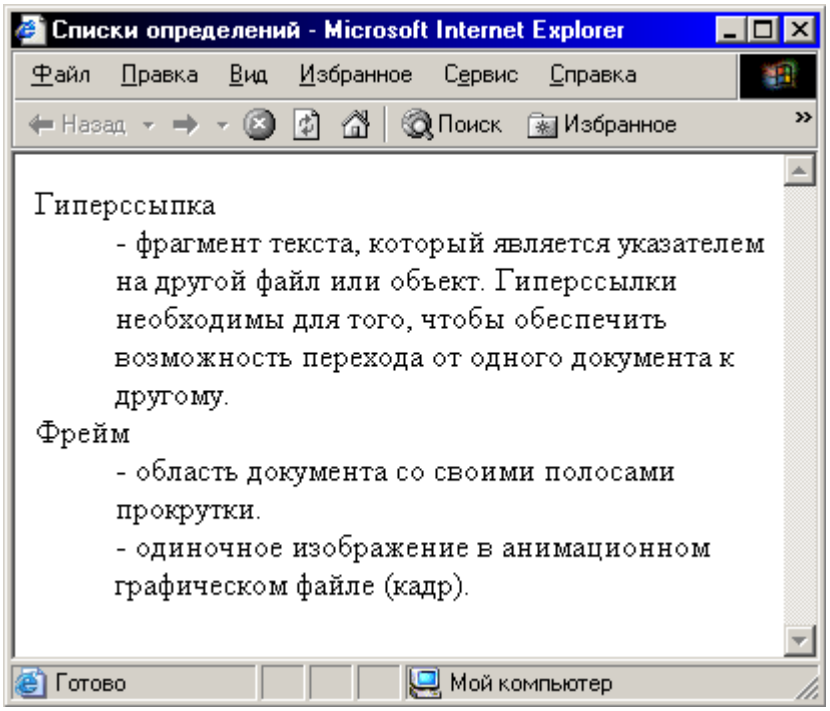
Список определений несколько отличается от других видов списков. Вместо меток <LI> в списках определений используются метки <DT> (от английского definition term — определяемый термин) и <DD> (от английского definition — определение определения).

Каждый пункт списка может быть дополнен одним или несколькими блоками текста при помощи тега (тегов) DD. Каждый блок автоматически размещается с новой строки. Термин «определение» носит условный характер. Абзацы, размещенные в списке, могут быть определениями, дополнениями, разъяснениями пунктов. По сути, пункт представляет собой заголовок, а определение — произвольный текст под заголовком.

Пример.

```
<HTML>
<HEAD>
<TITLE>Списки определений</title>
</head>
<BODY>
<BASEFONT SIZE=3>
<DL>
<DT>Гиперссыпка
<DD>- фрагмент текста,  который является указателем на другой файл или объект. Гиперссылки необходимы для того, чтобы обеспечить возможность перехода от одного документа к другому.
<DT>Фрейм
<DD>- область документа со своими полосами прокрутки.
<DD>- одиночное изображение в анимационном графическом файле (кадр).
</dl>
```

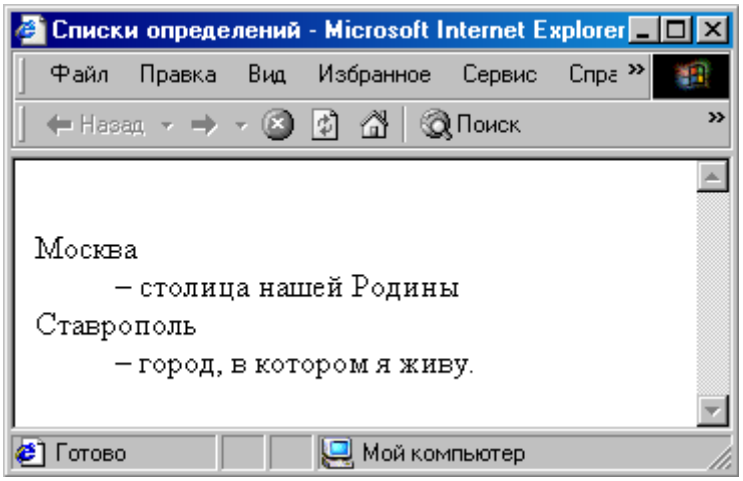
```
</body>
</html>
```



Если определяемые термины достаточно коротки, можно использовать модифицированную открывающую метку **<DL compact>**.

Пример.

```
<HTML>
<HEAD>
<TITLE>Списки определений</title>
</head>
<BODY>
<BR>
<DL compact>
<DT>Москва
<DD> - столица нашей Родины
<DT>Ставрополь
<DD> - город, в котором я живу.
</dl>
</body>
</html>
```

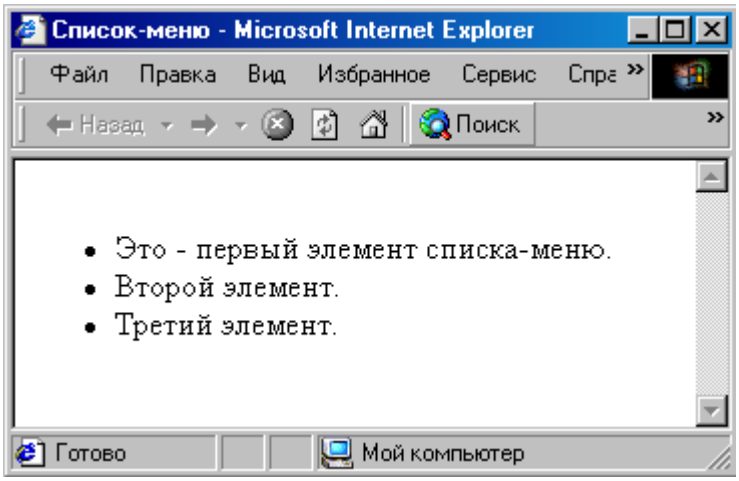


Список-меню

<MENU> (Menu List) используется для представления информации в виде списка-меню, в котором каждая запись занимает одну строку. Это позволяет сделать список более компактным по сравнению с конструкцией <UL>...</UL>. Формат такого списка зависит от используемой программы просмотра HTML-документа, однако многие программы просмотра используют <MENU> как "синоним" <UL>. Каждому элементу списка на экране обычно предшествует символ "пуля".

Пример

```
<HTML>
<HEAD>
<TITLE>Список-меню</title>
</head>
<BODY>
<BR>
<MENU>
<LI>Это - первый элемент списка-меню.
<LI>Второй элемент.
<LI>Третий элемент.
</menu>
</body>
</html>
```

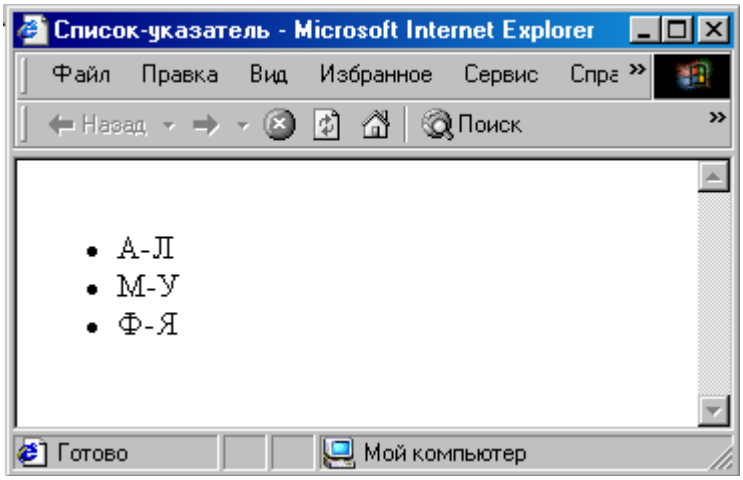


Список-указатель (список-индекс)

<DIR> (Directory List) используется для представления коротких записей (обычно менее 20 символов) в виде списка-индекса. Позволяет формировать список в несколько колонок (обычно с шагом в 24 символа). Формат такого списка зависит от используемой программы просмотра HTML-документа. Обычно каждому элементу списка на экране предшествует символ "пуля".

```
<HTML>
<HEAD>
<TITLE>Список-указатель</title>
</head>
<BODY>
```

```
<BR>
<DIR>
<LI>А-Л
<LI>М-У
<LI>Ф-Я
</dir>
</body>
</html>
```



Задания к лабораторной работе № 5
Работа со списками

Задание 1. Создание нумерованных списков.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе <TITLE> укажите название странички "Задание 1. Создание нумерованных и ненумерованных списков на Web-странице по образцу ".
3. Отформатируйте следующий текст согласно указаниям, данным в скобках курсивом.

Солнце должно быть:
(Шрифт размером 4, красного цвета, выравнивание по центру)

---

1. **Теплым.**
2. **Круглым.**
3. **Желтым.**

Снег должен быть:
(Шрифт размером 10, синего цвета, выравнивание по левому краю)

A. *Белым.*
B. *Холодным.*
C. *Пушистым.*

(Шрифт обоих, списков черный, размер 3).

Задание 2. Создание нумерованных и маркерованных списков, списков определений.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе <TITLE> укажите название странички "Задание 2. Создание нумерованных и ненумерованных списков на Web-странице по образцу ".

3. Создайте Ваш HTML-документ, который будет выглядеть следующим образом:

Списки

Ненумерованный список

- *Пункт 1 списка*
- *Пункт 2 списка*
- *Пункт 3 списка*

Вложенные списки

- *Пункт 1*
    1. Пункт 1.1.
    2. Пункт 1.2.
- *Пункт 2*
    1. Пункт 2.1.
    2. Пункт 2.2.
- *Пункт 3*
    1. Пункт 3.1.
    2. Пункт 3.2.

Нумерованный список

I. Пункт 1
II. Пункт 2
III. Пункт 3
IV. Пункт 4

Список с определениями

Пункт 1

Определение пункта 1

Другое определение пункта 1

Пункт 2

Определение пункта 2

Пункт 3

Определение пункта 3

Задание 3. Создание маркерованных списков.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе <TITLE> укажите название странички "Задание 3. Любимые актеры".
3. Создайте маркированный список, содержащий фамилии ваших любимых актеров.

Задание 4. Создание маркерованных списков.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.

2. В элементе <TITLE> укажите название странички "Задание 4. Цвета радуги".
3. Создайте нумерованный список, перечисляющий цвета радуги.

Задание 5. Создание списка определений.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе <TITLE> укажите название странички "Задание 5. Породы собак".
3. Создайте список определений, содержащий название и описание породы собак.

Задание 6. Создание списка определений.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе <TITLE> укажите название странички "Задание 6. Автомобили".
3. Создать 3-уровневый список. 1 уровень (верхний) – маркированный список стран; 2 уровень – нумерованный список автомобильных фирм; 3 уровень – маркированный список автомобилей (количество элементов на каждом из уровней – не меньше 2). Создать список определений (любой, количество элементов не меньше 3).

Задание 7. Создание списка определений.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе <TITLE> укажите название странички "Задание 7. Принтеры".
3. Создать 3-уровневый список. 1 уровень (верхний) – нумерованный список стран; 2 уровень – маркированный список фирм, производящих принтеры; 3 уровень – нумерованный список принтеров (количество элементов на каждом из уровней – не меньше 2).

Задание 8. Создание списка определений.

1. Создайте новую Web-страницу в текстовом редакторе Блокнот.
2. В элементе <TITLE> укажите название странички "Задание 8. Мониторы"
3. Создать 3-уровневый список. 1 уровень (верхний) – маркированный список стран; 2 уровень – нумерованный список фирм, производящих мониторы; 3 уровень – нумерованный список мониторов (количество элементов на каждом из уровней – не меньше 2). Создать список определений (любой, количество элементов не меньше 3).

Контрольные вопросы

1. Перечислите основные типы списков в HTML-документах.
2. Охарактеризуйте маркированный (ненумерованный) список.
3. Дайте характеристику нумерованных списков.
4. Дайте характеристику вложенным спискам.
5. Списки определений в HTML-документах.

Лабораторная работа № 6.
Вставка графических изображений

Цель: научиться размещать графические изображения на Web-странице, и изменять их размер и размещать горизонтальные линии на странице. Использовать карты-изображения..

Роль графики

Как интересный журнал или проспект теряет в своей привлекательности без цветных иллюстраций, так любой HTML-документ кажется сухим и невзрачным без использования графики. Значение графических изображений в аспекте создания электронных документов нельзя переоценить – реклама и коммерческие предложения компаний и юридических лиц

становятся более выразительными и яркими, иллюстрации и схемы способны превратить скучный перечень услуг или расценок в интересный информативный материал, любой художественный рассказ или произведение будет восприниматься легче и естественнее при наличии картинок или фотографий.

Однако всегда и во всем следует помнить о чувстве меры. HTML-документ, перенасыщенный иллюстративным материалом, будет неоправданно отвлекать внимание пользователя от истинного содержания страницы – информации. К тому же чрезмерное увлечение графикой влечет за собой увеличение времени загрузки электронного документа, что может сказаться на общей оценке посетителем вашего интернет-ресурса.

Также необходимо четко и внимательно отбирать графический материал для последующего размещения в HTML-документах, следить за его соответствием тематике вашего интернет-ресурса в целом и содержанию отдельной страницы в частности.

Следует помнить о том, что графика призвана привлечь внимание посетителя, заострить его интерес на конкретных моментах или формулировках, но ни в коем случае не должна отвлекать от основного содержания Web-сайта. Разумеется, данное утверждение имеет ряд оговорок и исключений из правил (например, в отношении компьютерных галерей и прочих ресурсов, где ставка сделана именно на графические изображения). И, тем не менее, в большинстве случаев необходимо строго следить за количеством графики на ваших HTML-документах.

Характеристика графических стандартов

Любая графическая информация может храниться в двух основных форматах – векторном и растровом.

Графический файл векторного формата состоит из отдельных математических данных, которые с помощью отрезков прямых, называемых векторами, отображают графический объект на экране. Преимуществом векторной графики является независимость качества изображения от масштаба рисунка, а минусом – недостаточная возможность для работы с фотографическими изображениями. Примером файлов векторного формата могут служить CDR (CorelDRAW), AI (Adobe Illustrator), FH (FreeHand), SVG. (Scalable Vector Graphics) и др. Векторный формат распространен, в основном, в полиграфии, предпечатной подготовке высококачественных документов и т. д.

Отображение файла растрового формата основано на обработке минимальной единицы рабочей области экрана – точки (пиксела). Изменение размеров растровых изображений может существенно повлиять на их качество, т. к. масштабирование осуществляется без вмешательства каких-либо сложных математических операций. Наиболее распространенными растровыми форматами являются PSD (Photoshop Document), BMP (Bitmap Image) и др.

Форматом, избранным для демонстрации Web-графики в Интернете, стал растровый формат, поскольку относительно малый размер мониторов пользователей не позволяет выводить на экран изображения больших размеров.

На сегодняшний день для создания графических изображений, предназначенных для размещения в электронных документах, используются три основных стандарта: GIF, JPEG и PNG.

Вставка графических изображений

Чтобы вставить изображение, нужно только иметь это самое изображение в формате GIF (файл с расширением *.gif) или JPEG (файл с расширением *.jpg или *.jpeg) и одну строчку в HTML-тексте.

<IMG> – элемент для создания ссылки на графический файл (image). Он не содержит конечного тэга – вся необходимая информация задается при помощи атрибутов. Этот элемент является универсальным: с его помощью можно использовать изображения в гиперссылках,

вставлять картинки в таблицы, просто размещать рисунки на Web-странице, решать задачи дизайна и т. д.

Необходимым атрибутом является **SRC** – указатель на графический файл:
SRC ="Ссылка на файл".

Допустим, нам нужно включить в документ изображение, записанное в файл picture.gif, находящееся в одном каталоге с HTML-документом. Тогда строчка будет вот такая:

```
<IMG SRC="picture.gif">
```

Очень полезным атрибутом является **ALT**. Он позволяет выводить текст в тех местах, где должны располагаться рисунки. Страница может загружаться достаточно долго, и пока графические файлы на «подходе», пользователь должен видеть, какие изображения он сможет получить.

Пример

```
<HTML>
<HEAD>
<TITLE>Вставка изображения</title>
</head>
<BODY>
<IMG SRC="picture.gif" ALT="Картинка">
</body>
</html>
```

Встретив такую метку, браузер покажет на экране текст “Картинка” и начнет загружать на его место картинку из файла picture.gif. Атрибут ALT может оказаться необходимым для старых браузеров, которые не поддерживают изображений, а также на случай, если у браузера отключена автоматическая загрузка изображений (при медленном подключении к Интернет это делается для экономии времени).

Файл, содержащий изображение, может находиться в другом каталоге или даже на другом сервере. В этом случае необходимо указать его полное имя (путь в файлу):

Можно использовать атрибуты выравнивания:

align ="bottom" — по нижнему краю;
align ="left" — влево;

align ="middle" — по центру;
align ="right" — вправо;
align ="top" — по верхнему краю.

Высоту и ширину области, в которой демонстрируется рисунок, задают при помощи атрибутов **height - высота и width - ширина**. В том случае, когда задается один из этих атрибутов, рисунок масштабируется таким образом, чтобы его высота или ширина соответствовали заданной. Второй размер устанавливается автоматически, в соответствующей пропорции.

Таким образом, применение только одного из атрибутов изменяет оба размера рисунка. Если задать явным образом оба атрибута, то рисунок будет масштабироваться по двум осям в соответствии с заданным размером. Масштабирование, как правило, ухудшает качество изображения.

Например, реальный размер рисунка 76x121 пикселов. Напишем два варианта отображения данного рисунка.

```
<IMG SRC = "picture.gif" width="76"  height="121">
<IMG SRC = "picture.gif" width="176”>
```

Обратите внимание, что во втором случае изменен размер рисунка (мы изменили ширину, высота будет пропорционально изменена автоматически) При этом происходит потеря качества изображения, поэтому желательно задавать эти атрибуты в соответствии с реальными размерами рисунка.

Атрибут **border** – задает рамку вокруг объекта. border="2" ширина рамки задается в пикселях. При наличии тегов **hspace** и **vspace** вокруг картинки образуется отступ в соответствующее количество пикселей от текста.

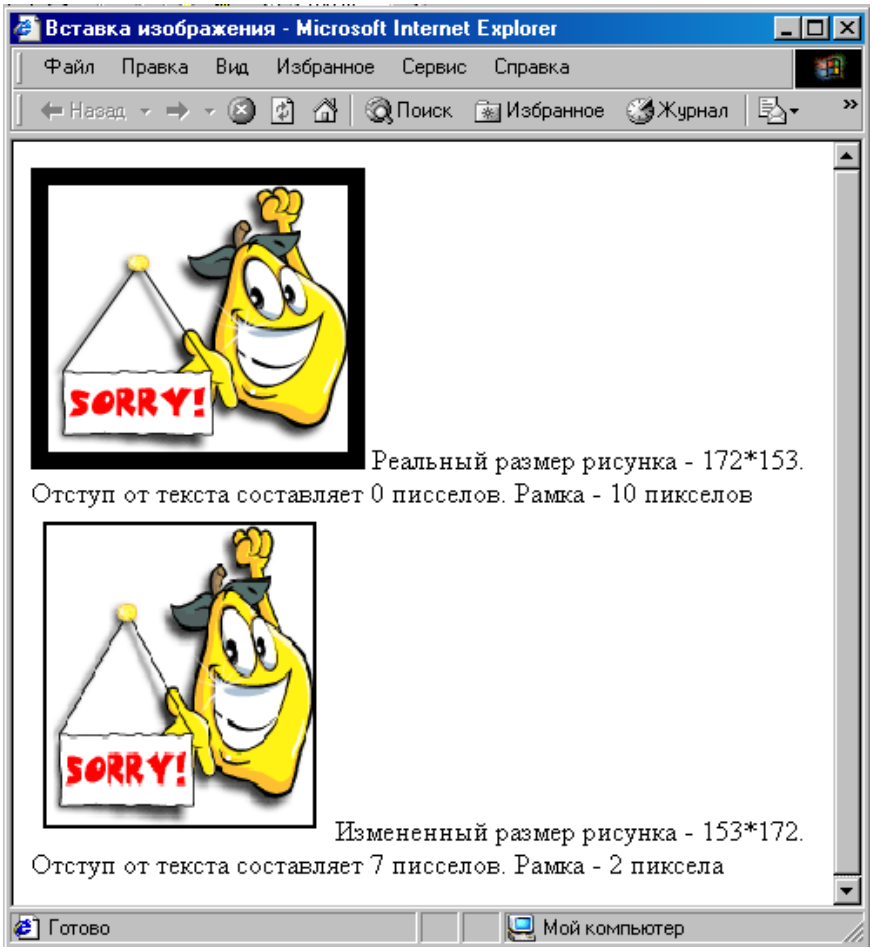
hspace = Число пикселов	Пустое пространство справа и слева от рисунка
vspace = Число пикселов	Пустое пространство выше и ниже рисунка

Полностью тэг **IMG** может выглядеть следующим образом:

```
<IMG src="путь к файлу рисунка" width="ширина картинки" height="высота картинки" hspace="число" vspace=" число" border=" число" align="left…" alt ="подпись к рисунку">
```

Пример.

```
<HTML>
<HEAD>
<TITLE>Вставка изображения</title>
</head>
<BODY>
<IMG SRC="picture.gif" border = "10">
Реальный размер рисунка - 172*153. Отступ от текста составляет  0 писселов. Рамка - 10 пикселов
<BR>
<IMG SRC="picture.gif" width="153" height="172" hspace="7" vspace="7" border = "2">
Измененный размер рисунка - 153*172.  Отступ от текста составляет 7 писселов. Рамка - 2 пиксела
</body>
</html>
```



Использование карт изображений

На многих HTML-документах сегодня с успехом используются так называемые карты-изображения (Imagemaps), которые представляют собой обычные графические файлы (как правило, стандарта GIF или JPEG) с привязанными к различным областям этого изображения гиперссылками.

Такие области описываются специальными координатами, в соответствии с которыми браузер переносит пользователя на нужную страницу.

О правилах конфигурации карт-изображений будет рассказано чуть позже, а сейчас рассмотрим преимущества и недостатки карт-изображений как средства навигации (перемещения) по HTML-документам.

Преимущества и недостатки карт-изображений

К основным преимуществам можно отнести следующие моменты:

- более удобного средства, чем карта-изображение, для создания сложных навигационных меню (в особенности географических, топографических и прочих карт) не найти. Процесс создания и пространственного размещения на странице нескольких десятков кнопок для обозначения, например, всех областей Российской Федерации, чрезвычайно сложен и потребует больших временных затрат;
- использование карты-изображения в качестве навигационных меню на каждой странице интернет-проекта может существенно сократить время загрузки электронных документов и сэкономить место на Web-сервере;
- для использования карты-изображения потребуется изготовить всего один рисунок;
- использование карт-изображений позволит разработчику HTML-документов реализовать

самые смелые дизайнерские задумки. Можно создавать графические объекты любой сложности и формы, не задумываясь об их пространственном размещении на странице, что способно придать интернет-ресурсу оригинальность и сделать его более запоминающимся для посетителей.

Однако без некоторых недостатков также не обошлось:

- графические заготовки для карт-изображений, имеющих большой размер файла, могут заметно увеличить время загрузки электронных документов по сравнению с обычными текстовыми ссылками;
- для посетителей, которые экономят свое время пребывания в Интернете, переход к HTML-документам по ссылкам, указанным в конфигурации карты-изображения, не позволяет отслеживать страницы, на которых они уже побывали, поскольку гиперссылки карт-изображений не изменяют цвет после посещения их пользователем;
- при наведении курсора мыши на определенную активную область карты-изображения в серверном варианте, в статусной строке браузера отображается не адрес электронного документа, а координаты области, которые хранятся в конфигурационном файле на самом Web-сервере;
- если параллельно с картой-изображением не предусмотрено дублирующее текстовое меню, то посетители, которые по каким-либо причинам не могут загрузить графику или отключили ее в своем браузере, останутся не у дел.

