МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
ОДЕССКИЙ ГОСУДАРСТВЕННЫЙ ЭКОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Ю. А. Кругляк
Л. С. Кострицкая

Программирование гипертекстовых документов

конспект лекций

Одесса – 2009
ТЭС

ББК 32.97
К 84
УДК 681.3

Печатается по решению Ученого совета Одесского государственного экологического университета

(протокол № 9 от 30.10 2008)

Кругляк Ю.А. , Кострицкая Л.С.
Программирование гипертекстовх документов: Конспект лекций.-Одесса: "ТЭС -2009.-150с.

Конспект лекций посвящен одному из самых популярных направлений в области информационных технологий – созданию Web-страниц.

В конспекте излагаются основные принципы работы в Internet, язык гипертекстовой разметки Web-страниц – HTML и каскадными таблицами стилей CSS.

Конспект лекций предназначен для студентов III курса специальности информационные управляющие системы и технологии, направление подготовки – компьютерные науки.

Введение

Появление всепланетной компьютерной сети Internet внесло существенные изменения в жизнь многих людей. В настоящее время для большинства компаний Internet - это быстрый и дешевый способ обмена информацией между сотрудниками, клиентами и деловыми партнерами. Для этого нужны собственные коммуникационные средства. Именно таким программным средством для World Wide Web и является язык HTML. Подавляющее большинство документов, с которыми вы сталкиваетесь в World Wide Web(Web-странички, Web-сайты,. Internet-магазины и т.д.)

HTML - это не язык программирования, а язык гипертекстовой разметки документа.

Идея гипертекстовой информационной системы состоит в том, что пользователь имеет возможность просматривать документы (страницы текста) в том порядке, в котором ему это больше нравится, а не последовательно, как это принято при чтении книг(как нелинейный текст). Достигается это путем создания специального механизма связи различных страниц текста при помощи гипертекстовых ссылок, т.е. у обычного текста есть ссылки типа "следующий-предыдущий", а у гипертекста можно построить еще сколь угодно много других ссылок.

Язык HTML позволяет определять структуру электронного документа с полиграфическим уровнем оформления; результирующий документ может содержать самые разнообразные теги: иллюстрации, аудио и видео фрагменты и так далее. Язык включает в свой состав развитые средства для специфицирования нескольких уровней заголовков, шрифтовых выделений, различных групп объектов, например, словари, каталоги или меню для размещения иллюстраций и других фрагментов, а также множество других возможностей.

Важным моментом, повлиявшим на судьбу HTML, стал выбор в качестве основы гипертекстовой базы данных обычного текстового файла, который хранится средствами файловой системы операционной среды компьютера.

Таким образом, гипертекстовая база данных в концепции WWW – это набор текстовых файлов, размеченных на языке HTML, который определяет форму представления информации (разметка) и структуру связей этих файлов (гипертекстовые ссылки).

Такой подход предполагает наличие еще одной компоненты технологии – интерпретатора языка. В World Wide Web функции

интерпретатора разделены между сервером гипертекстовой базы данных и интерфейсом пользователя.

Сервер, кроме доступа к документам и обработки гипертекстовых ссылок, осуществляет также препроцессорную обработку документов, в то время как интерфейс пользователя осуществляет интерпретацию конструкций языка, связанных с представлением информации.

В версию 4.0 включены дополнительные средства работы с мультимедиа, языки программирования, таблицы стилей(CSS), упрощенные средства печати изображений и документов, которые становятся более доступными для всех пользователей HTML 4.0. Эти дополнения служат интернационализации WWW и распространению ее по всему миру. Кроме этого, для управления сценариями просмотра страниц (гипертекстовой базы данных, выполненной в технологии World Wide Web) можно использовать языки программирования этих сценариев типа JavaScript, Java, PHP и VBScript.

Простая технология CSS расширяет возможности HTML, удобна в использовании существенно упрощает и ускоряет создания сайта.

Данный курс является базовым для изучения языков Internet-программирования: JavaScript и PHP.

1 Всемирная компьютерная сеть

Internet –это всемирная компьютерная сеть, объединяющая различные сети и отдельные компьютеры (они называются хост-компьютерами). Она обеспечивает обмен информацией между входящими в нее компьютерами независимо от их типа и используемой операционной системы. Для того чтобы все компьютеры сети «понимали» друг друга, они должны использовать единый набор правил, определяющий способ обмена информацией. Такой набор правил называется *протоколом.*

Протокол – это набор правил обмена информацией между компьютерами, установленных по взаимному соглашению.

Все компьютеры, подключенные к Internet должны использовать один и тот же протокол. В настоящее время для связи в Internet используется протокол TCP/IP. Этот протокол был разработан в 1983 году. Дата его создания считается днем рождения Internet, т. к. впервые стало возможным включить в процесс обмена информацией различные сети, каждая из которых использует свой способ передачи информации.

Фактически TCP/IP – это два протокола: IP (сокращение от англ. Internet Protocol–межсетевой протокол) и TCP (сокращение от англ. Transmission Control Protocol –протокол управления передачей). Протокол IP – это более ранний вариант, разработанный для пересылки данных, организованных в пакеты, между сетями. Система пакетной передачи информации или как ее называют система коммутации пакетов, была создана еще в 1968 г. Ее суть состоит в том, что сообщения, пересылаемые по сети, организуются в пакеты, куда входят не только сами сообщения, но и информация о маршруте: адреса компьютера отправителя и компьютера получателя (так же, как при почтовой пересылке сообщений, письма вкладываются в конверты, на которых должны быть указаны адреса получателя и отправителя).

Специальные компъютеры-маршрутизаторы, (англ. *router*) определяют путь, по которому пакеты должны следовать от одного компьютера к другому, пользуясь таблицами и алгоритмами маршрутизации.

Большие сообщения делятся на несколько пакетов, каждый со своим номером. По этим номерам компьютер-получатель должен полностью восстановить исходное сообщение после получения всех пакетов.

Недостатком протокола IP является его неспособность справиться с ошибками при передаче информации. Каждый пакет содержит контрольные данные (суммы), которые вычисляются определенным образом по исходной информации. Компьютер-получатель выполняет те же вычисления и находит контрольные суммы по полученной информации. Если они не совпадают с исходными, то в процессе передачи

произошла ошибка, и данные исказились. При искажении данных, нарушении последовательности доставки пакетов или прерывании процесса передачи компьютер получатель полностью отбрасывает пакет.

Этот недостаток устранен в протоколе управления передачей TCP. Он гарантирует, что все пакеты будут доставлены адресату, причем в том порядке, в котором были отправлены. Если пакет с каким-либо номером не пришел, т. е. полностью восстановитъ сообщение не удается, или данные в процессе передачи исказились, принимающий компьютер запрашивает повторную передачу. Таким образом протоколы TCP и IP «работают в паре»: TCP следит за целостностью данных, разбивает большие сообщения на последовательности более мелких, организует их нумерацию и последующее восстановление в единое сообщение; IP контролирует перемещение пакетов по Internet выбирая маршрут пересылки данных от одного компьютера к другому. (На самом деле под названием TCP/IP скрывается целое семейство протоколов, решающих те или иные частные задачи. Перечислим основные из них:

Транспортные протоколы TCP и UDP (User Datagram Protocol) управляют процессом передачи данных между машинами;

Протоколы маршрутизации IP, ICMP (Internet Control Message Protocol), RIP (Routing Information Protocol) обрабатывают адресацию данных, обеспечивают их физическую передачу и отвечают за выбор наилучшего маршрута до адресата;

Протоколы поддержки сетевого адреса DNS (Domain Name System), ARP (Address Resolution Protocol) обеспечивают идентификацию машины в сети по ее уникальному адресу.

Шлюзовые протоколы EGP (Exterior Gateway Protocol), GCP (Gateway-to-gateway protocol), IGP (Interior Gateway Protocol) отвечают за передачу информации о маршрутизации данных и состоянии сети, а также обрабатывают данные для взаимодействия с локальными сетями;

Протоколы прикладных сервисов FTP (File Transmission Protocol), Telnet и др. – сетевые программы, обеспечивающие доступ к различным услугам и службам Сети – например, передаче файлов между компьютерами;

Протокол SMTP (Simple Mail Transfer Protocol) отвечает за передачу сообщений электронной почты.)

Функция Internet заключается не только в том, чтобы связать компьютеры друг с другом. Основная задача Internet – обеспечить пользователя необходимой информацией и услугами. Для этого используется технология ***клиент/сервер*** (англ. *client/server).*

Клиент – это программа, принимающая информацию и услуги, предоставляемые другими программами – ***серверами.*** Клиенты

выступают в роли потребителей, а серверы – в роли поставщиков. Компьютеры, на которых запускают программы-серверы, должны обладать достаточно мощными ресурсами, т. к. им приходится «пропускать через себя» огромные объемы информации. Любой компьютер сети может выступить в роли клиента, как только на нем будет запущена какая-нибудь программа-клиент. Схема взаимодействия программы-клиента и программы-сервера приведена на рис.1.1

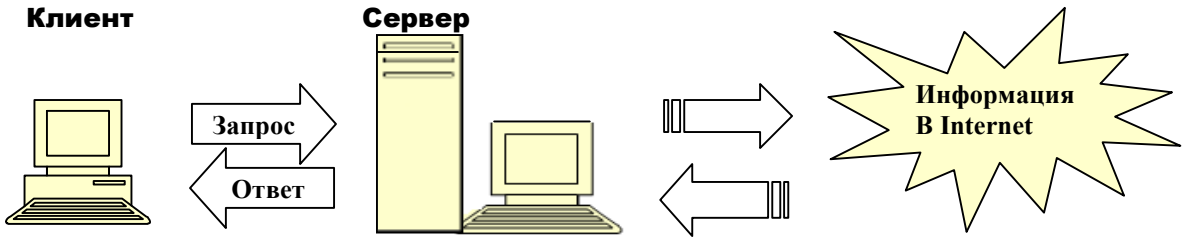


Рис.1.1

Доступ к Internet может получить любой пользователь компьютера, на котором установлен модем и необходимое ПО. Для подключения к Internet частных лиц, школ, институтов и любых других учреждений существуют специальные организации – ***провайдеры*** (от англ. *provide* – обеспечивать) услуг Internet. Если компьютер входит в состав локальной сети (например, школьной), которая подключена к Internet, то необходимую для работы информацию (адрес компьютера и др.) следует получить у администратора сети (человека, управляющего работой локальной сети).

Система адресации в Internet

Каждый компьютер, подключенный к Internet, должен иметь свой адрес. В Internet используется два типа адресов: *цифровые* или IP-адреса и ***доменные*** (от англ. *domain* – область, сфера). Рассмотрим структуру каждого из типов.

IP-адрес по смыслу аналогичен почтовому индексу, содержащему информацию о городе (первые три цифры) и почтовом отделении (вторые три цифры). IP-адрес представляет собой последовательность из четырех чисел, разделенных точками:

Например, 198.162.201.204.

Каждое из чисел занимает 1 байт = 8 битов (поэтому их часто называют *октетами),* т. е. может принимать значение от 0 до 255. Левая часть IP-дреса определяет конкретную сеть в Internet и называется

сетевым идентификатором (англ. *Network ID)*. Правая часть IP-адреса определяет конкретный компьютер в этой сети и называется ***идентификатором компьютера*** (англ. *Host ID)*. Используется 3 класса IP-адресов: A, B и C.

Класс IP-адреса определяет, сколько октетов отводится под адрес сети и сколько под адрес компьютера.

IP-адреса класса A предназначены для работы с небольшим количеством (до 126) сетей, содержащих большое число компьютеров (до 16777214). Поэтому в таких адресах один октет – самый левый – задает адрес сети, а три правых – адрес компьютера в этой сети(Рис.1.2).

.

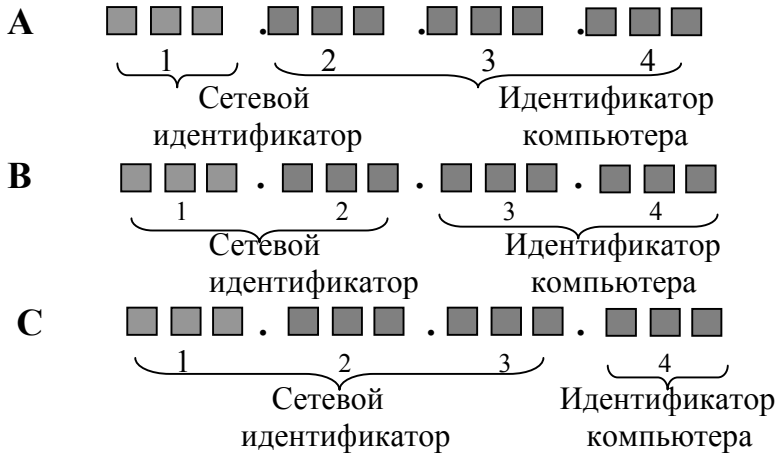


Рис.1.2

IP-адреса класса B предназначены для работы со средним количеством сетей (до 16384), содержащих среднее число компьютеров (до 65534). В таких адресах два левых октета – это сетевой идентификатор, а два правых – идентификатор компьютера.

IP-адреса класса C предназначены для работы с большим количеством сетей (до 2 097 092), содержащих малое число компьютеров (до 254). В таких адресах три левых октета содержат идентификатор сети, а самый правый – идентификатор компьютера.

К какому классу принадлежит IP-адрес, определяют по значению первого октета: если в первом октете число от 1 до 126 – это IP-адрес класса A, если от 128 до 191 – класса B, если от 192 до 223 – класса C.

Для передачи сообщений на конкретные компьютеры в Internet протокол TCP/IP и программы-клиенты используют IP-адреса.

Однако для восприятия пользователем IP-адрес неудобен. Человеку привычнее работать с именами. Поэтому пользователи обычно работают с ***доменными адресами*** – уникальными именами компьютеров в Internet. Доменный адрес, так же как и IP-адрес, состоит из частей, разделенных

точками. Однако в отличие от IP-адреса, уточняющего место назначения слева направо, доменный адрес делает это в обратном порядке – справа налево: вначале идет имя компьютера, а затем – имя сети, в которой он находится.

Чтобы пользователям Internet было проще связываться друг с другом, все пространство ее адресов разбито на области – домены. Возможно также разделение по определенным признакам внутри доменов. Доменный адрес компьютера включает в себя, как минимум, два уровня доменов(Рис.1.3):

Рис.1.3

Самый правый – домен первого уровня, следующий слева – его поддомен – домен второго уровня и т. д.

Домен первого уровня определяет страну или тип организации, которой принадлежит компьютер. Существуют установленные двухбуквенные сокращения для доменов стран. Например, Украина – ua; Россия – ru; США – us, Франция – fr и т. д.

Домены типов организаций обычно имеют трехбуквенные сокращения. Например, университеты и другие учебные заведения – edu, правительственные учреждения – gov, коммерческие организации – com, провайдеры услуг – net и т. д.

Домен второго уровня определяет организацию, которая владеет или управляет сетью, содержащей данный компьютер. Обычно имя этого домена совпадает с названием соответствующей фирмы или ее торговой маркой.

Имя компьютера указывает конкретный компьютер в сети, определенной доменами первого и второго (а, возможно, и следующих) уровней. Оно регистрируется только в этой сети и только эта сеть «ответственна» за передачу информации конкретному компьютеру-адресату.

Примеры доменных имен:

KIM.UNIVER.KHARCOV.UA

Для того чтобы сетевой компьютер «понял», куда следует передать сообщение, доменный адрес должен быть преобразован в IP-адрес. Значит где-то в Internet должны храниться таблицы соответствия доменных и IP-адресов. Естественно, что «обычный» компьютер пользователя сети не может и не должен знать все IP-адреса в Internet. Такие таблицы хранятся на специальных серверах, называемых DSN-серверами (сокращение от англ. *Domain Name System* –система доменных имен). DSN-серверы

разбросаны по всей Internet. Каждый из них хранит информацию о большом числе компьютеров Internet и способен мгновенно преобразовать доменное имя в IP-адрес. Если IP-адрес запрошенного компьютера не известен данному DSN-серверу, он обратится к ближайшему DSN-серверу и т. д. по цепочке, пока требуемый адрес не будет найден. Весь процесс займет несколько секунд. Адрес одного из DSN-серверов пользователь должен указать при настройке компьютера для работы в Internet (такой адрес можно получить у провайдера услуг Internet либо у администратора локальной сети).

World Wide Web

World Wide Web (WWW) или просто Web в переводе означает всемирная паутина (сеть). Web – это всемирная информационная сеть, представляющая собой огромный набор взаимосвязанных друг с другом документов. Они называются Web-*страницами* и расположены на сотнях тысяч компьютеров – Web-серверах, разбросанных по всему миру. Для просмотра Web -страниц используются программы-клиенты, называемые Web-*броузерами* (от англ. *to browse* – просматривать, пролистывать). Самыми популярными в настоящее время Web-броузерами являются Internet Explorer (от англ. *explorer* – исследователь) фирмы Microsoft и Netscape Navigator (от англ. *Navigator* – навигатор) фирмы Netscape.

Web – одна из новейших, но быстро завоевавших широкую популярность информационных служб Internet. Она была изобретена в 1980 году в европейской лаборатории физики элементарных частиц. Сотрудник этой лаборатории Тим Бернес Ли создал программу, позволившую связать документы между собой посредством введения в них ссылок друг на друга (именно так, он полагал, организована информация в человеческом мозге). Так появилось понятие ***гипертекста*** (англ. hypertext*)*. Приставка «гипер» означает «более чем», т. е. гипертекст – это более, чем обычный текст.

Гипертекст – это текст (документ), содержащий в себе ссылки на другие документы.

Справочная система Windows представляет собой гипертекст. Слово или фраза, являющаяся ссылкой или, как ее называют ***гиперссылкой,*** обычно подчеркивается и выделяется другим цветом, чтобы отличить ее от остального текста. Щелчок на гиперссылке выводит на экран связанный с ней текст (документ).

В 1989 г. автор идеи гипертекста предложил ее обобщить: связать гипертекстовые документы – Web страницы – по всему миру. С тех пор началось бурное развитие Web. Именно гипертекст лежит в основе структуры Web. Благодаря ему Web-страница становится интерактивной.

У гипертекста нет начала и нет конца, он напоминает паутину с неопределенным центром. В этом важная особенность гипертекста: информация никак не упорядочивается, документы просто связываются друг с другом с помощью ссылок. При этом могут связываться самые разные документы, находящиеся в самых разных местах общей структуры.

Поскольку современные документы могут содержать не только текст, но и изображения, звуковую и видеоинформацию, то для описания Web - страниц используется также термин ***гипермедиа*** (англ. *hypermedia)*.

Гипермедиа – это естественное обобщение понятия гипертекста, относящееся к документам, содержащим не только текст, но и информацию мультимедиа (графику, звук, видео).

Web-страница –это документ гипермедиа. По щелчку на гипермедиа-ссылке Web-броузер загрузит соответствующий мультимедийный файл и запустит приложение для воспроизведения его содержимого. Современные Web-броузеры поддерживают технологию, которая позволяет воспроизводить видео- и аудиозаписи непосредственно при отображении Web-страницы, т. е. так, как будто мультимедийный файл является частью документа. Элементы мультимедиа Web-страницы можно связать с другими документами гипермедиа, чтобы, например, при щелчке на изображении загружался соответствующий ему документ с информацией (возможно, мультимедийной).

Для создания Web-страниц используется специальный язык – ***HTML*** (сокращение от англ. *Hyper Text Mapkup Longuage* – язык гипертекстовой разметки). ***HTML-документ*** – это обычный текстовый файл с расширением ***htm***. или .***html.*** Web-страница хранится как HTML-документ. Главная цель HTML – описать внешний вид документа. Для этого в текст документа вставляются специальные коды – *дескрипторы* или, как их еще называют, *тэги* (от англ. *teg* – этикетка, бирка). Они определяют способы форматирования текста и позволяют связать слова и фразы документа с другими документами в Internet. Создать Web-страницу можно в любом текстовом редакторе. Существуют также специальные HTML-редакторы для создания HTML-документов.

У всех ресурсов в Internet в том числе и у Web-страниц, есть свой собственный адрес, заданный в виде ***URL*** (сокращение от англ. *Uniform Resourse Location* – унифицированный локатор ресурсов).

URl (Uniform Resource Locator) или универсальный указатель ресурса – адрес некоторого объекта в Интернете. Типичный URL для WWW имеет вид:

http: /www.название.домен/имя файла

URL – это стандарт, принятый в Internet для определения местонахождения любого ресурса, будь то документ или служба.

URL состоит из трех частей(Рис.1.4):

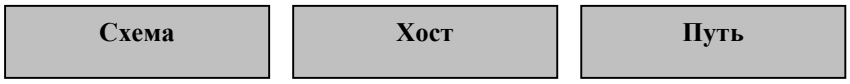


Рис.1.4

I**.** ***Схема*** описывает протокол прикладной программы(*В отличие от TCP/IP, работающего на «нижнем» уровне, прикладные программы, обеспечиеающие доступ к ресурсам Internet., используют протоколы «верхнего» уровня.*), который используется для доступа к ресурсу. Чаще всего это протокол ***HTTP*** (сокращение от англ. *Hyper Text Тгапз{ег Protoco1* – протокол передачи гипертекста). За протоколом ставятся символы *«://».* Если ресурсом является файл, то схема имеет вид «**file:**//**»,** если адрес электронной почты, то «**mailto**:», если новости, то «**news**:».

II. ***Хост*** – это доменное имя компьютера, на котором находится ресурс. Домен третьего уровня в этом имени обычно указывает на тип ресурса, например, www.host.com – это имя сервера Web-страниц, а ftp.host.com – имя FTP-сервера. **Web-хостинг** – услуга сторонней организации, которая заключается в размещении на жестком диске ее Web-сервера файлов, созданного вами Web-сайта.

III. ***Путъ*** — это полный путь к документу и, возможно, его имя. Имена каталогов отделяются друг от друга символом «/». Вот примеры различных типов URL.

Типичный URL для WWW имеет вид:

http: /www.название.домен/имя файла

- http://www.microsoft.com/windows
- http://www.boutell.com/fag/ (здесь содержатся ответы на часто задаваемые вопросы о Web)
- http://www.yahoo.com
- news:alt.internet.services
- mailto:irina.zaretskaya@univer.kharkov.ua
- file://NEW/prog.exe

Путь не является обязательным элементом URL. Если ввести URL Web-сервера, не указав при этом путь к HTML-документу, то сервер откроет свою *началъную* или, как ее еще называют *основную* или *домашнюю* страницу (от англ. *Home page* – домашняя страница). Каждый Web-сервер имеет свою начальную страницу, которая появляется по умолчанию при обращении к нему.

В последнее время появилась тенденция к «оживлению» Web-страниц: анимации изображений, использованию бегущих строк, движущихся пиктограмм на кнопках, мультипликации. Используются

также эффекты «превращения»: при помещении курсора мыши на некоторый объект он меняет свой вид. Появилось даже связанное с этим понятие: «активное содержимое». Например, при попадании курсора мыши на заголовок меняется его цвет и размер букв. Для каждого элемента Web-страниц можно написать свой сценарий поведения. Такие эффекты достигаются за счет использования при написании Web-страниц специальных средств: так называемых элементов управления ActiveX, программ-сценариев (их называют аплетами от англ. *ap1e)* на языке Java, надстроек (англ. *aplets)* и др.