Область применения

В принципе, карты-изображения можно применять в самых разнообразных областях компьютерных технологий. Наиболее распространенными из них являются:

- геоинформационные и картографические системы;
- баннерные рекламные сети и системы электронной коммерции;
- электронный и сотовый банкинг, платежные системы;
- игровые трехмерные и двумерные интернет-ресурсы;
- корпоративные серверы;
- интернет-ресурсы широкого профиля.

Как видно из списка, диапазон применения карт-изображений может охватывать практически все отрасли современных технологий.

Конфигурация карт-изображений

Прежде всего следует сказать, что карта-изображение не является принципиально новой технологией. Однако раньше их применение было осложнено рядом причин:

- |ранние версии некоторых браузеров (например, Netscape Navigator 1.x) не поддерживали технологию карт-изображений;
- часть Web-серверов не позволяло использовать графику для создания карт-изображений вообще или предоставляло такую возможность только лишь профессиональным разработчикам или крупным компаниям;
- раньше был возможен единственный вариант реализации карт-изображений – серверный, который подразумевал управление переходом на соответствующий документ исключительно со стороны сервера.

Сегодня существует клиентский вариант карт-изображений, что во многом облегчает применение данной технологии, которую в настоящее время поддерживает большинство современных браузеров.

Сама карта-изображение представляет собой обыкновенный графический |рисунок, а ее конфигурация определяется в виде значений координатных

кривых, которые прописываются в HTML-коде. Эти значения указывают: активные области изображения и содержат информацию о том, куда следует перейти браузеру после нажатия на одну из этих областей.

Активные области могут иметь форму прямоугольника, круга и многоугольника. Также позволительно комбинировать эти три варианта или определять область, расположенную вне активных областей карты-изображения.

Типы карт-изображений

Как уже было сказано, карты-изображения могут быть реализованы в двух вариантах – серверном и клиентском. Рассмотрим подробно каждый из них.

Серверный вариант

Серверный вариант реализации карты-изображения (Server-Side Imagemap) подразумевает, что доку+мент, в котором прописаны координаты областей, находится на самом Web-сервере.

Процесс перехода по гиперссылкам на карте-изображении осуществляется следующим образом. При нажатии на одной из активных областей браузером передаются координаты на сервер, который обращается к специальному конфигурационному файлу. Результат обработки данных возвращается браузеру назад в виде адреса, соответствующего данной активной области, который загружается в окне обозревателя.

В случае если конфигурационный файл не содержит данных об искомой активной области, выводится сообщение об отсутствии данного документа. Для того чтобы браузер определил графический объект как карту-изображение, необходимо в теге <img>, описывающем конфигурацию карты-изображения, указать параметр ismap, а файл конфигурации сохранить в формате MAP.

Серверный вариант поддерживает два формата реализации карт-изображений – CERN и NCSA.

Первый разработан научным центром European Organization for Nuclear Research и предлагает следующую запись координат активных областей:

```
тип_области координаты адрес
```

Значения пар координат разделяются запятой и заключаются в круглые скобки, например:

```
rect (54, 127) (445, 344) http://www.site.ru/
```

В начале конфигурации стоит значение rect (прямоугольное выделение для задания активной области). Другими значениями могут быть: circle (крут), poly (многоугольник) и default (значение по умолчанию).

Формат NCSA разработан центром приложений для суперкомпьютеров National Center for Supercomputing Applications и предлагает несколько иной формат записи конфигурации карт-изображений:

```
тип_области адрес координаты
```

Координаты x, y тоже разделяются запятыми, но в скобки не заключаются, например:

```
rect http://www.site.ru/ 54, 127 45, 344
```

Кроме типов областей, предложенных CERN, данный формат разрешает использование типа point (активизируется та ссылка после нажатия, которая обозначена ближе всего к точке соприкосновения).

Клиентский вариант

Клиентский вариант позволяет разместить все данные об активных областях карты-изображения в самом HTML-документе. В этом случае количество обращений к серверу сильно сокращается, а конфигурировать саму карту-изображение можно параллельно с изменением HTML-кода.

Размещать файл конфигурации на Web-сервере не нужно – координаты активных областей указываются в самом документе, куда встроен графический объект для карты-

изображения. При использовании клиентского варианта (Client-Side Imagemap) в тег <img> добавляется параметр usemap.

В связи с тем, что клиентский вариант на сегодня распространен более широко и к тому же более доступен, подробно рассмотрим процесс конфигурации карты-изображения на примере Client-Side Imagemap.

<MAP> </map>

Для определения конфигурации активных областей карты-изображения используется специальный тег-контейнер <MAP> с параметром name, который должен соответствовать свойству параметра usemap в теге <IMG>. Внутри него определяются области карты при помощи элементов <AREA>, и задается имя карты при помощи атрибута:

```
name="Имя"
```

Необходимо отметить, что описание карты-изображения следует сразу же после указания тега рисунка <IMG>:

```
<IMG scr=”picture.gif” usemap="#mymap"
<MAP name="mymap"
Координаты активных областей…
</MAP>
```

<**AREA**>

Описание активных областей карты-изображения осуществляется с помощью тега <AREA>, не требующего за собой закрывающего тега.

Тег <AREA> может включать следующие параметры:

- shape – определяет форму активной области.

Атрибуты, определяющие форму области на карте, являются обязательными. Существует три стандартных вида областей: *круг* (circle), *прямоугольник* (rect) и *многоугольник произвольной формы* (polygon).

Для круга необходимо задать координаты центра и радиус (R), выраженные в пикселях. Координаты центра отсчитываются от левого (X) и верхнего (Y) краев рисунка. Шаблон для задания круговой области таков:

```
shape = "circle" coords = X,Y,R
```

Для определения области произвольной конфигурации задают координаты (X,Y) каждого из углов многоугольника, который точно или приблизительно соответствует по форме этой области:

```
shape = "poly" coords = X1,Y1,X2,Y2,X3,Y3,...
```

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

```
shape = "rect" coords = X1,Y1,X2,Y2
```

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

- coords – осуществляет выбор конкретной активной области и содержит значения пар координат. Начало координат размещается в верхнем левом углу графического изображения, которому соответствует начальное значение 0, 0.
- target – используется при работе с фреймами.
- alt – действие параметра аналогично использованию его в обычных гиперссылках на основе графических указателей (альтернативный текст);

```
alt ="Текст подсказки"
```

- href – наличие гиперссылки для данной области.

```
href ="Протокол: //Адрес ссылки"
```

- nohref – отсутствие гиперссылки для данной области.

С изображениями карт удобно работать в стандартном для Windows редакторе Paint. Для него изображение должно быть представлено в формате **BMP**. Этот редактор позволяет использовать сетку в режимах увеличения. Ее можно включить или отключить при помощи комбинации клавиш **CTRL+G.** После выбора инструмента *Выделение* указатель мыши приобретает вид тонкого крестика. Таким образом, указатель можно легко установить с точностью до одного пикселя. В строке состояния редактора будут указаны координаты курсора относительно верхнего левого угла рисунка. Значения координат соответствуют требуемым для атрибута *coords* величинам и идут в том же порядке (X,Y).

Если первоначально изображение создано не в формате GIF, то его можно конвертировать в этот формат, используя любой графический редактор, поддерживающий такой тип файлов. Например, MS Photo Editor, входящий в состав Microsoft Office, или Paint Shop Pro фирмы JASK. Достаточно открыть графический файл в редакторе и сохранить его (выполнить команду Save As) в формате GIF.

Реальные карты, созданные с использованием самых разнообразных графических пакетов, смотрятся очень привлекательно. Часто области не имеют четких границ, и неискушенному пользователю «мышечувствительная» карта может показаться последним достижением в области разработки программ или, по крайней мере, хитро придуманным трюком. На самом же деле возможность использования карт была заложена в HTML с самых ранних версий.

Задания к лабораторной работе № 6
Вставка графических изображений

Задание 1. Размещение графики на Web-странице

- Создайте новую Web-страницу в редакторе Блокнот.
- В элементе <TITLE> укажите название странички «Задание 1. Размещение графики на Web-странице».
- Если в тэге <BODY> не указывать цвет странички, то по умолчанию фон будет белым. Сделайте цвет фона странички оранжевым.
- Разместите на страничке любой рисунок. Для размещения изображения вам требуется указать путь к файлу в элементе <IMG>. Чтобы упростить описание пути к этому графическому файлу, скопируйте этот файл в ту же папку, в которой будет сохранена Web-страничка.

Если файл находится не в одной папке с Web-страничкой, то необходимо указать относительный путь к этому файлу.

- Сделайте подпись к рисунку с помощью атрибута alt.
- Создайте рамку вокруг рисунка шириной 2 пиксела.
- Над рисунком поместите заголовок самого большого размера и выровняйте его по центру страницы.

Задание 2. Изменение размеров изображения на Web-странице.

- Создайте новую Web-страницу в редакторе Блокнот.
- В элементе <TITLE> укажите название странички «Задание 2. Изменение размеров изображения на Web-странице».
- Пусть фон странички будет зеленого цвета.
- Определите размер выбранного вами изображения.

- Вставьте это изображение на страничку, задав его размеры: ширина 100, высота 100 пикселов.
- Просмотрите в браузере полученную страничку. Обратите внимание на следующие вещи:

A. В связи с тем, что вы непропорционально изменили размеры изображения, произошло искажение рисунка;
B. Так как вы увеличили размер изображения, произошла потеря качества рисунка (обратите внимание на края изображения);
C. Для размещения изображений на Web-страничке, лучше использовать прозрачный фон рисунка.

Задание 3. Использование рисунка в качестве гиперссылки

• Выполнить задание вы можете, если успешно справились с *Заданием 3.*

• На страничках pagel.htm, page2.htm и page3.htm, вместо текстовых гиперссылок используйте рисунки.

Для этого:

• Вставьте на первую страничку одно изображение и создайте гиперссылку для перехода на вторую страницу:

<A href ="page2.htm"> <IMG src="*.gif">

• самостоятельно вставьте другое изображение для перехода с первой странички на третью;

• задайте бордюр обоих изображений, равным 2;

• Откройте в браузере первую страничку и, переходя по ссылкам со странички на страничку, убедитесь, что переходы работают верно.

• Самостоятельно вставьте изображения на страницы page2.htm и page3.htm и задайте соответствующие гиперссылки.

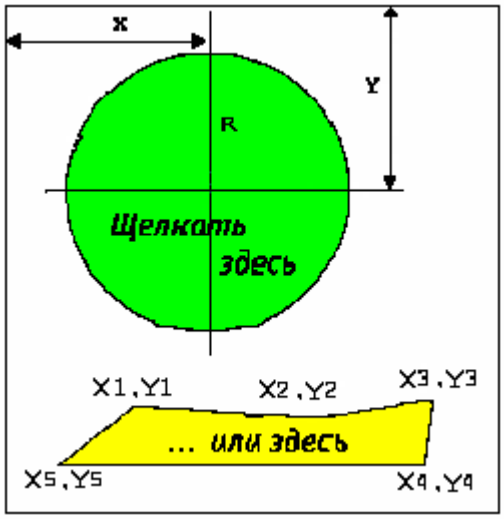
• Откройте в браузере первую страничку и, переходя по ссылкам со странички на страничку, убедитесь, что кнопки на всех страницах работают верно.

Задание 4. Использование рисунка в качестве гиперссылки (карты)

Создайте страницу Map.htm, на которой разместите две карты. Карта, имеющая имя kartal, содержит область в виде круга и область произвольной формы. Изображение karta2 содержит область прямоугольной формы. Между картами должен располагаться текст, объясняющий действие ссылок (щелчков). Ваш HTML-документ должен выглядеть следующим образом:

Примеры карт

Изображения карт иллюстрируют способы задания координат областей для переходов



Карта 1

На этой странице представлены две карты, которые позволяют выполнять различные переходы. Для правильного функционирования страницы необходимо проверить все ссылки на файлы, заданные с помощью атрибутов src и href.

Щелчок по зеленому кругу должен обеспечить переход к вашей Web-странице. Желтый многоугольник вернет вас на страницу Color.htm (название и коды цветов). Красный прямоугольник (см. ниже) обеспечивает переход в начало этой страницы.

Карта 2

2. Измените вид карт, используя другие геометрические фигуры.

Контрольные вопросы

1. Роль графики. Характеристика графических стандартов
2. Вставка графических изображений в HTML-документ.
3. Использование карт изображений

Лабораторная работа № 7.
Таблицы HTML-документов

Цель работы: познакомиться с основными приемами создания таблиц на языке гипертекстов, рассмотреть основные теги и атрибуты, научиться создавать, форматировать, заполнять таблицы, проектировать структуру таблицы.

Одним из наиболее мощных и гибких средств представления информационных данных в HTML по праву являются таблицы. В повседневной жизни, решая те или иные задачи, мы часто сталкиваемся с таблицами. Однако в HTML таблицы не ограничиваются удобным средством структурирования информации. Сегодня таблица становится основой большинства электронных документов, структура которых может включать самые разнообразные элементы HTML. Таблицы, изначально избранные в качестве визуального способа представления данных, сейчас имеют гораздо более важную функцию – управление структурой HTML-документа в целом. Удобство размещения данных в таблице неоспоримо, а преимущества перед другими средствами представления информации (например, списки) позволяют считать таблицы основополагающим структурным элементом любого HTML-документа.

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

Основные методы создания таблиц

<TABLE> </table>

Внешний элемент таблицы. Он позволяет задавать общие свойства таблицы и отделяет ее структуру от остальной части Web-страницы. Если начало и конец таблицы не определены, то никакие табличные команды не будут работать.

Рассмотрим параметры этого элемента.

Параметр align

Таблицу можно выровнять по горизонтали при помощи атрибута align. Данный параметр устанавливает тип выравнивания всей таблицы относительно ширины HTML-документа.

`align="left"` – влево;
`align="center"` – по центру;
`align="right"` – вправо.

Параметры width и height

Параметры width и height отвечают соответственно за ширину и высоту таблицы. Значение этих параметров может быть указано в процентах или пикселах:

Пример.

```
<TABLE WIDTH=''500" HEIGHT="50%">
```

Пример: width =250 задает таблицу шириной 250 пикселей; width = 50% задает таблицу на половину ширины страницы при любом размере изображения на экране.

В случае указания ширины или высоты в пикселах таблица будет иметь фиксированные размеры. Такая таблица при изменении размеров окна браузера останется неизменной. При использовании в качестве единицы измерения процентов ширина/высота таблицы будет варьироваться в зависимости от размеров окна браузера. В этом случае размер ячеек таблицы будет уменьшаться/увеличиваться пропорционально заявленному значению в процентах. Содержимое таблицы будет перемещаться в окне браузера соответственно изменению его размеров.

Для управления видом рамки используются два атрибута. Дело в том, что браузер создает изображение рамки, имитируя ее трехмерность (выпуклость). На рамке можно различить фронтальную и боковую наклонную грани. Кроме этого, имитируется различие в их освещенности.

Параметр border

Параметр **border** задает ширину рамки. Если рамка не задана, то получим таблицу без рамок, тот же результат дает <TABLE border=0>. Также как и ширину можно задать в процентах и пикселях.

```
border = Ширина в пикселях
```

Параметр bordercolor

Параметр **bordercolor** предназначен для определения цвета рамки вокруг таблицы (поддерживается только Internet Explorer). Значение указывается в шестнадцатеричном формате или в виде наименования (например, green).

Параметр cellspacing

Параметр cellspacing задает расстояние между соседними ячейками таблицы по вертикали и горизонтали. Расстояние между соседними ячейками (как по вертикали, так и по горизонтали) всегда будет одинаково, т. к. HTML не позволяет назначать разные значения в пределах данного параметра.

```
cellspacing = Ширина в пикселях
```

Если значение этого атрибута равно нулю, рамка получается тонкой, заостренной. Если данный параметр не задан, по умолчанию задается величина, равная 2 пикселям. Атрибут полезен для размещения текста и графики именно там, где нужно.

Параметр cellpadding

Для всех ячеек таблицы можно задать размер пустого пространства между содержимым таблицы и ее границами. Для создания и регулирования отступа между рамкой ячейки и ее содержимым (по вертикали и горизонтали) используется параметр **cellpadding.**

```
cellpadding = Число  пикселов
```

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

При значении параметра cellpadding, равном нулю, содержимое ячейки будет вплотную прижато к рамке таблицы, что ухудшит восприятие информации (это утверждение еще более актуально для таблиц с прозрачной рамкой, т. е. значением border="0", – в этом случае текст соседних ячеек сольется друг с другом).

Параметр unit

Параметр **unit** тега <TABLE> определяет единицы измерения, используемые при указании размеров всей таблицы и ее отдельных столбцов. Может принимать три значения:

- **unit = en** - это значение задает единицу измерения, равную EN-пробелу, т.е. ширине буквы n (зависит от размера шрифта);
- **unit = relative** используется для задания относительной ширины столбцов в процентах от общей ширины таблицы. (Числа при этом воспринимаются как ширина в процентах);
- **unit = pixels**- это значение применяется, когда нужно точно задать ширину столбца на экране. Например, тег <TABLE unit=pixels width=340> сформирует таблицу шириной 340 пикселей. Так как разные мониторы имеют разное разрешение, то при задании ширины таблиц и столбцов в пикселях ячейки на экранах с более высоким разрешением могут оказаться меньше. Например, таблица шириной 640 пикселей на мониторе с разрешением 640*768 займет весь экран, а на мониторе с разрешением 1024*768 немногим более половины ширины экрана.

Параметр colspan

Параметр **colspan** = перечисляет все столбцы и для каждого задает выравнивание и размер. Для столбца существует 5 способов выравнивания: ***L*** - по левому краю; ***C*** - по центру; ***R*** - по правому краю; ***J*** - и по левому и по правому; ***D*** - по десятичной точке или запятой.

Пример:

```
<TABLE UNIT= PIXELS COLSPAN=“L10 C15 R20 J25 D 30”>
```

Первый столбец имеет ширину 10 пикселей и его содержимое выравнивается по левому краю; второй столбец имеет ширину 15 пикселей и его содержимое выравнивается по центру; третий столбец имеет ширину 20 пикселей и его содержимое выравнивается по правому краю; четвертый столбец имеет ширину 25 пикселей и его содержимое выравнивается по левому краю и по правому краю; пятый столбец имеет ширину 30 пикселей и его содержимое выравнивается по десятичным запятым.

Параметр bgco1or

Для всей таблицы может быть задан цвет фона:

```
bgcolor="Цвет"
```

или

```
bgcolor="#RRGGBB".
```

В этом случае атрибут bgcolor помещается внутри элемента <TABLE>.

Пример.

```
<TABLE bgcolor="red" border="2" cellspacing="2" cellspacing="2">
<TR>
<TD>Ячейка 1</td>
<TD>Ячейка 2</td>
</tr>
</table>
```

Параметры тега <TABLE> перечислены в таблице.

Параметр	Функция
border	Создание рамки вокруг таблицы
bordercolqr	Определение цвета рамки вокруг таблицы
cellspacing	Указание расстояния между соседними ячейками
cellpadding	Создание отступа от границы ячейки до ее содержимого
align	Указание типа выравнивания таблицы по ширине окна браузера

Параметр	Функция
width	Определение ширины таблицы
height	Определение высоты таблицы
unit	Определяет единицы измерения, используемые при указании размеров всей таблицы и ее отдельных столбцов
colspan	Перечисляет все столбцы и для каждого задает выравнивание и размер

<CAPTION> </caption>

Элемент для задания заголовка таблицы. Заголовок выводится на экране вне рамки таблицы. Положением заголовка можно управлять:

`Align ="top"` – заголовок над таблицей;

`align = "bottom"` – заголовок под таблицей.

<TR> </tr>

Элемент, создающий строку таблицы (сокращение от Table Row – строка таблицы). Он не имеет конечного тега. Строка заканчивается там, где начинается следующая, то есть где стоит следующий элемент <TR>. Если в таблице два набора тегов <TR></tr>, то в ней будут две строки. Если N наборов<TR></tr>, то в таблице N строк.

<TH> </th>

Элемент ячейки, которая является заголовком столбца или строки таблицы (Table Header, заголовок таблицы). Этот элемент должен располагаться внутри элемента <TR>. Ячейка-заголовок отличается от обычной тем, что браузер выводит текст внутри нее выделенным (как правило, полужирным) шрифтом. Для элемента ячейки предусмотрено много атрибутов.

<TD> </td>

Этот элемент определяет обычную ячейку таблицы. Число тегов <TD></td> в строке определяет число ячеек. Строка с пятью наборами тегов <TD></td> содержит пять ячеек. Число ячеек в строке не должно быть постоянным. В одной строке может быть 5 ячеек, в другой 3.

Элемента – <**TR**>, <**TH**> и <**TD**> – не имеют конечных тегов. Функцию конечного тега выполняет следующий элемент, который определяет структуру таблицы. Однако все же рекомендуется их указывать для предотвращения ошибок, которые могут возникнуть при создании сложных вложенных таблиц.

Цветные таблицы

Параметры bgcolor и background

Параметры **bgcolor** и **background** используются для работы с фоном ячеек или целых рядов таблицы. **bgcolor** позволяет задать определенный цвет ячеек, а **background** – указать путь к графическому изображению, которое будет служить фоном.

Параметр bgcolor используется для тегов <TR>, <TD> и <TH>. Параметр background применим только к тегам ячеек <TD> и <TH>.

Пример.

```
<TR>
<TD bgcolor = “red” >
</tr>
</td>
```

Параметр **bordercolor** задает цвет границы каждой ячейки таблицы, например:

```
<TD bordercolor=yellow> </td>
```

Здесь цвет не в кавычках.

Атрибуты **bordercolordark** = и **bordercolorlight** = задают два цвета для окраски краев таблицы (создает “трехмерное” изображение),

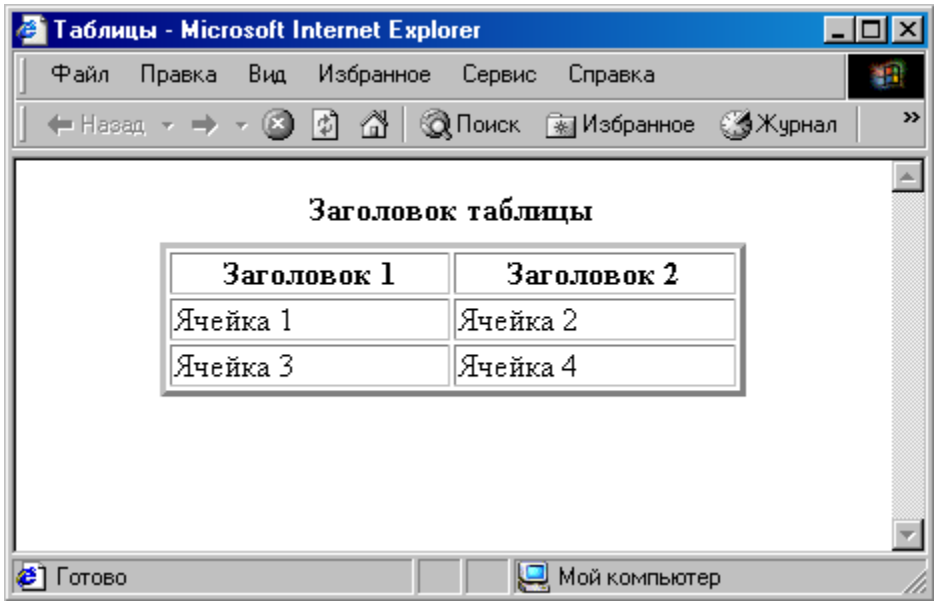
Пример.

```
<TR>
<TD bordercolorlight = yellow bordercolordark = blue>
```

Теперь, зная, какие элементы используются для построения таблицы, можно создать простейшую таблицу:

Пример.

```
<HTML>
<HEAD>
<TITLE>Таблицы</title>
</head>
<BODY>
<BASEFONT SIZE=3>
<TABLE CELLPADDING=0 CELLSPASING=3 BORDER=3 ALIGN=CENTER WIDTH=70%>
<CAPTION VALIGN="top"><B>Заголовок таблицы</B></caption>
<TR> <TH>Заголовок 1</th>
     <TH>Заголовок 2</th></tr>
<TR> <TD>Ячейка 1</td>
     <TD>Ячейка 2</TD></tr>
<TR> <TD>Ячейка 3</td>
     <TD>Ячейка 4</td></tr>
</table>
</body>
</html>
```

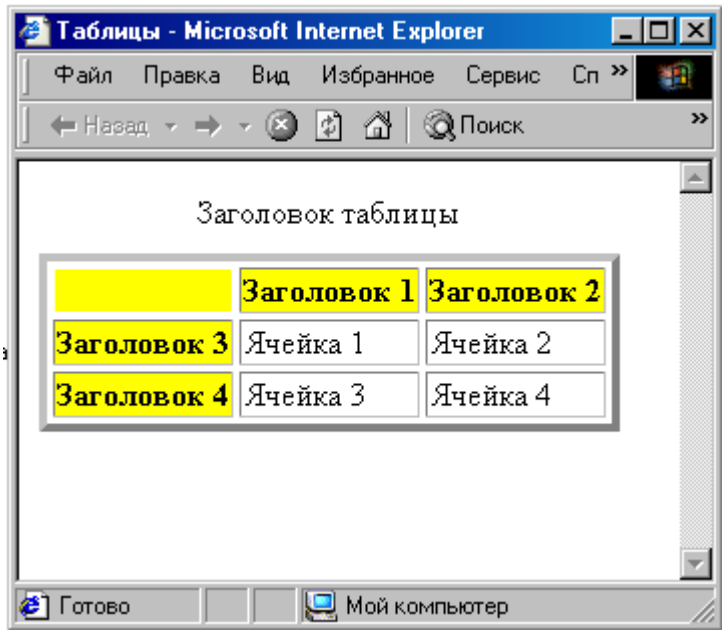


Первая строка таблицы содержит только ячейки-заголовки. Текст, расположенный после элементов <**TD**>, представляет собой содержимое ячейки. Таблица может форматироваться автоматически (если не заданы атрибуты), с учетом объема данных в ячейках.

Последний пример можно несколько усложнить. При необходимости можно создать заголовки и для столбцов, и для строк:

Пример.

```
<HTML>
<HEAD>
<TITLE>Таблицы</title>
</head>
<BODY>
<TABLE border=4 cellspacing=3>
<CAPTION> Заголовок таблицы </caption>
<TR><TH bgcolor="yellow">
<TH bgcolor="yellow">Заголовок 1
<TH bgcolor="yellow"> Заголовок 2
<TR><TH bgcolor ="yellow"> Заголовок 3
<TD> Ячейка 1
<TD> Ячейка 2
<TR><TH bgcolor="yellow"> Заголовок 4
<TD> Ячейка 3
<TD> Ячейка 4
</table>
</body>
</html>
```



Обратите внимание: несмотря на то, что левая верхняя ячейка не используется, для нее задан цвет фона так же, как и для других ячеек-заголовков. Это необходимо сделать для того, чтобы рамка таблицы в этом месте была правильно прорисована.

Основные параметры тегов <TR>, <TH> и <TD>

Параметр align

Если в случае использования этого параметра в теге <TABLE> вся таблица выравнивается определенным образом, то в данном случае назначается тип выравнивания для конкретной ячейки или ряда таблицы. Если необходимо, чтобы каждая ячейка содержала разные типы выравнивания, придется указывать соответствующее значение параметра align в каждом теге <TD> или <TH>. Если требуется задать один тип выравнивания для всего ряда (включающего все ячейки), значение параметра align прописывается в теге <TR>.

Параметр **align** определяет выравнивание текста и графики по горизонтали несколькими способами:

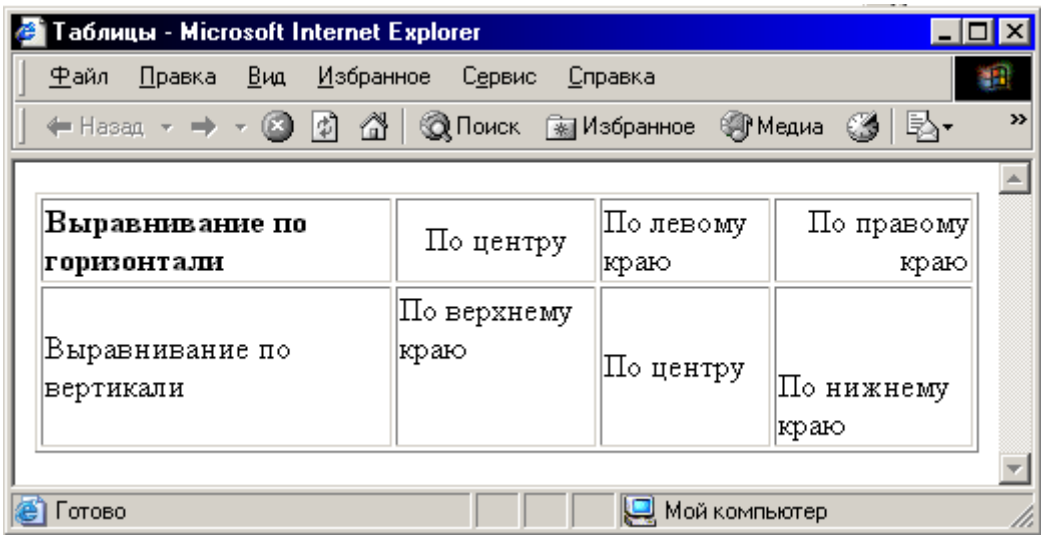
а**)** align = left выравнивает по левому краю с учетом отступа, заданного атрибутом cellpadding;
б) align = center располагает содержимое по центру;
в) align = right выравнивает по правому краю с учетом отступа, заданного атрибутом cellpadding .

Параметр valign

Параметр **valign** осуществляет выравнивание текста и графики по вертикали несколькими способами.
а) valign = top - выравнивает содержимое ячейки по ее верхней границе;
б) valign = middle - центрирует содержимое ячейки по ее вертикали;
в) valign = bottom - выравнивает содержимое ячейки по ее нижней границе.

Пример. Выравнивание текста в таблице

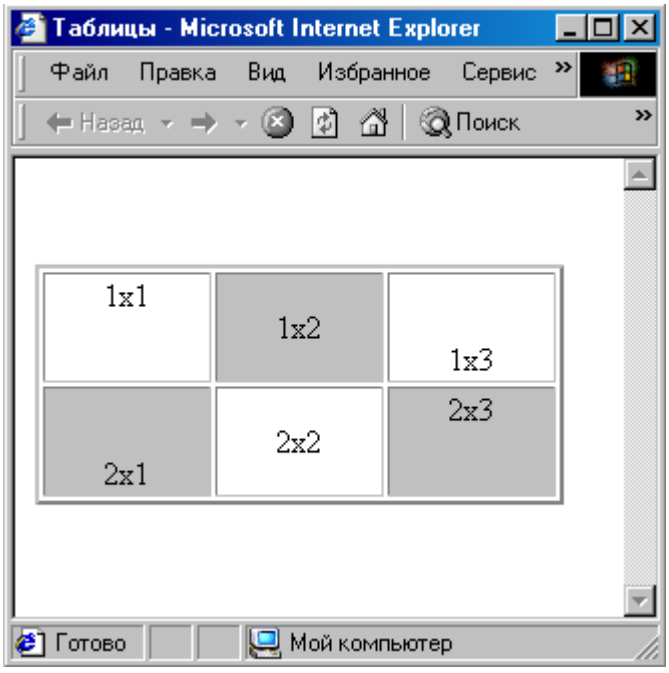
```
<HTML>
<HEAD>
<TITLE>Таблицы</title>
</head>
<BODY>
<TABLE width="100%" border="l" align="center">
<TR>
<TD><B>Выравнивание по горизонтали</b></td>
<TD align="center"> По центру </td>
<TD align="left"> По левому краю </td>
<TD align="right"> По правому краю </td>
 </tr>
<TR>
<TD <B>Выравнивание по вертикали </b></td>
<TD valign="top"> По верхнему краю <BR><BR></td>
<TD valign="middle"> По центру </td>
<TD valign="baseline"> <BR><BR>По нижнему краю </td>
</tr>
</table>
</body>
</html>
```



Выравнивание по горизонтали	По центру	По левому краю	По правому краю
Выравнивание по вертикали	По верхнему краю	По центру	По нижнему краю

Пример.

```
<HTML>
<HEAD>
<TITLE>Таблицы</title>
</head>
<BODY>
<TABLE border=2>
<TR>
     <TD height="55" width="80" valign="top">
     <center>1x1</center> </td>
     <TD width="80" bgcolor="silver"> <center>1x2</center> </td>
     <TD width="80" valign="bottom">
     <center>1x3</center> </td>
</tr>
<TR>
     <TD height="55" width="80" bgcolor="silver" valign="bottom">
     <center>2x1</center> </td>
     <TD width="80" > <center>2x2</center> </td>
     <TD width="80" bgcolor="silver" valign="top">
     <center>2x3</center>
</td>
</tr>
</table>
</body>
</html>
```



Параметры width и height

Параметры width и height отвечают соответственно за ширину и высоту таблицы. Значение этих параметров может быть указано в процентах или пикселах:

`width` = Ширина в пикселях (процентах)

`height` =Высота в пикселях (процентах)

Параметр nowrap

Обычно любой текст, не помещающийся в одну строку ячейки, переходит на следующую строку. Параметр **nowrap** запрещает принудительный перенос строки в ячейке или табличном ряде. При использовании атрибута **nowrap** (он не имеет параметров) с тегами <TH> и <TR> длина ячейки расширяется настолько, чтобы заключенный в ней текст поместился в одну строку. В этом случае в ячейке будет создана одна строка, а таблица может уйти за край окна. Например, если название фирмы ее главный дизайнер хочет записать в одну строку.

Не рекомендуется использовать данный параметр во всех ячейках, т. к. это может сильно понизить уровнь масштабируемости таблицы

Параметр rowspan и colspan

Иногда при построении таблиц возникает необходимость в объединении нескольких соседних ячеек в одну. Атрибуты **rowspan** и **colspan** позволяют объединять ячейки в таблице.

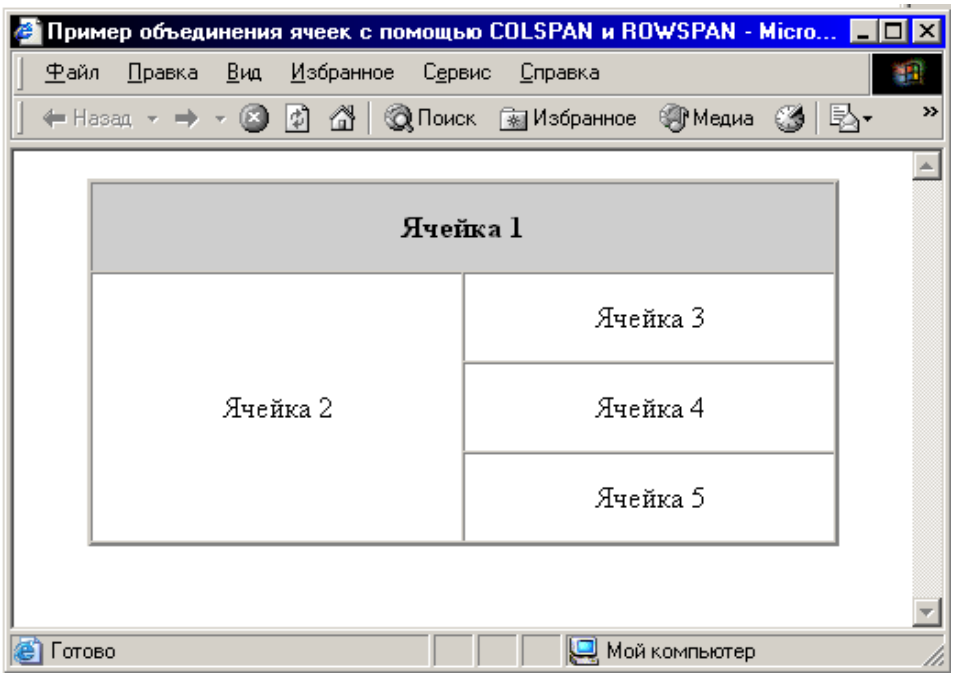
Параметр **rowspan**, используемый в тегах <TH> и <TR> задает число строк, на которые растягивается ячейка. Если указано rowspan = n, то n строк должно находиться ниже. НЕЛЬЗЯ поместить этот атрибут в последней строке таблицы.

При задании атрибута rowspan = n и условии, что $n > 1$, соответствующая ячейка займет не одну, а n строк и, соответственно, будет иметь размер в n раз больший, чем обычная ячейка данного столбца.

Параметр **colspan (Column Span**, соединение столбцов) используется для того, чтобы сделать какую-нибудь ячейку шире, чем верхняя или нижняя и растянуть ячейку над любым количеством обычных ячеек. При помощи атрибута **colspan** можно создавать ячейки, расположенные сразу в нескольких столбцах. Например, атрибут colspan = 2 растянет ячейку на ширину 2 столбцов.

Пример.

```
<HTML>
<HEAD>
<TITLE>Пример объединения ячеек с помощью COLSPAN и ROWSPAN</title>
</head>
<BODY bgcolor="#FFFFFF" text="black"  leftmargin="40"
rightmargin="40" marginwidth="40">
<TABLE align="center"  border="2"  cellspacing="0"  cellpadding="5"
width="100%"  height="200">
<TR align="center" bgcolor="#CECECE">
<TH colspan="2">Ячейка 1</th>
</tr>
<TR>
<TD align="center" rowspan="3">Ячейка 2</td>
<TD align="center">Ячейка 3</td>
</tr>
<TR>
<TD align="center">Ячейка 4</td>
</tr>
<TR>
<TD align="center">Ячейка 5</td>
</tr>
</table>
</body>
</html>
```



Пустые ячейки используются для того, чтобы у ячейки были границы, но не было содержимого. Если нужна рамка, можно воспользоваться пустой строкой **
** или   non-breaking space - неразрывающий пробел. Можно задать ширину ячеек в пикселях или относительных единицах, но не вводить значений.

Перечень параметров тегов <TR>, <TD> и <TH> приведен в таблице

Параметр	**Функция**	**Применение**
align	Выравнивание содержимого ячейки или ряда по горизонтали	<TR>, <TD>, <TH>
valign	Выравнивание содержимого ячейки или ряда по вертикали	<TR>, <TD>, <TH>
width	Определение ширины ячейки или ряда	<TR>, <TD>, <TH>
height	Определение высоты ячейки или ряда	<TR>, <TD>, <TH>
bgcolor	Указание цвета для фона ячейки или ряда	<TR>, <TD>, <TH>
background	Указание рисунка для фона ячейки	<TD>, <TH>
nowrap	Запрет принудительного переноса строки в ячейке или ряду	<TR>, <TD>, <TH>
colspan	Объединение соседних ячеек по горизонтали	<TD>, <TH>
rowspan	Объединение соседних ячеек по вертикали	<TD>, <TH>

Группировка данных

При построении таблиц мы можем легко задать единый тип выравнивания для отдельной ячейки и даже целого табличного ряда. Однако гораздо, чаще может возникнуть необходимость в едином выравнивании информационных данных конкретного столбца. Столбец в таблице – это последовательность ячеек, располагающихся в разных рядах.

Стандартными средствами HTML нам пришлось бы задавать один и тот же тип выравнивания для отдельно взятой ячейки, формирующей нужный столбец:

```
<TABLE>
<TR>
<TD align="right">Ячейка 1 с выравниванием вправо</td>
<TD align="center">Ячейка 1 с выравниванием по центру</td>
</tr>
<TR>
<TD align="right">Ячейка 2 с выравниванием вправо</td>
<TD align="center">H4euKa 2 с выравниванием по центру</td>
</tr>
</table>
```

При работе с браузером Internet Explorer, труд разработчика электронного документа может быть значительно облегчен за счет таких тегов, как <col> и <colgroup>.

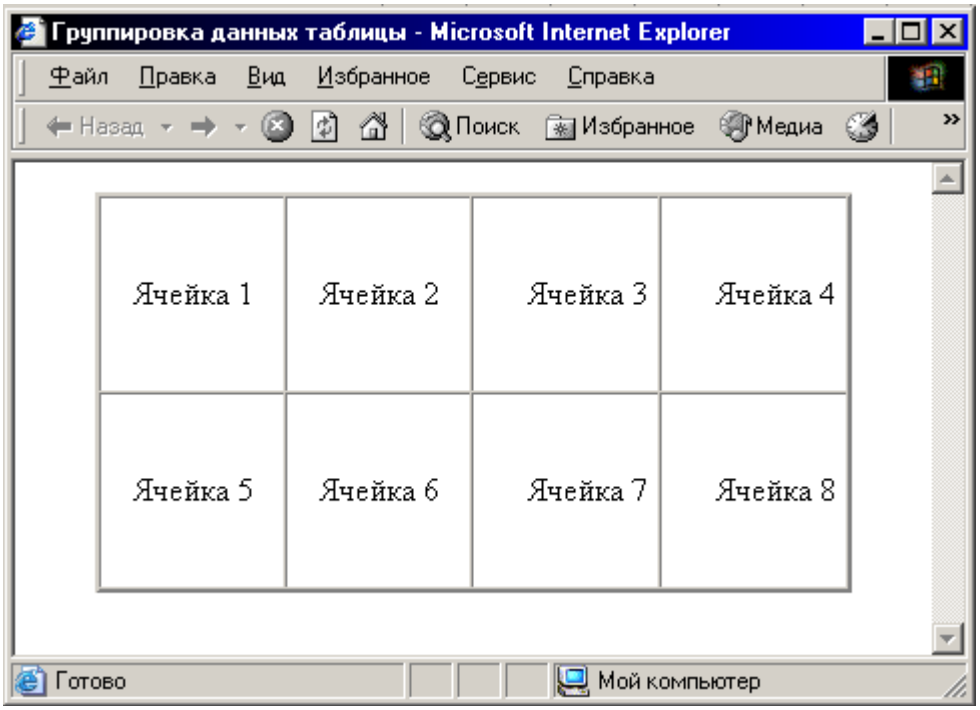
Теги <col> и <colgroup> предназначены для определения свойств отображения табличных данных, сгруппированных по конкретному признаку. Они оба могут иметь параметр span, задающий количество соседних столбцов, и параметр align, станавливающий единый тип выравнивания для выбранного столбца (возможные значения: по левому краю, по правому краю, по центру; формат записи аналогичен тегам <TD> и <TH>).

Помимо этого тег <colgroup> может содержать дополнительный параметр вертикального выравнивания данных – valign (возможные значения: по верхнему краю, по нижнему краю, по середине; формат записи аналогичен тегам <TD> и <TH>).

Разница между тегами <col> и <colgroup> заключается в условном объединении (группировке) взаимосвязанных данных отдельно взятого столбца таблицы.

Пример.

```
<HTML>
<HEAD>
<TITLE>Группировка данных таблицы</title>
</head>
<BODY BGCOLOR="#FFFFFF" TEXT="black"  LEFTMARGIN="40"
RIGHTMARGIN="40" MARGINWIDTH="40">
<TABLE align="center" border="2" cellspacing="0"
cellpadding="5" width="100%" height="200">
<COLGROUP align="center"  span="2">
<COLGROUP align="right"  span="2">
<TR>
<TD>Ячейка 1</td>
<TD>Ячейка 2</td>
<TD>Ячейка 3</td>
<TD>Ячейка 4</td>
</tr>
<TR>
<TD>Ячейка 5</td >
<TD>Ячейка 6</td>
<TD>Ячейка 7</td>
<TD>Ячейка 8</td>
</tr>
</table>
</body>
</html>
```



Колонтитулы таблицы

Браузер Internet Explorer позволяет использовать дополнительные теги структурирования табличных данных, а именно – <thead>, <tbody> и <tfoot>. Эти теги предназначены для создания колонтитулов таблицы различных уровней (соответственно для верхнего, основного (содержательного) и нижнего уровней таблицы).

Теги верхнего и нижнего колонтитулов <thead> и <tfoot> могут быть использованы в структуре таблицы лишь единожды, причем для них необязательно ншшчие закрывающих тегов.

Тег основного колонтитула <tbody> может встречаться неоднократно в пределах одной таблицы, однако требует своего закрывающего тега. Верхний и нижний колонтитулы функционально очерчивают логические заголовки соответствующего уровня и применимы, в основном, в больших таблицах, не помещающихся в пределах одной страницы электронного документа.

Основные колонтитулы выполняют функцию, аналогичную тегам группировки <соь> и <colgroup>, расставляя логические метки по ходу изложения основной содержательной части таблицы.

Прорисовка структуры таблицы

Еще одной замечательной возможностью нестандартного представления таблиц, работающей только в браузере Internet Explorer, является прорисовка внутренней структуры таблицы, а именно – рамок и линеек. Другими словами, возможно оформить таблицу таким образом, что от рамки останутся только верхняя и нижняя границы, а между ячейками останется только вертикальная линейка.

За изменение свойств рамки отвечает параметр **frame**, а с помощью параметра **rules** варьируется внешний вид линеек таблицы (оба параметра применимы внутри тега <TABLE>).

Возможные значения параметра frame приведены в таблице.

значение	**Функция**
border	Рамка с четырех сторон
above	Рамка только сверху

значение	Функция
below	Рамка только снизу
hsides	Верхняя и нижняя части рамки
vsides	Левая и правая части рамки
lhs	Только левая часть рамки
rhs	Только правая часть рамки
void	**Нет рамок**

Возможные значения параметра rules приведены в таблице.