Служба FTP

FTP (сокращение от англ. *File Transfer Protocol* — протокол передачи файлов) — это принятый в Internet протокол для передачи файлов между компьютерами. Служба ***FTP*** – одна из первых, но она широко используется и в настоящее время. Файлы, предназначенные для открытого доступа, содержатся на множестве серверов. Программа ***FTP-клиент*** (т. е. программа, использующая протокол ***ftp***) позволяет установить связь с одним из таких серверов и использовать набор команд для работы с ним: просмотра каталога, поиска файлов, управления их перемещением. Служба FTP может использоваться как для получения файлов с удаленных серверов, так и для отправки своих файлов на сервер, чтобы другие пользователи могли ими воспользоваться. Возможна пересылка как текстовых, так и двоичных файлов. Поскольку большинство FTP-серверов работает под управлением операционной системы UNIX.

2 Работа в сети Internet с общей позиции

Схема клиент-сервер

После того, как вы набрали адрес в **браузере** (программе, которая позволяет просматривать страницы в Internet), с вашего компьютера посылается запрос на компьютер, адрес которого вы написали. При удачном соединении тот компьютер посылает результат – обычно, текст в формате HTML.

Схема работы ***клиент-сервер***, очень широко распространена в программировании и имеет название ***технология клиент-сервер***. Работа с ***базами данных*** (совокупность связанных сведений, представленных в пригодной для хранения, передачи и автоматизированной обработки) осуществляется также по этой технологии. В этом случае приложение-клиент посылает запрос ***системе управления базами данных*** (СУБД), а она посылает результат-набор данных. Представленная выше схема для наших целей требует уточнения.

1. Обработка запроса WEB-сервером. В простейшем случае в адресной строке после имени компьютера (rsc.pp.ru) расположен путь (может быть пустой), указывающий расположение документа, который надо послать клиенту. Однако, подобное встречается хоть и часто, но не всегда. Довольно часто в строке указывается путь не к файлу, который надо переслать, а к файлу, содержащему программу, которую надо запустить. Результатом же работы этой программы является тот документ, который надо послать клиенту. Указанная программа часто называется ***скриптом***, точнее, скриптом, запускаемым на стороне сервера.

2. Иногда, строка после названия сервера вообще никак не связана с расположением документов на сервере и обрабатывается программой, написанной WEB-программистом, которая возвращает в качестве результата документ. Такая схема повышает секретность хранения информации

Для чего нужны скрипты? Довольно часто WEB-сервер должен возвращать разную информацию для различных людей, а также информацию, которая изменяется с течением времени. Для нас наиболее актуальный пример – *форум.*

В случае, если на стороне сервера скрипты не используются сайт называют ***статическим***, иначе ***динамическим***.

Однако, в использовании скриптов имеется ещё одно удобство. Часто почти все страницы сайта сделаны по одному *шаблону*. Если сайт является статическим, то при изменении меню нужно изменять все файлы, содержащие HTML-текст, если же сайт динамический, то можно меню подключать ко всем страницам автоматически (что серьёзно упрощает работу администратора).

Было сказано, что существуют скрипты, выполняемые на стороне сервера. Естественно, такое название означает, что существуют ***скрипты, выполняемые на стороне клиента***. Часто при вводе информации перед посылкой её на сервер происходит проверка её корректности на стороне клиента (с помощью скрипта). Такой приём серьёзно снижает нагрузку на сервер и повышает удобство для пользователя.

На стороне сервера для хранения информации используются специальные хранилища (базы данных) и программы (СУБД). В этом случае скрипты, выполняют запросы к СУБД, последняя возвращает результаты запросы-данные, а скрипт формирует документ, который передаётся браузеру.

Гипертекст породил много специальных терминов:

Элемент(element) – конструкция языка HTML.Это контейнер, содержащий данные и позволяющий отформатировать их определенным образом. Любая страница представляет собой набор элементов. Одной из основных идей гипертекста – возможность вложения элементов.

Тег – начальный и конечный маркеры элемента. Теги определяют границы действия элементов и отделяют элементы друг от друга. В тексте Web-страницы теги заключаются в угловые скобки, а конечный тег всегда снабжается косой чертой.

Атрибут(attribute) – параметр или свойство элемента.

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

Фрейм(frame) – этот термин имеет два значения. Первое – область документа со своими полосами прокрутки. Второе – одиночное изображение в сложнос (анимационном) графическом файле (аналогии с кадром кинофильма).

HTML-файл или ***HTML***-страница – документ, созданный в виде гипертекста на основе языка HTML. Такие файлы имеют, как правило, расширение ***html*** или ***htm***. В гипертекстовых редакторах и броузерах эти файлы имеют общее название «документ».

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

GGI(Common Galery Interface) – общееназвание для программ, которые работают на сервере, позволяют расширить возможности Web-страниц. Например, без таких программ невозоможно создание интерактивных страниц.

Код HTML – гипертекстовый документ в своем первоначальном виде, когда видны все элементы и атрибуты.

World Wide WWW или просто **Web –** Всемирная паутина, распределенная система доступа к гипертектовым документам, существующая в Интернете. HTML является основным языком для создания документов в WWW. Изучая его, мы, фактически, изучаем часть этой системы, хотя область применение языка гораздо шире.

Web-страница – документ (файл), подготовленный в формате гипертекста и размещенный в Word Wide Web.

Сайт (site) – набор Web-страницы, принадлежащий одному владельц**у.**

Браузер (brawser) **–** программа для просмотра Web-страниц.

Web-хостинг – услуга сторонней организации, которая заключается в размещении на жестком диске ее Web-сервера файлов, созданного вами Web-сайта.

CSS – язык описания стилей, описываются типы шрифтов, способы расположения информации в документе HTML и другие дизайнерские аттрибуты.

JavaScript (JScript), VBScript, Java, COM-объекты, Flash – представленные средства (первые три из них языки программирования)

предоставляют возможность для написания скриптов, исполняемых на стороне клиента.

PHP, Perl, любые другие языки (включая Prolog) – используются для написания скриптов, исполняемых на стороне сервера.

СУБД **(**например, ***MySQL***) **–**программа, организующая работу с базами данных (на сервере)

WEB-сервер **(**например**,** ***Apache***) **– п**рограмма, обрабатывающая запросы браузеров.

3 Общие принципы создания Web –узла

Рекомендации по создания Web –узла

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

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

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

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

Для того, чтобы правильно ответить на поставленные вопросы, необходимо сформировать категории пользователей, на которые рассчитан

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

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

Типичная структура Web-узла фирмы обычно представлена так:

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

2. Информация о продукции и услугах**.** Разместите на Web-странице фотографии или рисунки своей продукции. Опишите ее свойства и преимущества, приведите примеры использования. Если имеется бумажный каталог продукции, то можно перенести его структуру и содержание в Web-узел. Это облегчит создание и дальнейшее обновление электронного варианта каталога. Если планируется прием заказов на продукцию или услуги через Internet, то нужно разместить здесь бланк заказа, который будет поступать по электронной почте.

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

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

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

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

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

помощью поисковых серверов. Что касается авторитетности, то все зависит от того, насколько тщательно вы подберете информацию, проверите ее и будете своевременно обновлять.

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

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

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

После того, как определены цели, задана структура и собрана текстовая и графическая информация, необходимо разработать внешний вид Web-узла. Он также зависит от целей, которых необходимо достичь. Спектр возможных решений здесь очень широк: от просмотра уже существующих страниц и создания подобных до обращения за помощью к профессиональным дизайнерам и художникам. В то же время, необходимо помнить о некоторых уже сложившихся правилах построения Web-документов, из которых состоит Web-узел.

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

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

3. Навигационная панель**.** Одним из наиболее важных разделов Web-документа является навигационная панель или панель управления. WWW завоевала весь мир во многом благодаря тому, что гипертекстовые ссылки обеспечивают полную связность публикуемых материалов. Но эти же ссылки таят в себе опасность погружения в полный хаос, когда, пройдя цепочку из трех-четырех документов, вы уже не сможете вернуться обратно, запутавшись в обилии ссылок. Ваш Web-узел должен обеспечивать пользователю ясные и интуитивно понятные навигационные маршруты. Многочисленные исследования показали, что посетители Web-серверов очень нетерпеливы и дальше, чем на два уровня документов, углубляться в содержание сервера не хотят. Поэтому, создавая Web-узел большого объема, следует предусмотреть промежуточные документы, обычно находящиеся на первом-втором уровнях, от которых любая информация находится не далее, чем в двух переходах. Навигационная панель вашего Web-узла должна присутствовать в каждом документе. В первую очередь, она должна включать в себя направляющие ссылки типа "Вперед"-"Назад" ("Следующий"-"Предыдущий"), указывающие на соседние документы в структуре Web-узла. Далее от панели управления обязательно должны идти ссылки на все крупные разделы Web-узла – так называемые разделы первого уровня. И, наконец, пользователь всегда должен иметь возможность мгновенно вернуться на главную страницу Web-узла. Помимо ссылок следует указать путь к локальной поисковой системе и индексу.

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

Часто возникает вопрос о размерах документа: какое число страниц является оптимальным? Ответ на первый взгляд может показаться странным: одна экранная страница или вообще никаких ограничений. Многочисленные исследования показали, что пользователи не любят работать с полосами прокрутки браузеров. Больше всего им нравятся документы, которые размещаются на одной экранной странице. Так и в WWW – вы никоим образом не сможете дать пользователю больше

информации, чем в концентрированном изложении на одной странице. Если все-таки вы не укладываетесь в эти рамки, создайте еще один документ.

Одна экранная страница оказалась подходящей мерой представления информации. Если размер документа превышает одну страницу, то в большинстве случаев он может быть поделен на несколько логических частей, каждая из которых будет занимать не более одной страницы. Если же логического деления информации произвести не удается, то необходимо переработать стиль изложения, а может быть, и сами материалы. Сейчас выработалось единое мнение, что Web-сервер необходимо строить на основе одноэкранных документов. Есть только два исключения из этого правила. Оно не распространяется на статьи, публикуемые в WWW, и второе исключение – анкетные формы, которые, естественно, нельзя разрывать.

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

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

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

Рассмотрим второй вариант. Правильный выбор провайдера, предоставляющего доступ к Web-странице, позволит вашим клиентам с

максимальным удобством получать необходимую информацию. Кроме того, поддержка Web-сервером специальных возможностей значительно облегчит разработку Web-узла.

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

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

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

3. Поддержка сервером провайдера CGI-сценариев. CGI (Common Gateway Interface, общий шлюзовой интерфейс) – спецификация, позволяющая Web-серверу выполнять произвольные прикладные программы. В результате работы таких программ (сценариев, или "скриптов") создаются HTML-документы. С помощью CGI-сценариев могут приниматься данные от пользователя, они позволяют организовать диалог на Web-страницах, запросы к базам данных и т.д. Создать CGI-сценарий можно с помощью любого популярного языка программирования: Perl, Basic, C, C++, Pascal и т.п.

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

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

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

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

4 Язык HTML

История развития HTML

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

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

Причем последняя технология постепенно перемещается на первое место.

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

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

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

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

Разработчики HTML смогли решить две задачи:

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

Первая задача была решена за счет выбора теговой модели описания документа. Такая модель широко применяется в системах подготовки документов для печати.

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

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

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

Таким образом, гипертекстовая база данных в концепции WWW – это набор текстовых файлов, размеченных на языке HTML, который определяет форму представления информации (разметка) и структуру

связей между этими файлами и другими информационными ресурсами (гипертекстовые ссылки). Гипертекстовые ссылки, устанавливающие связи между текстовыми документами, постепенно стали объединять самые различные информационные ресурсы, в том числе звук и видео; в результате возникло новое понятие – гипермедиа.

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

Сейчас World Wide Web Consortium (W3C) – международная организация, которая занимается подготовкой и распространением документации на описание новых версий HTML – уже опубликовала материалы спецификации HTML 4.01. Кроме возможностей разметки текста, включения мультимедиа и формирования гипертекстовых связей, уже существовавших в предыдущих версиях HTML, в версию 4.01 включены дополнительные средства работы с мультимедиа, языки программирования, таблицы стилей, упрощенные средства печати изображений и документов. Для управления сценариями просмотра страниц Website (гипертекстовой базы данных, выполненной в технологии World Wide Web) можно использовать языки программирования этих сценариев, например, JavaScript, Java и VBScript. Усложнение HTML и появление языков программирования привело к тому, что разработка Web-узлов стала делом высокопрофессиональным, требующим специализации по направлениям деятельности и постоянного изучения новых Web-технологий.

Принципы гипертекстовой разметки

HTML является описательным языком разметки документов, в нем используются указатели разметки (теги). Теговая модель описывает документ как совокупность контейнеров, каждый из которых начинается и заканчивается тегами, то есть документ HTML представляет собой не что иное, как обычный ASCII-файл, с добавленными в него управляющими HTML-кодами (тегами). Поскольку HTML произошел от SGML, в нем разрешено использовать только три управляющих символа: горизонтальную табуляцию, перевод каретки и перевод строки. Это облегчает взаимодействие с различными операционными системами.

Теги HTML-документов в большинстве своем просты и понятны, ибо они образованы с помощью общеупотребительных слов английского

языка, понятных сокращений и обозначений. HTML-тег состоит из имени, за которым может следовать необязательный список атрибутов тега. Текст тега заключается в угловые скобки ("<" и ">"). Простейший вариант тега – имя, заключенное в угловые скобки, например <HEAD> или <I>. Для ряда тегов характерно наличие атрибутов, которые могут иметь конкретные значения, устанавливаемые автором для изменения функции тега.

Например, при описании таблицы открывающий тег с атрибутами может выглядеть так:

```
<TABLE WIDTH=570 ALIGN=center CELLPADDING=10
       CELLSPACING=2 BORDER=16>
```

Эта запись означает следующее: таблица шириной 570 пикселов, выровнена по центру, поле между рамкой и содержимым ячеек 10 пикселов, поле рамки 2 пиксела, ширина бордюра 16 пикселов.

Атрибуты тега следуют за именем и отделяются друг от друга одним или несколькими знаками табуляции, пробелами или символами возврата к началу строки. Порядок записи атрибутов в теге значения не имеет. Значение атрибута, если таковое имеется, следует за знаком равенства, стоящим после имени атрибута. Если значение атрибута – одно слово или число, то его можно просто указать после знака равенства, не выделяя дополнительно. Все остальные значения необходимо заключать в одинарные или двойные кавычки, особенно если они содержат несколько разделенных пробелами слов. Длина значения атрибута ограничена 1024 символами. Регистр символов в именах тегов и атрибутов не учитывается, чего нельзя сказать о значениях атрибутов. Например, особенно важно использовать нужный регистр при вводе URL (Universe Resource Locator, унифицированный указатель ресурса), других документов в качестве значения атрибута HREF.

Чаще всего элементы разметки HTML или HTML-контейнеры состоят из начального и конечного компонентов, между которыми размещаются текст и другие элементы документа. Имя конечного тега идентично имени начального, но перед именем конечного тега ставится косая черта (/) (например, для тега стиля шрифта – курсив <I> закрывающая пара представляет собой </I>, для тега заголовка <TITLE> закрывающей парой будет </TITLE>). Конечные теги никогда не содержат атрибутов. По своему значению теги близки к понятию скобок "begin/end" в универсальных языках программирования, которые задают области действия имен локальных переменных и т.п. Теги определяют область действия правил интерпретации текстовых документов.

При использовании вложенных элементов разметки в документе следует соблюдать особую аккуратность. Вложенные теги нужно

закрывать, начиная с последнего. Некоторые элементы разметки не имеют конечного компонента, поскольку являются автономными элементами. Например, тег изображения <IMG>, который служит для вставки в документ графического изображения, конечного компонента не требует. К автономным элементам разметки также относятся разрыв строки (
), горизонтальная линейка (<HR>) и теги, содержащие такую информацию о документе, которая не влияет на его отображаемое содержимое, например теги <META> и <BASE>.

В некоторых случаях конечные теги в документе можно опускать. Большинство браузеров устроено так, что при обработке текста документа начальный тег воспринимается как конечный тег предыдущего. Самый распространенный тег такого типа – тег абзаца <P>. Поскольку он используется в документе очень часто, его обычно ставят только в начале каждого абзаца. Когда один абзац заканчивается, следующий тег <P> сигнализирует браузеру о том, что нужно завершить данный абзац и начать следующий. Большинство авторов тегом конца абзаца не пользуются.

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

Для краткости и образности мы будем в ряде случаев вместо словосочетания "элемент разметки" применять термин "контейнер".

Общая схема построения контейнера в формате HTML может быть записана в следующем виде:

```
"контейнер"=
<"имя тега" "список атрибутов">
содержание контейнера
</"имя тега">
```

Следует отметить, что в литературе кроме термина "контейнер" еще используется и термин "элемент". Следует быть внимательным, чтобы не путать контейнер (например, BODY) и тег (BODY), используемый при формировании контейнера.

Кроме тегов, элементами HTML являются CER (Character Entity Reference), они предназначены для представления специальных символов в документе HTML, которые могут быть неверно обработаны браузером. Предположим, создается документ HTML, речь в котором идет об элементах данного языка. Если указать имя тега <BODY> просто в документе, браузер может воспринять его как непосредственно старт-тег. Для вывода таких символов и используется CER.

Например, чтобы представить символ "<" в документе HTML, нужно заменить его на <, а символ ">" – на >. То есть, если указать в тексте HTML строку <BODY>, она будет выглядеть на экране как текст <BODY>.

Может возникнуть вопрос: как быть с символами "</>", "&" и со специальными символами, типа знака ударения? Можно выводить их, используя соответствующие CER, например для "&" это будет &, и т.д.

CER легко обнаружить, если посмотреть на структуру любого документа HTML, поскольку каждый из них начинается с амперсанта "&". В отличие от наименований тегов HTML, наименования CER чувствительны к регистру символов. Также наименования CER могут задаваться не в виде имени, а с помощью трехзначных кодов символов в виде &#nnn;. Далее в таблице 4.1 приведены наиболее часто используемые CER и соответствующие им числовые коды.

Таблица 4.1.

Числовой код	**Именная замена**	**Символ**	**Описание**
"	"	"	Кавычка
&	&	&	Амперсант
<	<	<	Меньше
>	>	>	Больше
			Неразрывный пробел
¡	¡	¡	Перевернутый восклицательный знак
¢	¢	¢	Цент
£	£	£	Фунт
¤	¤	¤	Валюта
¥	¥	¥	Йена
¨	¨	¨	Умляут
©	©	©	Копирайт
«	«	«	Левая угловая кавычка
®	®	®	Зарегистрированная торговая марка
±	±	±	Плюс или минус
»	»	»	Правая угловая кавычка

Группы тегов HTML

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

– определяющие структуру документа;

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

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

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

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

Типичным примером использования такой записи можно считать следующий пример (рис. 4.1):

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

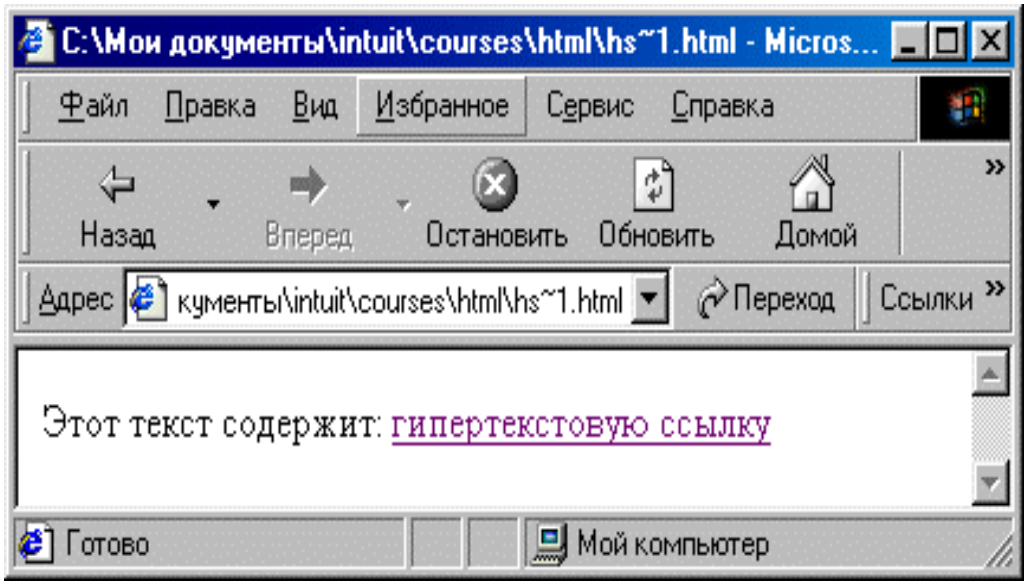


Рис. 4.1

В приведенном выше примере тег "A", который в HTML называют якорем (anchor), использует атрибут HREF, обозначающий гипертекстовую ссылку (Hypertext Reference), для записи этой ссылки в форме URL. Данная

ссылка указывает на документ с именем "index.html" в каталоге "help" на сервере "www.intuit.ru", доступ к которому осуществляется по протоколу HTTP.

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

Структура HTML-документа позволяет задействовать вложенные друг в друга контейнеры. Собственно, сам документ – это один большой контейнер, который начинается с тега <HTML> и заканчивается тегом </HTML>.

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

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

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

Структура HTML-документа и элементы разметки заголовка документа

Язык HTML (HyperText Markup Language - язык разметки гипертекста) предназначен для описания вида документов, содержащих текст и некоторые дополнительные структуры (такие как таблицы, списки). Кроме того, можно вставлять ссылки между частями документа, с которыми удобно работать в браузере.

В мире существует огромное количество версий языка HTML. Во-первых, существуют спецификации языка HTML, принятые консорциумом W3C. Во-вторых, в каждом браузере реализованы не все возможности, имеющиеся в спецификации, однако, существуют свои собственные "фишки". Поэтому профессиональные сайты необходимо тестировать на максимально возможном количество браузеров (в настоящее время уместно использовать Internet Explorer 4.x-6.x, Mozilly, Opery и Netscape).

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

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

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

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

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

```
<HTML>
<HEAD>
<TITLE>Документ с фреймами</TITLE>
</HEAD>
```

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

Назначение заголовка

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

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

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

Отображение содержания элемента TITLE

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

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

Общие гипертекстовые связи определяются не частью документа (контекстом), а всем документом целиком. Например, быть предыдущим

по отношению к другому документу или следующим – это общая гипертекстовая связь, которая позволяет организовать так называемый "линейный" просмотр информационных узлов гипертекстовой сети.

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

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

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

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

Основные контейнеры заголовка

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

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

- HEAD (элемент разметки HEAD);
- TITLE (заглавие документа);
- BASE (база URL);
- ISINDEX (поисковый шаблон);
- META (метаинформация);

– LINK (общие ссылки);
– STYLE (описатели стилей);
– SCRIPT (скрипты).

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

Элемент разметки HEAD

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

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

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

Элемент разметки TITLE

Элемент разметки TITLE служит для именования документа в World Wide Web. Состоит контейнер из тега начала, содержания и тега конца. Наличие тега конца обязательно. Тег начала элемента не имеет специфических атрибутов.