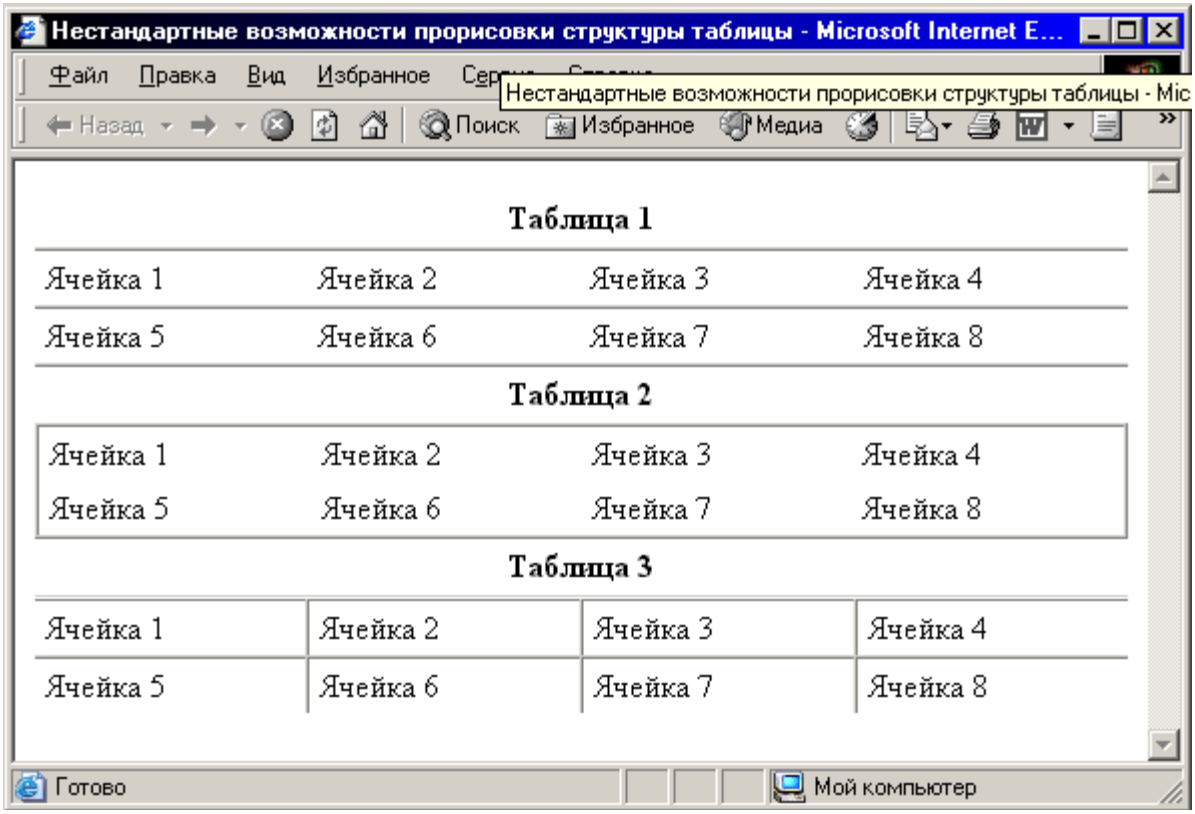
Значение	Функция
all	Отображение линейки целиком
groups	Часть линейки, разделяющая сгруппированные данные
cols	Часть линейки, разделяющая столбцы
rows	Часть линейки, разделяющая строки
none	Отсутствие линейки

Пример.

```
<HTML>
<HEAD>
<TITLE>Нестандартные возможности прорисовки структуры
таблицы</title>
</head>
<BODY BGCOLOR="#FFFFFF" TEXT="black">
<TABLE ALIGN="center" BORDER="1"  CELLSPACING="0"
CELLPADDING="4" WIDTH="100%"  FRAME="hsides" RULES="rows">
<CAPTION><B>Таблица 1</B></caption>
<TR>
    <TD>Ячейка 1</td>
    <TD>Ячейка 2</td>
    <TD>Ячейка 3</td>
    <TD>Ячейка 4</td>
</tr>
<TR>
    <TD>Ячейка 5</td>
    <TD>Ячейка 6</td>
    <TD>Ячейка 7</td>
    <TD>Ячейка 8</td>
</tr>
</table>

<TABLE ALIGN="center"  BORDER="1"  CELLSPACING="0"  CELLPADDING="4"
WIDTH="100%"'FRAME="box" RULES="groups">
<CAPTION><B>Таблица 2</B></caption>
<TR>
    <TD>Ячейка 1</td>
    <TD>Ячейка 2</td>
    <TD>Ячейка 3</td>
    <TD>Ячейка 4</td>
</tr>
<TR>
    <TD>Ячейка 5</td>
```

```
    <TD>Ячейка 6</td>
    <TD>Ячейка 7</td>
    <TD>Ячейка 8</td>
</tr>
</table>
<TABLE ALIGN="center" BORDER="1" CELLSPACING="0" CELLPADDING="4"
WIDTH="100%" FRAME="above" RULES="all">
<CAPTION><B>Таблица 3</B></caption>
<TR>
    <TD>Ячейка 1</td>
    <TD>Ячейка 2</td>
    <TD>Ячейка 3</td>
    <TD>Ячейка 4</td>
</tr>
<TR>
    <TD>Ячейка 5</td>
    <TD>Ячейка 6</td>
    <TD>Ячейка 7</td>
    <TD>Ячейка 8</td>
</tr>
</table>
</body>
</html>
```



Вложенные таблицы

Одной из замечательных особенностей HTML-таблиц по праву считается поддержка многоуровневой вложенности. Другими словами, одна таблица может включать другую, та, в свою очередь, еще одну и т. д.

Преимущества вложенных таблиц

Особенность вложенных таблиц, в отличие от других способов представления данных в электронном документе, позволяет более точно размещать отдельные элементы страницы относительно друг друга и относительно границ самого документа, отображаемого браузером.

Например, два разнородных блока текста и нумерованный список, размещенные внутри тега <BODY>, невозможно разместить на одном уровне, а тем более на одном уровне со смещением в какую-либо сторону. Использование таблиц с легкостью решает эту проблему, позволяя располагать различные элементы и их комбинации в разных местах документа посредством видимых и невидимых ячеек рядов таблицы.

Вот почему в последнее время преобладающее большинство HTMЬ-документов создается на основе таблиц, где в качестве несущей основы берется таблица с невидимыми краями, содержащая вложенные таблицы с разным оформлением, отличающимися начениями параметров.

Подводя итог сказанному, можно выделить следующие преимущества вложенных таблиц:

- гибкая масштабируемость структуры электронного документа в целом;
- широкие возможности позиционирования отдельных элементов страницы;
- многоуровневое представление разнородных информационных данных;
- расширенные оформительские возможности;
- поддержка популярными браузерами.

Правила построения вложенных таблиц ничем не отличаются от создания таблиц одного уровня – используются те же теги и параметры, задаются те же свойства и значения.

Единственное, о чем нельзя забывать в ходе создания сложных вложенных таблиц, это:

- каждая таблица последующего уровня размещается внутри тега-контейнера <TD> или <TH> таблицы предыдущего уровня;
- вложенная таблица не может быть создана за пределами вышеназванных тегов ячейки таблицы;
- таблица одного уровня может содержать любое количество вложенных таблиц другого уровня, идущих друг за другом в пределах тега ячейки таблицы верхнего уровня;
- количество тегов таблиц всех уровней должно соответствовать количеству закрывающих тегов этих же таблиц.

Задания к лабораторной работе № 7.
Таблицы HTML-документов

Задание 1. Создание таблицы по образцу.

- В элементе <TITLE> укажите название странички "Задание 1. Создание таблицы по образцу"
- Создайте таблицу, как показано на рис.

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

- Текст в первой строке выровняйте по центру по горизонтали и по вертикали.
- Текст во второй строке выделите жирным шрифтом.

Задание 2. Создание таблицы по образцу.

- В элементе <TITLE> укажите название странички "Задание 2. Создание таблицы по образцу"
- Создайте HTML-документ, соответствующий следующему образцу:

Примеры таблиц

Стандартная таблица

Заголовок таблицы

Заголовок 1	Заголовок 2
Ячейка 1	Ячейка 2
Ячейка 3	Ячейка 4

Объединение ячеек

Заголовок 1	Заголовок 2	Заголовок 3
Ячейка 1		
		Ячейка 2
		Ячейка 3
		Ячейка 4
Ячейка 5		

Вложенная таблица

Ячейка 1.1.1 / **Ячейка 1.1.2** / **Ячейка 1.1.3** / **Ячейка 1.1.4**	***Ячейка 3.1.1*** / ***Ячейка 3.1.2*** / ***Ячейка 3.3.1***
	Ячейка 3.2.1 / Ячейка 3.2.2 / Ячейка 3.2.3
Ячейка 2.1.1 / **Ячейка 2.1.2**	Ячейка 4.1.1 / Ячейка 4.1.2
Ячейка 2.2.1 / Ячейка 2.2.2	***Ячейка 4.2.1*** / ***Ячейка 4.2.2***
	Ячейка 4.3.1 / Ячейка 4.3.2

Табличка

Петровым 2 звонка

Задание 3. Форматирование таблицы

- Для выполнения этого задания воспользуйтесь результатами выполнения Задания 1.
- В элементе <TITLE> укажите название странички "Задание 3. Форматирование таблицы".
- Сделайте фон первого столбца зеленого цвета, второго – красного, третьего – синего.
- Задайте ширину бордюра таблицы, равным 3 пикселям.
- Ширину таблицы сделайте равной ширине экрана.

- Шрифт, которым написан текст внутри таблицы, сделайте белым.

Задание 4. Заполнение таблицы

- Для выполнения этого задания воспользуйтесь результатами выполнения Задания 1.
- В элементе <TITLE> укажите название странички "Задание 4. Заполнение таблицы".
- В первую ячейку первой строки вставьте рисунок из файла *.jpg.
- Во второй ячейке второй строки напишите ваше любимое четверостишие.
- Сделайте фон третьей ячейки второй строки красного цвета.

Задание 5. Самостоятельное проектирование и создание таблицы

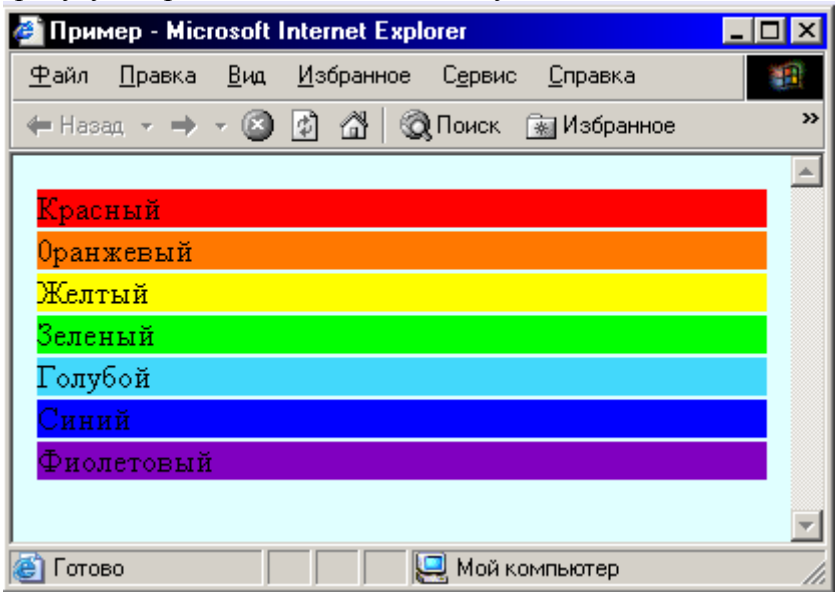
- Составьте таблицу для расписания ваших занятий. Пусть эта таблица состоит из восьми столбцов и девяти строк. Ширина таблицы 100%.
- В первом столбце укажите время начала и конца ваших занятий. Выравнивание данных в ячейках должно быть по левому краю.
- В остальных столбцах укажите дисциплины по дням недели.
- Выравнивание дней недели по центру ячейки и жирным шрифтом.
- Выравнивание названий дисциплин в ячейках должно быть по левому краю.
- У всех столбцов фон сделайте разным цветом.
- Перед таблицей поместите заголовок "РАСПИСАНИЕ ЗАНЯТИЙ", выделив его тэгами <H1> и </h1>. Цвет заголовка красный.
- Между заголовком и таблицей поместите рисунок.

Задание 6.

Составьте справочную таблицу “Цветовая гамма HTML-документа” размером 5*3. Заполните ячейки таблицы записями кодов различных оттенков красного (строка 1), зеленого (строка 2) и синего (строка 3). Фон ячеек закрасьте теми же цветами, коды которых записаны в них.

Задание 7.

Составьте радугу, закрасив ячейки таблицы нужным цветом.



Задание 8.

Составьте выписку из своей зачетной книжки, оформив ее в виде таблицы.

Задание 9.

Изучите примеры и создайте WEB-страницу, на которой посередине разместите таблицу следующего вида (ширина таблицы=90%)

Наиболее популярные модели струйных принтеров				
Изготовитель, название модели	**Формат печати**	**Количество цветов**	**Макс. разрешение (dpi)**	**Макс. скорость печати (стр./мин.)**
Hewlett Packard				
HP DeskJet 690C+	A4	6	600x600	5
HP DeskJet 970Cxi		4	2400x1200	12
Epson				
Epson Stylus Color 440	A4	4	720x720	4
Epson Stylus Color 900			1440x720	12

Контрольные вопросы

1. Основные методы создания таблиц.
2. Параметры, которые используются при создании таблиц в HTML-документах.
3. Группировка данных
4. Колонтитулы таблицы
5. Прорисовка структуры таблицы
6. Вложенные таблицы
7. Создание цветных таблиц.

Лабораторная работа № 8.
Формы

Цель: познакомиться с разными видами пользовательских форм.

Пользовательская форма – это совокупность стандартных HTML-конструкций ввода текстовой и прочей информации и программы-обработчика этой информации, работающей на Web-сервере. Иными словами, пользовательская форма (или HTML-форма) служит для передачи информационных данных серверу.

Результат конструкций языка разметки HTML интерпретируется браузером, с помощью которого пользователь электронного документа получает информацию. Таким образом, объединив все эти формулировки, можно сказать, что HTML-форма выступает в роли посредника между пользователем и сервером.

Посетитель Web-страницы вводит в HTML-форму определенные данные, которые обрабатываются программой и отсылаются на Web-сервер. Все эти действия укладываются в три стадии:

1. Ввод пользователем информации.
2. Обработка введенной информации программой, установленной на сервере.
3. Получение результата отправления введенной информации на Web-сервер (открытие нового HTML-документа, переадресация на предыдущую страницу и пр.).

В качестве программы-обработчика чаще всего выступает CGI-сценарий (скрипт, который обычно разрабатывается на языке Perl или C/C++ и который взаимодействует со специальным компонентом Web-сервера -Common Gateway Interface) или программы, написанные на основе таких серверных языков программирования, как PHP, ASP, JSP и др.

Значение пользовательских форм трудно переоценить – они являются особым средством HTML, дающим посетителю возможность не только пассивно просматривать информацию, но и быть задействованным в содержании Web-сайта. Такое свойство принято называть интерактивностью, которая на сегодняшний день встречается практически во всех электронных документах.

Диапазон функций, выполняемых HTML-формами, чрезвычайно широк, поэтому остановимся лишь на некоторых, наиболее распространенных направлениях.

1. Организация обратной связи

Форма обратной связи – это неотъемлемый атрибут Web-сайта любой тематики и функциональной направленности. Размещая электронный документ во Всемирной сети, мы уже подразумеваем реакцию пользовательской аудитории – похвалу и одобрение или же несогласие и критику.

В качестве средства обратной связи между пользователем и автором электронного документа могут выступать следующие.

1. *Почтовая форма*. Посетителя просят указать свои персональные данные и изложить суть обращения. Текст обращения обрабатывается программой и высылается автору сайта по электронной почте. При этом другие посетители сайта не могут получить доступ к содержанию отправленного сообщения;
2. *Гостевая книга*. Электронная книга жалоб и предложений, в которой любой желающий может оставить свой отзыв, поделиться впечатлениями о сайте, изложить критические замечания и указать на ошибки автора. В этом случае сообщения одного посетителя видимы для остальных пользователей гостевой книги;
3. *Интернет-форум*. Более мощный инструмент стимулирования оценки пользовательской аудитории, нежели почтовая форма или гостевая книга. В интернет-форумах происходит многосторонний диалог, причем не только между человеком, задающим вопрос, и автором Web-сайта, но и между всеми пользователями данного интернет-ресурса. На вопрос одного пользователя может ответить другой, третий подскажет то, чего не упомянул второй, наконец, автор сайта подытожит высказывания всех пользователей и даст свой развернутый ответ. Часто в рамках интернет-форума работают дополнительные функциональные средства общения: личные сообщения, скрытые разделы форума для привилегированных пользователей, опросы общественного мнения и пр.;
4. *Чат*. Средство чата (от англ, "to chat" — разговаривать, болтать) подразумевает единовременное общение многих пользователей в режиме реального времени. В отличие от гостевой книги или форума, где ответ на поставленный вопрос может быть выслан и через час, и через день, общение в чате происходит между двумя (или более) собеседниками так, будто они сидят в одной комнате. Собеседники могут находиться в разных городах, государствах и временных зонах, однако, присутствуя в чате, все пользователи способны общаться одновременно. В чатах также могут присутствовать некоторые дополнительные функции: приваты (отдельный разговор между двумя пользователями), система почтовых сообщений, голосования и т. д.

2. Авторизация

Для получения доступа к информации на многих интернет-ресурсах существует необходимость регистрации в сервисе. После регистрации, которая подразумевает указание

уникального пользовательского имени и пароля, пользователь должен пройти процедуру авторизации, т. е. проверки на наличие в базе данных сайта заявленного имени и пароля.

Система авторизации распространена на информационных ресурсах (новостные ленты, афиши событий), корпоративных серверах (закрытые разделы для клиентов, партнеров, дистрибьюторов), сайтах электронной почты и т, д.

3. Проведение исследований

Для изучения пользовательского спроса на тот или иной вид услуг, выявления и последующего анализа реакции посетителей, на многих сайтах устраиваются всевозможные опросы, анкетирования, голосования, тесты и пр.

Набор стандартных HTML-конструкций облегчает участие пользователя в подобных исследованиях. Все, что ему потребуется, это отметить подходящие варианты ответа и подтвердить участие в опросе.

4. Пользовательский профиль

Регистрируясь на каком-либо интернет-ресурсе, человек указывает всякого рода информацию: имя, город и страну проживания, увлечения, место работы, контактные данные и пр. Совокупность таких данных составляет пользовательский профиль, который впоследствии возможно будет изменить путем редактирования соответствующих полей HTML-формы.

Структура пользовательских форм

Структура любой пользовательской формы описывается тегом-контейнером <FORM>, внутри которого могут располагаться теги и параметры составляющих элементов HTML-формы. В пределах тега <FORM> не может содержаться другая пользовательская форма. Закрывающий тег рбязателен для указания. Тег <FORM> может содержать четыре основных параметра: ACTION, METHOD, NAME и ENCTYPE. Рассмотрим каждый из них.

<FORM> </form>

Этот элемент необходим для построения сложных форм. Все формы начинаются тегом <FORM> и завершаются тегом </form>. После заполнения формы и подтверждения ввода со стороны пользователя введенная информация пересылается на сервер и обрабатывается при помощи CGI-программы, связанной с формой.

Параметр ACTION

Параметр **ACTION** является единственным обязательным параметром тега <FORM> и предназначен для указания пути на Web-сервере к программе обработчику данных пользовательской формы, например:

```
<FORM ACTION = "http://www.название.домен/имя программы">
```

Путь к программе-обработчику может быть как относительный, так и абсолютный.

Значение этого параметра играет важную роль с точки зрения работоспособности HTML-формы. Если путь или название программы указаны неверно или же указанный файл не является исполняемым на стороне сервера, то обработка введенных пользователем данных может быть нарушена, произведена некорректно или вовсе не быть осуществленной.

Наиболее распространенными типами исполняемых на стороне сервера файлами являются CGI-программы (с расширениями pi, cgi, fcgi), PHP-скрипты (php, php3, phtm, phtml), ASP-приложения (asp), JSP-обработчики (jsp) и др.

Одним из вариантов обработки формы может быть пересылка данных по электронной почте:

```
<FORM ACTION ="mailto:Адрес@название.домен">
```

Параметр METHOD

Параметр **METHOD** применяется для указания протокола, используемого для пересылки данных на сервер. Протокол GET выбран по умолчанию, но в большинстве случаев он не удовлетворяет разработчиков, поэтому чаще используется протокол POST.

При использовании типа GET данные пользовательской формы пересылаются в составе адреса запроса браузера: после имени программы-обработчика ставится знак вопроса (?), обозначающий вывод запроса браузера к переменным HTML-формы, а также последовательность переменных и их свойств из самой формы. Последовательность переменных формы разделяется символом амперсанда "&".

Пример адреса запроса браузера с использованием типа передачи данных GET:

```
http://www.site.ru/ogi-bin/form.cgi?Name=Vasya&Email=vasya@vasya.ru
```

Из структуры ссылки, образовавшейся в ходе обработки данных формы, понятно, что пользователь ввел свое имя ("Vasya") и адрес электронной почты ("vasya@vasya.ru").

При использовании типа POST данные формы не отображаются в адресной строке браузера, а передаются в составе самого запроса профаммы-обработчйка. Таким образом, используя этот тип, мы получим следующую гиперссылку (учитывая те же данные формы, что и в случае с типом GET):

```
http://www.site.ru/cgi-bin/form.cgi
```

Следует отметить, что пользовательская форма может включать информацию о пользователе, носящую конфиденциальный характер (например, пароли доступа), которая будет отображена в запросе браузера при использовании типа GET. В этом случае вся секретная информация, введенная пользователем, будет доступна для просмотра любому пользователю данного компьютера (просмотр истории перехода по Web-сайтам в браузере легко обнаружит ссылку с конфиденциальными данными).

Но в конечном итоге выбор типа передачи данных на Web-сервер определяется конкретной задачей, стоящей перед автором HTML-формы, особенностями сервера и самой профаммы-обработчйка.

Параметр NAME

Параметр NAME присваивает HTML-форме уникальное имя, которое используется в программе-обработчике для идентификации пользовательских данных, например:

```
<FORM NAME="mail">
```

Параметр ENCTYPE

Параметр **ENCTYPE** предназначен для определения типа данных при кодировании информации, введенной пользователем, и последующей ее передаче на Web-сервер. Кодирование осуществляется браузером и используется для предотвращения разного рода искажений в процессе передачи на сервер.

Возможными значениями параметра могут быть: appiication/x-www-form-urlencoded (по умолчанию) и multipart/form-data.

Первое значение используется, если помимо текста необходимо передать на сервер данные иного типа (к примеру, фафику или запакованные файлы). Формат записи состоит из указания типа и его подтипа. Тип данных – это определение общего типа данных (текст, графика, архив, программа и т. д.), например, text, image, application. Подтип – это вид денных внутри определенного общего типа (image/gif, text/html).

Значение multipart/form-data используется в редких специфических случаях, например, при необходимости предоставить пользователю возможность зафузки на сервер любого файла со своего локального компьютера.

Пример.

```
<FORM METHOD="get | post" ACTION="URL">
Элементы_формы_и_другие_элементы_HTML </form>
```

<INPUT>

Этот элемент позволяет создавать различные элементы управления в форме:

- ♦ текстовую строку;
- ♦ поле ввода пароля;
- ♦ поле выбора локального файла для загрузки на Web-сервер;
- ♦ опцию выбора (флажки);
- ♦ опцию переключения;
- ♦ кнопку отправления пользовательских данных;
- ♦ графический вариант кнопки отправления пользовательских данных;
- ♦ кнопку сброса введенных пользовательских данных.

Элемент не имеет конечного тега, так как все параметры задаются при помощи атрибутов.

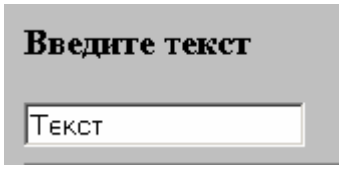
Атрибут **type**.позволяет указывать один из перечисленных элементов формы, каждому из которых соответствует конкретное значение.

type ="text" – это элемент текстовой строки.

Параметры текстовой строки

Параметр	Описание
size	Определение максимального количества символов, вводимых в текстовой строке без перемещения курсора.
maxlength	Определение максимального количества символов, допустимых для ввода в текстовой строке. По умолчанию количество вводимых символов не ограничено.
name	Присвоение идентификационного имени для программы-обработчика.
value	Указание значения текстовой строки (при просмотре в браузере выводится в виде обычного текста в самой строке).

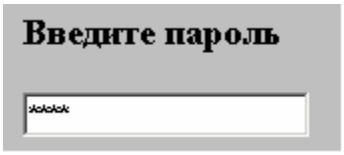
```
<H3> Введите текст </h3>
<INPUT type ="text"  name="S001" value="Текст">
```



type="password" – это элемент ввода пользовательского пароля. Ничем не отличается от обыкновенной текстовой строки, за исключением того, что введенная информация отображается звездочками. Такая мера связана с сохранение конфиденциальности пользовательских данных, но данные, вводимые в это поле, при использовании типа передачи get будут отображаться в ссылке запроса браузера.

<H3> Введите пароль </h3>

<INPUT type ="password" name="S001" value="один">



type="checkbox" – это элмент опции выбора, устанавливающий (снимающий) флажок для конкретного поля пользовательской формы.

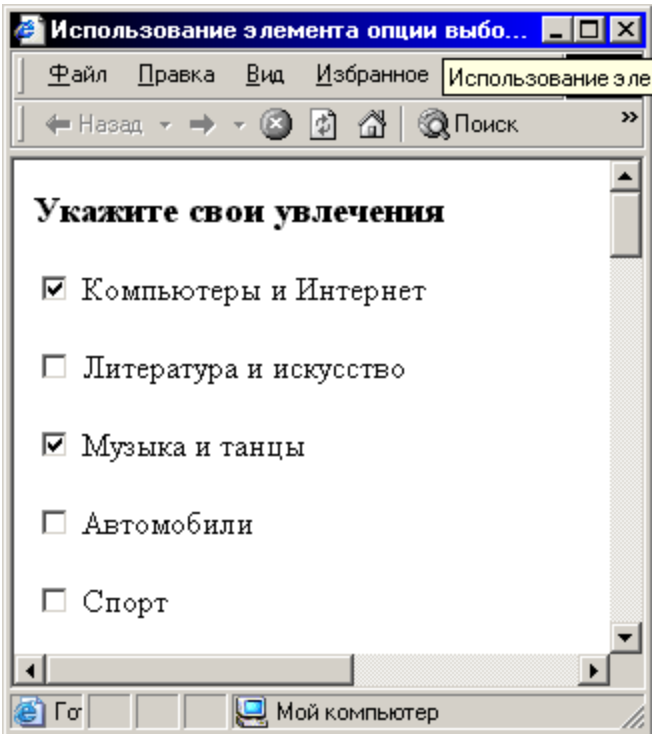
Этот элемент используется для простых логических (BOOLEAN) значений. Значение, ассоциированное с именем данного поля, которое будет передаваться в вызываемую CGI-программу, может принимать значение ON или OFF.

Параметры опции выбора

Параметр	Описание
name	Указание общего для всех вариантов выбора идентификационного имени.
value	Определение значения для конкретного варианта выбора (обязательный параметр). Не должен повторяться, так как при установке флажка передается на Web-сервер.
checked	Данный вариант является выбранным по умолчанию.

Пример.

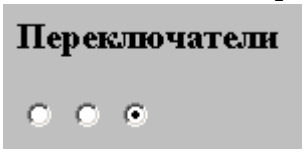
```
<HTML>
<HEAD>
<TITLE> Использование элемента опции выбор </title>
</head>
<BODY>
<H3> Укажите свои увлечения</h3>
<FORM name="select_hobby" action="/cgi-bin/hobby.cgi" metod="get">
<INPUT   type="checkbox"  name="hobby"  value="computers"  checked>
Компьютеры и Интернет
<BR> <BR>
<INPUT  type="checkbox"  name="hobby"  value="art"> Литература  и
искусство
<BR> <BR>
<INPUT type="checkbox"  name="hobby" value="music" checked> Музыка
и танцы
<BR> <BR>
<INPUT type="checkbox"  name="hobby" value="avto"> Автомобили
<BR> <BR>
<INPUT type="checkbox"  name="hobby" value="sport"> Спорт
</form>
</body>
</html>
```



type="radio" – это элемент опции переключения между различными вариантами выбора. В отличии от checkbox, вариант выбора может быть только один. Имеет дополнительные параметры name, value, checked. Для создания группы переключателей необходимо использовать несколько элементов INPUT.

Пример.

```
<H3> Переключатели </h3>
<INPUT type="radio" name="S001" value="Первый">
<INPUT type="radio" name="S001" value="Второй">
<INPUT type="radio" name="S001" value="Третий" checked>
```

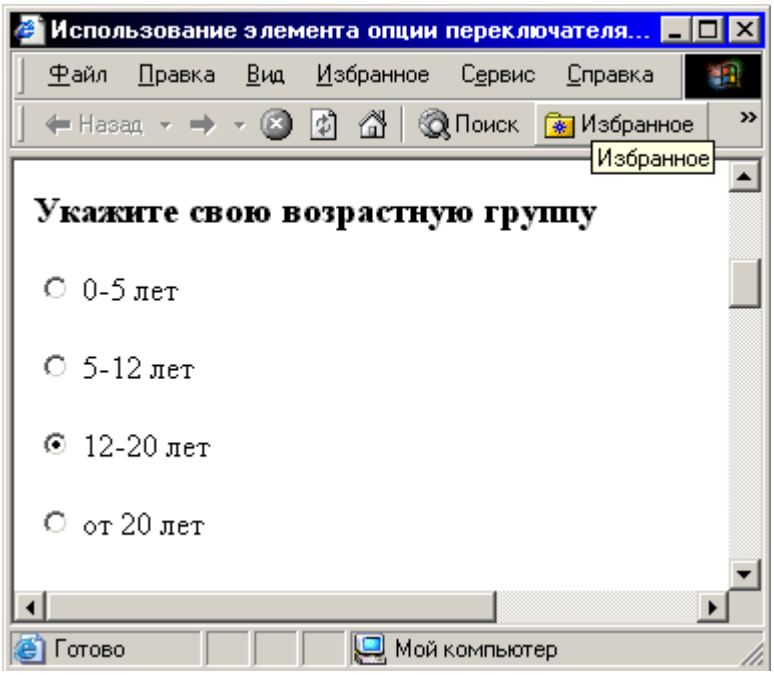


Атрибут **checked** определяет, какой из переключателей должен быть выбран по умолчанию.

Пример.

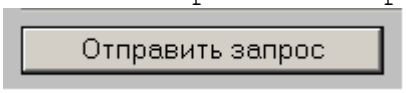
```
<HTML>
<HEAD>
<TITLE> Использование элемента опции переключателя</title>
</head>
<BODY bgcolor="silver">
<H3> Укажите свою возрастную группу</h3>
<FORM name="select_age" action="/cgi-bin/hobby.cgi" metod="get">
<INPUT  type="radio" name="age" value="baby" checked> 0-5 лет
<BR> <BR>
<INPUT type="radio" name="age" value="child"> 5-12 лет
<BR> <BR>
```

```
<INPUT type="radio"  name="age" value="junior" checked> 12-20 лет
<BR> <BR>
<INPUT type="radio"  name="age" value="adult"> от 20 лет
</form>
</body>
</html>
```



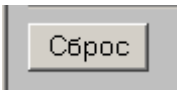
type="submit" – это кнопка отправления пользовательских данных на Web-сервер. При нажатии на нее запускается программа-обработчик, которая анализирует введенные пользовтелем данные и отправляет результат на ервер. Атрибут value используется для определения надписи на кнопке. По умолчанию используется надпись «Подача запроса».

```
<INPUT type ="submit"  value="Отправить запрос">
```



type="reset" – это кнопка сброса введенных пользователем данных HTML-формы. При нажатии на нее восстанавливаются все установленные по умолчанию значения полей формы. Атрибут value используется для определения надписи на кнопке. По умолчанию используется надпись «Сброс».

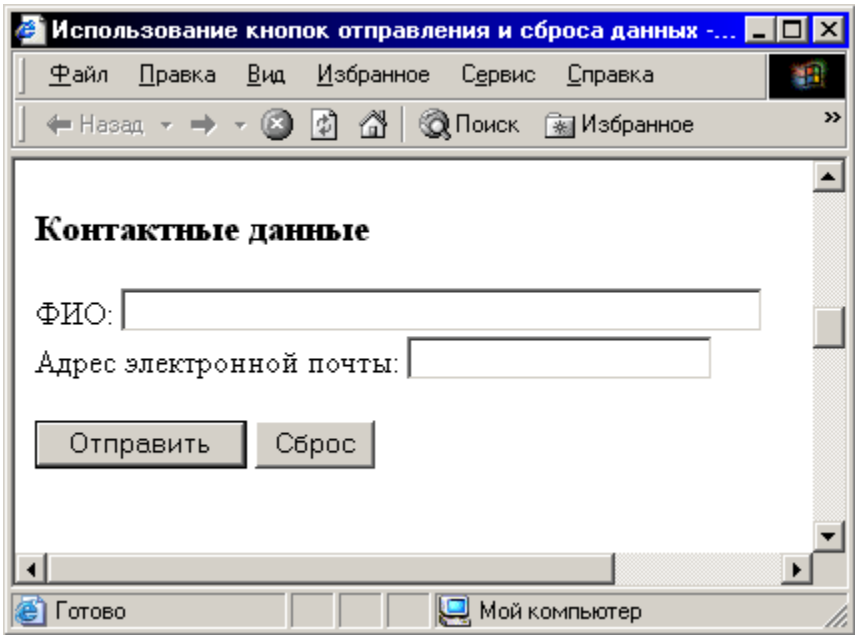
```
<INPUT type="reset">
```



Пример. Использование кнопок отправления и сброса данных.

```
<HTML>
<HEAD>
<TITLE> Использование кнопок отправления и сброса данных</title>
</head>
<BODY bgcolor="silver">
<H3>  Контактные данные</h3>
<FORM>
ФИО: <INPUT NAME ="name" SIZE="44">
```

```
<BR>
Адрес электронной почты: <INPUT NAME ="email" SIZE =:"40">
</FORM>
<INPUT type ="submit"  value="Отправить">
<INPUT type="reset" >
</body>
</html>
```



type="image" – то графический аналог стандартной кнопки отправления данных формы на Web-сервер. (кнопка с рисунком).

Параметры кнопки с рисунком

Параметр	Описание
src	Указание поти к фаулу графического изображения, служащего кнопкой передачи данных формы
align	Указание типа выравнивания текста относительно графической кнопки отправления данных формы
border	Определение толщины рамки кнопки (как правило, значение равно нулю)
alt	Указание альтернативного текста/подсказки для кнопки отправления данных

Атрибуты необходимы для определения свойств элемента. Многие из них являются обязательными, так как обеспечивают обработку данных формы на стороне сервера.

Атрибут **name** должен присутствовать во всех элементах INPUT, кроме кнопок подтверждения и сброса.

```
name = имя поля ввода
```

Его значение определяет имя поля формы, то есть блока данных, введенных в это поле. Программа сервера, используя это имя, может выделить необходимые данные.

Область применения атрибута **value** нам уже известна. Значение атрибута присваивает полю значение по умолчанию или значение, которое будет выбрано при использовании типа RADIO (для типа RADIO данный атрибут обязателен) или определяет надпись на кнопке.

Атрибут **checked** используется при создании группы переключателей, этот атрибут может быть использован и для создания флажков. Его наличие показывает, что флажок

должен быть установлен по умолчанию. В отличие от переключателей, любое количество флажков может быть установлено одновременно.

Атрибут **size** позволяет задать длину поля ввода. Длина выражается в символах, но эта величина может быть задана только приближенно. Для того чтобы разместить в поле ввода определенное количество символов, потребуется подбор значения атрибута. При этом никто не может дать гарантию, что все броузеры обеспечат требуемую длину строки, а не меньшую. Таким образом, длину поля ввода необходимо выбирать с запасом.

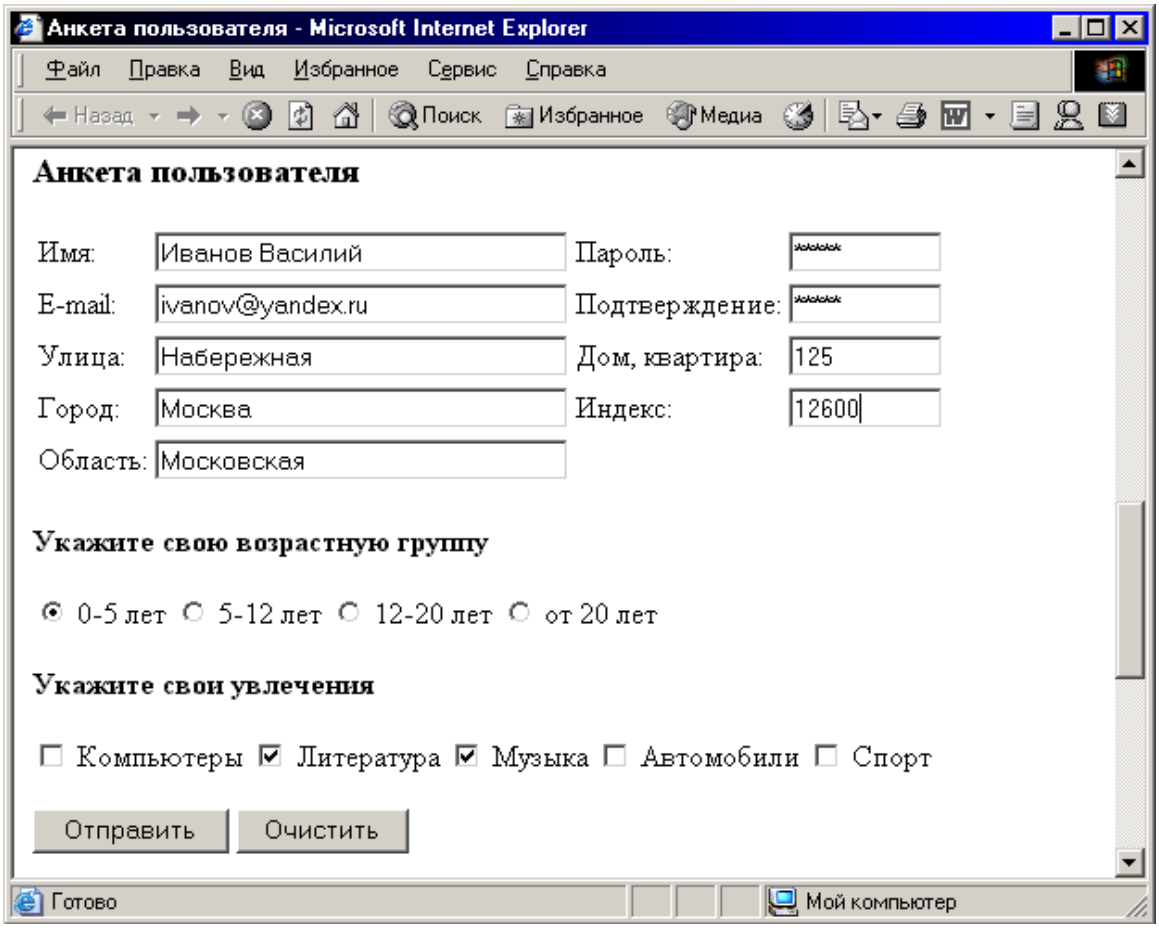
Атрибут **maxlenght** – определяет количество символов, которое пользователи могут ввести в поле ввода. При превышении количества допустимых символов броузер реагирует на попытку ввода нового символа звуковым сигналом и не дает его ввести. Не путать с атрибутом size. Если maxlength больше чем size, то в поле осуществляется скроллинг. По умолчанию значение maxlength равно бесконечности. При помощи этого атрибута можно ограничить размер файла, присоединяемого к форме.

Пример. Использование таблиц при использовании форм.

```
<HTML>
<HEAD>
<TITLE> Анкета пользователя </title>
</head>
<BODY bgcolor="silver">
<FORM>
<H3>  Анкета пользователя</h3>
<TABLE>
<TR> <TD> Имя: </td>
     <TD> <INPUT TYPE = "name"
     NAME = "name"  SIZE = "30"></td>
     <TD> Пароль: </td>
     <TD> <INPUT TYPE = "password"
     NAME = "name"  SIZE = "10"></tr>
<TR> <TD> E-mail: </td>
     <TD><INPUT TYPE = "name"
     NAME = "name"  SIZE = "30"></td></td>
     <TD> Подтверждение: </TD>
     <TD><INPUT TYPE = "password"
     NAME = "name"  SIZE = "10"></td></tr>
<TR><TD> Улица: </td>
     <TD><INPUT TYPE = "name"
     NAME = "street1"  SIZE = "30"></td>
     <TD> Дом, квартира:</td>
     <TD> <INPUT TYPE = "name"
     NAME = "street2"  SIZE = "10"></td></tr>
<TR><TD> Город: </td>
     <TD><INPUT TYPE = "text"
     NAME = "sity" SIZE = "30"></td>
     <TD> Индекс: </td>
     <TD><INPUT TYPE = "text"
     NAME = "zip" SIZE = "10"></td></tr>
<TR><TD> Область: </td>
     <TD><INPUT TYPE = "text"
     NAME = "state" SIZE = "30"></td></tr>
</table>
```

```
<H4> Укажите свою возрастную группу</h4>
<FORM name = "select_age" action = "/cgi-bin/hobby.cgi" metod =
"get">
<INPUT type="radio" name="age" value="baby" checked>0-5 лет
<INPUT type="radio" name="age" value="child"> 5-12 лет
<INPUT type="radio" name="age" value="junior"> 12-20 лет
<INPUT type="radio" name="age" value="adult"> от 20 лет
</form>

<H4> Укажите свои увлечения</h4>
<FORM name="select_hobby" action="/cgi-bin/hobby.cgi" metod="get">
<INPUT type="checkbox" name="hobby" value="computers"> Компьютеры
<INPUT type="checkbox" name="hobby" value="art"> Литература
<INPUT type="checkbox"  name="hobby" value="music"> Музыка
<INPUT type="checkbox"  name="hobby" value="avto"> Автомобили
<INPUT type="checkbox"  name="hobby" value="sport"> Спорт
</form>
<INPUT type ="submit" value="Отправить">
<INPUT type="reset"  value="Очистить">
</form>
</body>
</html>
```



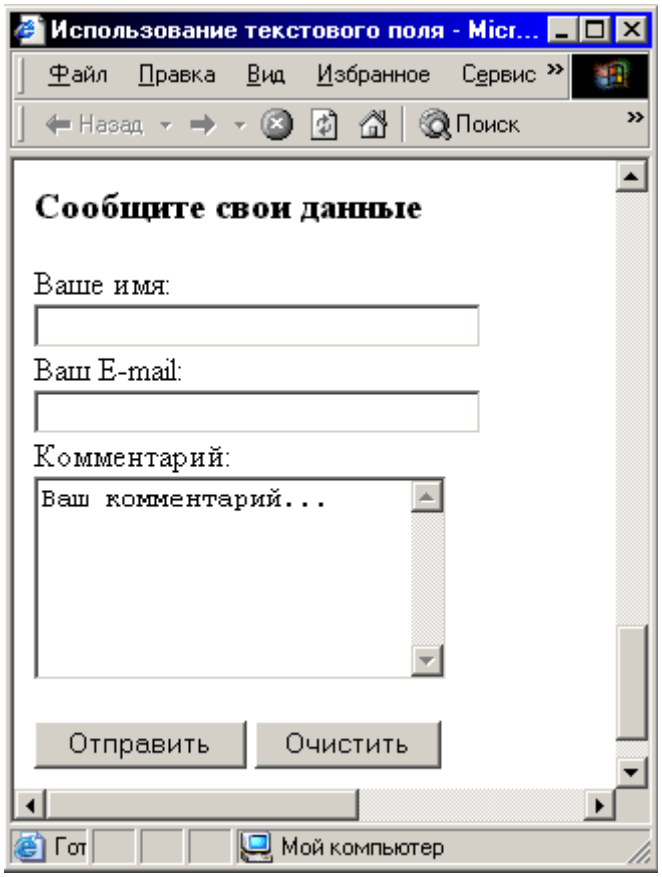
<TEXTAREA> </textarea>

При помощи этого элемента создается область заданной ширины и высоты для ввода или просмотра текста.

Параметры текстового поля

Параметр	Описание
cols	Определение количества столбцов текстового поля
rows	Определение количества строк текстового поля
name	Присвоение уникального имени, необходимого для идентификации программой обработчиком

```
<HTML>
<HEAD>
<TITLE> Использование текстового поля </title>
</head>
<BODY bgcolor="silver">
<H3>  Сообщите свои данные</h3>
<FORM>
Ваше имя:
<BR>
<INPUT name ="name" SIZE="30">
<BR> Ваш E-mail:
<BR>
<INPUT name ="e-mail" SIZE="30">
<BR> Комментарий:
<BR>
<TEXTAREA   name="comment"   rows=6   cols=23>Ваш   комментарий...
</textarea>
</form>
<INPUT type ="submit"  value="Отправить">
<INPUT type="reset" value="Очистить">
</body>
</html>
```



<SELECT> <OPTION> </select>

Элемент **<SELECT>** предназначен для компактной группировки большого количества элементов пользовательской формы. Такой вид формы может быть представлен ниспадающим меню или списком наименований.

Параметры

Параметр	Описание
name	Уникальное имя, предназначенное для идентификации программой обработчиком. Является обязательным параметром, значение которого передается на Web-сервер.
size	Параметр, значение которого определяет число позиций ниспадающего меню, состоящего из списка наименований. Когда size не использован, список выглядит обычным образом: вначале видна только первая строка, а позиционирование на ней указателя мыши раскрывает список. Если значение атрибута задано, то список не раскрывается, а прокручивается, причем пользователь видит только указанное количество строк.
multiple	Параметр, разрешающий выбор нескольких позиций из списка наименований

Элемент **OPTION** предназначен для создания пункта списка.

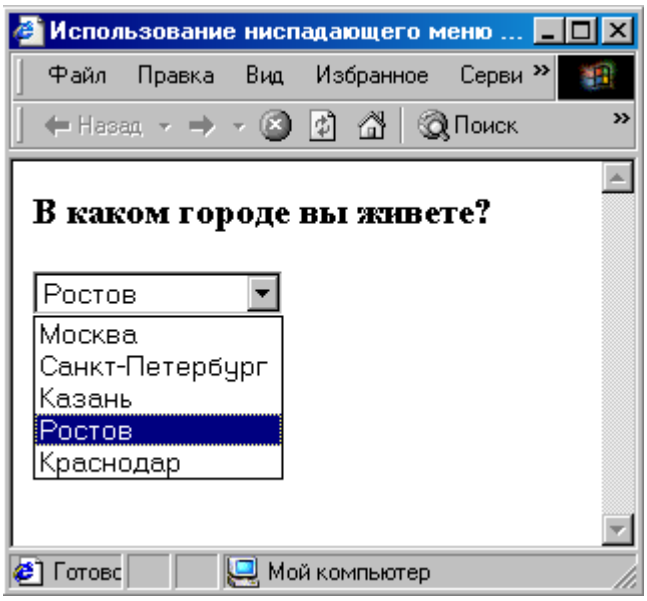
Параметры текстового поля

Параметр	Описание
value	Параметр, значение которого передается программой-обработчиком на Web-сервер.
selected	Параметр отмечает текущую позицию ниспадающего меню или списка как выбранную. Таким образом можно сделать визуальный акцент на любой по

Параметр	Описание
	счету позиции меню или списка.

Пример.

```
<HTML>
<HEAD>
<TITLE>Использование ниспадающего меню</TITLE>
</HEAD>
<BODY>
<H3> В каком городе вы живете?
<BR> <BR>
<SELECT>
<OPTION selected=2 value=a> Москва
<OPTION value=b> Санкт-Петербург
<OPTION value=c> Казань
<OPTION value=d> Ростов
<OPTION value=d> Краснодар
</select></h3>
</HTML>
```



В результате мы получим список. Оба элемента, создающие список, имеют собственные атрибуты.

Элементы **SELECT** и **TEXTAREA** могут быть использованы не в составе формы, а как самостоятельные детали Web-страницы. Их применение оправдано в тех случаях, когда надо увеличить плотность размещения данных. При помощи элемента SELECT можно создавать списки, которые получаются более компактными, чем стандартные списки, рассмотренные ранее. Область ввода текста также поможет сэкономить место на странице за счет того, что сколь угодно большой текст будет прокручиваться в окне фиксированного размера.

Задания к лабораторной работе № 6
Формы

№	Общие задания
1.	Создайте HTML – форму "Личная", в которой пользователь будет задавать свое имя, фамилию, адрес, интересы (может быть несколько), username, password.

№	Общие задания
	Предусмотреть текстовое поле для комментария из нескольких строк. Следует учитывать следующие моменты компоновки и дизайна HTML – форм выбор даты, месяца и года рождения должен производиться различными способами. Предусмотрите в данной форме гиперссылку для перехода на другой Web – документ.
2.	В новом HTML – документе разработайте HTML-форму, в которой на примере своей зачетной книжке (дисциплина и оценка), покажите использование различных видов списков для компоновки форм, использование флажков и переключателей.