В различных браузерах алгоритм отображения элемента TITLE может отличаться.

При выборе текста для содержания контейнера TITLE следует учитывать, что отображается он системным фонтом, так как является заголовком окна браузера. В нелокализованных версиях операционных

систем и графических оболочек русский текст содержания элемента TITLE будет отображаться абракадаброй.

Синтаксис контейнера TITLE в общем виде выглядит следующим образом:

```
<TITLE>название документа</TITLE>
```

Заголовок не является обязательным контейнером документа. Его можно опустить. Роботы многих поисковых систем используют содержание элемента TITLE для создания поискового образа документа. Слова из TITLE попадают в индекс поисковой системы. Из этих соображений элемент TITLE всегда рекомендуется использовать на страницах Web-узла.

Элемент разметки BASE

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

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

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

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

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

```
<BASE HREF=http://intuit.ru/start/>
<HTML>
<HEAD>
```

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

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

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

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

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

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

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

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

Потребность в этом возникает при организации постоянно отображаемых меню. Такое меню может быть реализовано либо во фрейме, либо в окне. При этом информационные страницы Web-узла,

которые загружаются при активизации гипертекстовых ссылок, будут загружаться в другое окно или фрейм.

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

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

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

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

Элемент разметки ISINDEX

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

Шаблон ввода ключевых слов при наличии данного контейнера в заголовке HTML-документа отображается в виде дополнительного поля ввода рабочей области браузера, что нарушает компоновку HTML-страниц, выполненных с применением современных средств разметки. Больше всего ISINDEX подходит для документов с компоновкой в стиле HTML 2.0.

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

Применение элемента ISINDEX

В классическом варианте при использовании ISINDEX список ключевых слов, которые вводятся в поисковом шаблоне и разделены

символом "+", присоединяется к базовому адресу HTML-документа после символа "?".

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

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

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

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

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

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

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

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

Во втором случае в документ встраивается обращение к CGI-скрипту, который реализует функции поисковой программы. Такое совмещение –

свойство современного подхода к компоновке поисковых страниц. Как правило, и поисковый шаблон, и результаты поиска отображаются на одной странице, так как это позволяет корректировать запрос по мере получения результатов поиска. Встроеный в страницу скрипт анализирует переменные окружения сервера, и в случае отсутствия запроса может вообще никак не обнаруживать свое присутствие внутри документа.

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

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

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

```
Введите ключевые слова:
```

Применение атрибута PROMPT

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

Элемент разметки META

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

Впервые контейнер META был задействован при принудительной перезагрузке документа браузером через заголовок HTTP-сообщения. В заголовке HTTP-сообщения можно указать оператор refresh. Время, заданное как параметр этого оператора, определяет интервал в секундах, после которого браузер загружает документ, определенный атрибутом URL данного оператора.

В контейнере META подобный механизм реализуется следующим образом:

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

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

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

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

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

В Windows 95 и Windows NT 4.0 с поддержкой таблиц UNICODE появилась возможность указывать тип кодировки документа – CHARSET. Для перекодировки на стороне клиента (документ подготовлен в кодировке cp1251) в заголовок документа необходимо включить META-тег следующего вида:

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

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

Кроме Content-type, можно указать и другие операторы. Например, запретить кэширование документа. Необходимость в этом возникает при частом обновлении документа или наличии в нем изменяющихся SSI-

вставок. Для запрета кэширования достаточно вставить в заголовок META-тег вида:

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

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

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

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

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

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

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

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

При индексировании такого документа содержимое контейнера TITLE и атрибутов CONTENT контейнеров META после фильтрации попадет в индекс поисковой машины и может быть использовано для составления

запросов. Процесс фильтрации отбракует так называемые stop-слова и общие слова. Они не попадут в индекс поисковой машины. В частности, будут отбракованы предлоги или, если речь идет о тематическом поисковом индексе, например по технологиям World Wide Web, то в него не попадут: web, Web-технология и т.п.

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

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

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

Элемент разметки LINK

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

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

Общие ссылки нельзя привязать по контексту. Например, два информационных узла находятся в отношении следования, т.е. при "линейном" просмотре одна Web-страница является следующей для другой Web-страницы. В этом случае речь идет о страницах целиком, а не об

отдельных их частях. Такой же общей связью является принадлежность к Web-узлу, который ассоциируется со своей домашней страницей.

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

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

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

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

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

В общем случае контейнер LINK может имеет следующий вид:

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

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

Элемент разметки STYLE

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

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

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

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

Элемент разметки SCRIPT

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

```
<script language="JavaScript">
<SCRIPT LANGUAGE="JavaScript"
      SRC=script.code>
```

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

В общем виде запись контейнера выглядит следующим образом:

```
<SCRIPT [TYPE=тип_языка_программирования]
```

```
      [SRC=URL]>
JavaScript/VBScript-код
</SCRIPT>
```

Теги тела документа

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

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

Тело документа – контейнер BODY

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

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

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

Как видно из этого примера, в качестве значения данного атрибута используется адрес в сокращенной форме URL. В данном случае это адрес локального файла. Следует заметить, что разные интерфейсы пользователя поддерживают различные дополнительные атрибуты для тега BODY (таблица 4.2).

Таблица 4.2

Атрибут	Значение
BGCOLOR=#FFFFFF	Цвет фона
TEXT=#0000FF	Цвет текста
VLINK =#FF0000	Цвет пройденных гипертекстовых ссылок
LINK =#00FF00	Цвет гипертекстовой ссылки

В данной таблице строка #XXXXXX определяет цвет в терминах RGB в шестнадцатеричной нотации. Также имеется возможность задавать цвета по названию. Далее в таблице приведены названия цветов, определенные в стандарте HTML 4 и соответствующие им RGB-коды. Отметим, что многие современные браузеры выходят за рамки стандартов и поддерживают гораздо больше названий цветов представленных в таблице 4.3.

Таблица 4.3

Название	**Код**	**Название**	**Код**
aqua	#00FFFF	navy	#000080
black	#000000	olive	#808000
blue	#0000FF	purple	#800080
fuchsia	#FF00FF	red	#FF0000
gray	#808080	silver	#C0C0C0
green	#008000	teal	#008080
lime	#00FF00	white	#FFFFFF
maroon	#800000	yellow	#FFFF00

Так, значения атрибутов в таблице 4.1 определяют цвет текста как синий, фона – белый, пройденные ссылки красные, а новые ссылки зеленые. Если в качестве атрибутов тега BODY указать:

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

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

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

Теги управления разметкой

Заголовки

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

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

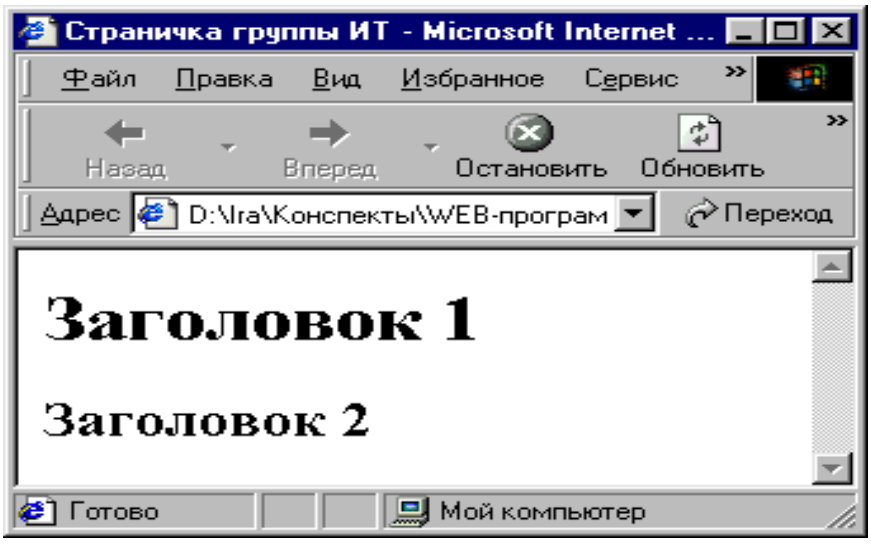


Рис.4.2

Тег <P>

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

Атрибут ALIGN

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

Далее приведены возможные значения атрибута ALIGN:

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

2. ALIGN=left выравнивание по левому краю. По умолчанию текст HTML выравнивается по левому краю и не выравнивается по правому, то есть начало строк находится на одном уровне по вертикали, а концы – на

разных. Чаще всего, получающийся при этом текст с равными промежутками между словами выглядит лучше. Поскольку выравнивание по левому краю задается автоматически, атрибут ALIGN=left можно опустить.

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

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

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

Использование тега

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

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

Элемент разметки <NOBR>

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

Теги управления отображением символов

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

Теги, управляющие формой отображения

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

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

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

Верхние и нижние индексы

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

Атрибут SIZE

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

Атрибут COLOR

Если вы хотите сделать свою страницу более красочной, можете воспользоваться атрибутом COLOR в теге FONT, и тогда единственным ограничением будет цветовая палитра на компьютере пользователя. Теги, управляющие формой отображения, приведены в таблице 4.4, а теги, характеризующие тип информации, приведены в таблице 4.5.

Таблица 4.4

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

Таблица 4.5.

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

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

Блоки цитат – элемент <BLOCKQUOTE>

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

Создание списков в HTML

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

Неупорядоченные списки – тег <UL>

Ненумерованный список. Ненумерованный список предназначен для создания текста типа:

– первый элемент списка;
– второй элемент списка;
– третий элемент списка.
–

Записывается данный список в виде последовательности:

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

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

Атрибуты маркеров в ненумерованном списке

Чтобы не применять одни и те же маркеры на разных уровнях вложенности, можно использовать атрибут TYPE. Вы можете задать любой тип маркера в произвольном месте списка. Можно даже смешивать разные типы маркеров в одном списке. Ниже перечислены теги с атрибутами стандартных маркеров:

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

Упорядоченные списки – тег <OL>

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

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

Список определений – тег <DL>

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

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

Определяемый термин записывается на одной строке, а его определение – на следующей, с небольшим отступом вправо. Тег <DL> позволяет создавать отдельные абзацы с отступом без нумерации или маркеров. Отступ делается от левого края. Если на странице несколько тегов <DL>, то текст постепенно сдвигается все больше вправо. В конце определения поместите закрывающий тег </DL>. Помните, что тег <DL> сдвигает только левую границу абзаца.

Горизонтальные линейки – тег <HR>

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

Преформатированный вывод – тег <PRE>

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

Применение тега <BLINK>

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

Комментарии в языке HTML

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

Гипертекстовые ссылки

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

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

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

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

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

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

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

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

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

Естественно, что браузер при обращении к серверу будет восстанавливать полную форму URL, опираясь на некоторую схему по умолчанию, которая называется базовым URL. Иногда неполную форму

URL называют относительным URL, подразумевая, что адрес задается относительно некоторого базового адреса.

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

В HTML есть элемент разметки BASE (рассмотренный ранее), который позволяет задать или переопределить базовый адрес.Его применяют как за пределами документа (например, при создании документов HTML-редакторами), так и в теле документа.

Содержание контейнера гипертекстовой ссылки, заключенное между тегом начала и тегом конца, выделяется в тексте цветом (таблица 4.6), определенным для контекстных гипертекстовых ссылок в атрибутах тега <BODY>.

Таблица 4.6

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

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

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

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

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

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

Для ссылки на такую точку используют следующую форму URL:

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

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

Использование графики в HTML

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

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

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

Теперь рассмотрим, как вставить изображение в Web-страницу. Тегом HTML, который заставляет браузер выводить изображение, является <IMG> с обязательным атрибутом SRC (SouRCe, источник). Имя файла

представляет собой имя выводимого графического файла. Замыкающего тега не требуется. Пример вставки изображения:

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

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

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

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

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

Атрибут SRC

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

Атрибут ALT

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

Атрибут ALIGN

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

1. ALIGN="top" выравнивает верх изображения по верхнему краю самого высокого элемента в строке окружающего текста.

2. ALIGN="middle" выравнивает центр изображения по базовой линии строки окружающего текста.

3. ALIGN="bottom" выравнивает нижний край изображения по базовой линии строки окружающего текста.

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

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

1. ALIGN="left" определяет огибаемое текстом изображение. Изображение располагается вдоль левой границы документа, а последующие строки текста огибают его справа.

2. ALIGN="right" определяет огибаемое текстом изображение. Изображение располагается вдоль правой границы документа, а последующие строки текста огибают его слева.

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

4. ALIGN="texttop" выравнивает верх изображения по верхнему краю самого высокого текстового символа в строке окружающего текста. Действие этого аргумента в большинстве случаев, но не всегда, подобно действию аргумента ALIGN="top".

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

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

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

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

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

Атрибут USEMAP

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

Атрибут BORDER

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

Атрибут HSPACE

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

Атрибут VSPACE

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

Атрибуты WIDTH и HEIGHT

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

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

Форматы графических файлов

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

Однако у этого формата есть одно серьезное ограничение: он не поддерживает изображения с глубиной цвета больше восьми бит на пиксел. Обычно этого оказывается достаточно для контурных

изображений типа комиксов и рисунков, где используется ограниченное количество цветов, или для небольших картинок, где для цветопередачи хватает 256 оттенков. Однако для больших изображений фотографического качества больше подходит формат JPEG.

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

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

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

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

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

Метод сжатия "с потерями" имеет много реализаций. JPEG достигает существенного сжатия за счет отбрасывания той графической информации,

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

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

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

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

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

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

Активные изображения

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

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

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

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

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

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

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

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

Изображения в миниатюре

Часто для иллюстрации какой-то темы требуются изображения большого размера, загружаться они будут достаточно долго. В том месте, где требуется разместить, большой рисунок, можно поместить маленькую его копию и сделать ссылку на полномасштабное изображение. Те

посетители, которым это действительно интересно, смогут посмотреть изображение полностью, а все остальные пролистнут страницу, не задерживаясь. Такая методика особенно хороша для обложек книг, фотографий, рекламных листков, которые не все читатели захотят изучить в деталях.

Средства описания таблиц в HTML

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

Создание таблиц в HTML

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

Создание строки таблицы – тег <TR>

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

Определение ячеек таблицы - тег <TD>

Внутри строки таблицы обычно размещаются ячейки с данными. Каждая ячейка, содержащая текст или изображение, должна быть окружена тегами <TD></TD>. Число тегов <TD></TD> в строке определяет число ячеек(Рис.4.4).

```
<HTML>
<BODY>
  <H1 ALIGN=center>Таблица</H1>
  <CENTER>
  <TABLE BORDER>
    <TR>
```

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

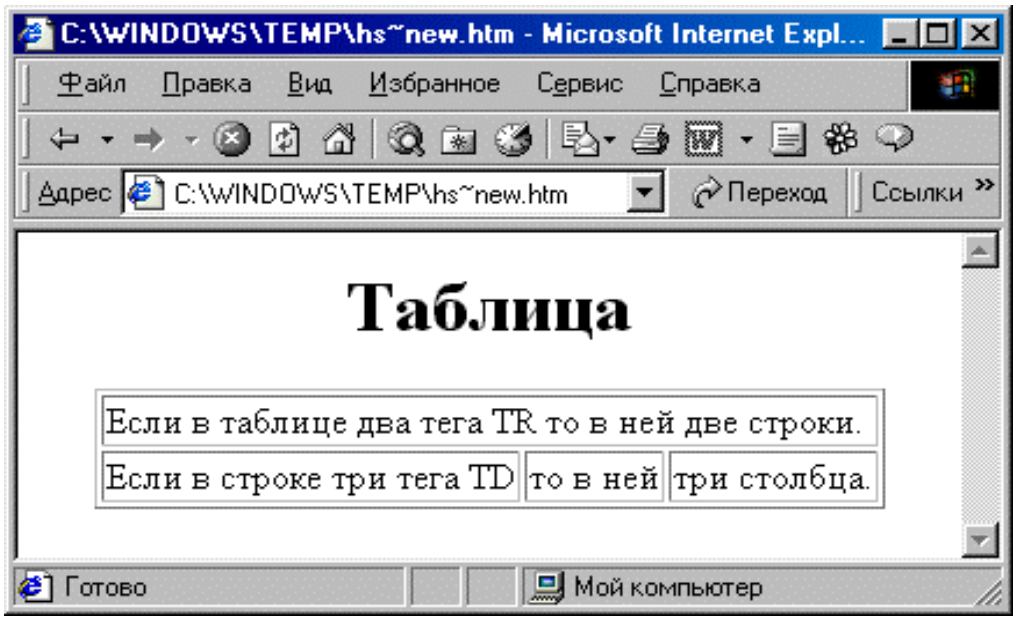


Рис.4 3

Заголовки столбцов таблицы – тег <TH>

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

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

Использование заголовков таблицы – тег <CAPTION>

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

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

Атрибут NOWRAP

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

Атрибут COLSPAN

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

```
<HTML>
<BODY>
  <CENTER>
  <TABLE BORDER=3>
```

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

Атрибут ROWSPAN

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

Атрибут WIDTH

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

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

Применение пустых ячеек

Если ячейка не содержит данных, она не будет иметь границ. Если требуется, чтобы у ячейки были границы, но не было содержимого, необходимо поместить в нее что-то, что не будет видно при просмотре. Можно воспользоваться пустой строкой
. Можно даже задать пустые столбцы, определив их ширину в пикселах или относительных единицах и не введя в полученные ячейки никаких данных. Это средство может оказаться полезным при размещении на странице текста и графики.

Атрибут CELLPADDING

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

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

Атрибуты ALIGN и VALIGN

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

1.ALIGN=bleedleft прижимает содержимое ячейки вплотную к левому краю.

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

3.ALIGN=center располагает содержимое ячейки по центру.

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

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

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

1. VALIGN=top выравнивает содержимое ячейки по ее верхней границе.

2. VALIGN=middle центрирует содержимое ячейки по вертикали.

3. VALIGN=bottom выравнивает содержимое ячейки по ее нижней границе.

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

Атрибут BORDER

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

Атрибут CELLSPACING

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

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

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

Атрибут BGCOLOR

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

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

```
    </TABLE>
    </CENTER>
</BODY>
</HTML>
```

Атрибут BACKGROUND

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

Использование таблиц в дизайне страницы

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

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

```
<HTML>
<BODY>
  <CENTER>
  <TABLE CELLPADDING="10" CELLSPACING="0"
         BORDER="16">
    <TR>
      <TD ALIGN="center">
   <H2> Одесский  государственный
экологический университет
 </H2>
        <H3>Добро пожаловать!</H3>
        <TABLE BORDER WIDTH="100%">
          <TR>
            <TD ALIGN="center"><I>Учебный курс
             " Web-программирование"</I></TD>
          </TR>
        </TABLE>
      </TD>
    </TR>
  </TABLE>
```

```
  </CENTER>
</BODY>
```

5 HTML-формы

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

Задание формы – элемент FORM

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

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

Определение элементов управления формы – тег <INPUT>

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

Атрибут TYPE=text

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

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

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

Атрибут TYPE=checkbox

При создании форм часто требуется получить ответ пользователя на вопрос типа "Да/Нет". Для создания независимых кнопок в формах HTML используется тег <INPUT> с атрибутом TYPE=checkbox. В зависимости от содержания формы пользователь может отметить несколько флагов. Когда форма использует тег <INPUT> с атрибутом CHECKBOX, в нем должны присутствовать и атрибуты NAME, и VALUE. Атрибут NAME указывает на наименование данного поля (флага) ввода. В атрибуте VALUE будет содержаться значение поля.

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

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

Атрибут TYPE=radio

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

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

Атрибут TYPE=image

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

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

Атрибут TYPE=password

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

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

Атрибут TYPE=reset

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

используется атрибут reset, тег <INPUT> может дополнительно содержать атрибут VALUE. Данный атрибут определяет надпись на изображении кнопки.

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

Атрибут TYPE=submit

Используя форму HTML для ввода информации от пользователя, необходимо обеспечить пользователю возможность завершить ввод данных. Для этого используется тег <INPUT> с атрибутом TYPE=submit. Браузер, в свою очередь, выводит данный элемент, как кнопку, по которой пользователь может щелкнуть, чтобы завершить процесс редактирования. Когда в форме используется тег <INPUT> с атрибутом submit, данный элемент может содержать два дополнительных атрибута: NAME и VALUE. Атрибут NAME хранит значение переменной поля в вашей форме. Атрибут VALUE – указывает наименование кнопки Submit.

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

Атрибут TYPE=hidden

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

Создание многострочных областей ввода текста –

тег <TEXTAREA>

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

Атрибут COLS

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

Атрибут NAME

Определяет наименование поля.

Атрибут ROWS

Задает количество видимых строк текстовой области.

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

Использование списков в форме – тег <SELECT>

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

Атрибут MULTIPLE

Позволяет выбрать более чем одно наименование.

Атрибут NAME

Определяет наименование объекта.

Атрибут SIZE

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

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

Атрибут SELECTED

Используется для первоначального выбора значения элемента по умолчанию.

Атрибут VALUE

Указывает на значение, возвращаемое формой после выбора пользователем данного пункта. По умолчанию значение поля равно значению тега <OPTION>(Рис.5.1).

```
<BR>Выбор
```

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

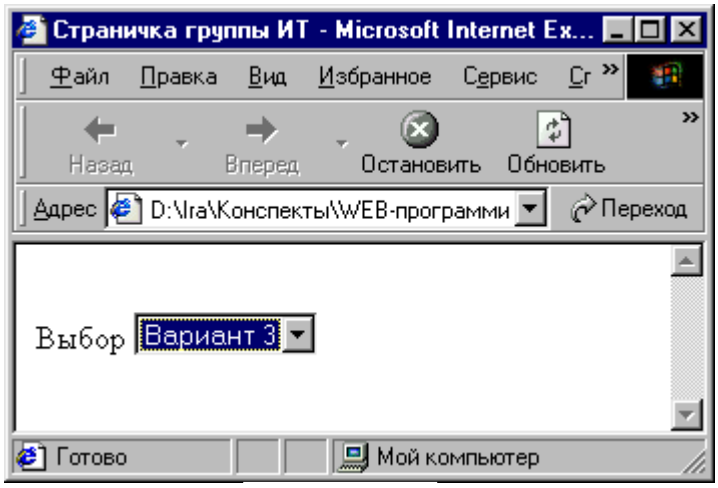


Рис.5.1

6 Фреймы

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

На первый взгляд, фреймы – это нечто сложное, но их легче понять, если провести аналогию с ячейками таблицы. Расположение фреймов на экране и ячеек в таблице задается почти одинаково: теги и атрибуты работают так же, как их табличные "родственники". Однако, хотя аналогия между единичным фреймом на странице и ячейкой таблицы верна, нужно помнить, что есть и отличия. Содержимое ячейки задано в коде HTML-страницы с таблицей. Текст или графика, составляющие содержимое

таблицы, фактически вводятся на той же странице HTML, что и тег или атрибут, описывающие таблицу. Напротив, экран с фреймами описывается в HTML-странице, в контейнере FRAMESET. Содержимое же фрейма – это отдельная HTML-страница, которая может находиться где угодно: в другом каталоге, на локальном сервере или на удаленном узле где-то в сети. Фреймовая структура определяет только способ организации экрана с фреймами и указывает, где находится начальное содержимое каждого фрейма. Для всех фреймов задаются URL, описывающие местонахождение их данных. Как правило, на странице с фреймовой структурой содержимого фреймов нет. Такая страница обычно невелика – она описывает только кадровую структуру экрана. Когда документ загружается во фрейм, вы можете щелкать мышкой на ссылке в этом документе, чтобы увидеть связанные документы в других кадрах, заданных во фреймовой структуре.