№	Вариант задания
1.	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта автомобилей. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.
2.	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта алкогольной продукции. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.
3.	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта сигарет. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit. Сформировать HTML – форму, в которой выбор информации осуществляется из различных видов списков.
4.	Создать опрос общественного мнения, ориентированный на исследование рынка сбыта компьютерной техники. Выполнить в виде формы, предусмотреть создание как минимум 4 страниц, кнопок Clear и Submit

Контрольные вопросы

1. Перечислите основные теги для создания HTML–форм.
2. Тег (дескриптор) FORM.
3. Ввод информации в HTML–форму. Тег INPUT.
4. Создание паролей в HTML–формах.
5. Осуществление выбора информации с помощью флажков и переключателей.
6. Назначение кнопки RESET.
7. Назначение кнопки SUBMIT.
8. Выбор информации в HTML–форме. Основные теги и их характеристика.
9. Характеристика дескриптора TEXTAREA.
10. Какой тег следует указывать при поиске из нескольких массивов информации?
11. Если форма, проектируемая пользователем достаточно велика, то какой способ используют для размещения информации?
12. Какой тег используется для формирования горизонтальной строки?
13. Какой тег используют для перевода строки?
14. Что улучшает совместное использование в Web документе таблиц и форм?
15. Какие элементы управления позволяют передать данные с компьютера клиента на сервер?
16. Напишите пример одновременного использования HTML-форм, таблиц и кнопки передачи запроса.

Задания к работе.

1.Создайте страницу Form.htm следующего вида.

Лабораторная работа № 9.
Фреймы

Цель работы: познакомится с основными приемами работы с фреймами в HTML-документах.

Фреймы в HTML позволяют авторам представлять документы в нескольких разделах, которые могут быть независимыми или вложенными окнами. Это обеспечивает дизайнерам способ оставлять некоторую информацию видимой, в то время как другая информация прокручивается или заменяется. Например, в одном окне в одном фрейме может отображаться статический баннер, во втором навигационное меню, а в третьем - сам документ, который можно прокручивать или переходить к другому с помощью навигации во втором фрейме.

Используя фреймы, можно значительно улучшить внешний вид и функциональность информационных систем и Web-приложений. Каждое подокно, или фрейм, может иметь следующие свойства:

- Каждый фрейм имеет свой URL, что позволяет загружать его независимо от других фреймов
- Каждый фрейм имеет собственное имя (параметр NAME), позволяющее переходить к нему из другого фрейма
- Размер фрейма может быть изменен пользователем прямо на экране при помощи мыши (если это не запрещено указанием специального параметра)

Данные свойства фреймов позволяют создавать продвинутые интерфейсные решения, такие как:

- Размещение статической информации, которую автор считает необходимым постоянно показывать пользователю, в одном статическом фрейме. Это может быть графический логотип фирмы, copyright, набор управляющих кнопок
- Помещение в статическом фрейме оглавления всех или части WEB-документов, содержащихся на WEB-сервере, что позволяет пользователю быстро находить интересующую его информацию
- Создавать окна результатов запросов, когда в одном фрейме находится собственно запрос, а в другом результаты запроса

Синтаксис фреймов

Формат документа, использующего фреймы, внешне очень напоминает формат обычного документа. Вместо тэга BODY используется контейнер FRAMESET, содержащий описание внутренних HTML-документов, в которых находится собственно информация, размещаемая во фреймах.

```
<HTML>
<HEAD>...</head>
<FRAMESET>...</frameset>
</html>
```

Однако, фрейм-документ является специфичным видом HTML-документа, поскольку не содержит элемента BODY и какой-либо информационной нагрузки соответственно. Он описывает только фреймы, которые будут содержать информацию (кроме случая двойного документа).

Атрибуты:

- **border**=толщина рамки (по умолчанию 1)
- **bordercolor**=цвет рамки
- **framespasing**=отступ от края
- **rows="**высота строк**"**

Данный тэг содержит описания некоторого количества подокон, разделенные запятыми. Каждое описание представляет собой числовое значение размера подокна в пикселях, процентах от всего размера окна или связанное масштабное значение. Количество подокон определяется количеством значений в списке. Общая сумма высот подокон должна составлять высоту всего окна (в любых измеряемых величинах). Отсутствие атрибута ROWS определяет один фрейм, величиной во все окно браузера.

Синтаксис используемых видов описания величин подокон:

высота строк в виде числа – (ROWS=10)

Простое числовое значение определяет фиксированную высоту подокна в пикселях. Это далеко не самый лучший способ описания высоты подокна, поскольку различные браузеры имеют различный размер рабочего поля, не говоря уже о различных экранных разрешениях у пользователя. Если вы, все же, используете данный способ описания размера, то настоятельно рекомендуется сочетать его с каким-либо другим, чтобы в результате вы точно получили 100%-ное заполнение окна браузера вашего пользователя.

высота строк в виде процентного отношения (%) – (ROWS=15%)

Значение величины подокна в процентах от 1 до 100. Если общая сумма процентов описываемых подокон превышает 100, то размеры всех фреймов пропорционально уменьшаются до суммы 100%. Если, соответственно, сумма меньше 100, то размеры пропорционально увеличиваются.

высота строк*

Символ "*" указывает на то, что все оставшееся место будет принадлежать данному фрейму. Если указывается два или более фрейма с описанием "*" (например "*,*"), то оставшееся пространство делится поровну между этими фреймами. Если перед звездочкой стоит цифра, то она указывает пропорцию для данного фрейма (во сколько раз од будет больше аналогично описанного чистой звездочкой). Например, описание "3*,*,*", говорит, что будет создано три фрейма с размерами 3/5 свободного пространства для первого фрейма и по 1/5 для двух других.

COLS="ширина колонок"

(ширина колонок может быть задана в виде числа (COLS=10) или в виде процентного отношения (COLS=15%), знак умножения (*) – означает, что фрейм занимает все оставшиеся колонки независимо от изменения ширины окна браузера)

Примеры:

```
<FRAMESET COLS=“60,60,60,60”>
...продолжение определения...
</FRAMESET>
```

– описывает 4 вертикальных фрейма шириной по 60 пикселей. Задавать в пикселях ширину всех рамок не рекомендуется, так как разрешение мониторов различное, и рамки на разных мониторах будут выглядеть по-разному. Размер одной из рамок рекомендуется указывать символом “*”.

```
<FRAMESET COLS="50,*,50">
...продолжение определения...
</FRAMESET>
```

– описывает три вертикальных фрейма, два по 50 точек справа и слева, и один внутри этих полосок.

```
<FRAMESET ROWS="20%,3*,*">
...продолжение определения...
</FRAMESET>
```

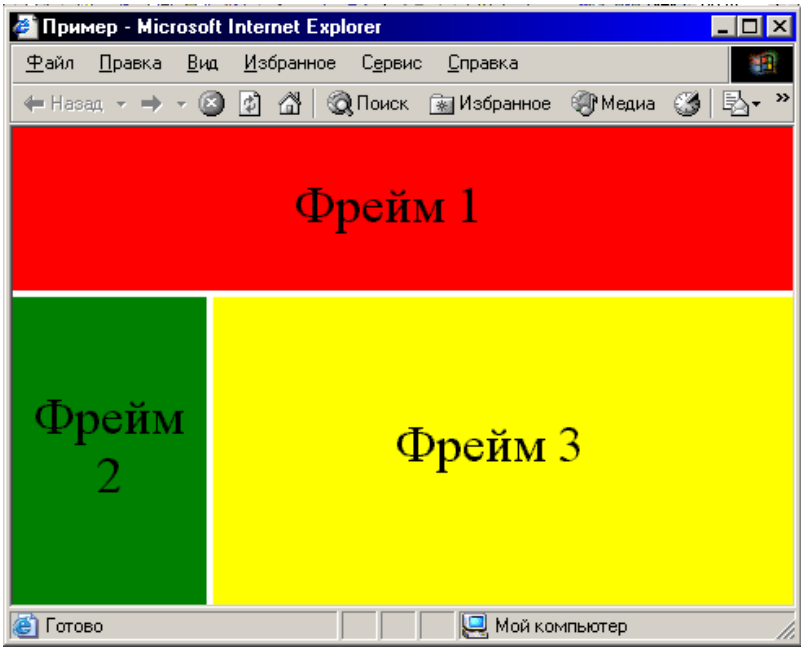
– описывает три горизонтальных фрейма, первый из которых занимает 20% площади сверху экрана, второй 3/4 оставшегося от первого фрейма места (т.е. 60% всей площади окна), а последний 1/4 (т.е. 20% всей площади окна.

```
<FRAMESET ROWS="*,60%,*">
...продолжение определения...
</FRAMESET>
```

– аналогично предыдущему примеру.

Пример.

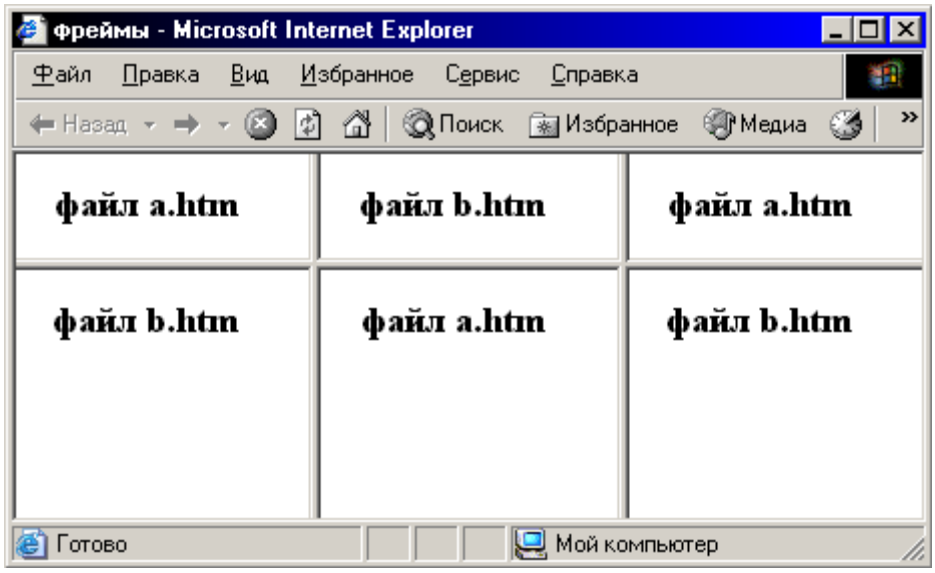
```
<HTML>
<HEAD>
<TITLE>Пример</title>
<META name="Author" content="Denis">
</head>
<FRAMESET BORDER=1 FRAMESPASING=0 ROWS="100,*">
  <FRAME NAME="Frame1" SRC="L7Frame1.htm" SCROLLING="Auto">
  <FRAMESET BORDER=1 FRAMESPASING=0 COLS="25%,*">
    <FRAME NAME="Frame2" SRC="L7Frame2.htm" SCROLLING="Auto">
    <FRAME NAME="Frame3" SRC="L7Frame3.htm" SCROLLING="Auto">
  </frameset>
</frameset>
</html>
```



Пример.

```
<HTML>
<HEAD>
<TITLE>фреймы </title>
</head>
<FRAMESET rows="30%,70%" COLS = "33%,34%,33%">
    <FRAME SRC= "a.htm">
    <FRAME SRC="b.htm">
    <FRAME SRC= "a.htm">
    <FRAME SRC="b.htm">
    <FRAME SRC= "a.htm">
    <FRAME SRC="b.htm">
</frameset>
</html>
```

– создается сетка 2x3.



<NOFRAME> позволяет строить двойные документы для браузеров, поддерживающих фреймы и не поддерживающих фреймы. Выводит альтернативный текст, если фреймы не поддерживаются браузером.

<NOFRAMES> текст HTML **</NOFRAMES>**

Представим общий синтаксис фреймов:

```
<FRAMESET COLS=" ширина колонок" | ROWS=" высота строк">
     <FRAME src="url1">
     <FRAME ...>
     ...
</frameset>
```

Тэг **FRAME** описывает каждый фрейм в отдельности. Между тегами **<FRAMESET>** и **</FRAMESET>** не должно быть никаких других тегов или атрибутов, кроме <FRAME>, <FRAMESET> и <NO FRAMES>.

FRAME

```
<FRAME     src="ur1"     [NAME="frame_name"]     [MARGINWIDTH="nw"]
[MARGINHEIGHT="nh"] [SCROLLING=yes|no|auto] [NORESIZE]>
```

Данный тэг определяет фрейм внутри контейнера FRAMESET.

src="url"

Описывает URL документа, который будет отображен внутри данного фрейма. Если он отсутствует, то будет отображен пустой фрейм.

name="frame_name"

Данный параметр описывает имя фрейма. Атрибут **NAME** используется для того, чтобы указать, в каком именно кадре нужно загружать страницу, на которую сделана ссылка. Кадровая структура позволяет делать ссылки в одном кадре, а загружать страницы в другой кадр. Имя обязательно должно начинаться с символа.

Если для разработки информационной системы использовать два вертикальных кадра, например, узкий слева для оглавления (индекса ключевых слов), а правый большой кадр для того, чтобы в нем появлялись выбранные страницы, то этот большой кадр можно назвать “MAIN”.

Пример:

```
<FRAME src=“a.htm” name=“main”>
```

Содержимое поименованных фреймов может быть задействовано из других документов при помощи специального атрибута TARGET.

MARGINWIDTH="value"

Это атрибут может быть использован, если автор документа хочет указать горизонтальный отступ между содержимым кадра и его границами. (Этот атрибут аналогичен атрибуту таблиц CELLPADDING). Значение value указывается в пикселях и не может быть меньше единицы. Нельзя указать 0. Если ничего не присваивать, то по умолчанию атрибут=6.

Пример:

```
<FRAME MARGINWIDTH=“30” src=“c.htm”>
```

MARGINHEIGHT="value"

Атрибут **MARGINHEIGHT** = используется для того, чтобы указать вертикальный отступ между содержимым кадра и его границами.

SCROLLING="yes | no | auto"

Этот атрибут позволяет задавать наличие полос прокрутки у фрейма.

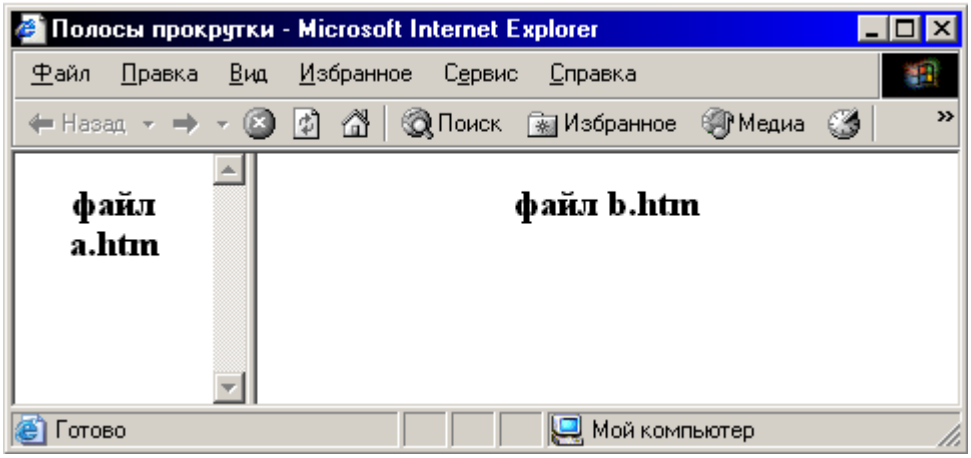
При задании **SCROLLING = NO** полос прокрутки не будет, даже если они необходимы.

При задании **SCROLLING = YES** в кадре всегда будет полоса прокрутки.

При задании **SCROLLING = AUTO** предоставляется возможность браузеру самому решать, нужна ли полоса прокрутки (значение по умолчанию).

Пример:

```
<HTML>
<HEAD>
<TITLE>Полосы прокрутки </title>
</head>
<FRAMESET COLS = “25%, 75%”>
<FRAME SCROLLING = YES SRC= “a.htm”>
<FRAME SCROLLING = NO SRC=“b.htm”>
</frameset>
</html>
```



NORESIZE

Данный атрибут позволяет создавать фреймы без возможности изменения размеров. По умолчанию, размер фрейма можно изменить при помощи мыши так же просто, как и размер окна Windows. NORESIZE отменяет данную возможность. Если у одного фрейма установлен атрибут NORESIZE, то у соседних фреймов тоже не может быть изменен размер со стороны данного.

Планирование фреймов и взаимодействия между фреймами

С появлением фреймов сразу возникает вопрос: "А как сделать так, чтобы нажимая на ссылку в одном фрейме инициировать появление информации в другом?"

Ответом на данный вопрос является планирование взаимодействия фреймов (далее – планирование). Каждый фрейм может иметь собственное имя, определяемое параметром NAME при описании данного фрейма. Существует, также, специальный атрибут – TARGET, позволяющий определять, к какому фрейму относится та или иная операция. Формат данного атрибута следующий:

```
TARGET="windows_name"
```

Данный атрибут может встречаться внутри различных тэгов:

TARGET в тэге A

Это самое прямое использование TARGET. Обычно, при активизации пользователем ссылки соответствующий документ появляется в том же окне (или фрейме), что и исходный, в котором была ссылка. Добавление атрибута TARGET позволяет произвести вывод документа в другой фрейм. Например:

```
<A href="mydoc.php" TARGET="Frame1"> Переход в фрейм №1 </a>
```

TARGET в тэге BASE

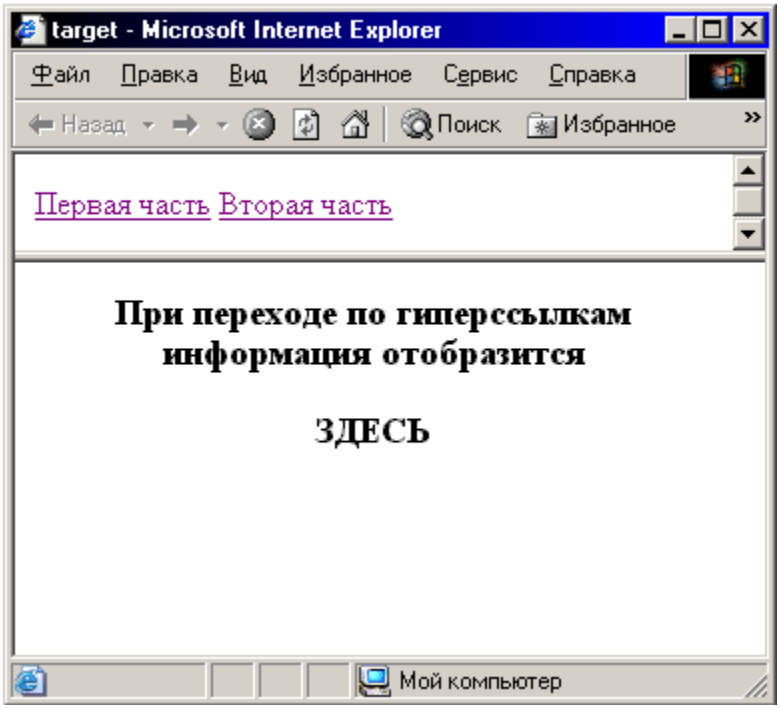
Размещение TARGET в тэге BASE позволит вам не указывать при описании каждой ссылки фрейм-приемник документов, вызываемых по ссылкам. Это очень удобно, если в одном фрейме у вас находится меню, а в другой - выводится информация. Например:

Документ № 1.

```
<FRAMESET ROWS="20,*">
<FRAME SRC="doc2.php" name="Frame1">
<FRAME SRC="doc3.php" name="Frame2">
</frameset>
```

Документ № 2 (doc2.php).

```
<HTML>
<HEAD>
<BASE TARGET="Frame2">
</head>
<BODY>
<A HREF="url1"> Первая часть</a>
<A HREF="url2"> Вторая часть</a>
</body>
</html>
```



TARGET в тэге AREA

Также можно включать тэг TARGET в описание ссылки при создании карты изображения.

Пример:

```
<AREA SHAPE="circle" COORDS="100,100,50"
REF="http://www.softexpress.com" TARGET="Frame1">
```

TARGET в тэге FORM

То же относится и к определению формы. В данном случае, после обработки переданных параметров формы результирующий документ появится в указанном фрейме.

```
<FORM action="url" TARGET="window_name">
```

Внимание! Имя окна (фрейма) в параметре TARGET должно начинаться с латинской буквы или цифры. Также необходимо помнить, что существуют зарезервированные имена для разрешения специальных ситуаций.

Зарезервированные имена фреймов

Зарезервированные имена фреймов служат для разрешения специальных ситуаций. Все они начинаются со знака подчеркивания. Любые другие имена фреймов, начинающиеся с подчеркивания, будут игнорироваться браузером.

TARGET="_blank"

Данное значение определяет, что документ, полученный по ссылке, будет отображаться в новом окне браузера.

TARGET="_self"

Данное значение определяет, что документ, полученный по ссылке, будет отображаться в том же фрейме, в котором находится ссылка. Это имя удобно для переопределения окна назначения, указанного ранее в тэге **BASE**.

TARGET="_parent"

Данное значение определяет, что документ, полученный по ссылке, будет отображаться в родительском окне, вне зависимости от параметров **FRAMESET**. Если родительского окна нет, то данное имя аналогично "_self".

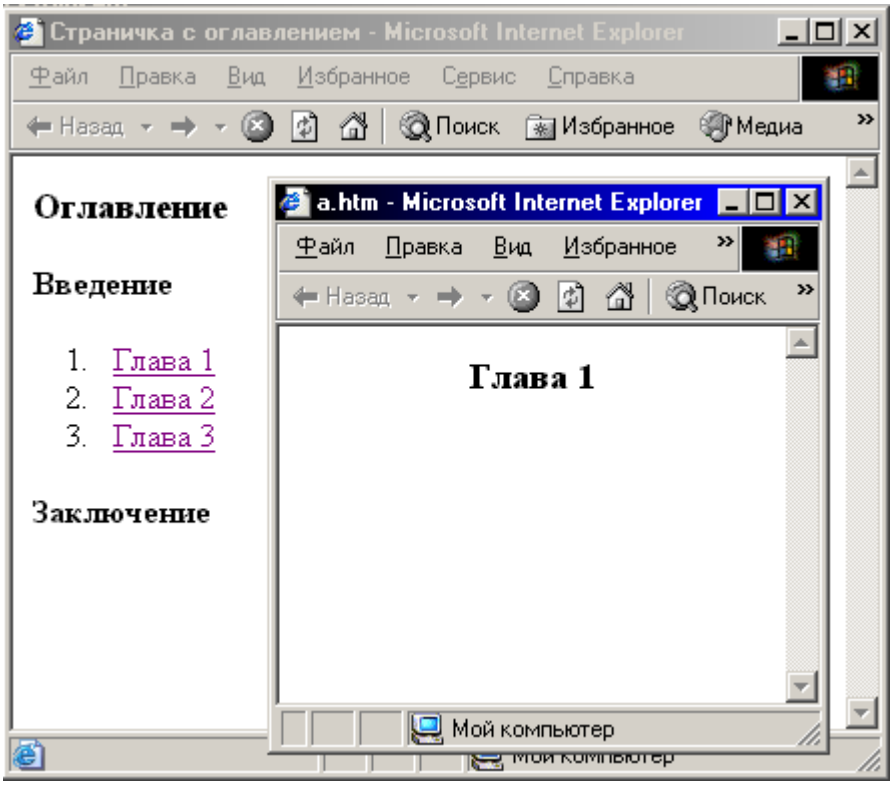
TARGET="_top"

Данное значение определяет, что документ, полученный по ссылке, будет отображаться на всей поверхности окна, вне зависимости от наличия фреймов. Использование данного параметра удобно в случае вложенных фреймов.