Создание простой страницы с фреймами

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

Задание фреймовой структуры

Для начала мы должны представить себе общий вид страницы – где расположить фреймы и какого они будут размера. Затем можно подумать об их содержании. Ниже приводится код простой фреймовой структуры с использованием тега <FRAMESET>. Обратите внимание: страница с фреймовой структурой не содержит тега <BODY>(Рис. 6.1).

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

```
</HTML>
```

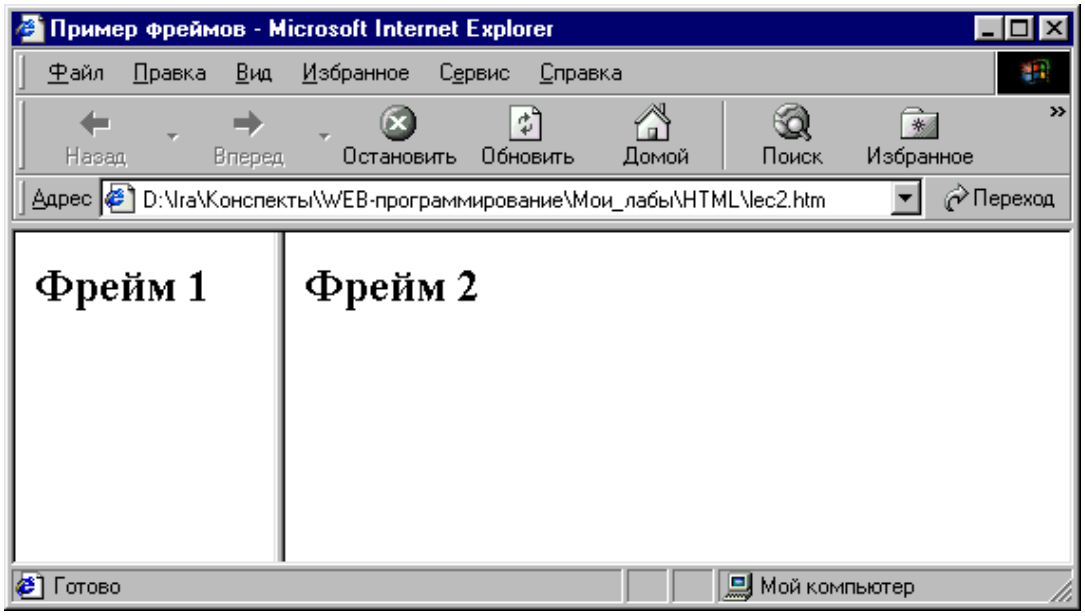


Рис. 6.1

Вот и весь код, необходимый для того, чтобы задать фреймовую структуру. Обратите внимание на тег <NOFRAMES>. Через некоторое время мы к нему вернемся. В результате мы получили экран, разделенный на два окна. Левое окно занимает 25% экрана и содержит страницу с названием menu.html. Окно справа займет 75% экрана и содержит файл main.html. Пока у нас их нет, так что вы увидите страницу с двумя пустыми фреймами. Прежде чем она появится, нам придется пару раз щелкнуть мышкой в ответ на сообщения об ошибках, потому что браузер будет пытаться найти несуществующие страницы. Заметьте, что правую страницу мы назвали "main" (<главная>) с помощью строки:

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

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

Подготовка содержимого фрейма

Теперь загрузим фреймы с содержимым. Зададим страницу menu.html в левом фрейме, где мы собираемся щелкать мышью, переключаясь между двумя страницами в правом фрейме. menu.html – это обычная HTML-страница, построенная как оглавление. На самом деле мы можем взять готовую страницу с оглавлением и использовать ее. Имейте в виду, что этот фрейм узкий и высокий, так что страница, которая будет в него

загружаться, должна быть спроектирована соответствующим образом. Теперь мы должны определить, где будут появляться другие страницы при щелчке мышкой на ссылке. Поскольку мы хотим, чтобы они отображались в правом фрейме, добавим атрибут TARGET (TARGET="main") в тег ссылки. Это означает, что, когда пользователь щелкает на ссылке, вызываемая страница появляется в фрейме main. Мы отображаем все страницы в фрейме main, поэтому давайте добавим атрибут TARGET="main" во все теги ссылок в оглавлении. Если мы не определим атрибут TARGET, то страница появится там, где мы щелкнули мышкой, – в левом фрейме.

Подготовка фрейма main

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

Использование тега <NOFRAMES>

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

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

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

```
</HTML>
```

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

Макетирование фреймов – тег <FRAMESET>

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

Атрибуты ROWS и COLS

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

Атрибут ROWS

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

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

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

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

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

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

Атрибут COLS

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

Задание содержимого фрейма – элемент FRAME

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

Атрибут NAME

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

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

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

Атрибут MARGINWIDTH

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

Атрибут MARGINHEIGHT

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

Атрибут SCROLLING

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

Атрибут NORESIZE

Как правило, пользователь может, перемещая границу фрейма мышкой, изменить его размер. Это удобно, но не всегда. Иногда требуется атрибут NORESIZE. Помните: все границы фрейма, для которого вы задали NORESIZE, становятся неподвижными – соответственно, может оказаться так, что размеры соседних фреймов тоже станут фиксированными. Пользуйтесь этим атрибутом с осторожностью.

Атрибут SRC

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

Атрибут TARGET

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

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

Вложенные и множественные кадровые структуры

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

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

7 Назначение и применение каскадных таблиц стилей (CSS)

Дизайн Web-узлов – это точное размещение компонентов HTML-страниц относительно друг друга в рабочей области окна браузера.

Недостатки такого определения Web-дизайна очевидны. В нем не учтены ни цвет, ни форма, ни другие свойства компонентов HTML-страниц. Главное в этом определении – показать ограниченность возможностей HTML-разметки. Позиционирование компонентов на странице является одним из самых слабых мест в HTML.

К компонентам страницы относятся: блоки текста, графика и встроенные приложения. Размер и границы каждого из этих компонентов в рамках HTML-разметки задаются с разной степенью точности. Размер графики и приложений можно задать с точностью до пиксела. Размеры текстовых блоков в HTML задать нельзя: они вычисляются браузером исходя из относительного размера шрифта по умолчанию.

Автор страницы не может заранее определить настройки браузера пользователя, что существенно ограничивает число вариантов представления информации на странице.

Нельзя сказать, что разработчики браузеров не пытались изменить данную ситуацию. В ранних версиях браузеров CERN для платформы NEXT и в браузерах WWWC автор страницы имел возможность переопределять настройки браузера по умолчанию через HTML-разметку. Но этот подход не получил продолжения в коммерческих продуктах.

Другой способ управления настройками браузера – программирование на JavaScript. Бурное развитие этого языка позволяет говорить о возможности полного контроля над процессом отображения HTML-страниц. Недостаток JavaScript – отказ от декларативного характера разметки и относительно большой объем кода для переопределения свойств элементов разметки.

Спецификация CSS (Cascading Style Sheets) позволяет остаться в рамках декларативного характера разметки страницы и полностью контролировать форму представления элементов HTML-разметки.

Каскадные таблицы стилей призваны разрешить противоречие между точностью определения размеров картинок и приложений, с одной стороны, и точностью определения размеров блоков текста и его начертания – с другой.

Таблицы стилей также позволяют определить цвет и начертание текстового фрагмента, изменять эти параметры внутри текстового блока, выполнять выравнивание текстового блока относительно других блоков и компонентов страницы.

Наличие подобных возможностей позволяет говорить о CSS как о средстве разделения логической структуры документа и формы его

представления. Логическая структура документа определяется элементами HTML-разметки, в то время как форма представления каждого из этих элементов задается CSS-описателем элемента.

CSS позволяет полностью переопределить форму представления элемента разметки по умолчанию. Например, <I>...</I> определяет отображение текста курсивом:

```
<I>Отобразим текст курсивом</I>
```

А теперь переопределим стиль отображения для элемента разметки I.

```
<I STYLE="text-decoration:underline; font-
style:normal;">
Отобразим текст курсивом
</I>
```

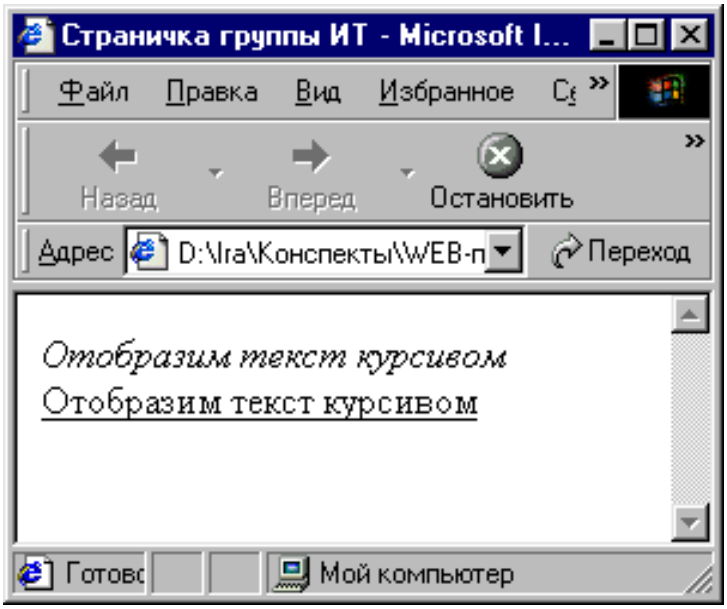


Рис. 7.1

Этот пример показывает, что привычный стиль отображения элементов может быть полностью изменен при помощи CSS. В данной технологии HTML-разметка носит чисто декларативный характер.

Практическое значение CSS для Web-инжиниринга (совокупности технологий разработки и сопровождения Web-узлов) заключается в том, что процесс создания узла можно формализовать и представить в виде последовательности действий:

– необходимо определиться с номенклатурой страниц, т.е. все страницы проектируемого Web-узла разбить на типы, например домашняя страница, навигационные страницы, информационные

страницы, коммуникационные страницы и т.п. У каждого узла этот перечень может быть свой;
- для каждого типа страниц требуется разработать определенную логическую структуру (стандартный набор компонентов страницы);
- следует создать навигационную карту узла и форму ее реализации на страницах;
- для каждого стандартного компонента страницы нужно разработать стиль его отображения (CSS-описатель);
- теперь остается только рисовать картинки, создавать анимацию, писать программы, вручную вводить текст и графику или генерировать содержимое страниц автоматически при обращении к ним.

Объяснив таким образом роль и назначение CSS среди многообразия Web-технологий, мы переходим непосредственно к обсуждению применения каскадных таблиц стилей.

Способы применения CSS

Под способами применения CSS мы в данном разделе понимаем форму декларирования стиля на HTML-странице и форму связывания описания стиля отображения элемента разметки с самим элементом. Речь идет о том, где и в какой форме автор страницы (или дизайнер) описывает стиль, и как и в какой форме на него ссылается. Итак, различают четыре способа применения стилей:

- переопределение стиля в элементе разметки;
- размещение описания стиля в заголовке документа в элементе STYLE;
- размещение ссылки на внешнее описание через элемент LINK;
- импорт описания стиля в документ.

Здесь мы следуем за Джорджем Янгом из Microsoft (Cascading Style Sheets in Internet Explorer 4.0. Microsoft, 1997). Важно отметить, что в браузерах других производителей импорт стиля не поддерживается. Однако, поскольку патент на CSS1 принадлежит Microsoft, опустить импорт в нашем описании было бы неправильно.

Переопределение стиля

```
<H1 STYLE="font-weight:normal;
font-style:italic;
font-size:10pt;">
Заголовок первого уровня
</H1>>
```

Атрибут style можно применить внутри любого элемента разметки. Например, мы можем через style определить ширину и выравнивание элемента hr (горизонтальное отчеркивание):

```
<HR STYLE="width:100px;">
```

Очевидно, что не все параметры стиля можно установить для конкретного элемента разметки. О типах элементов и соответствующих параметрах стилей мы поговорим в разделе "Понятия блочного и строкового элементов".

Здесь же нужно отметить следующее: стили разработаны в первую очередь для управления отображением текста. Не следует увлекаться стилями при управлении отображением нетекстовых элементов HTML-разметки.

Элемент STYLE

Применение элемента STYLE – это основной способ внедрения каскадных таблиц стилей в ткань HTML-документа. Помимо управления отображением элементов разметки, элемент STYLE позволяет описывать стилевые свойства элементов, которые можно изменять при программировании на JavaScript.

Элемент STYLE дает возможность определить стиль отображения для:

- стандартных элементов HTML-разметки;
- произвольных классов (селектор CLASS);
- HTML-объектов (селектор ID).

К сожалению, работа с селекторами в браузерах различных производителей может преподносить различного рода сюрпризы. Особенно это касается селектора ID. В данном случае мы будем рассматривать Microsoft как держателя патента на спецификацию CSS.

Понятие селектора, применение селекторов и формальный синтаксис CSS мы обсудим в разделах "Синтаксис" и "Наследование и переопределение".

Стандартные элементы разметки описываются в элементе STYLE следующим образом:

```
<HEAD>
<STYLE>
p { color:darkred;text-align:justify;
    font-size:8pt; }</STYLE>
</HEAD>
<BODY>
```

```
...
<P>
Этот параграф мы используем как пример
применения описания стиля для стандартного
элемента HTML-разметки.
</P>
...
</BODY>
```

Теперь все параграфы документа будут отображаться стилем из элемента STYLE, если только стиль не будет каким-либо способом переопределен. В STYLE можно определить стиль любого элемента разметки.

Ссылка на внешнее описание

Ссылка на описание стиля, расположенное за пределами документа, осуществляется при помощи элемента LINK, который размещают в элементе HEAD. Внешнее описание может представлять собой файл, содержащий описание стилей. Описание стилей в этом файле будет по синтаксису в точности совпадать с содержанием элемента STYLE.

Ниже приведен пример ссылки на внешнее описание стилей:

```
<LINK TYPE="text/css" REL="stylesheet"
      HREF="http://intuit.ru/my_css.css">
```

Здесь важны, значения атрибутов REL и TYPE. Атрибут REL должен иметь значение stylesheet. Type может принимать значения text/css или text/javascript. Второй тип описания стилей введен Netscape. Его мы в данном учебном курсе не обсуждаем.

Атрибут HREF задает универсальный локатор ресурса (URL) для внешнего файла описания стилей. Это может быть ссылка на файл с любым именем, а не только на файл с расширением *.css.

Импорт описания стилей

Импорт описателей стилей в некотором смысле составляет конкуренцию представленному выше указанию на внешний описатель стиля.

Импортировать стиль можно либо внутрь элемента STYLE, либо внутрь внешнего файла, который представляет собой описатель стиля. Оператор импорта стиля должен предшествовать всем прочим описателям стилей:

```
<STYLE>
@import:url(http://intuit.ru/style.css)
A { color:cyan;text-decoration:underline; }
</STYLE>
```

Импортируемый стиль можно переопределить либо через описатель элемента в STYLE, либо через атрибут элемента STYLE.

Синтаксис

Формально стиль отображения элементов разметки задается ссылкой в элементе разметки на селектор стиля. Синтаксис описания стилей в общем виде представляется следующим образом:

```
selector[, selector[, ...]]
        { attribute:value;
        [atribute:value;...] }
```

или

```
selector selector [selector ...]
         { attribute:value;
         [atribute:value;...] }
```

В первом варианте перечислены селекторы, для которых действует данное описание стиля. Второй вариант задает иерархию вложенности селекторов, для совокупности которых определен стиль. Напомним, что речь в данном случае идет об описаниях стилей в нотации text/css. Описания стилей размещаются либо внутри элемента STYLE, либо во внешнем файле.

В качестве селектора можно использовать имя элемента разметки, имя класса и идентификатор объекта на HTML-странице.

Атрибут (attribute) определяет свойство отображаемого элемента, например левый отступ параграфа (margin-left), а значение (value) – значение этого атрибута, например, 10 типографских пунктов (10 pt).

Селектор – имя элемента разметки

Когда автор Web-узла хочет определить общий стиль всех страниц, он просто прописывает стили для всех элементов HTML-разметки, которые будут использоваться на страницах. Это дает возможность скомпоновать

страницы из логических элементов, а стиль отображения элементов описать во внешнем файле.

Такой способ создания сайта позволяет автору изменять внешний вид всех страниц путем внесения изменений в файл описания стилей, а не в файлы HTML-страниц.

Внешний файл при этом может выглядеть следующим образом:

```
I, EM { color:#003366,font-style:normal }
A I { font-style:normal;font-weight:bold;
      text-decoration:line-through }
```

В первой строке этого описания перечислены селекторы-элементы, которые будут отображаться одинаково:

```
<I>Это курсив</I> и это тоже <EM>курсив</EM>
```

Последняя строка определяет стиль отображения вложенного в гипертекстовую ссылку курсива:

```
<A NAME=empty><I>intuit</I></A>
```

В данном случае переопределение состоит в том, что текст отображается внутри гипертекстовой ссылки перечеркнутым, причем жирным шрифтом.

Селектор – имя класса

Имя класса не является каким-либо стандартным именем элемента HTML-разметки. Оно определяет описание класса элементов разметки, которые будут отображаться одинаково. Для того, чтобы отнести элемент разметки к тому или иному классу, нужно воспользоваться его атрибутом CLASS:

```
<STYLE>
.test {color:white;background-color:black;}
</STYLE>
...
<P CLASS="test">
Этот параграф мы отобразим белым цветом по
черному фону
</P>
...
<P>
```

```
Эту <A CLASS="test">гипертекстовую ссылку</A>
мы отобразим белым цветом по черному фону.
</P>
```

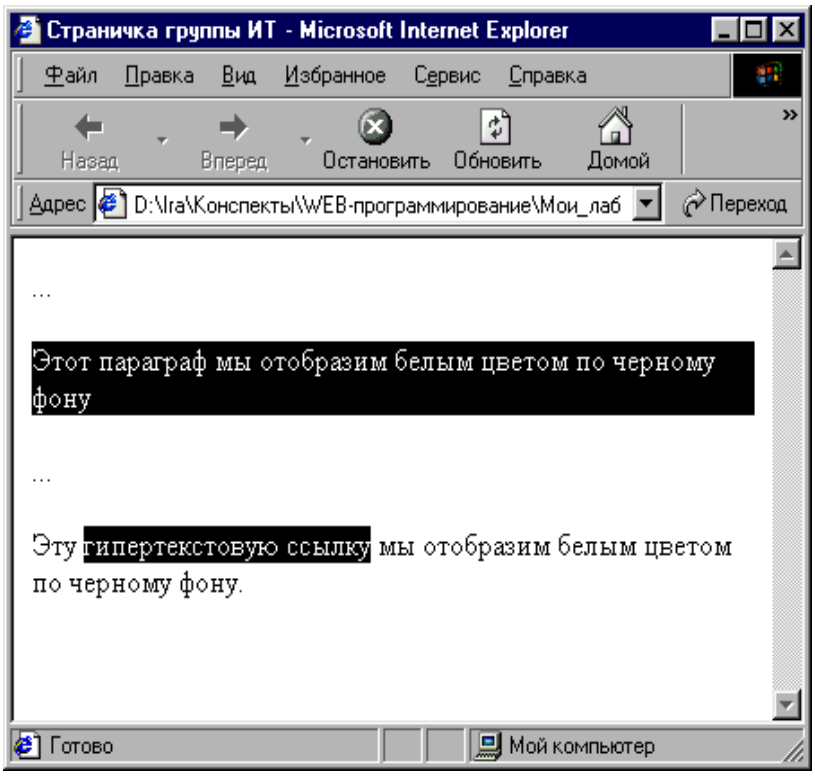


Рис.7. 2

Таким образом, в любом элементе разметки можно сослаться на описание класса отображения. При этом совершенно необязательно, чтобы элементы разметки были однотипными. В примере к одному классу,отнесены и параграф, и гипертекстовая ссылка в другом параграфе.

Лидирующую точку в имени класса можно опустить. Она задается из соображений сохранения единства описания. Например, можно определить классы отображения однотипных элементов разметки:

```
a.menu { color:red;background-color:white;
         text-decoration:normal; }
a.paragraph { color:navy;
              text-decoration:underline; }
```

В данном примере класс гипертекстовых ссылок menu имеет одно описание стиля, а класс гипертекстовых ссылок paragraph – совершенно другое. При этом каждый из этих классов нельзя применить к другим элементам разметки, например, параграфу или списку. Если имя элемента разметки не задано, это означает, что класс можно отнести к любому

элементу разметки – корневой класс описания стилей. Это очень похоже на обозначение имени корневого домена в системе доменных имен. Собственно ничего удивительного здесь нет, т.к. система классов объектов на HTML-странице представляет собой дерево. Элементы разметки – это узлы дерева.

Селектор – идентификатор объекта

Объектная модель документа (Document Object Model) описывает документ как дерево объектов. Объектами являются: сам документ, его разделы (элемент DIV), картинки, параграфы, приложения и т.п. Каждый из объектов можно поименовать и обращаться к нему по имени. Данная возможность используется при программировании страниц на стороне клиента.

Применение идентификатора объекта оправдано еще и в случае модификации атрибута описания стиля для данного объекта в его CSS-описании. Вместо двух описаний классов, которые отличаются только одним из параметров, можно создать одно описание класса и описание идентификатора объекта. Описание стиля для объекта задается строкой, в которой селектор представляет собой имя этого объекта с лидирующим символом "#":

```
a.mainlink { color:darkred;
             text-decoration:underline;
             font-style:italic; }
#blue { color:#003366 }
...
<A CLASS=mainlink>основная гипертекстовая
ссылка</A>
<A CLASS=mainlink ID=blue>модифицированная
гипертекстовая ссылка</A>
```

Интерпретация идентификаторов объектов в Internet Explorer связана с атрибутом ID.

Различия в интерпретации ID в браузерах при декларативном использовании CSS не очень страшны. Другое дело, если автор решится программировать стили, т.е. изменять значения атрибутов описателей стилей. Придется, для каждого из браузеров разрабатывать совершенно разные страницы.

Наследование и переопределение

При обсуждении технических спецификаций часто бывает полезно вникнуть в смысл названия. В названии принято точно определять суть и назначение стандарта или спецификации. Описание стилей отображения элементов HTML-разметки носит название "Каскадные таблицы стилей". Со словом "стилей" все более-менее понятно. Под словом "таблицы" следует понимать набор свойств элемента разметки, который можно представить в виде строки в таблице свойств, т.е. элементы разметки – строки, а свойства – столбцы. А вот слово "каскадные" требует пояснения.

Во-первых, существует иерархия элементов разметки (дерево объектов на странице). Во-вторых, свойства этих объектов могут наследоваться. Таким образом в дереве объектов образуется ветвь, которая ведет к листу дерева – элементу разметки, например, элементу списка или параграфу. Его свойства определяются элементами разметки, в которые вложен элемент, и описателями стиля для данного элемента:

```
Это начало первого раздела, который сдвинут
на 10 пикселов вправо относительно левого
края параграфа и на 10 пикселов вниз
относительно стандартной границы параграфа.

   Это начало второго раздела, который
  сдвинут относительно предыдущего раздела
  на 10 пикселов, а относительно параграфа –
  на 20 пикселов. Данный раздел имеет красную
  строку с отступом в 10 пикселов и смещен
  относительно предыдущего раздела на 20
  пикселов.

   - первый элемент списка
   - второй элемент списка

Список сдвинут относительно второго раздела
на 10 пикселов, а относительно текущего
параграфа – на 30 пикселов. Первая строка не
является строкой начала параграфа, поэтому на
нее отступ не распространяется (только в
Netscape).
```

Предыдущий текст закодирован в терминах разделов и списка следующим образом:

```
<DIV STYLE="margin-left:10px;margin-top:10px;">
```

```
 Это начало первого раздела, который сдвинут
 на 10 пикселов вправо относительно левого
 края параграфа и на 10 пикселов вниз
 относительно стандартной границы параграфа.
<DIV STYLE="margin-left:10px;margin-top:20px;
 text-indent:10px;font-style:italic;">>

    Это начало второго раздела, который
   сдвинут относительно предыдущего раздела
   на 10 пикселов, а относительно параграфа –
   на 20 пикселов. Данный раздел имеет красную
   строку с отступом в 10 пикселов и смещен
   относительно предыдущего раздела на 20
   пикселов.
<UL STYLE="margin-left:10px;">
<LI>первый элемент списка
<LI>второй элемент списка
</UL>
 Список сдвинут относительно второго раздела
 на 10 пикселов, а относительно текущего
 параграфа – на 30 пикселов. Первая строка не
 является строкой начала параграфа, поэтому на
 нее отступ не распространяется (только в
 Netscape).
</DIV>
</DIV>
```

Таким образом, отступы отсчитываются относительно элемента, в который вложен текущий элемент. Все параметры, которые не были переопределены в текущем элементе, наследуются из старшего по иерархии элемента. Последнее хорошо продемонстрирует применение стилей отображения списка, который вложен в раздел и поэтому отображается курсивом.

Когда объяснение некоторого феномена HTML-разметки растягивается на несколько параграфов, имеет смысл воспользоваться приведенной ниже графической схемой построения страницы.

При использовании стилей действуют следующие правила старшинства стилей отображенные на рисунке 7.1:

– сначала применяются стили браузера по умолчанию;
– стили браузера по умолчанию переопределяются прилинкованными стилями (элемент LINK заголовка документа);

– прилинкованные стили переопределяются описаниями стилей в элементе STYLE;
– стили элемента STYLE переопределяются атрибутом STYLE в любом из элементов разметки.

Рис. 7.3

Не все атрибуты стиля могут наследоваться. Например, "набивка" (отступ содержания элемента от его границ) элемента BODY не наследуется вложенными в него элементами и определяется по умолчанию или прописывается для каждого элемента отдельно. Алгоритмы наследования в Internet Explorer и в Netscape Navigator разные, поэтому для единства отображения элементов следует прописывать стиль по максимуму атрибутов.

Блочные и строковые элементы

В описании элементов разметки языка HTML существует понятие строкового (in-line) элемента разметки и блочного (block) элемента разметки. Формально они определены в DTD SGML-описания языка HTML. Объяснить различие между блочным и строковым элементами можно на примере:

– параграф – это блочный элемент разметки;
– выделение курсивом – это строковый элемент разметки.

Блочные элементы можно вкладывать друг в друга, но они не должны пересекаться. Строковые элементы можно как вкладывать, так и пересекать (согласно DTD и практике старых версий браузеров), но этого делать не рекомендуется.

Очевидно, что по набору атрибутов управления отображением (атрибуты описания стиля) строковые и блочные элементы отличаются. Упрощенно можно сказать, что атрибуты описания стиля строкового элемента являются подмножеством атрибутов описания стиля блочного элемента. Обобщениями блочного и строкового элементов, с точки зрения стилей, являются элементы DIV и SPAN, соответственно.

Элемент DIV

DIV играет роль универсального блока. Блочный элемент всегда отделен от прочих элементов страницы (контекста) пустой строкой. DIV не несет никакой смысловой нагрузки. Часто говорят, что DIV – это раздел страницы. Но на самом деле его применение имеет смысл только в контексте CSS. Никаких правил по умолчанию для отображения DIV не существует. Это просто новая строка текста.

DIV позволяет применить атрибуты стиля, связанные с границей блока и отступами блока от границ старшего элемента, а также "набивку", т.е. отступ от границы блока до границы вложенного элемента:

```
<DIV STYLE="border-color:#003366;
border-width:1px;
margin:20px;padding:10px;">
Блочный элемент, заданный элементом разметки
DIV.
</DIV>
<P>Для него определена граница и отступы как
от границ старшего элемента разметки, так и
для вложенных в него элементов разметки.
</P>
```

Если текст будет просматриваться браузерами, не поддерживающими CSS, элемент DIV использовать не рекомендуется. В этом случае лучше применить параграф или другой подходящий по смыслу элемент разметки из стандартного набора HTML.

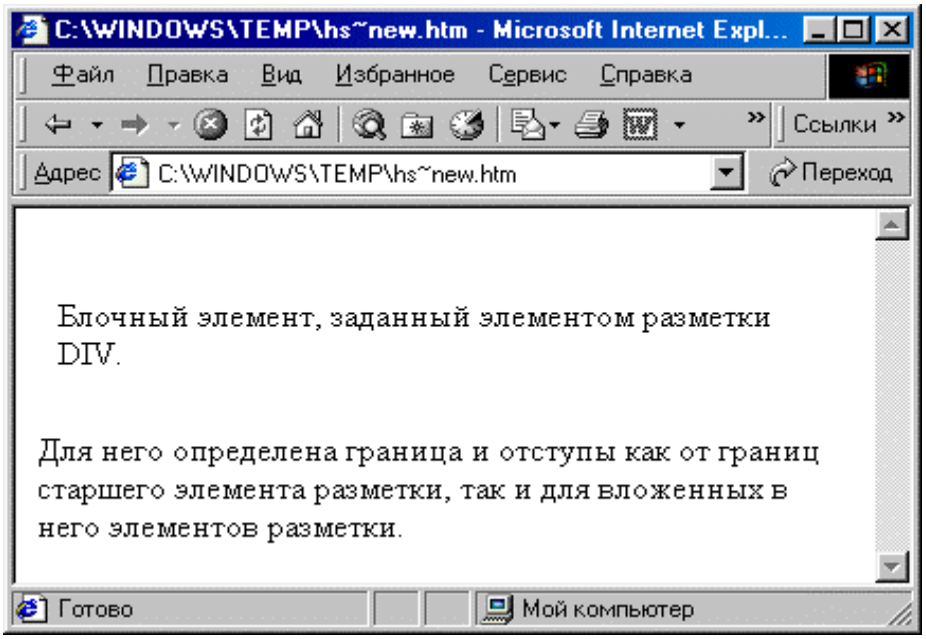


Рис.7. 4

Элемент SPAN

Элемент разметки SPAN – это обобщенный строковый элемент разметки, применение которого не приводит к образованию блока текста. Он может заменить элементы FONT, I, B, U, SUB, SUP и т.п. Приведем примеры таких соответствий в таблице 5.1:

Таблица7.1

HTML-элемент	CSS-аналог
<FONT COLOR=red> ...</FONT>	<SPAN STYLE="color:red; ">...</SPAN>
<I>...</I>	<SPAN STYLE="font-style:italic; ">...</SPAN>
<B>...</B>	<SPAN STYLE="font-weight:bold; ">...</SPAN>
<U>...</U>	<SPAN STYLE="text-decoration:underline; ">...</SPAN>

В новых версиях браузера Netscape описания строковых стилей пересекаться не должны. Тэг конца элемента строкового типа закрывает ближайший элемент, а не тот, который открыт тэгом начала данного строкового стиля. Также и в случае применения элемента SPAN, где тэг конца можно соотнести только с ближайшим тэгом начала элемента SPAN:

```
<B> предложение <I> с пересекающимися </B> стилями
</I>
```

Предложение с пересекающимися стилями

```
<SPAN STYLE="font-weight:bold;"> предложение
<SPAN STYLE="font-style:italic;">
с пересекающимися </SPAN> стилями </SPAN>
```

Применение элемента SPAN ограничено браузерами, которые поддерживают CSS. При этом не все атрибуты спецификации CSS поддерживаются в браузерах. Например, атрибут vertical-align, который призван заменить элементы SUP и SUB, не поддерживается ни одним из браузеров.

Свойства блоков

Блочные элементы(блоки текста или box) позволяют оперировать с текстом в терминах прямоугольников, которые этот текст занимает. При этом блок текста становится элементом дизайна страницы с теми же свойствами, что и картинка, таблица или прямоугольная область приложения.

Блок текста обладает свойствами: высоты (height), ширины (width), границы (border), отступа (margin), набивки (padding), произвольного размещения (float), управления обтеканием (clear).

Графически свойства можно представить следующим образом:

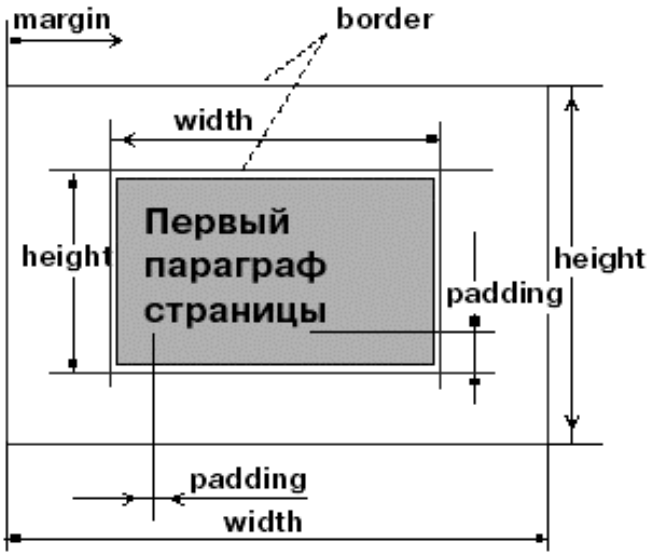


Рис. 7.5

В данном примере внутри окна браузера расположен блочный элемент (DIV), внутрь которого помещен еще один блочный элемент(P). DIV имеет белый фон и границу. Параграф имеет границу и желтый фон.

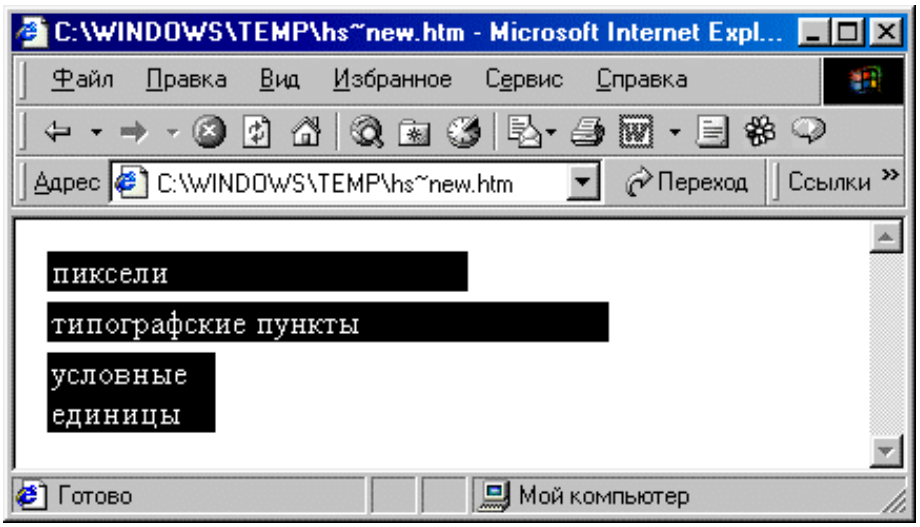


Рис. 7.6

С шириной и высотой блока текста все более или менее понятно. Задаваться они могут в типографских пунктах (pt), пикселах (px) и условных единицах (em):

```
<DIV STYLE="width:200px;">пикселы</DIV>
<DIV STYLE="width:200pt;">типографские
пункты</DIV>
<DIV STYLE="width:5em;">условные единицы</DIV>
```

Расстояние от границы блочного элемента до границы вложенного в него блочного элемента называется padding. В рамках данного курса лекций для обозначения этого свойства используется слово "набивка" или словосочетание "внутренний отступ".

Отступ от "набивки" внешнего блочного элемента до границы вложенного элемента называется margin. Для его обозначения мы будем употреблять термин "отступ" или словосочетание "внешний отступ".

Таким образом padding и margin характеризуют отступы блочного элемента относительного начала его содержания и относительно границы охватывающего его элемента разметки, соответственно.

Отступы и "набивка" могут быть левыми, правыми, верхними и нижними. CSS позволяет изменять любые из них(Риис.7.7).

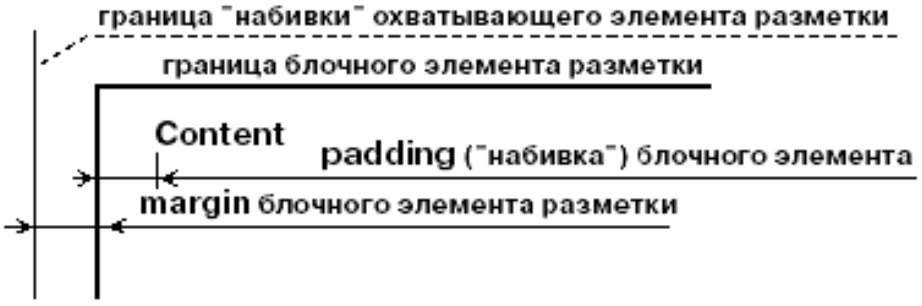


Рис. 7.7

При отображении блока текста можно показать его видимую границу. CSS позволяет определить ее стиль, ширину и цвет. При использовании видимой границы следует учитывать специфику браузеров. Одним из возможных способов применения границы является видимое ограничение "плавающих" блоков текста.

"Плавающий" текстовый блок позволяет реализовать возможность обтекания этого блока текстом.

Прижмем блок текста вправо. Слева его будет обтекать другой текст.

Обтекание одного текста другим происходит в том же ключе, что и Обтекание текстом картинки или таблицы. Текст охватывающего блока стремится "втиснуться" на свободное место, оставленное "плавающим" блоком. Когда граница "плавающего" блока кончается, охватывающий блок распространяется на всю ширину отведенного для текста пространства.

Таким образом, блок текста с точки зрения размещения на странице равноценен картинкам или прямоугольным областям приложений.

Отступы (margin)

При отображении блока текста на бумаге вокруг него обычно оставляют поля. Поля можно задавать либо относительно границы страницы, либо относительно самого блока текста. В первом случае мы имеем дело с "набивкой" (padding), а во втором – с отступом (margin). Собственно, ширина поля будет определяться суммой ширины "набивки" и ширины отступа:

Обычно пунктирная линия и граница блока являются невидимыми линиями. Они угадываются по выравненному краю текста. Вернее, угадывается суммарная ширина полей. Стрелки указывают направление отсчета отступа. Padding отсчитывается от внешней границы блока внутрь блока, в то время как margin – от внешней границы блока в область охватывающего его блока (наружу)(Рис.7.8).

Внешний отступ (margin) может отсчитываться по любому направлению относительно сторон блока:

1. margin-left – левый внешний отступ. Определяет расстояние от левой границы блока текста до левой границы внутреннего отступа ("набивки", padding) охватывающего элемента;
2. margin-right – правый внешний отступ. Определяет расстояние от правой границы блока текста до левой границы внутреннего отступа ("набивки", padding) охватывающего элемента;
3. margin-top – верхний внешний отступ. Определяет расстояние от верхней границы блока текста до верхней границы внутреннего отступа ("набивки", padding) охватывающего элемента;
4. margin-bottom – нижний внешний отступ. Определяет расстояние от нижней границы блока текста до нижней границы внутреннего отступа ("набивки", padding) охватывающего элемента;
5. margin – задает общий внешний отступ от всех сторон блока текста. Применяется в том случае, если блок текста равноудален от всех границ внутреннего отступа охватывающего элемента.

Графически эти отступы можно представить следующим образом(Рис.7.8):

Рис.7.8

В данном случае для параграфа использовалось следующее описание стиля:

```
P { margin-left:50px;margin-right:5px;
    margin-top:15px;margin-bottom:50px;
    padding:0px;text-align:left; }
```

Нужно иметь в виду, что браузеры могут отображать эти параметры по-разному.

Если размер всех внешних отступов одинаковый, то можно просто воспользоваться атрибутом margin:

```
P { margin:5px; }
```

При применении внешнего отступа следует помнить, что он отсчитывается от границы элемента до границы внутреннего отступа ("набивки", padding) охватывающего элемента. Если этот факт не учитывать, то общая ширина видимых полей может оказаться больше, чем указано во внешнем отступе.

Набивка (padding)

Текст внутри блока начинается не от самой его границы. Между границей и содержанием блока есть свободное пространство. Оно называется внутренний отступ текстового блока или padding. Совместно с внешним отступом (margin) текстового блока padding образует общее поле отступа от границы охватывающего блок элемента разметки.

Padding можно проиллюстрировать на примере левого внутреннего отступа текста в параграфе(Рис.7.9):

Для этого примера при описании параграфа использовался стиль:

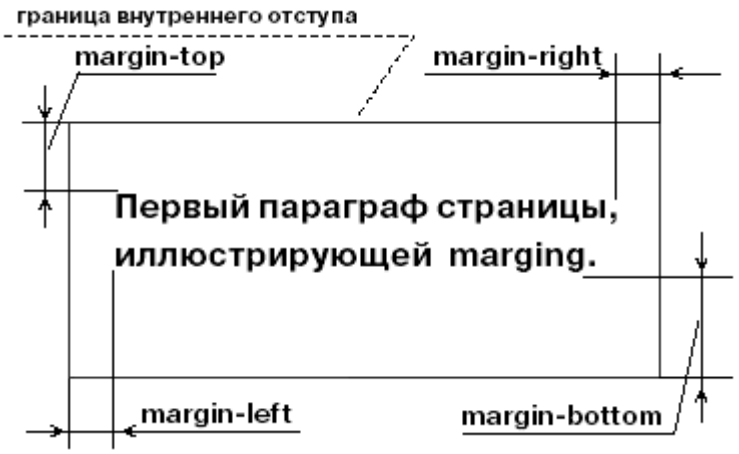


Рис. 7.9

```
P { padding-left:100px;text-align:left;border-
width:1px; }
```

Чтобы браузер правильно отображал стили, не следует размещать описание стиля на нескольких строчках, как это сделано в примере. Для Internet Explorer это не имеет значения.

У блока текста существует четыре стороны. Соответственно, padding может быть:

1. padding-left – левый внутренний отступ, который определяет расстояние от левого края блока до его содержания(Рис.7.10);

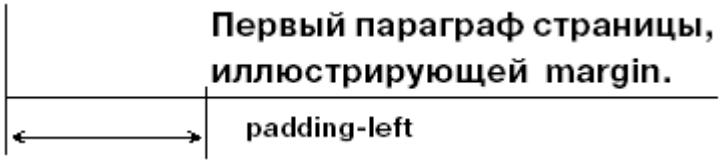


Рис.7.10

2. padding-right – правый внутренний отступ, который определяет расстояние от правого края блока до его содержания;
3. padding-top – верхний внутренний отступ, который определяет расстояние от верхнего края блока до его содержания;
4. padding-bottom – нижний внутренний отступ, который определяет расстояние от нижнего края блока до его содержания;
5. padding – определяет единый размер внутреннего отступа блока. Этот параметр задается в случае одинакового размера отступа от всех сторон блока.

Проиллюстрируем применение padding на примере(Рис. 7.11):

```
P { padding-left:100px;padding-right:50px;
    padding-top:20px;padding-bottom:10px;
    text-align:left;border-width:1px; }
```

При установке padding следует помнить, что этот параметр задает размер отступа от границы блока до границы внешнего отступа (margin) содержания блока. По этой причине общий размер поля может оказаться больше, чем задано в параметре padding.

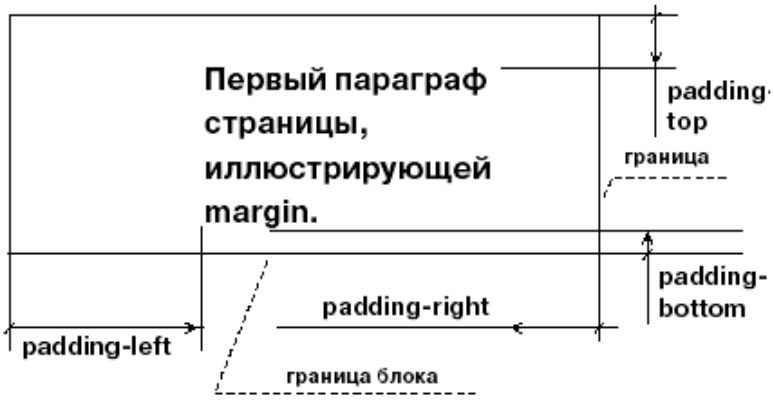


Рис.7.11

Граница (border)

У каждого блочного элемента разметки есть граница. От границы отсчитываются отступы (margin и padding). Вдоль границы "плавающего" блока его обтекает текст.

Для описания границ блоков применяются следующие атрибуты:

1. border-top-width – ширина верхней границы блока;
2. border-bottom-width – ширина нижней границы блока;
3. border-left-width – ширина левой границы блока;
4. border-right-width – ширина правой границы блока;
5. border-width – ширина границы блока. Задается в том случае, если ширина границы блока одинаковая по всему периметру блока;
6. border-color – цвет границы блока. Согласно спецификации, CSS1 может быть задан для каждой из границ блока. Например, border-right-color:red. Может задаваться как мнемоникой (red, blue, navy и т.п.), так и в нотации RGB (border-color:#003366). Указание цвета для каждой из границ поддерживается не всеми браузерами;
7. border-style – тип линии границы блока. Может принимать значения: none, dotted, dashed, solid, double, groove, ridge, inset, outset. Согласно спецификации CSS1, может быть задан для каждой из границ блока.

Например, border-right-style:dotted. Указание типа линии границы поддерживается не всеми браузерами.