Атрибут **TARGET =** используем для составления оглавления:

Пример.

```
<HTML>
<HEAD>
<TITLE>Страничка с оглавлением</title>
</head>
<BODY>
<H3> Оглавление </h3>
<H4>Введение</h4>
<OL>
<li><A HREF=“a1.htm” TARGET=“main”>Глава 1 </a>
<li><A HREF=“a2.htm” TARGET=“main”>Глава 2 </a>
<li><A HREF=“a3.htm” TARGET=“main”>Глава 3 </a>
</ol>
<H4>Заключение</h4>
</body>
</html>
```



Бегущая строка (элемент MARQUEE)

<MARQUEE> </marquee>

Элемент, создающий бегущую строку. Сам по себе прием интересен, наилучшего эффекта можно добиться, удачно подобрав атрибуты. Фоновый цвет задается так:

bgcolor="Цвет"

Если фон задан, то броузер рисует на экране цветную полосу, вдоль которой бежит текст. Высоту полосы можно регулировать двумя способами:

height=Высота в пикселах

height=Число %

Если используются пикселы, то можно порекомендовать диапазон 30...50. Высоту можно указывать и в процентах. Процент определяет долю от высоты видимой части гипертекста внутри окна броузера. Эта величина, разумеется, не является постоянной и зависит от размера окна. Если высота полосы достаточно большая, имеет смысл использование атрибута align для выравнивания текста по верхнему краю, по середине или по нижнему краю полосы:

- align=top
- align=middle
- align=bottom

Правда, не все браузеры поддерживают этот атрибут. Вот пример полосы зеленого цвета, высотой 50 пикселов, с выравниванием бегущего текста по середине:

```
<MARQUEE bgcolor="green" height=50 align="middle"> Бегущая строка
</MARQUEE>
```

Направление движения строки тоже можно менять:

```
direction="left"
direction="right"
```

Самым удачным атрибутом элемента marquee является behavior, управляющий поведением (behavior) строки. По умолчанию создается обычная бегущая строка, какие бывают на табло. Дойдя до края экрана (окна), она уходит из поля зрения, а затем появляется с противоположной стороны. Этому варианту поведения соответствует следующее значение атрибута:

behavior="scroll"

Второй вариант движения заключается в следующем - строка появляется из-за края окна, достигает противоположного и останавливается. Значение атрибута таково:

behavior="slide"

По третьему сценарию строка не исчезает с экрана, но и не останавливается. Она движется вправо или влево, "отражаясь" от краев окна и меняя направление движения. Атрибут в этом случае должен быть задан так:

behavior="alternate"

Всю полосу можно сдвинуть по горизонтали вправо:

hspace=Смещение в пикселах

Выше и ниже полосы можно создать пустое пространство:

vspace=Высота в пикселах

Количество проходов строки по экрану можно ограничить:

loop=Число

Выполнив необходимое число проходов, строка остановится. Отсчет начнется только после того, как пользователь увидит ее на экране. Скорость движения задает следующий атрибут:

```
scrollamount=Число
```

Если число = 1, то строка будет еле ползти по сравнению с режимом по умолчанию. Если число > 10, то она будет двигаться очень быстро. Данный атрибут задает скорость движения как число пикселов, которые проходит строка за каждый шаг.

Существует второй атрибут скорости, определяющий временной интервал (в миллисекундах) между шагами:

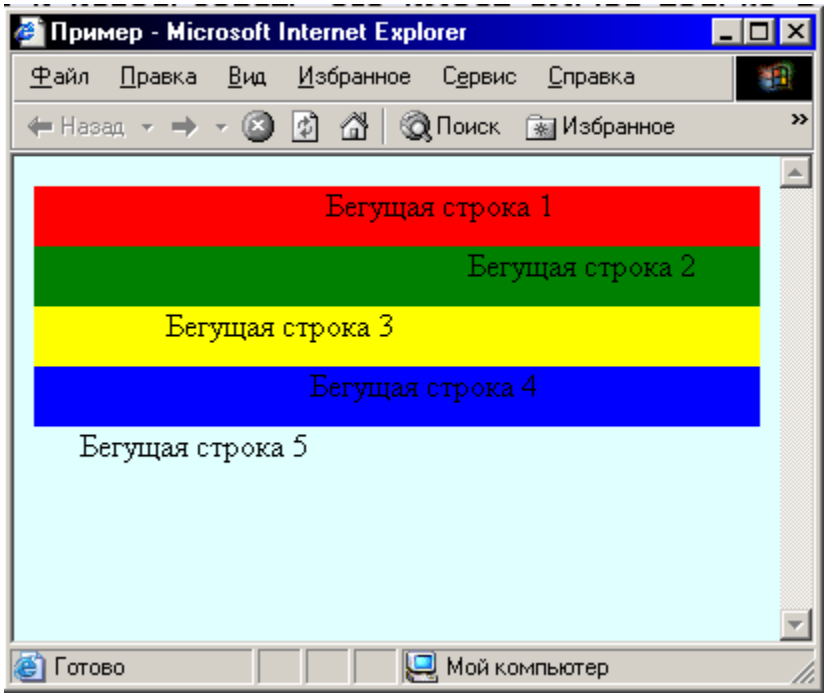
```
scrolldelay=Число
```

С помощью этого атрибута можно заставить строку двигаться рывками.

Пример.

```
<HEAD>
<TITLE>Бегущая строка</title>
</head>
<BODY bgcolor=#E0FFFF>
```

```
<BASEFONT size=3>
<MARQUEE bgcolor="RED" height=30 direction="right"
behavior="scroll"> Бегущая строка 1 </marquee>
<MARQUEE BGCOLOR="GREEN" height=30 direction="right"
behavior="slide" scrollamount=3> Бегущая строка 2 </marquee>
<MARQUEE BGCOLOR="YELLOW" height=30 direction="right"
behavior="alternate" scrollamount=4> Бегущая строка 3 </marquee>
<MARQUEE BGCOLOR="BLUE" height=30 direction="right"
behavior="alternate" scrolldelay=500 scrollamount=10> Бегущая
строка 4 </marquee>
<MARQUEE> Бегущая строка 5 </marquee>
</body>
</html>
```



Фиксация информации на экране (элемент BANNER)

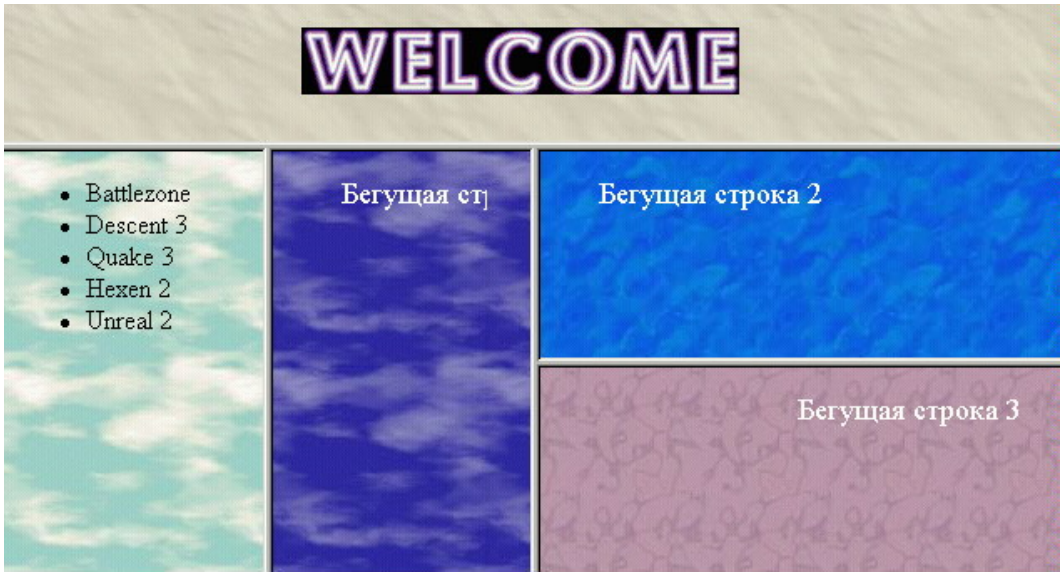
<BANNER> </banner>

Элемент, позволяющий зафиксировать некоторую информацию на экране вне зависимости от того, какая часть документа просматривается. Такие "постоянные" детали страницы, называемые баннерами, удобны для размещения логотипов, подсказок, заголовков и т. д. Обсуждаемый элемент поддерживается не всеми браузерами, и использовать его имеет смысл только в том случае, когда заранее известно, что он будет функционировать.

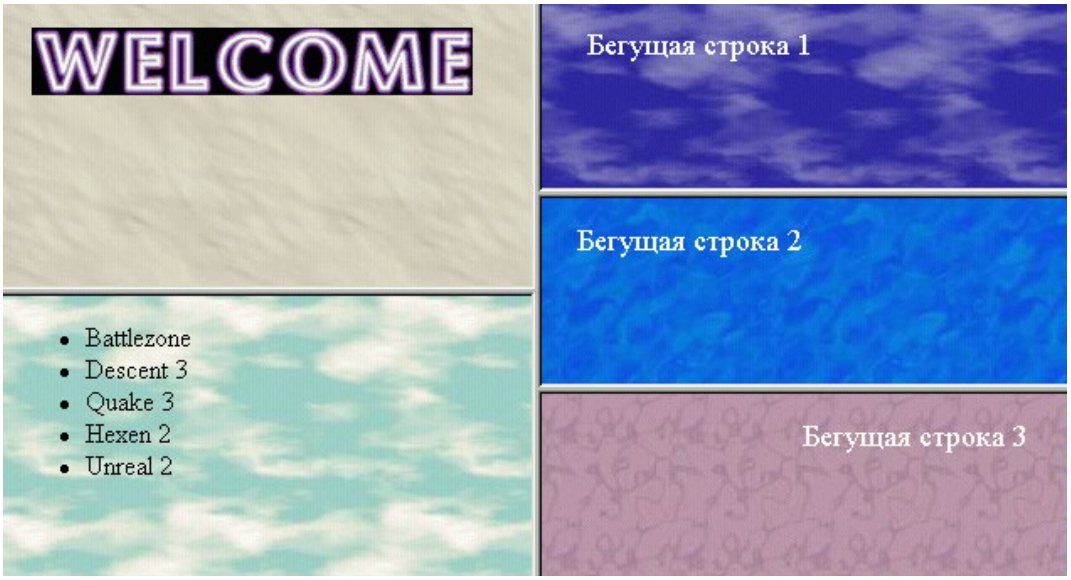
Задания к лабораторной работе № 9.
Фреймы

1. На базе приведенных примеров задайте структуру гипертекста с кадрами, закрасив фон различными цветами. Разделите вертикальный кадр на горизонтальные полосы.
2. В один из кадров вставьте графическое изображение или текстовый файл.
3. Разработайте оглавление к курсовой работе по информатике.
4. Изучите примеры и создайте WEB-страницу следующего вида: в трех фреймах должно быть по одной бегущей строке (на картинке - бегущая строка 1, бегущая

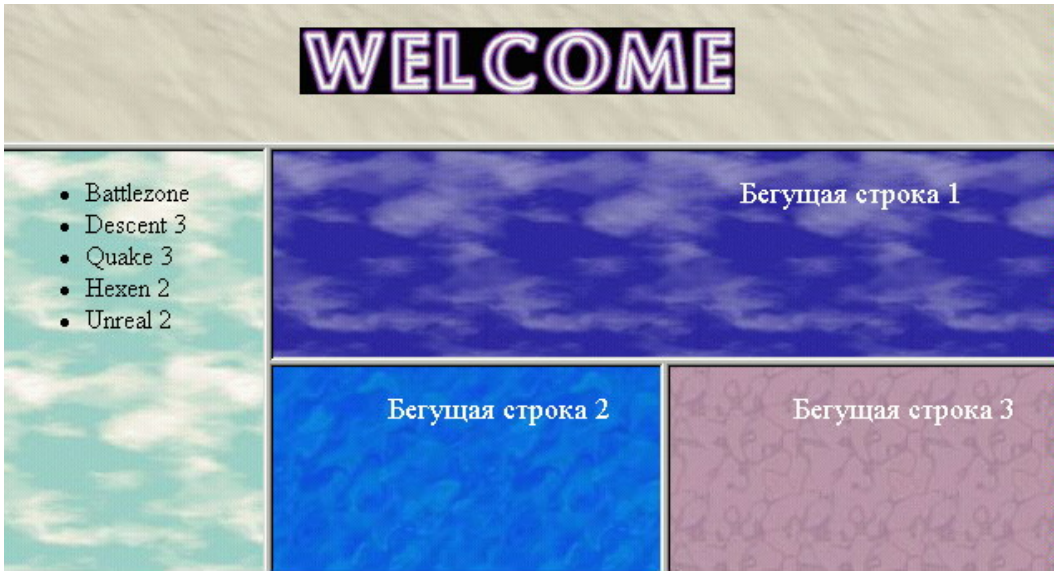
строка 2, бегущая строка 3); должны быть использованы 3 варианта движения, т.е. каждая из строк должна иметь свой вариант)



5. Изучите примеры и создайте WEB-страницу следующего вида: в трех фреймах должно быть по одной бегущей строке (на картинке - бегущая строка 1, бегущая строка 2, бегущая строка 3); должны быть использованы 3 варианта движения, т.е. каждая из строк должна иметь свой вариант)



6. Изучите примеры и создайте WEB-страницу следующего вида: в трех фреймах должно быть по одной бегущей строке (на картинке - бегущая строка 1, бегущая строка 2, бегущая строка 3); должны быть использованы 3 варианта движения, т.е. каждая из строк должна иметь свой вариант)



Контрольные вопросы

1. Создание фреймов в HTML-документах.
2. Синтаксис фреймов. Перечислите и охарактеризуйте атрибуты тега <FRAMESET>.
3. Тег <FRAME>. Его основные атрибуты. Дайте их характеристику.
4. Планирование фреймов и взаимодействия между фреймами
5. Создание оглавлений в HTML - документах.
6. Создание бегущей строки.

Лабораторная работа № 10.
Элементы APPLET и SCRIPT

<APPLET> </applet>

Этот элемент поддерживается броузерами, в которые встроен интерпретатор Java. Простейший шаблон, который позволит включить в HTML-страницу апплет на языке Java, показан ниже:

```
<APPLET code="Имя_файла.class" width=n height=m> Произвольный текст комментария
</applet>
```

Атрибут **code** необходим для указания имени файла, содержащего откомпилированную Java-программу. В отличие от других ссылок на ресурсы, значение этого атрибута может быть только относительным. При выполнении апплета создается окно шириной *n* и высотой *m* пикселов. Внутри элемента APPLET может размещаться произвольный гипертекст. Броузеры, поддерживающие Java, игнорируют все элементы этого гипертекста, включая собственно текст. Исключение составляют элементы **PARAM.** Броузеры, не поддерживающие Java, игнорируют элементы **PARAM** и воспроизводят «понятный» для них гипертекст. Стартовый тег элемента **APPLET** может быть снабжен дополнительными атрибутами.

Атрибут **codebase** предназначен для указания места расположения апплетов:

codebase="URL_*для_апплетов*"

Если этот атрибут не задан, то по умолчанию используется URL страницы.

Атрибут **alt** выполняет ту же функцию, что и при выводе изображений. Если броузер не может воспроизвести апплет, то выводится текст, заданный при помощи этого атрибута:

alt="*Произвольный текст*"

Атрибут name задает имя апплета. Это необходимо только в том случае, когда несколько апплетов, расположенных на одной странице, взаимодействуют между собой:

name="*Имя апплета*"

Атрибуты **width** и **height** задают параметры окна апплета: ширину и высоту соответственно.

Поскольку для выполнения апплета создается окно, предусмотрено использование уже хорошо известного нам атрибута **align** для управления размещением этого окна. Атрибут может иметь следующие назначения: **bottom, left, middle**, **right**, **top.**

Вокруг окна можно создать пустое пространство:

vspace=*Число пикселов выше и ниже*

hspace=Число *пикселов слева и справа*

<**param**>

Этот элемент используется для передачи параметров Java-программе и размещается только внутри элемента APPLET. Шаблон его использования таков:

<APPLET code = "*Имя_файла.* class" width=n height=m> <PARAM name="Имя *параметра.*" value=Значение> *Произвольный текст комментария*

</applet>

Элемент PARAM должен содержать одну или несколько пар атрибутов, определяющих имя параметра (name) и его значение (value). При выполнении апплета прием параметров осуществляется по следующему шаблону:

Переменная=get Parameter ("*Имя_параметра*");

<**SCRIPT**>

Если Java-программа передается на сторону клиента в откомпилированном виде, то программы на языках JavaScript и Visual Basic Script включаются непосредственно в HTML-документ в виде исходного текста. Контейнером для этих программ служит элемент SCRIPT. Он должен включать атрибут **language**, который определяет используемый язык и может принимать значения JavaScript или VBScript. Ниже приведен шаблон Web-страницы, в которой использована программа на JavaScript.

```
<HTML>
<HEAD>
<TITLE>Страница с программой на JavaScript </title>
<SCRIPT language = "JavaScript">
function Имя(параметр)
{
Текст программы на JavaScript
}
</script>
</head>
<BODY onload ="Имя(параметр)">
Текст Web-страницы
</body>
</html>
```

В приведенном выше листинге в элементе BODY был использован атрибут onload для того, чтобы запустить программы после загрузки страницы.

Лабораторная работа №11.
Основы JavaScript

Цели работы: создать первый скрипт и ознакомится с основами создания и размещения JavaScript на веб-странице.

Что такое JavaScript?

Во-первых, это не Java. Таким образом Java и JavaScript – это не одно и то же. Java – это язык программирования, разработанный в Sun Microsystems. А JavaScript является продуктом Netscape. Но это не единственное отличие.

Оба языка представляют собой ООР (Object Orientated Programming - объектно-ориентированный язык программирования). Это значит, что с их помощью можно строить небольшие объекты, из которых потом складывается целое. Главное отличие в том, что Java позволяет создавать совершенно самостоятельные события. «Java-applet» («приложеньице») может запускаться с веб-страницы, но на самом деле это полностью независимая программа, хоть и маленькая. К тому же ее нельзя просмотреть в виде текста. Для запуска ее необходимо «транслировать» в то, что называется «машинным языком».

Netscape как бы упростил Java до набора более простых команд. JavaScript не может существовать сам по себе, он должен находиться внутри веб-страницы, а веб-страницу необходимо просматривать в браузере, который понимает язык JavaScript (скажем, Netscape Communicator и Internet Explorer).

JavaScript — это не HTML. У JavaScript и HTML очень похожие правила. Во-первых, JavaScript располагается внутри документа HTML. JavaScript сохраняется в виде текста вместе с документом HTML. Главная же разница в том, что в HTML имеет довольно расплывчатые правила. Не имеет значения, сколько пробелов вы оставляете между словами или абзацами. HTML можно было бы писать одной сплошной строкой. У JavaScript имеется четкая форма. И пренебрегать ею можно лишь изредка.

Пример.

```
<SCRIPT LANGUAGE="javascript">
document.write("<FONT COLOR='RED'>Это красный текст</FONT>")
</SCRIPT>
```

Например, вторая строка нашего скрипта выглядит следующим образом:

```
document.write("<font color='red'>Красный текст</font>")
```

То есть, целиком находится на одной линии и должна сохранять свою форму. Предположим, вы скопировали ее в текстовый редактор с узкими страницами, и поля разорвали строку:

```
document.write("<font color='red'>Красный текст</font
                                  >")
```

Таким образом записанный скрипт имеет ошибки.

При написании скрипта желательно использовать текстовый редактор, например NotePad поскольку в нем нет полей. При написании скрипта обращайте внимание на регистр букв.

Ниже разбирается этот скрипт.

Начнем с первой строки:

```
<SCRIPT LANGUAGE="JavaScript">
```

Это код HTML, который дает браузеру понять, что с этого места начинается JavaScript. Все скрипты начинаются с такой команды. Запсись LANGUAGE(язык)="JavaScript" является необходимой.

Этим:

```
</SCRIPT>
```

...заканчивается любой JavaScript без исключений.

Скрипт начинается с <SCRIPT LANGUAGE="javascript"> и заканчивается </SCRIPT>. Основная часть скрипта имеет следующий вид:

```
document.write("<FONT COLOR='RED'>Это красный текст</FONT>")
```

Вот из чего состоит скрипт: указывается DOCUMENT (документ HTML) и те изменения, которые в нем произойдут – что-то будет написано (WRITE). То, что будет написано, находится в скобках.

DOCUMENT представляет собой object (объект). Слово WRITE (писать), отделенное точкой, называется method (методом объекта). Таким образом, скрипт попросту говорит: «Возьмите объект (что-то, уже существующее) и припишите что-то к нему».

Текст в скобках называется instance (примером метода), он передает то, что происходит, когда метод воздействует на объект. Имейте в виду, что текст внутри скобок находится в кавычках. Никогда нельзя про них забывать.

Текст в кавычках представляет собой простой HTML. Обратите внимание, что дальше идут одинарные кавычки. Если поставить двойные, JavaScript решит, что это конец строки, и получится, что только часть вашего текста будет применена к объекту, а это уже ошибка.
Запомните: внутри двойных кавычек ставятся одинарные. JavaScript не перекрашивает текст, а текст перекрасил HTML.

Пример. Работа с датами

```
<SCRIPT LANGUAGE="JavaScript">
//Скрипт отмечает точную дату и время вашего прибытия на страницу
Now = new Date();
document.write("Сегодня " + Now.getDate()+
"-" + Now.getMonth() + "-" + Now.getFullYear() + ".
Вы зашли на мою страницу ровно в: " + Now.getHours() +
":" + Now.getMinutes() + " и " + Now.getSeconds() +
" секунд.")
</SCRIPT>
```

Обратите внимание, что синтаксис программы очень похож на программы написанные на C++.

Пример. Работа с мышью

```
<A HREF="http://www.newmail.ru"
onMouseOver="window.status='Бесплатный хостинг';
return true">Ссылка</A>
Пример 4. Команда onClick
<a href="http://www.jsp.newmail.ru" onClick="alert('Attention!');">
JavaScript it's easy</a>
```

Пример. Команда onFocus

```
<form>
<input type="text" size="30"
onFocus="window.status='Текст в строке состояния';">
</form>
```

Пример. Команда onBlur

```
<form>
<input type="text" size="45" value="Впишите свое имя и щелкните по
другой строке"
onBlur="alert('Вы изменили ответ — уверены, что он правильный?');">
</form>
```

Пример. Команда onChange

```
<form>
<input TYPE="text" size="45"
value="Измените текст и щелкните по другой строке"
onChange="window.status='Текст был изменен';">
</form>
```

Пример. Команда onSubmit

```
<form>
<input TYPE="submit"
onSubmit="parent.location='thanksalot.html'";>
</form>
```

Пример. Получение информации от пользователя

```
<SCRIPT LANGUAGE="javascript">
/* Скрипт предназначен для того, чтобы получить
от пользователя информацию и поместить ее на страницу */

var user_name = prompt ("Напишите свое имя","Здесь");
document.write("Добрый день, " + user_name + "! Добро
пожаловать!");
</SCRIPT>
```

Пример. Функции в JavaScript

```
<SCRIPT LANGUAGE="javascript">
<!-- Скрыть от браузеров, не читающих Javascript
function dateinbar()
{
var d = new Date();
var y = d.getFullYear();
var da = d.getDate();
var m = d.getMonth() + 1;
var t = da + '/' + m + '/' + y;

defaultStatus = "Вы прибыли на страницу " + t + ".";
}
// не скрывать -->
</SCRIPT>
...и команда onLoad в <BODY>:
<BODY BGCOLOR="xxxxxx" onLoad="dateinbar()">
```

Индивидуальные задания.