Для описания границы нет необходимости указывать в стиле все атрибуты. Существует сокращенная запись атрибутов. Например, для описания верхней линии границы можно использовать запись типа:

```
P { border-top:1px dotted red; }
атрибут: ширина_линии тип_линии цвет_линии код
```

Если необходимо ограничить блок текста границей, то это может выглядеть примерно так(Рис.7.12):

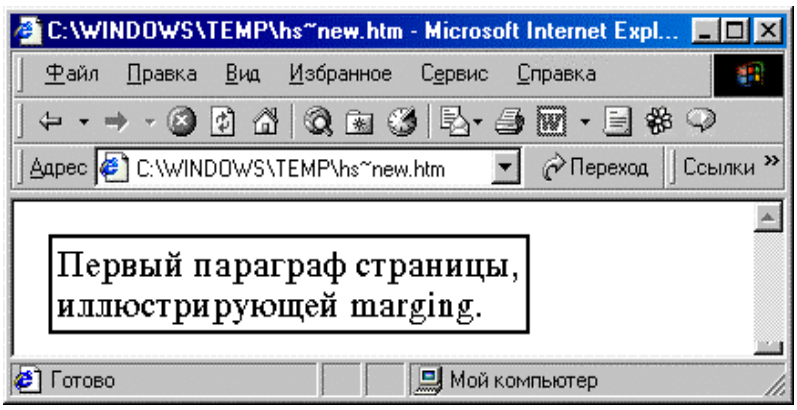


Рис. 7.12

В этом примере мы использовали следующее описание стиля отображения границы:

```
P { text-align:left;border-width:2px;
    border-color:darkred;border-style:solid; }
```

Применение границы для обозначения блока – не самый лучший способ оформления документа. Во всяком случае, его применяют нечасто.

Указывая границу в Internet Explorer, нужно обязательно определять ее тип, в противном случае она не будет отображаться.

Обтекание блока текста

Под обтеканием блока текстом понимают тот же эффект, который можно реализовать для графики, когда картинка не разрывает блок текста, а встраивается в него. Текст в этом случае "обтекает" картинку с одной стороны – там, где есть свободное поле между границей страницы (элемента) и картинкой. Обтекание картинки текстом от обычного встраивания картинки в текст документа отличается тем, что вдоль

вертикальной границы картинки располагается несколько строк текста, а не одна.

Обтеканием блока текста другим текстом управляют два атрибута CSS: float и clear.

Атрибут float определяет "плавающий" блок текста. Он может принимать значения:

1. left – блок прижат к левой границе охватывающего элемента;
2. right – блок прижат к правой границе охватывающего элемента;
3. both – текст может обтекать блок с обеих сторон.

Проиллюстрировать обтекание позволяет следующий пример(Рис.7.13):

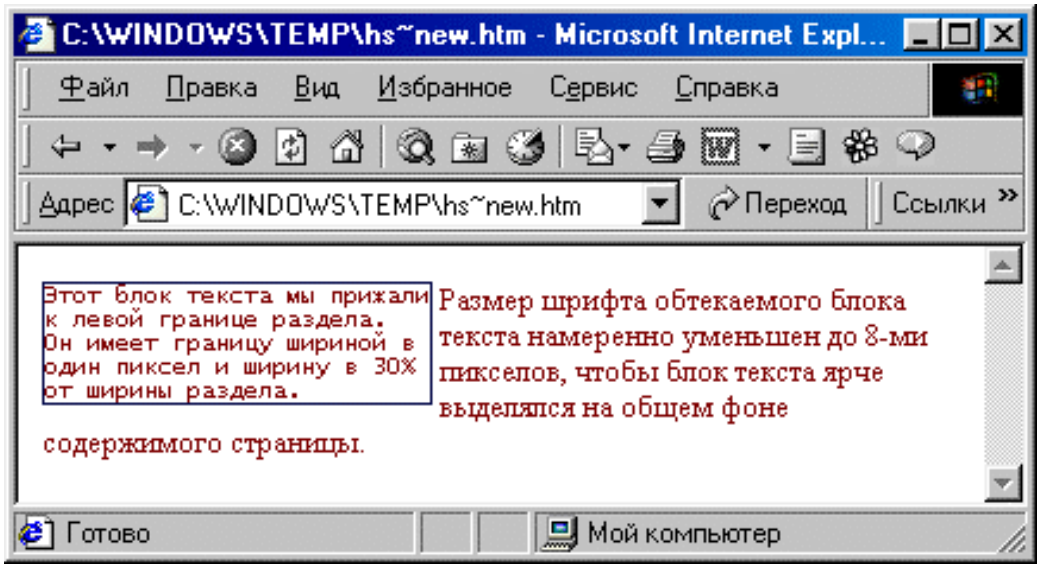


Рис.7. 13

При использовании значения right блок текста будет прижат вправо.

Второй атрибут описания стилей clear позволяет управлять собственно обтеканием. Он не допускает наличия "плавающих" блоков около блока текста. Атрибут может принимать значения: right, left, none, both:

Управление цветом в CSS

Каскадные таблицы стилей (CSS) в первую очередь описывают свойства текста. Это касается как текстовых блоков, так и строковых элементов разметки содержания страницы. В данном разделе речь пойдет об управлении отображением цвета текста (color) и цвета фона (background-color), на котором отображается текст.

Кроме цвета текста и цвета фона CSS позволяет определять цвет границы текстового блока (border-color).

Атрибуты стилей, которые мы собираемся рассмотреть, согласно спецификации Microsoft, относятся к группе атрибутов Color and Background Properties. Всего в эту группу входит семь атрибутов, шесть из которых определяют свойства фона. Кроме цвета фона и его прозрачности, можно управлять фоновой картинкой (координатами ее размещения и способами повторения). К сожалению, Netscape Navigator большинство из этих атрибутов не поддерживает, поэтому мы не будем рассматривать их подробно.

В Internet Explorer различна фоновый цвет заливает весь блок или строковый элемент вне зависимости от наличия в нем текста.

Цвет текста

В HTML для управления цветом отображаемого текста используется элемент FONT. Его аналогом в CSS является атрибут color. Этот атрибут можно применять как для блочных, так и для строковых элементов разметки. Рассмотрим в качестве блочного элемента разметки ячейку таблицы(Рис.7.14):

```
TD { color:darkred; }
```

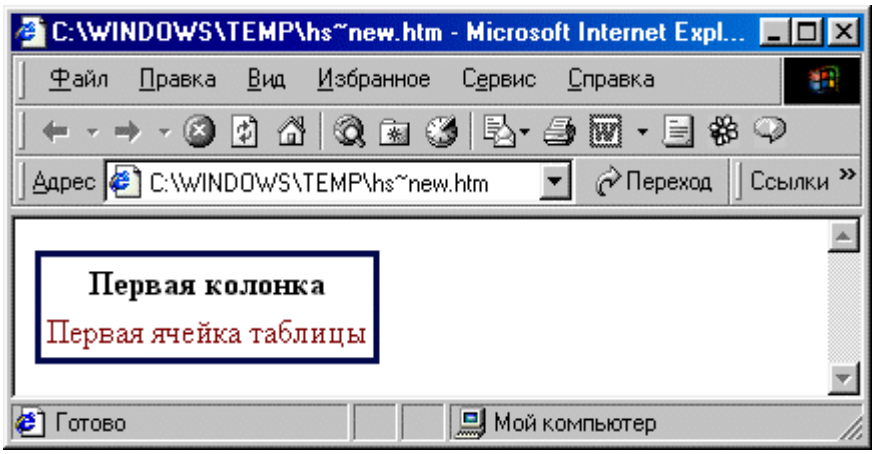


Рис. 7.14.

В данном примере цвет текста определен только для обычной ячейки, поэтому содержание заголовка колонки отображается основным цветом (#003366).

При определении цвета текста для блочного элемента весь текст этого элемента отображается заданным цветом(Рис.7.15). Частичное изменение цвета возможно, если поместить строковый элемент разметки внутрь блочного:P { color:darkred; }

```
I { color:#003366;font-style:normal; }
```

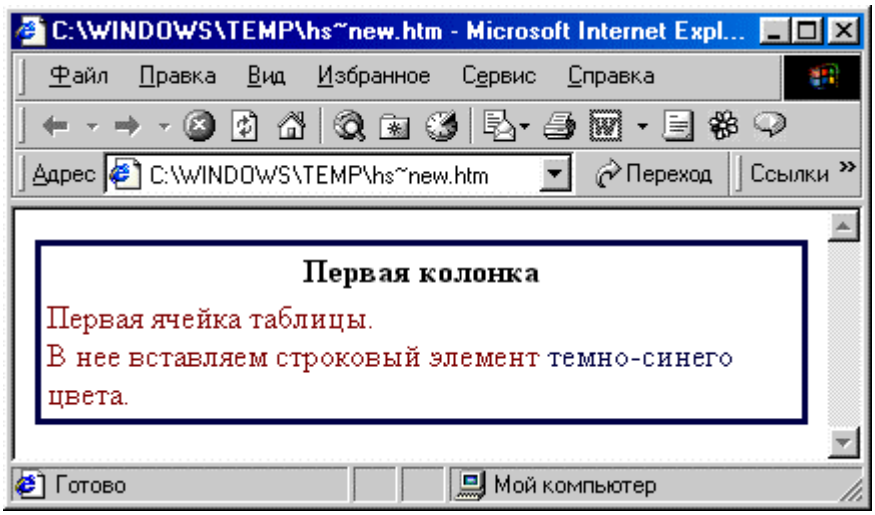


Рис. 7.15

В данном примере в качестве блочного элемента мы используем параграф, а в качестве строкового элемента (in-line) применяем I. Таблица в данном случае большого значения не имеет, но используется для единообразия с предыдущим примером. В нее мы помещаем параграф со встроенным в него in-line элементом разметки(Рис.7.16).

Цвет фона текста

В HTML цветом фона можно управлять только для конкретного блочного элемента разметки. Таким элементом может быть вся страница:

```
<BODY BGCOLOR=...>...</BODY>
```

Или, например, таблица:

```
<TABLE BGCOLOR=...>...</TABLE>
```

В приведенном ниже примере для выделения текста применено инвертирование цвета фона и цвета текста:

```
<SPAN STYLE="background-color:black;color:white;">
как строковые элементы разметки
```

```
</SPAN>
```

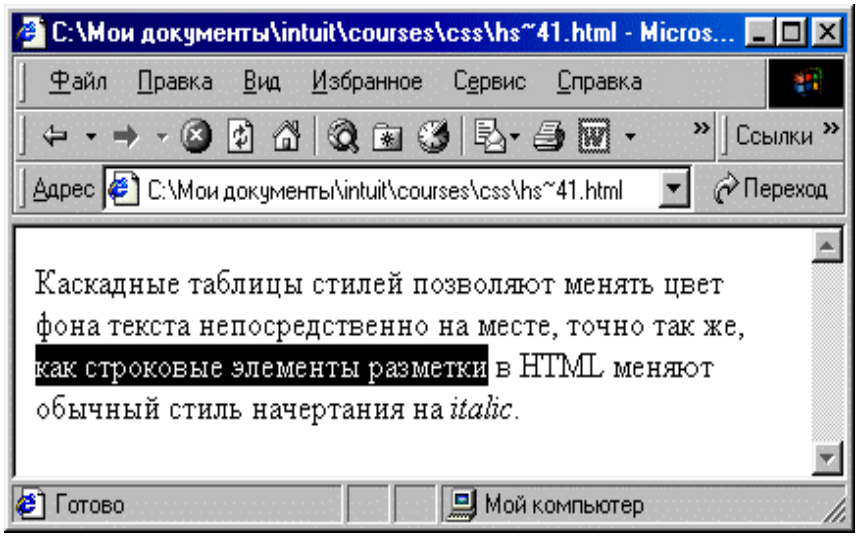


Рис. 7.16

При использовании цвета фона следует помнить, что поддержка этого атрибута реализована для всех блочных элементов разметки только в Internet Explorer 4.0. Для работы с фоном элементов существует несколько атрибутов, которые поддерживаются только в Internet Explorer, начиная с версии 4.0: background-image; background-repeat; background-attachment; background-position. Все свойства фона можно описать в атрибуте background:

```
background:transparent|color url repeat
           scroll position
```

Пример:

```
P { background: gray
    http://intuit.ru/intuit.gif
    no-repeat fixed center center; }
```

Однако при всем изобилии возможностей, злоупотреблять ими не стоит.

Шрифт

Шрифтам в компьютерной графике всегда уделялось много внимания, и World Wide Web не является исключением. Но все богатство и разнообразие существующих шрифтов для русского языка ограничено фактически тремя шрифтами: serif (обычно Times или другой шрифт с засечками), sans-serif (Arial, Helvetica или другой шрифт без засечек) и

monospace (Courier). Если быть точным, то здесь перечислены семейства шрифтов. Обычно каждое из этих семейств представлено только одним кириллическим шрифтом.

Гарнитура шрифта - это набор начертаний одного шрифта. Шрифт может иметь "прямое" начертание (normal), курсив (italic), "скошенное" (oblique), усиленное по насыщенности ("жирное", bold), "мелкое" (капитель, small-caps) и т.п.

Наиболее распространенные гарнитуры в российской части Web - это Times, Arial, Courier. Причем все они принадлежат к разным группам шрифтов. Times - это пропорциональный шрифт "с засечками" (serif), Arial - это пропорциональный шрифт "без засечек" (sans-serif), а Courier - это моноширинный шрифт (monospace). В Unix вместо Arial чаще применяется Helvetica.

Автор документа для управления отображением букв может применить несколько атрибутов, влияющих на шрифт:

1. font-family - семейство начертаний шрифта (гарнитура);
2. font-style - прямое начертание или курсив;
3. font-weight - "усиление" (насыщенность) шрифта, "жирность" букв;
4. font-size - размер шрифта (кегль). Задается в пикселах (px) и типографских пунктах (pt; 0,35 мм). Кегль можно задавать не только в абсолютных единицах, но и в относительных. Кроме процентов, существует еще несколько условных единиц измерения кегля, которые можно применять в CSS: small, x-small и xx-small, существуют размеры large, x-large и xx-large. Кроме того, есть larger, smaller и medium.
5. font-variat - вариант начертания (обычный или мелкими буквами - капитель).

Все эти параметры можно совместить в одном атрибуте font:

```
font:bold 12pt sans;
```

Правда, нет никакой уверенности в том, что последнее определение шрифта будет работать во всех браузерах.

При использовании различных гарнитур (font-famaly) следует помнить, что наличие или отсутствие необходимой автору гарнитуры всецело зависит от предпочтений пользователя. Для кириллицы это может вылиться в появление абракадабры там, где автор применяет отсутствующие у пользователя шрифты.

Самое неприятное, с чем можно столкнуться при использовании шрифтов - это несоответствие моноширинных шрифтов, которые

применяются в HTML-формах. Обратная связь с пользователем в этом случае невозможна.

Спецификация CSS предусматривает перечисление шрифтов в описаниях стилей, что позволяет частично решить проблему подбора шрифта. К сожалению, в Unix и Windows шрифты не согласованы. Фактически, при разработке страниц в CSS используются только классы шрифтов (serif, sans-serif и monospace).

Текст

В этом разделе мы рассмотрим те свойства текстового фрагмента, которые остались без внимания в разделах, посвященных блокам текста и шрифтам.

При обсуждении свойств блочных элементов разметки речь шла о параметрах, относящихся к блоку как целому. Мы не рассматривали внутренние характеристики текста.

Рассказывая о шрифтах, мы акцентировали внимание на начертаниях символов как таковых, а не на их соотношении.

Тем не менее в стороне остались такие важные характеристики текстового фрагмента, как:

- межбуквенные расстояния;
- высота строк;
- выравнивание;
- отступ в первой строке параграфа;
- преобразования начертания.

Все эти атрибуты сгруппированы в свойства текстовых фрагментов (Text Properties).

Межбуквенные расстояния

Расстояние между буквами автоматически регулируется размером шрифта – кеглем. Чем больше размер шрифта, тем больше расстояние между буквами. На пропорциональном шрифте (моноширинном) межбуквенное расстояние зависит от начертания букв и показать его как расстояние между буквами достаточно сложно. У моноширинного шрифта размер символа фиксирован, поэтому и расстояние между буквами прослеживается четко.

Однако не всегда удобно управлять межбуквенным растоянием через кегль (font-size). Бывают случаи, когда нужно либо уплотнить строку, либо увеличить расстояния между буквами(Рис.7.17).

```
<P STYLE="font-family:monospace;
letter-spacing:5pt;
color:darkred">
Межбуквенное расстояние 5pt</P>
<P STYLE=
```

```
"font-family:monospace;
letter-spacing:10pt;
color:darkred">
Межбуквенное расстояние 10pt
</P>
```

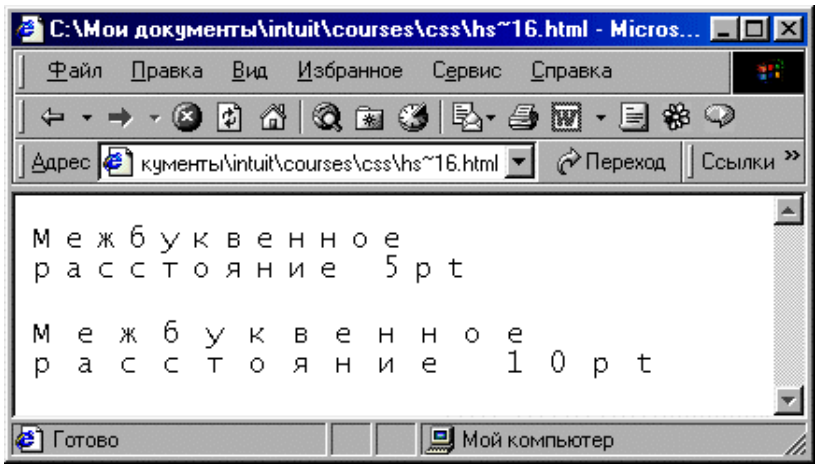


Рис. 7.17

Выравнивание

По умолчанию все слова в параграфе прижаты влево. Левый край параграфа таким образом оказывается выравненным. Точно так же может быть выравнен правый край параграфа или блока текста, и даже оба края вместе.

В обычной HTML-разметке такой эффект достигается за счет применения атрибута ALIGN. Аналогичный результат в CSS достигается за счет атрибута text-align (Рис.7.18):

```
<P STYLE="text-align:right;color:darkred;">
Этот параграф выравнен по правому краю. Все
строки справа кончаются на границе раздела.
А вот слева они начинаются с различным
отступом от левого края.</P>
<P STYLE="text-align:justify;color:darkred;">
Этот параграф выравнен левому и правому краю.
Все строки справа кончаются на вертикальной
границе раздела. Все строки слева теперь
начинаются также с вертикальной границы
раздела.</P>
```

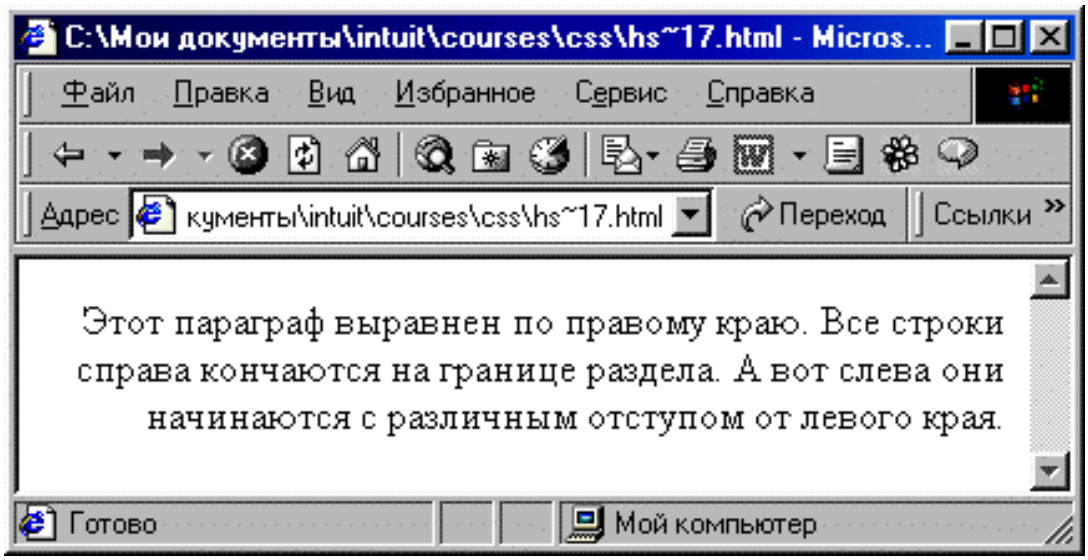


Рис. 7.18

Выравнивать текст можно в любом блочном элементе. Причем можно не только выравнивать текст по краям блочного элемента, но и центрировать его (<P STYLE="text-align:center;">...</P>).

Преобразование шрифта

Преобразование шрифта подразумевает капитализацию слов, перевод всех "больших" и "маленьких" букв в большие, или, наоборот, получение одних строчных. Рассмотрим несколько примеров(Рис.7.26):

```
<P STYLE="text-transform:uppercase;">
Сделать заглавными</P>
<P STYLE="text-transform:lowercase;">
Сделать строчными</P>
<P STYLE="text-transform:capitalize;">
Сделать заглавными первые буквы в словах</P>
```

Обратите внимание, что выполнение преобразований зависит от алгоритма преобразования символов. В нелокализованных программах переход от строчных букв к прописным осуществляется путем простого смещения по таблице ASCII, что для русского алфавита не приемлемо.

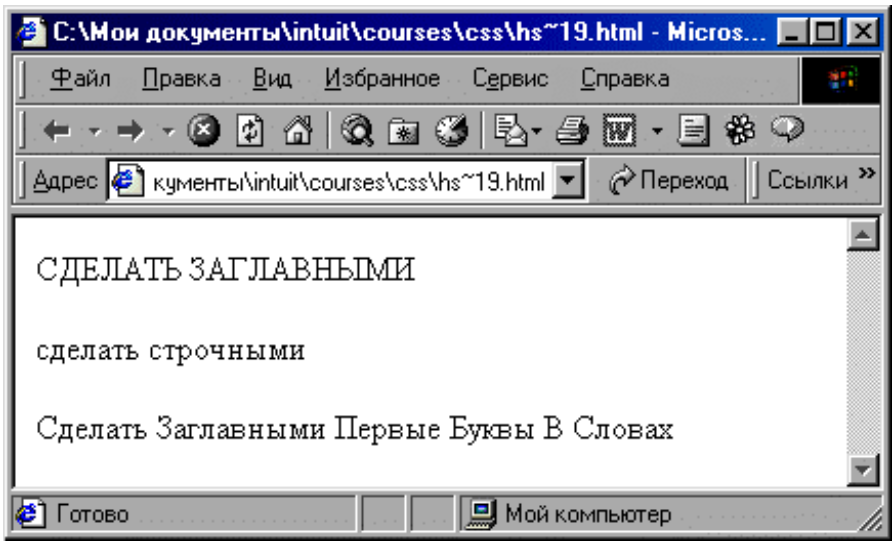


Рис.7 19

Еще один вид преобразования шрифта – это подчеркивание, перечеркивание или надчеркивание слов. Выполняется такое преобразование с помощью атрибута text-decoration(Рис.7.20):

```
<P STYLE="text-decoration:line-through;">
Перечеркнем это предложение.</P>
<P STYLE="text-decoration:underline;">
Подчеркнем это предложение.</P>
```

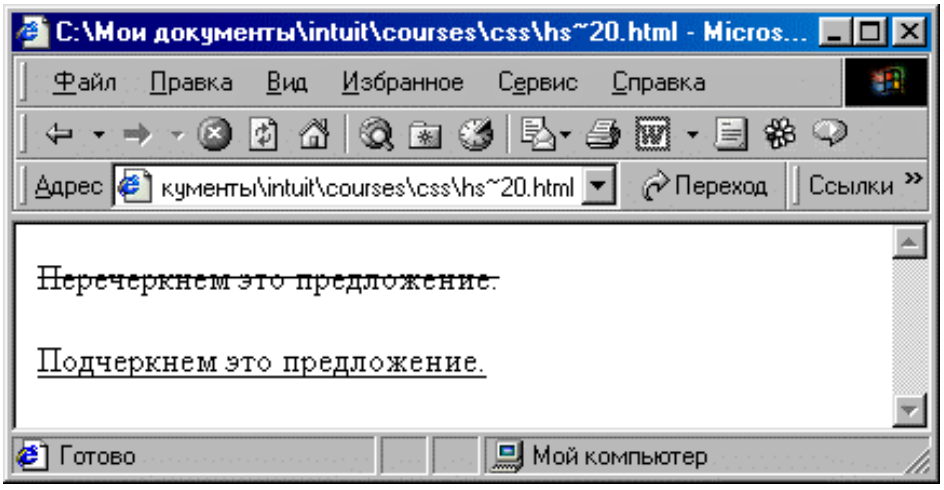


Рис 7.20

Для того, чтобы преобразование работало, необходимо соответствующее начертание (подчеркнутые или перечеркнутые начертания букв). Очень сложно найти гарнитуру, в которой было бы

начертание с надчеркнутыми буквами. Отмена декора происходит, если использовать в text-decoration значение none.

Первая строка параграфа

При оформлении параграфов в технологии CSS автор может воспользоваться "красной" строкой, такую возможность предоставляет ему атрибут text-indent. Речь идет о горизонтальном отступе в первой строке параграфа относительно его левого края(Рис.7.21):

```
<P STYLE="text-indent:20pt;">
Этот параграф мы начнем со строки с
горизонтальным отступом в двадцать
типографских пунктов от левого края параграфа.
</P>
<P STYLE="text-indent:-10pt;">
А в этом параграфе мы применим отрицательный
горизонтальный отступ в первой строке параграфа.
</P>
```

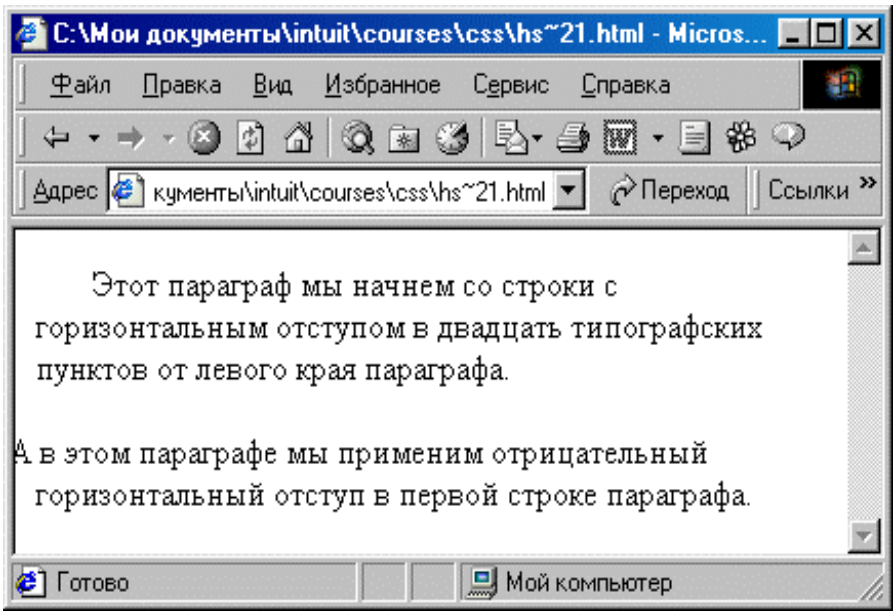


Рис. 7.20

Отрицательные значения атрибутов – это нормальная практика CSS. Там, где применение отрицательного значения оправдано, например, в случае смещения вложенного блока текста относительно охватывающего элемента разметки, можно указывать отрицательные атрибуты смещения.

Кроме text-indent в CSS для оформления первой строки параграфа зарезервирован модификатор стиля first-line. Он позволяет не только

задать горизонтальное смещение, но и определить другие параметры параграфа:

```
P:first-line { color:red; }
```

Еще один параметр, который влияет на отображение первой строки параграфа – первая буква первой строки. Ее отображением управляет модификатор first-letter:

```
P:first-letter { font-size:20pt; }
```

К сожалению, оба названных модификатора реализованы не во всех версиях браузеров, поэтому для верности применяют элементы разметки FONT и TABLE.

Межстрочное расстояние

В CSS межстрочное расстояние определяется параметром line-height. Он задает расстояние не между строками, а между базовыми линиями строк. Проще говоря, если, например, взять букву "н" и напечатать ее последовательно друг под другом, то line-height будет равно расстоянию между двумя одинаковыми точками букв.

Посмотрим, как этот параметр влияет на взаимное расположение строк(Рис.7.21).

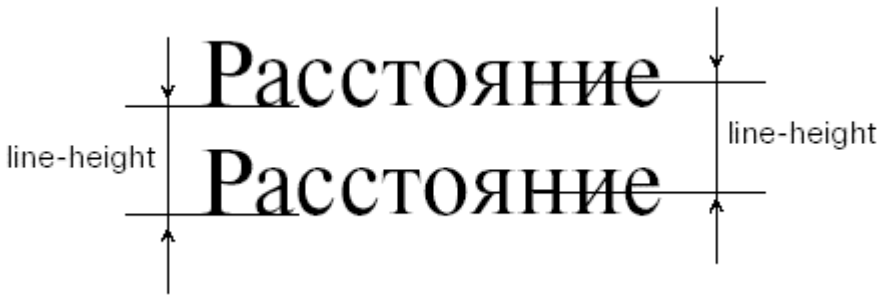


Рис.7. 21

```
<P STYLE="line-height:12pt;font-size:12pt;
color:darkred;">
Этот параграф мы набрали кеглем 12 pt.
Line-height задан в 12 pt.</p>
<P STYLE="line-height:24pt;font-size:12pt;
color:darkred;">
Этот параграф мы набрали кеглем 12 pt.
Line-height задан в 24 pt.</P>
<P STYLE="line-height:6pt;font-size:12pt;
color:darkred;">
Этот параграф мы набрали кеглем 12 pt.
```

```
Line-height задан в 6 pt.</P>
```

Первый пример набран со значением line-height, равным размеру кегля. Во втором примере значение line-height вдвое превышает размер кегля. В третьем примере значение line-height в два раза меньше размера кегля – строки стали "наползать" друг на друга.

В связи с использованием line-height следует обратить внимание на применение in-line картинок на HTML-страницах. Под in-line картинкой здесь имеется в виду картинка, которая встроена в тело документа при помощи элемента IMG, но не с новой строки и не как элемент таблицы(Рис. 7.22).

```
<P STYLE="color:white;background-color:black;
font-size:20px;">
В эту строку мы встраиваем картинку - <IMG
SRC="inline.gif" BORDER="0" WIDTH="24"
HEIGHT="24" ALIGN="top">,
на которой изображены концентрические круги.
</P>
```

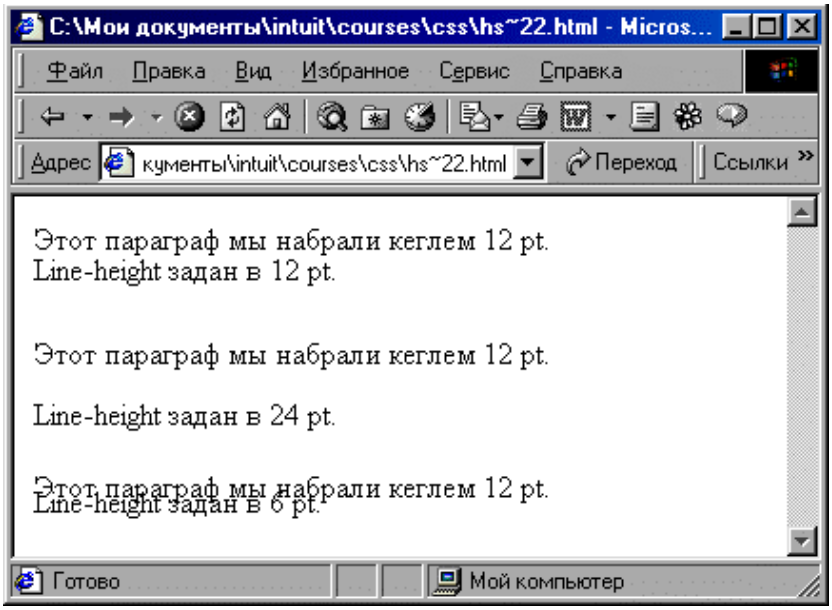


Рис. 7.22

Картинка имеет размеры 24x24 пиксела и выравнена по верхнему краю строки. Ее размер больше размера кегля (20 px), поэтому межстрочное расстояние увеличено браузером автоматически. Таким образом, можно точно позиционировать текст и графику в строке (Рис. 7.22).

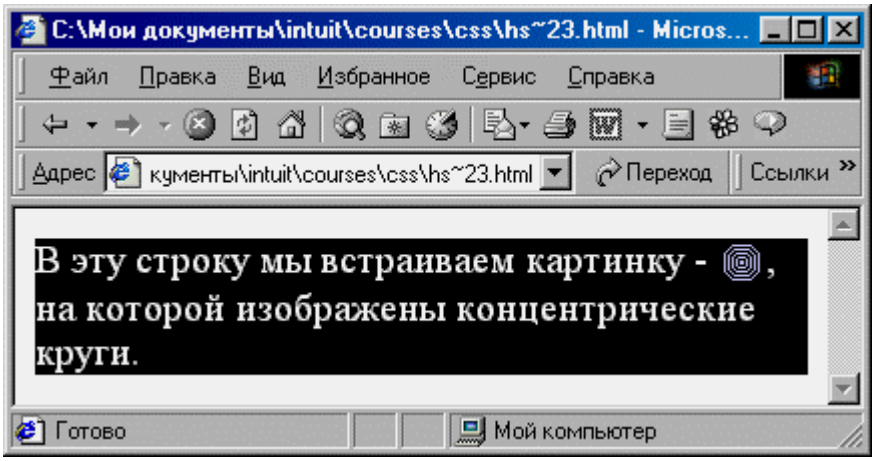


Рис. 7.22

Списки

При отображении списков CSS позволяет управлять формой и изображением "пулек" (bullets) списка. "Пулька" (bullet) – это символ, стоящий перед элементом списка. Например, в неупорядоченном списке (unordered list) перед элементом списка ставится "жирная" точка:

- Первый элемент списка
- Второй элемент списка
- Третий элемент списка

CSS позволяют управлять формой "пулек" и заменять "пульки" картинками. Любопытно, что управление отображением элементов списка отнесено к набору свойств, в который входит атрибут display. У этого атрибута может быть только одно значение – none. Если элемент в своем описании имеет атрибут display, и этот атрибут равен none, то он не отображается браузером вообще:

```
<UL STYLE="display:none;">
<LI>Первый элемент списка
<LI>Второй элемент списка
<LI>Третий элемент списка
</UL>
```

Если посмотреть HTML-код данного документа, то за примером описания списка следует код, который браузер не отобразил.

Атрибут display управляет отображением документа на дисплее компьютера, но не распространяется на другие среды отображения документа. Например, при печати скрытый список должен быть отображен. Однако, на самом деле он не отображается и при печати.

Форма "пулек"

Форма "пульки" в виде "жирной" точки несколько непривычна. Обычно в машинописных документах используют черту. С другой стороны, в рекламных материалах часто в качестве "пульки" применяют квадрат или другой символ типографского набора, а также графическую картинку. CSS позволяет управлять формой "пульки" через атрибут list-style-type(Рис.7.23):

```
<UL STYLE="list-style-type:square;">
<LI>В виде "пульки" используем квадрат
</UL>
<UL STYLE="list-style-type:disk;">
<LI>В виде "пульки" используем диск
</UL> <UL STYLE="list-style-type:circle;">
<LI>В виде "пульки" используем круг
</UL>
```

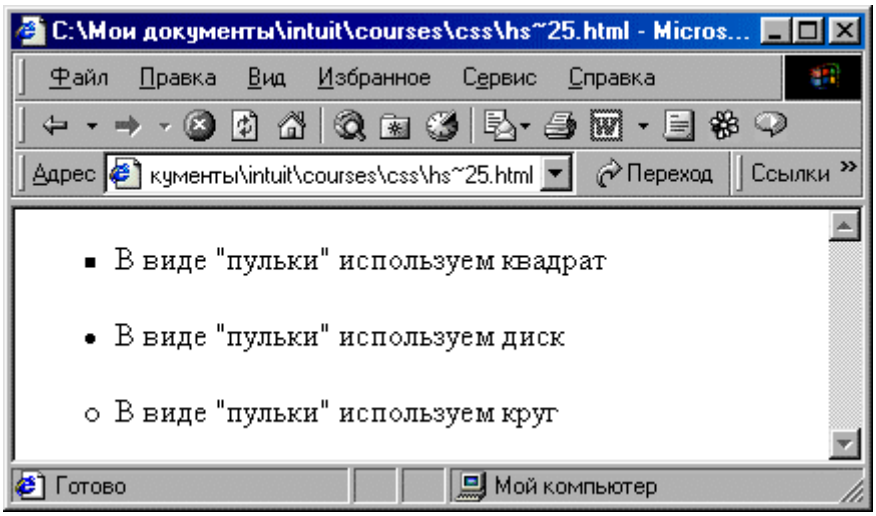


Рис. 7.23

До сих пор мы обсуждали только неупорядоченные списки (UL), но управлять отображением "пулек" можно и в упорядоченных списках (OL)(Рис.7.24):

```
<OL STYLE="list-style-type:lower-roman;
color:darkred;">
<LI>...
```

```
...
</OL>
<OL STYLE="list-style-type:upper-alpha;
color:darkred;">
<LI>...
...
</OL>
<OL STYLE="list-style-type:lower-alpha;
color:darkred;">
<LI>...
...
</OL>
```

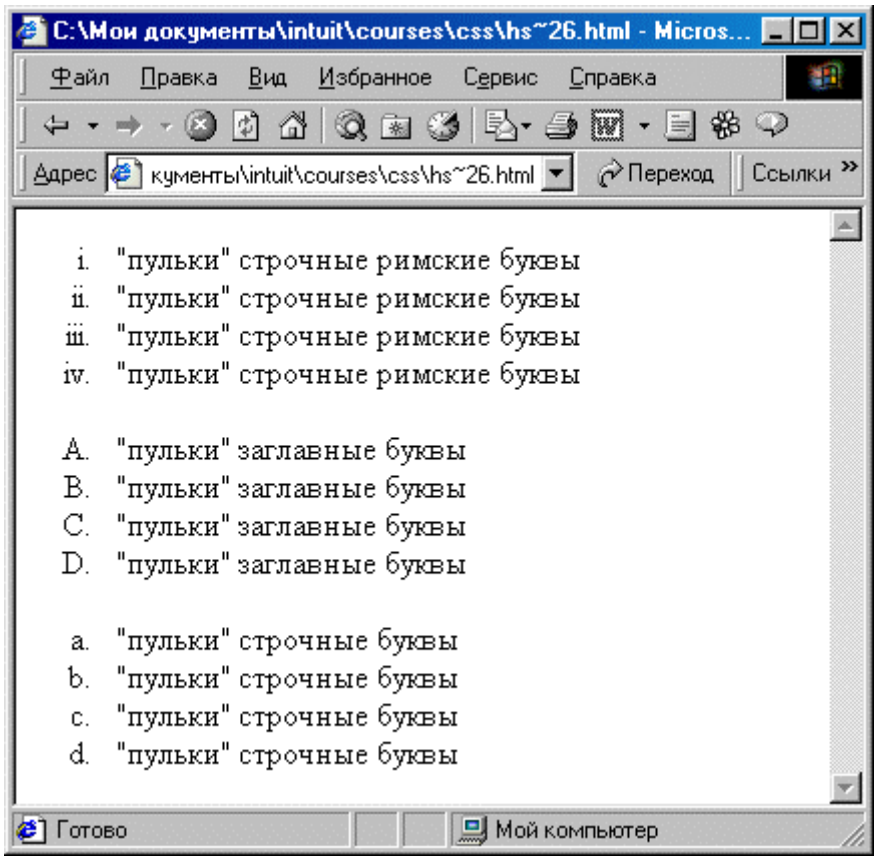


Рис.7.24

CSS позволяют вообще отказаться от "пулек". Для этого нужно указать значение атрибута list-style-type равным none.

"Пульки"-картинки

Если стандартные формы "пулек" автора страницы не устраивают, он может использовать нестандартные. Для этого ему придется "пульку" нарисовать самому и в виде графического файла разместить на Web-узле.

У такой "пульки" есть URL, который используется в CSS для обращения к ней.

```
    <UL STYLE="list-style-image:url(bimage.gif);">
<LI>Элемент списка расположен за чертой
</UL>
```

Картинка может быть и более замысловатой. Это уже зависит от фантазии автора документа. Например, можно создать картинку-стрелочку(Рис. 7.34):

```
    <UL STYLE="list-style-image:url(barrow.gif);">
    <LI>Элемент списка расположен за стрелкой
    </UL>
```

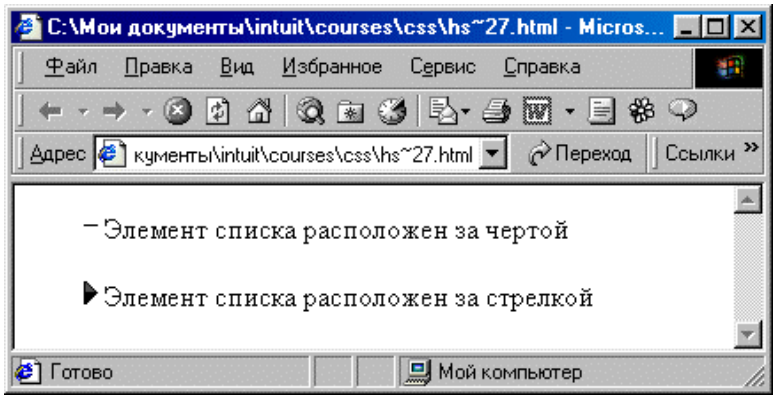


Рис. 7.34

Здесь надо признаться в маленьком обмане. Если вы пользователь Internet Explorer, то все, что здесь написано – верно. Фрагмент кода, представленный перед примером, является его точной копией.

8 JAVA аплеты

Основные понятия

Основная идея, положенная в основу использования аплетов на Web-страницах довольно проста: страница, описанная на языке HTML, должна сообщать броузеру, какой аплет загрузить и где его разместить. Дескриптор, необходимый для использования аплета, должен сообщать броузеру следующую информацию:

1. Откуда получить файлы с классами.
2. Как расположить аплет на странице (его размеры, координаты и т.д.).

Броузер загружает файлы с классами из сети (или из каталога на компьютер пользователя) и автоматически запускает аплет на выполнение, используя виртуальную машину языка Java (Java Virtual Machine).

Кроме аплетов, Web-страница может содержать все остальные элементы HTML: разнообразные шрифты, маркированные списки, графические изображениям и т.д. Аплеты – это всего лишь часть гипертекстовой страницы. Всегда стоит иметь в виду, что язык Java не является средством для разработки гипертекстовых страниц, это инструмент, позволяющий их оживлять. Отсюда вовсе не следует, что элементы графического пользовательского интерфейса в аплете, написанном на языке Java важны. Однако эти элементы должны быть согласованы с соответствующим дизайном Web-страницы, созданным с помощью языка HTML.

Для того чтобы решить эту проблему, компания Sun разработала и распространила инструмент под названием "Java Plug-In" (ранее известный под именем “Activator”). Используя различные механизмы расширения возможностей броузеров Internet Explorer, этот инструмент можно совершенно свободно встроить в броузер. Это позволяет использовать их для запуска аплетов с помощью внешней среды поддержки выполнения программ на языке Java (Java Runtime Environment), разработанной компанией Sun. Предполагается, что версии виртуальных машин в этих броузерах будут постоянно обновляться, обеспечивая программистам доступ к самым современным и мощным свойствам языка Java.

На рисунке 8.1 представлено место аплетов в иерархии класов.

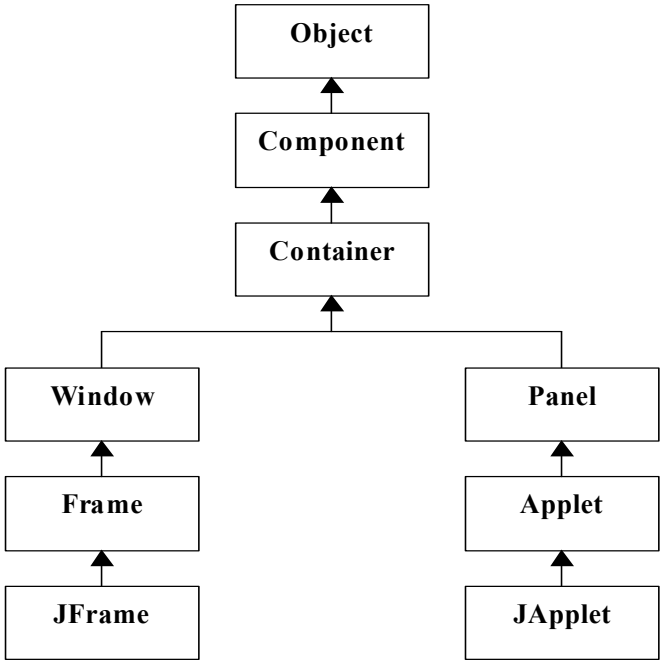


Рис. 8.1

Итак из вышесказанного следует, что аплеты – это маленькие приложения, которые размещаются на серверах в Интернет, транспортируются клиенту по сети, автоматически устанавливаются и запускаются на месте как часть документа всемирной паутины.

По традиции перепишем программу HelloWorld в виде аплета. Прежде, чем сделать это, нужно пояснить, что, с точки зрения программиста, аплет не представляет собой ничего удивительного. Это просто класс, который (прямо или косвенно расширяет класс java.applet.Applet. Отметим, что, хотя пакет java.applet не входит в пакет AWT, он представляет собой компонент библиотеки AWT. Соответствующая цепочка наследования показана на рисунке 36. В рамках данного курса для реализации аплетов мы будем пользоваться библиотекой Swing. Все наши аплеты будут расширять класс JApplet, являющийся суперклассом всех аплетов, создаваемых с помощью библиотеки Swing. Как показано на рис. 8.1, класс JApplet является непосредственным подклассом обычного класса Applet.

```
import java.awt.Graphics;
class Hello1 extends java.applet.Applet {
        public void paint( Graphics g ) {
                g.drawString( "Hello, World!", 50,
25 );
        }
}
```

Для того, чтобы запустить этот апплет необходимо в HTML-документ поместить следующие строки:

```
<applet code="Hello1" width=200 height=50>
</applet>
```

Эти строки говорят броузеру , что он должен загрузить оттранслированный код java-апплета HelloWorld.class, котрый находиться в том же каталоге, что и html-файл. Начальные размеры апплета указываются в теге <applet>. В данном случае 200x50 пикселов.