Напишите скрипт который делал бы следующее

№	Вариант задания
1.	Вывести две строки текста произвольного содержания друг под другом. Цвет первой строки должен быть красным, а второй синим.
2.	Поместить на вашу страницу дату, разделенную дробями /. Приветственный текст должен быть зеленого цвета.
3.	Метод, alert() (предупредить) вызывает небольшое диалоговое окно с текстом и кнопкой

	OK. Сделайте так, чтобы окно предупреждения всплывало при наведении курсора на ссылку.
4.	Создать форму, которая будет взаимодействовать с пользователем. Форма должна иметь три элемента: поле ввода с просьбой ввести имя; два поля для флажков с вопросом о том, что пользователю больше нравиться учиться или отдыхать; кнопку отправки данных. При вводе имени, в строке состояния должны появиться слова: «Впишите сюда свое имя»
5.	Создать форму, которая будет взаимодействовать с пользователем. Форма должна иметь три элемента: поле ввода с просьбой ввести имя; два поля для флажков с вопросом о том, что пользователю больше нравиться учиться или отдыхать; кнопку отправки данных. Два поля с флажками должны отослать в строку состояния слова: «Вы выбрали...» и выбор пользователя.
6.	Создать форму, которая будет взаимодействовать с пользователем. Форма должна иметь три элемента: поле ввода с просьбой ввести имя; два поля для флажков с вопросом о том, что пользователю больше нравиться учиться или отдыхать; кнопку отправки данных. При нажатии на кнопку должно выскочить окно предупреждения, благодарящее пользователя за участие в опросе.
7.	Создать ссылку на страницу на вашем сервере. Например, если вы находитесь на www.you.ru, JavaScript создаст ссылку на www.you.ru/index.html. Также, каким бы свойством вы ни воспользовались, присвойте ему переменную.
8.	Создайте функцию, которая вызовет два запроса (prompt). (Подсказка: один следует за другим с новой строки.) Первый попросит пользователя ввести свое имя, второй — отчество. Затем та же функция должна вызвать окно предупреждения (alert) с текстом: **Hello, имя отчество, Welcome to web page, it's great site WWW!**
9.	Напишите функцию, которая возвращает дату ее вызова и после пишет дату в TextBox.
10.	Напишите функцию, которая при ее вызове меняет состояние у CheckBox и пишет некоторое содержание в TextBox.

Контрольные вопросы

1. Какая разница между HTML и JavaScript ?
2. Опишите работу с датами
3. Опишите работу с мышью
4. Опишите команду onClick
5. Опишите команду onFocus
6. Опишите команду onBlur
7. Опишите команду onChange
8. Опишите команду onSubmit
9. Как получить информацию от пользователя
10. Дайте пример представления функции в JavaScript.

Лабораторная работа № 12. Создание собственного сайта по индивидуальному заданию

"Создание Web-сайта на заданную тему в текстовом редакторе Блокнот"

Задание: Необходимо создать сайт на одну из предложенных тем, состоящий не менее чем из пяти страниц.

Обязательные элементы:

- графические элементы оформления;
- гиперссылки на другие аналогичные ресурсы;

- гиперссылка на почтовый адрес автора.

Оцениваются:

- владение основными элементами языка HTML;
- гармоничность цветовой гаммы Web-сайта;
- содержание (контент) Web-сайта;
- продуманность структуры Web-сайта.

Темы:

1. Компьютерные преступления.
2. Компьютерные вирусы.
3. Вопросы безопасности в компьютерных системах.
4. Интернет сегодня.
5. Студенческая жизнь.
6. Наш факультет.
7. Любимые дисциплины.
8. Мои увлечения (хобби).
9. Спортивные соревнования.

СПРАВОЧНИК

HTML - документ (страничка)	документ, написанный на языке разметки гипертекста (HTML).
Web - сайт, Web – сервер	цепочка логически связанных документов, написанных на языке HTML.
Теги разметки	специальные команды для расположения на экране текста, графики, видео и аудио фрагментов, а также команды, служащие для связи с другими HTML - документами и ресурсами Интернет.

Теги разметки документа	**Значения тегов разметки документа**
<HTML> … </html>	Начало и окончание HTML - документа
<HEAD> … </head>	Между этими тегами располагается информация о документе (на экран не выводится)
<TITLE> … </title>	Между этими тегами заключается название странички, которое будет выведено в рамке окна программы просмотра.
<BODY> … </body>	Между этими тегами располагается "тело" документа (текст, графика и т.д.).
Параметры тега <BODY>	
<BODY bgcolor ="#FFFFFF">	BGCOLOR - цвет фона
background	"Обои" или бэкграунд
text	Цвет текста
link	Цвет гипертекстовой связи (ссылки)
vlink	Цвет ссылки, уже посещенной в прошлом
alink	Цвет активной ссылки
Теги, служащие для форматирования текста	
<P> … </p>	Теги, служащие для выделения абзацев. Новый абзац всегда отделяется от предыдущего пустой строкой

Теги разметки документа	Значения тегов разметки документа
 	Тег, служащий для переноса текста на другую строку. Может также служить для отделения графики от текста на интервал.
<HR>	Тег, служащий для логического разделения текста горизонтальной линией.
<PRE> … </pre>	Между этими тегами располагается предварительно отформатированный текст. На экран он выводится шрифтом типа "курьер".
Заголовки, служащие для выделения логических частей текста	
<H1> … </h1>	Заголовок первого уровня
<H2> … </h2>	Заголовок второго уровня
<H3> … </h3>	Заголовок третьего уровня
<H4> …</h4>	Заголовок четвертого уровня
<H5> … </h5>	Заголовок пятого уровня
<H6> … </h6>	Заголовок шестого уровня
Параметры выравнивания (используются в тегах <P> и <H_1>)	
align = left	Выравнивание по левому полю
align = right	Выравнивание по правому полю
align = center	Выравнивание по центру
Теги выравнивания	
<left> … </left>	Выравнивание по левому полю
<right> … </right>	Выравнивание по правому полю
…	Выравнивание по центру
Теги для выделения текста и шрифта	
<B> … 	Теги для выделения текста (слов, букв) жирным шрифтом
<I> … </i>	Теги для выделения текста (слов, букв) курсивным шрифтом, типа Italic
<U> … </u>	Текст, расположенный между двумя этими тегами, будет подчеркнут
<BLINK> … </blink>	Текст, расположенный между двумя этими тегами, будет мигать
<FONT size = +n> … <FONT size = -n> … 	Теги для изменения размера шрифта
<FONT color="#0000FF> … 	Теги для изменения цвета шрифт а
Теги для формирования списков	
<OL> …	Теги, показывающие начало и конец нумерованного списка
<UL> …	Теги, показывающие начало и конец маркированного списка.
<LI>	Элемент списка
<DL> и </dl>	Теги, показывающие начало и конец глоссария
<DT>	Термин глоссария, располагается без отступа от левого поля страницы
<DD>	Описание термина, располагается с отступом от левого поля страницы
Теги-команды для вставки в текст объектов нетекстовой информации	

Теги разметки документа	Значения тегов разметки документа
<IMG src ="file.gif"> или <IMG src =" file.jpg">	Команда для вставки графического изображения
<IMG src =" file.wav">	Команда для вставки звукового фрагмента
<IMG src =" file.avi">	Команда для вставки видео фрагмента
Параметры графического изображения	
widht	Ширина картинки в пикселях
height	Высота картинки в пикселях
align - выравнивание	
align = left	Выравнивание по левому полю
align = right	Выравнивание по правому полю
align = top	По верхней границе
align = bottom	По нижней границе
align = middle или center	По центру
hspace	Горизонтальный отступ от графического изображения
vspace	Вертикальный отступ
alt	Альтернативный текст, служит для обозначения изображения
Команды, служащие для гиперсвязи с другими HTML - документами и ресурсами Интернет	
<A href="fail.htm"> … или <A href="http://www.ru">…	Гиперсвязи (на определенный файл или адрес)
<ADDRESS> <A href=mailto:person@firm.ru person@firm.ru </address>	Гиперсвязь с адресом электронной почты
Таблица - сетка для показа данных в строках и столбцах, а также средство для форматирования текста	
<TABLE> …</table>	Теги для вставки таблицы в HTML - документ
Параметры тега <TABLE>:	
bgcolor	Цвет фона
border	Ширина бордюра
widht	Ширина таблицы
Теги разметки таблицы	
<CAPTION> … </caption>	Название таблицы
Параметры: align = top	Выравнивание над таблицей
align = bottom	Выравнивание под таблицей
<TR> … </tr>	Строчка таблицы
Параметры: bgcolor	цвет фона внутри строки
align = left	Выравнивание внутри строки (по левому краю)
align = right	Выравнивание внутри строки (по правому краю)
align = center	Выравнивание внутри строки (по центру)
valign = top	Вертикальное выравнивание внутри строки таблицы - по верхнему краю
valign = bottom	Вертикальное выравнивание внутри строки таблицы - по нижнему краю

Теги разметки документа	Значения тегов разметки документа
valign = middle	Вертикальное выравнивание внутри таблицы - по середине
<TD> … </td>	Столбец таблицы
Параметры: bgcolor	Цвет фона под столбцом
align = left, right, center	Выравнивание внутри столбца
valign = top, bottom, middle	Вертикальное выравнивание
colspan	Растягивание клетки на несколько столбцов
rowspan	Растягивание клетки на несколько строк
<TH> и </th>	Заголовок столбца
параметры: bgcolor	Цвет фона под названием
align = left, right, center	Выравнивание
valign = top, bottom, middle	Вертикальное выравнивание
colspan, rowspan	Растягивание клетки на несколько столбцов или строк
widht	Ширина названия
Рамки (фреймы) - средство для разделения экрана на несколько областей, в каждой из которых отображается содержимое отдельной Web-странички или даже целого Web-сайта	
<FRAMESET> ... </frameset>	Теги для создания рамки
Параметры тега <FRAMESET>	
cols	Деление экрана на определенное количество вертикальных колонок
rows	Подразделяют экран на определенное количество горизонтальных колонок
bordcolor	Цвет рамки
border	Ширина бордюра
frameborder	Граница рамки - FRAMEBORDER=YES - есть граница, FRAMEBORDER=NO - нет границы
framespacing = n	Ширина границы
<FRAME>	Тег для описания рамки (<FRAME SRC="file.htm">)
Параметры тега <FRAME>	
SCROLING	Параметр для регулировки полосы прокрутки
SCROLING = YES	Полоса прокрутки будет всегда
SCROLING = NO	Полосы прокрутки не будет
SCROLING = AUTO	Полоса прокрутки появляется только в случае необходимости
MARGINWIDHT и MARGINHEIGHT	Параметры, которые управляют отступом внутри рамок, служат для выравнивания графического изображения внутри рамки
NORESIZE	Параметр, указывающий на то, что размер рамки-фрейма никогда не будет меняться
A link to <A href="file.htm" TARGET="frame2"> file.htm	Связь между фреймами
TARGET	Атрибут связи между фреймами. Имеет несколько значений:
blank	Загружает содержимое страницы, заданной ссылкой, в новое пустое окно

Теги разметки документа	Значения тегов разметки документа
self	Содержимое страницы, заданной ссылкой, в окно, которое содержит ссылку
parent	Загружает содержимое страницы, заданной ссылкой, в окно, являющееся непосредственным владельцем набора фреймов
top	Содержимое страницы, заданной ссылкой, в окно, игнорируя используемые фреймы
Бегущая строка	
<MARQUEE> ТЕКСТ </marquee>	Тег, создающий бегущую строку
<MARQUEE direction = left > tekct</marquee>	Если бегущую строку нужно направить справа налево
<MARQUEE direction=right> tekct</marquee>	Движение слева направо
scroll	Стандартное движение от правого края к левому
slide	Надпись один раз пробегает от правого края к левому, где и остается
alternate	Движение от правого края страницы к левому и обратно; бесконечный цикл.
<MARQUEE loop = n behavior = scroll > текст </marquee>	Ограничение числа циклов. Значение n оператора LOOP указывает число повторений цикла
<MARQUEE widht = n > текст </marquee>	Указать ширину участка, занимаемого бегущей строкой, где n - ширина той части страницы, на которой расположена бегущая строка. Значение n указывается как в пикселях, так и в процентах от общей ширины видимой части страницы
<MARQUEE crollamount = n > текст </marquee>	Регулировка движения надписи по экрану. Здесь n - число пикселей
<MARQUEE scrolldelay = t > текст </marquee>	В данном случае переменная величина – время t (измеряется в миллисекундах). Метод задания скорости состоит в указании времени, спустя которое текст будет перерисован на экране заново
<FONTSIZE=n><MARQUEE> текст </marquee>	Возможность указывать величину шрифта текста в строке
<MARQUEE bgcolor=n> текст </marquee>	Окрасить поверхность бегущей строки в какой-либо цвет, где n, как это бывало и раньше, можно указать в вид шестнадцатеричного числа, либо написав его название
<MARQUEE height = n > текст </marquee>	Указать высоту бегущей строки, задавая величину n в пикселях

ПРИЛОЖЕНИЕ 1.
Таблица спецсимволов

Ниже приведены значения специальных символов, которые можно использовать при создании веб-сайтов. Можно включать в HTML как имя символа, так и его код - результат должен быть одинаков.

Имя	Код	Вид	Описание
			неразрывный пробел
&iexcl;	&#161;	¡	перевернутый восклицательный знак
&cent;	&#162;	¢	цент
&pound;	&#163;	£	фунт стерлингов
&curren;	&#164;	¤	денежная единица
&yen;	&#165;	¥	иена или юань
&brvbar;	&#166;	¦	разорванная вертикальная черта
&sect;	&#167;	§	параграф
&uml;	&#168;	¨	трема (знак над гласной для произнесения ее отдельно от предшествующей гласной; напр., na?ve)
&copy;	&#169;	©	знак copyright
&ordf;	&#170;	ª	женский порядковый числитель
&laquo;	&#171;	«	левая двойная угловая скобка
&not;	&#172;	¬	знак отрицания
&shy;	&#173;		место возможного переноса
&reg;	&#174;	®	знак зарегистрированной торговой марки
&macr;	&#175;	¯	знак долготы над гласным
&deg;	&#176;	°	градус
&plusmn;	&#177;	±	плюс-минус
&sup2;	&#178;	²	верхний индекс 'два' - "в квадрате"
&sup3;	&#179;	³	верхний индекс 'три' - "в кубе"
&acute;	&#180;	´	знак ударения
&micro;	&#181;	µ	микро
&para;	&#182;	¶	символ параграфа
&middot;	&#183;	·	точка
&cedil;	&#184;	¸	седиль (орфографический знак)
&sup1;	&#185;	¹	верхний индекс 'один'
&ordm;	&#186;	º	мужской порядковый числитель
&raquo;	&#187;	»	правая двойная угловая скобка
&frac14;	&#188;	¼	дробь - одна четверть
&frac12;	&#189;	½	дробь - одна вторая
&frac34;	&#190;	¾	дробь - три четверти
&iquest;	&#191;	¿	перевернутый вопросительный знак
&Agrave	&#192;	À	латинская заглавная буква A с тупым ударением

;			
Á	Á	Á	латинская заглавная буква A с острым ударением
Â	Â	Â	латинская заглавная буква A с циркумфлексом (диакритический знак над гласной)
Ã	Ã	Ã	латинская заглавная буква A с тильдой
Ä	Ä	Ä	латинская заглавная буква A с тремой (знак над гласной для произнесения ее отдельно от предшествующей гласной)
Å	Å	Å	латинская заглавная буква A с верхним кружком
Æ	Æ	Æ	латинские заглавные символы AE
Ç	Ç	Ç	латинская заглавная буква C с седилем
È	È	È	латинская заглавная буква E с тупым ударением
É	É	É	латинская заглавная буква E с острым ударением
Ê	Ê	Ê	латинская заглавная буква E с циркумфлексом (диакритический знак над гласной)
Ë	Ë	Ë	латинская заглавная буква E с тремой
Ì	Ì	Ì	латинская заглавная буква I с тупым ударением
Í	Í	Í	латинская заглавная буква I с острым ударением
Î	Î	Î	латинская заглавная буква I с циркумфлексом
Ï	Ï	Ï	латинская заглавная буква I с тремой
Ð	Ð	Ð	латинские заглавные символы ETH
Ñ	Ñ	Ñ	латинская заглавная буква N с тильдой
Ò	Ò	Ò	латинская заглавная буква O с тупым ударением
Ó	Ó	Ó	латинская заглавная буква O с острым ударением
Ô	Ô	Ô	латинская заглавная буква O с циркумфлексом
Õ	Õ	Õ	латинская заглавная буква O с тильдой
Ö	Ö	Ö	латинская заглавная буква O с тремой
×	×	×	знак умножения
Ø	Ø	Ø	латинская заглавная буква O со штрихом
Ù	Ù	Ù	латинская заглавная буква U с тупым ударением
Ú	Ú	Ú	латинская заглавная буква U с острым ударением
Û	Û	Û	латинская заглавная буква U с циркумфлексом
Ü	Ü	Ü	латинская заглавная буква U с тремой
Ý	Ý	Ý	латинская заглавная буква Y с острым ударением
Þ	Þ	Þ	латинская заглавная буква THORN
à	à	à	латинская строчная буква A с тупым ударением
á	á	á	латинская строчная буква A с острым ударением
â	â	â	латинская строчная буква A с циркумфлексом
ã	ã	ã	латинская строчная буква A с тильдой

ä	ä	ä	латинская строчная буква A с тремой
å	å	å	латинская строчная буква A с верхним кружком
æ	æ	æ	латинская строчные буквы AE
ç	ç	ç	латинская строчная буква A с седилем
è	è	è	латинская строчная буква E с тупым ударением
é	é	é	латинская строчная буква E с острым ударением
ê	ê	ê	латинская строчная буква E с циркумфлексом
ë	ë	ë	латинская строчная буква E с тремой
ì	ì	ì	латинская строчная буква I с тупым ударением
í	í	í	латинская строчная буква I с острым ударением
î	î	î	латинская строчная буква I с циркумфлексом
ï	ï	ï	латинская строчная буква I с тремой
ð	ð	ð	латинская строчные символы eth
ñ	ñ	ñ	латинская строчная буква N с тильдой
ò	ò	ò	латинская строчная буква O с тупым ударением
ó	ó	ó	латинская строчная буква O с острым ударением
ô	ô	ô	латинская строчная буква O с циркумфлексом
õ	õ	õ	латинская строчная буква I с тильдой
ö	ö	ö	латинская строчная буква I с тремой
÷	÷	÷	знак деления
ø	ø	ø	латинская строчная буква O со штрихом
ù	ù	ù	латинская строчная буква U с тупым ударением
ú	ú	ú	латинская строчная буква U с острым ударением
û	û	û	латинская строчная буква U с циркумфлексом
ü	ü	ü	латинская строчная буква U с тремой
ý	ý	ý	латинская строчная буква Y с острымударением
þ	þ	þ	латинская строчная буква thorn
ÿ	ÿ	ÿ	латинская строчная буква Y с тремой
ƒ	ƒ	*ƒ*	знак функции
Греческие буквы			
Α	Α	A	греческая заглавная буква альфа
Β	Β	B	греческая заглавная буква бета
Γ	Γ	Γ	греческая заглавная буква гамма
Δ	Δ	Δ	греческая заглавная буква дельта
Ε	Ε	E	греческая заглавная буква эпсилон
Ζ	Ζ	Z	греческая заглавная буква дзета
Η	Η	H	греческая заглавная буква эта
Θ	Θ	Θ	греческая заглавная буква тета
Ι	Ι	I	греческая заглавная буква иота

Κ	Κ	Κ	греческая заглавная буква каппа
Λ	Λ	Λ	греческая заглавная буква лямбда
Μ	Μ	Μ	греческая заглавная буква мю
Ν	Ν	Ν	греческая заглавная буква ню
Ξ	Ξ	Ξ	греческая заглавная буква кси
Ο	Ο	Ο	греческая заглавная буква омикрон
Π	Π	Π	греческая заглавная буква пи
Ρ	Ρ	Ρ	греческая заглавная буква ро
Σ	Σ	Σ	греческая заглавная буква сигма
Τ	Τ	Τ	греческая заглавная буква тау
Υ	Υ	Υ	греческая заглавная буква ипсилон
Φ	Φ	Φ	греческая заглавная буква фи
Χ	Χ	Χ	греческая заглавная буква хи
Ψ	Ψ	Ψ	греческая заглавная буква пси
Ω	Ω	Ω	греческая заглавная буква омега
α	α	α	греческая строчная буква альфа
β	β	β	греческая строчная буква бета
γ	γ	γ	греческая строчная буква гамма
δ	δ	δ	греческая строчная буква дельта
ε	ε	ε	греческая строчная буква эпсилон
ζ	ζ	ζ	греческая строчная буква дзета
η	η	η	греческая строчная буква эта
θ	θ	θ	греческая строчная буква тета
ι	ι	ι	греческая строчная буква иота
κ	κ	κ	греческая строчная буква каппа
λ	λ	λ	греческая строчная буква лямбда
μ	μ	μ	греческая строчная буква мю
ν	ν	ν	греческая строчная буква ню
ξ	ξ	ξ	греческая строчная буква кси
ο	ο	ο	греческая строчная буква омикрон
π	π	π	греческая строчная буква пи
ρ	ρ	ρ	греческая строчная буква ро
ς	ς	ς	греческая строчная буква сигма (final)
σ	σ	σ	греческая строчная буква сигма
τ	τ	τ	греческая строчная буква тау
υ	υ	υ	греческая строчная буква ипсилон

φ	&#966;	φ	греческая строчная буква фи
&chi;	&#967;	χ	греческая строчная буква хи
&psi;	&#968;	ψ	греческая строчная буква пси
&omega;	&#969;	ω	греческая строчная буква омега
Стрелки			
&larr;	&#8592;	←	стрелка влево
&uarr;	&#8593;	↑	стрелка вверх
&rarr;	&#8594;	→	стрелка вправо
&darr;	&#8595;	↓	стрелка вниз
&harr;	&#8596;	↔	стрелка влево-вправо
Прочие символы			
&spades;	&#9824;	♠	знак масти 'пики'
&clubs;	&#9827;	♣	знак масти 'трефы' - shamrock
&hearts;	&#9829;	♥	знак масти 'червы' - valentine
&diams;	&#9830;	♦	знак масти 'бубны'
&quot;	&#34;	"	двойная кавычка
&amp;	&#38;	&	амперсанд
&lt;	&#60;	<	знак 'меньше'
&gt;	&#62;	>	знак 'больше'
&circ;	&#710;	ˆ	символ циркумфлекса (диакритический знак над гласной)
&tilde;	&#732;	˜	тильда
&trade;	&#8482;	™	знак торговой марки
Знаки пунктуации			
&bull;	&#8226;	•	bullet - маленький черный кружок
&hellip;	&#8230;	…	многоточие ...
&prime;	&#8242;	′	одиночный штрих - минуты и футы
&Prime;	&#8243;	″	двойной штрих - секунды и дюймы

&oline;	&#8254;	‾	надчеркивание
&frasl;	&#8260;	⁄	косая дробная черта
Общая пунктуация			
&ndash;	&#8211;	–	тире
&mdash;	&#8212;	—	длинное тире
&lsquo;	&#8216;	‘	левая одиночная кавычка
&rsquo;	&#8217;	’	правая одиночная кавычка
&sbquo;	&#8218;	‚	нижняя одиночная кавычка
&ldquo;	&#8220;	“	левая двойная кавычка
&rdquo;	&#8221;	”	правая двойная кавычка
&bdquo;	&#8222;	„	нижняя двойная кавычка

Литература

1. Попов И.И. Информационные ресурсы и системы: реализация, моделирование, управление // М.: ТПК “Альянс”, 1996.
2. Максимович Г.Ю., Романенко А. Г., Самойлюк О.Ф. Информационные системы / Под общей редакцией К. И. Курбакова // М.: Изд-во Рос. экон. акад., 1999. - 198 с.
3. Морис Б. HTML в действии / Пер. с анг. -СПб.: Питер Пресс, 1997.- 256 с.
4. Internet шаг за шагом // СПб.: Питер Пресс, 1997.