Синтаксис тега <applet>

```
<applet
        [CODEBASE= codebase URL]
        CODE=appletFile
        [ALT=alternateText]
        [NAME=appletInstanceName]
```

```
            WIDTH=pixels HEIGHT=pixels
            [ALIGN=alignment]
            [VSPACE=pixels] [HSPACE=pixels]
    >
            [<PARAM   NAME=AttributeName1
VALUE=AttributeName1>]
            [<PARAM   NAME=AttributeName2
VALUE=AttributeName2>]
...     [HTML-текст , отображаемый при отсутствие
поддержки Java]
</applet>
```

CODEBASE= codebase URL

CODEBASE -необязательный атрибут,здающий базовый URL кода апплета, являющийся каталогом в котором будет выполнятся поиск исполняемого файла апплета (задаваемого в признаке CODE).Если этот атрибут не задан, по умолчанию используется каталог данного html-документа.

CODE=appletFile

CODE-обязательный атрибут, задающий имя файла, в котором содержиться оттранслированный код апплета.Имя апплета задается относительно codebase, то есть либо от текущего каталога, либо от катологa,указанного в CODEBASE.

ALT=alternateText.

Признак ALT-необязательный атрибут, задающий короткое текстовое сообщение , которое должно быть выведено в том случае , если используемый броузер распознает синтаксис тега <applet>,но выполнять аплеты не умеет.

NAME=appletInstanceName

NAME необязательный атрибут, используемый для задания имени данного экземпляра апплета . Присвоение апплетам имен необходимо для того, чтобы другие аплеты на этой же странице могли находить их и общаться с ними. Для того, чтобы получить доступ к подклассу MyApplet класса Applet с именем "Duke",

нужно написать:

```
MyApplet a=getAppletContext().getApplet("Duke");
```

ALIGN=alignment

ALIGN-необязательный атрибут, задающий стиль выравнивания апплета. Возможные значения атрибута-LEFT, RIGHT, TOP, TEXTTOP, MIDDLE, ABSMIDDLE, BASELINE, BOTTOM, ABSBOTTOM.

[VSPACE=pixels] [HSPACE=pixels]

Эти необязательные атрибуты задают ширину свободного пространства в пикселях сверху и снизу апплета (VSPACE), и слева и справа от него (HSPACE).

PARAM NAME=appletAttribute VALUE=value

Этот тег дает возможностьпередавать из html -страницы апплету необходимые ему аргументы. Аплеты получают эти атрибуты, вызывая метод getParameter() . Метод getParameter() возвращает значение типа String.

Пример :

```
<applet code=testing width=40 height=40>
<param  name=fontName value=Univers>
<param  name=fontSize value=14>
</applet>
```

Эти параметры извлекаются из апплета следующим образом:

```
String FontName=getParameter("fontName");
int
FontSize=Integer.parseInt(getParameter("fontSize"));
```

При создание апплета программист должен заместить некоторые методы класса Applet(Таблица 8.1)

.

Порядок вызова методов класса Applet

Таблица 8.1

init	Вызывается первым. В нем инициализируются все переменные. Вызывается только один раз при загрузке апплета.
start	Вызывается после init. Используется в качестве стартовой точки для возобновления работы, после того как апплет был остановлен.start вызывается при каждом вызове html- документа, содержащего данный апплет.

paint	Вызывается каждый раз при повреждении апплета. Например перекрытие окна апплета другим окном. В таких случаях после того, как апплет становится видимым, для восстановления его изображения вызывается метод paint.
update	Используемый по умолчанию метод update закрашивает апплет цветом фона по умолчанию, после чего вызывает метод paint.
stop	Вызывается в тот момент, когда броузер покидает html-дакумент содержащий апплет. При вызове метода stop апплет еще работает. Этот метод используется для приостановления тех процессов, работа которых необязательна при невидимом апплете. После того, как к этой странице снова обратятся необходимо возобновить их работу в методе start.
destroy	Вызывается тогда, когда среда решает, что апплет надо полностью удалить из памяти. В этом методе надо освободить все ресурсы , которые использовал апплет.
repaint	Используется для принудительного перерисовывания апплета. Вызывает метод update.
repaint(time)	Можно вызывать метод repaint, устанавливая крайний срок перерисовки.
repaint(x,y,w,h)	Перерисовывает данный прямоугольник.
repaint(time,x,y,w,h)	Вызывает update для перерисовки заданного прямоугольника, до того как истечет заданное время time.

9 Некоторые встроенные классы Java

Класс Graphics

Основные методы: drawOval, fillRect, fillOval, drawArc, drawPolygon, fillPolygon...

Класс Color

Вы можете использовать статические переменные этого класса (например, Color.black). В классе Color для установки цвета предусмотрены статические переменные: black, white, red, green, blue, cyan, yellow, magenta, orange, pink, gray, darkGray, lightGray. Для создания нового цвета используется один из трех конструкторов класса:

```
Color(int,int,int)
```

Значение каждого параметра(0-255) есть соответственно количество красного ,зеленого и синего цвета.

```
Color(int)
```

Здесь задается значение красной, зеленой и голубой компоненты в упакованном виде.

Пример:

```
int newRed=(0xff000000|(0xc0<<16)|(0x00<<8)|0x00);
Color darkRed=new Color(newRed);
```

Color(float,float,float)

Значение каждого параметра задает количество красного,зеленого и голубого в диапазоне от 0 до 1. На примере покажем как работают эти методы.

```
import java.applet.*;
import java.awt.*;

public class Graphic extends Applet {
  public void paint(Graphics g){
         g.setColor(Color.blue);
         g.drawLine(0,0,100,100);
         g.setColor(Color.red);
         g.drawOval(5,5,50,50);
         g.setColor(Color.green);
         g.drawRect(50,50,20,20);
         g.setColor(Color.yellow);
         g.drawRoundRect(0,50,20,20,10,10);
         g.setColor(Color.magenta);

g.fillArc(50,0,20,20,(int)3.14/3,(int)3.14/2);
    }
}
```

Результаты работы апплета(Рис.9.1):

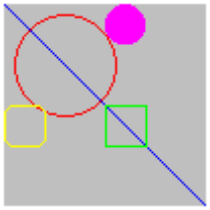


Рис.8.1

Шрифты

Консруктор класса Font создает новый шрифт с указанным именем. Пример:

```
import java.applet.*;
import java.awt.*;
public class whatfont_1 extends Applet{
 public void init(){
  Graphics g;
 }
 public void paint(Graphics g){
  String font_1="TimesRoman";
  Font font= new Font(font_1,Font.PLAIN,20);
  g.setFont(font);
  g.drawString(font_1,10,20);
  Font font1= new Font(font_1,Font.BOLD,20);
  g.setFont(font1);
  g.drawString(font_1,10,45);
  Font font2= new Font(font_1,Font.ITALIC,20);
  g.setFont(font2);
  g.drawString(font_1,10,70);
  Font font3= new
Font(font_1,Font.ITALIC|Font.BOLD,20);
  g.setFont(font3);
  g.drawString(font_1,10,95);
 }
 }
```

Результаты работы апплета(Рис.8.2):

Рис.8.2

FontMetrics

Позволяет программисту задавать положение выводимого в апплете текста.

Методы класса FontMetrics:

1. stringWidth - возвращает длину заданной строки для заданного шрифта.
2. bytesWidth, charsWidth - возвращает ширину указанного массива байтов для текущего шрифта.
3. getAscent, getDescent, getHeight - возвращает подъем, снижение и ширину шрифта.

Набор абстрактных классов для работы с окнами

Класс Component

Component - абстрактный класс, который инкапсулирует все атрибуты визуального интерфейса. Все элементы интерфейса пользователя, которые отображаются на экране и осуществляют взаимодействие с пользователем, являются его подклассами. Его компоненты отвечают за обработку ввода с клавиатуры, управление фокусом, взаимодействие с мышью, уведомлением о входе или выходе из окна, изменением размера, положение окон , а также за прорисовку своего собственного графического представления.

Класс Button

Класс Button - подкласс класса Component. Используется для того чтобы вызвать какое-либо действие, когда пользователь нажмет и отпустит нарисованную кнопку. Пример работы с классом Button:

```
import java.applet.*;
import java.awt.*;

public class button_1 extends Applet {
```

```
      public void init(){
       setLayout(null);
       int
width=Integer.parseInt(getParameter("width"));
       int
height=Integer.parseInt(getParameter("height"));
       Button yes=new Button("yes");
       Button no=new Button("no");
       Button maybe=new Button("undecided");
       add(yes);add(no);add(maybe);
       yes.reshape(0,0,width,height/3);
       no.reshape(0,height/3,width,height/3);
       maybe.reshape(0,2*height/3,width,height/3);
      }
    }
```

Результаты работы апплета(Рис.8.3):

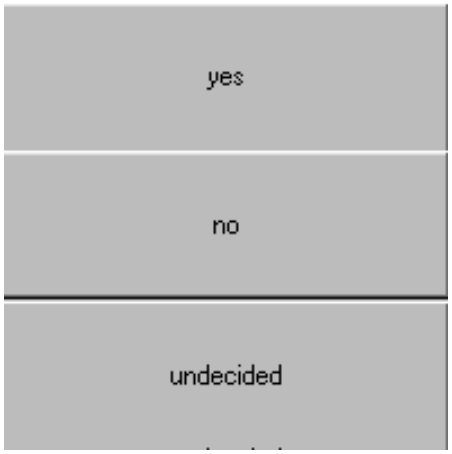


Рис.8.3

Класс Checkbox

Этот класс используется для выбора одной из двух и более возможностей. Для установки окна с пометкой используются методы setState и getState. Пример:

```
import java.awt.*;
import java.applet.*;

public class check_box extends Applet{
//кнопки располагаются по умолчанию
```

```
            setLayout(null);
    //параметры берем из html-документа
            int
width=Integer.parseInt(getParameter("width"));
            int
height=Integer.parseInt(getParameter("height"));
    //создаем объекты типа Checkbox
            Checkbox win95=new
Checkbox("windows95",null,true);
            Checkbox solaris=new Checkbox("solaris
2.x");
            Checkbox mac=new Checkbox("MacOS 7.5");
    //рисуем в заданных координатах
            win95.reshape(0,0,width,height/3);

    solaris.reshape(0,height/3,width,height/3);
            mac.reshape(0,2*height/3,width,height/3);
            }
    }
```

Класс Choice

Класс Choice используется при создании раскрывающихся списочных меню. Метод countItems возвращает количество пунктов в меню выбора . Вы можете задать пункт , который выбран в данный момент с помощью метода select , передав ему либо индекс (пункты меню перечисляются с нуля), либо строку , которая совпадает с меткой нужного пункта меню. С помощью getSelectedItem и getSelectedIndex можно получить соответственно, строку-метку и индекс выбранного в данный момент пункта меню. Пример:

```
    import java.applet.*;
    import java.awt.*;

    public class choice_1 extends Applet{
      public void init(){
       setLayout(null);
       int
width=Integer.parseInt(getParameter("width"));
       int
height=Integer.parseInt(getParameter("height"));
       Choice os=new Choice();
       Choice browser=new Choice();
       os.addItem("windows 95");
```

```
    os.addItem("Solaris 2.x");
    os.addItem("MacOS 7.5");
    browser.addItem("Netscape 1.1");
    browser.addItem("Netscape 2.0");
    browser.select("Netscape 2.0");
    add(os);
    add(browser);
    os.reshape(0,0,width,height/2);
    browser.reshape(0,height/2,width,height/2);
    }
}
```

Результаты работы апплета(Рис.8.4):

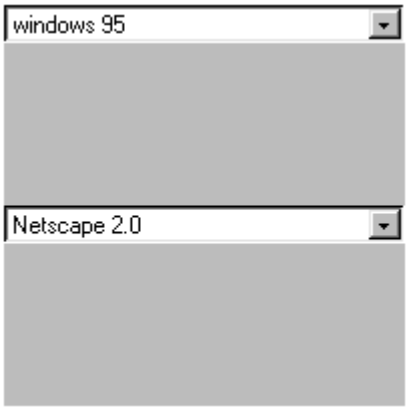


Рис.8.4

Класс Scrollbar

Используется для выбора подмножества значений между заданными минимальным и максимальным числами.Конструктор этого класса позволяет задавать ориентацию линейки прокрутки. Для этого предусмотрены константы VERTICAL и HORIZONTAL. Также с помощью конструктора можно задать начальное положение и размер движка, а также минимальное и максимальное значение , в пределах которых линейка проктрутки может изменять параметр. Для получения и установки текущего состояния линейки прокрутки используются методы getValue и setValue. Воспользовавшись методами getMinimum и getMaximum можно получить рабочий диапазон объекта. Пример:

```
import java.applet.*;
import java.awt.*;

public class Scroll extends Applet {
  public void init(){
   setLayout(null);
```

```
        int
width=Integer.parseInt(getParameter("width"));
        int
height=Integer.parseInt(getParameter("height"));
        Scrollbar                                    hs=new
Scrollbar(Scrollbar.HORIZONTAL,50,width/10,0,100);
        Scrollbar                                    vs=new
Scrollbar(Scrollbar.VERTICAL,50,height/2,0,100);
        add(hs);add(vs);
        int thickness=16;
        hs.reshape(0,height-thickness,width-
thickness,thickness);
        vs.reshape(width-thickness,0,thickness,height-
thickness);
      }
    }
```

Результаты работы апплета(Рис.8.5):

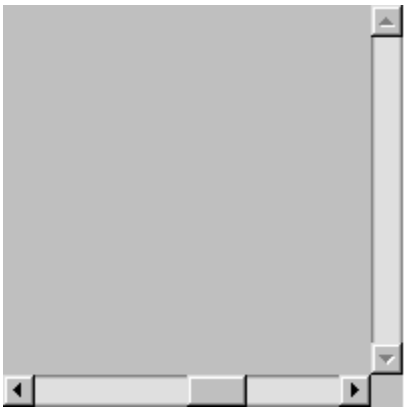


Рис.8.5

Класс Event

Любой компонент может обработать событие, заместив определенные методы, вызываемые используемой по умолчанию реализацией метода handleEvents класса Component. Этот метод вызывается с объектом класса Event, описывающего все возможные типы событий. В классе Event определены десятки констант , позволяющих использовать символические имена, например PGUP и HOME. Для работы со специальными событиями, например, с обратными вызовами из компонентов класса Button, Scrollbar и Menu придется замещать метод action. Этот метод вызывается с исходным событием и со вторым параметром, который представляет собой компонент пользовательского интерфейса, создавшим это событие.

Пример:

```
import java.awt.*;
import java.applet.*;

public class EventDemo extends Applet{
        static final int n=4;
        Label lab;
        public void init(){
//кнопки будут выводиться слева направо по 4 в ряд
        setLayout(new GridLayout(n,n));
//устанавливаем шрифт
        setFont(new
Font("Helvetica",Font.BOLD,24));
//берем высоту и ширину апплета как параметр
        int
width=Integer.parseInt(getParameter("width"));
        int
height=Integer.parseInt(getParameter("height"));
//определяем и рисуем кнопки
        for(int i=0;i0) add(new Button(""+k));
                }
        }
//определяем объект типа Label
          lab=new Label("?",Label.CENTER);
//устанавливаем шрифт
          lab.setFont(new
Font("Helvetica",Font.ITALIC,24));
// добавляем объект к нарисованным кнопкам
              add(lab);
    }
    public boolean action(Event e,Object o){
//если кнопка была нажата-выводим ее название
              if(o instanceof String){
                lab.setText((String) o);
          }
          return false;
    }
}
```

Результаты работы апплета(Рис.8.6):

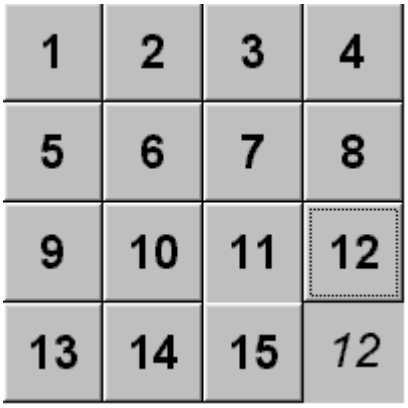


Рис.8.6

Обработка событий мыши

Для обработки событий мыши используются следующие методы класса **Mouse**:

1. mouseDrag - когда пользователь двигает мышь с нажатой кнопкой.
2. mouseDown - когда пользователь нажимает на клавишу мыши.
3. mouseUp - пользователь освобождает клавишу мыши

Пример.

```
import java.applet.*;
import java.awt.*;

public class Mouse extends Applet{
 String MouseDownEvent=null;
 String MouseUpEvent=null;
 String MouseDragEvent=null;
 public boolean mouseUp(Event event,int x,int y){
    MouseUpEvent="mouseUp:"+x+","+y;
    repaint();
    return(true);
 }
public boolean mouseDown(Event event,int x,int y){
    MouseDownEvent="mouseDown:"+x+","+y;
    repaint();
    return(true);
 }
```

```
    public boolean mouseDrag(Event event,int x,int y){
        MouseDragEvent="mouseDrag:"+x+","+y;
        repaint();
        return(true);
     }
     public  void paint(Graphics g){
        if(MouseDragEvent!=null)
g.drawString(MouseDragEvent,5,15);
        if(MouseUpEvent!=null)
g.drawString(MouseUpEvent,5,45);
        if(MouseDownEvent!=null)
g.drawString(MouseDownEvent,5,75);
     }
    }
```

Результаты работы апплета(Рис.8.7):

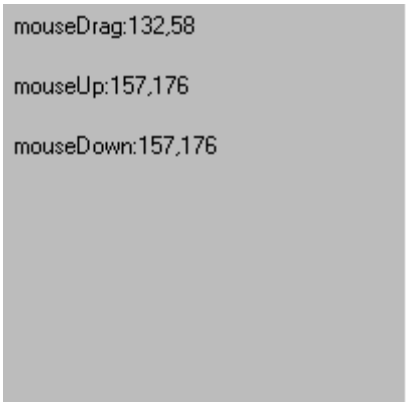


Рис.8.7

Работа с изображениями

Перед тем как апплет сможет загрузить изображение, он должен объявить объект Image. Например:

```
Image picture;
```

Программа должна использовать функцию getImage, чтобы связать данный объект с графическим файлом. Например:

```
picture=getImage(getCodeBase(),"ImageFile.gif");
getCodeBase возвращает URL апплета
ImageFile.gif-имя загружаемого файла(*.gif,*.jpg)
```

Если графическое изображение находиться не в том же каталоге, что и апплет, строка с именем файла должна включать путь к его каталогу.

Апплет использует функцию drawImage класса Graphics для вывода изображения. Например:

```
g.drawImage(picture,x,y,this);
```

где x,y-координаты верхнего левого угла изображения

Пример простого загрузчика изображения.

```
import java.applet.*;
import java.awt.*;

public class SimplImage extends Applet {
  Image Art;
  public void init(){

Art=getImage(getDocumentBase(),getParameter("img"));
  }
  public void paint(Graphics g){
   g.drawImage(Art,0,0,this);
  }
}
```

Когда апплет запускается в методе init(), он начинает загрузку art. Процесс загрузки изображения по сети хорошо заметен, поскольку встроенный интерфейс ImageObserver вызывает процедуру paint при каждом поступлении новой порции данных из сети. Но можно использовать ImageObserver для отслеживания загрузки изображения. Можно загружать изображения в память, а в это время выводить на экран другую информацию и только когда изображение целиком загрузится , вывести его на экран. Такой прием называется двойной буферизацией изображения. Целая переменная infoflags в imageUpdate поразрядно проверяется на наличие одного или нескольких флагов. Возможные флаги и информация которую они несут(Таблица 8.2).

:

Таблица 8.2

width	ширина изображения доступна и может быть взята из аргумента width.
height	высота изображения доступна и может быть взята из

	аргумента height.
properties	свойства изображения доступны их можно получить посредством art.properties.
somebits	доступны пиксели необходимые для рисования масштабированного варианта изображения. Область, содержащая новые пиксели , задается параметрами x,y,width и height.
framebits	еще один кадр ранее нарисованного изображения с несколькими кадрами готов для перерисовки. Параметры x, y ,width, height не содержат информацию.
allbits	обработка перерисовываемого изображения окончена , и оно может быть отрисовано в конечном виде.Параметры x, y ,width, height не содержат информацию.
error	при пересылке изображения возникла ошибка.Загрузка прервана.
abort	пересылка изображения была прервана.

Пример демонстрирующий двойную буферизацию графического изображения.

```
import java.applet.*;
import java.awt.*;

public class backgrou extends Applet {
Image picture;
 boolean  ImageLoaded=false;
 public void init()

{picture=getImage(getCodeBase(),getParameter("img"));
     Image
offScreenImage=createImage(size().width,size().height
);
     Graphics
offScreenGC=offScreenImage.getGraphics();
     offScreenGC.drawImage(picture,0,0,this);
     }
     public void paint(Graphics g)
     { if(ImageLoaded)
       {g.drawImage(picture,0,0,null);
        showStatus("Done");
        }
       else showStatus("Loading image");
```

```
   }
public boolean imageUpdate(Image img,int
infoflags,int x,int y,int w,int h)
{if(infoflags==ALLBITS)
  {ImageLoaded=true;
    repaint();
    return false;
  }
  else return true;
}
}
```

Результаты работы апплета(Рис.8.8):

Рис.8.8

Класс MemoryImageSource

Используется для создания нового изображения из массива пикселей. Пример:

```
import java.applet.*;
import java.awt.*;
import java.awt.image.*;
public class MemorImage extends Applet{
   Image art;
   Dimension d;
   public void init(){
   generatImage();
 }
 public void generatImage(){
 int pixels[]=new int[d.width*d.height];
 int i=0;
 int r,g,b;
            for(int y=0;y<h;y++){
                        for(int x=0;x<h;x++){
                          r=(x^y)&0xff;//red is x
XOR y
```

```
            g=(x*2^y*2)&0xff;//green is 2x XOR 2y
            b=(x*4^y*4)&0xff;//blue is 4x XOR 4y
            pixels[i++]=(2555<<24)|(r<<16)|(g<<8)|b;
 }
}
          art=createImage(new
MemoryImageSource(d.width,d.height,pixels,0,d.widt
h));
         }
         public void paint(Graphics g){
                 g.drawImage(art,0,0,this);
         }
}
```

Результаты работы апплета(Рис.8.9):

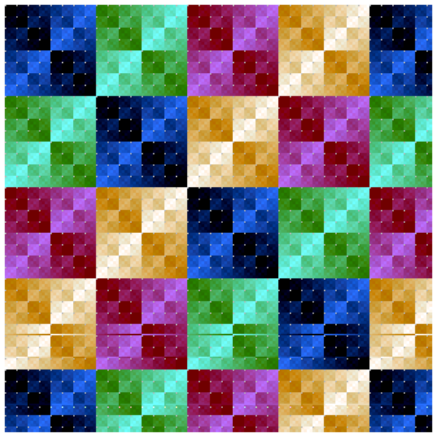


Рис.8.9

Преобразование приложения в аплет

Графическое приложение (т.е. приложение, использующее библиотеку AWT) легко преобразовать в аплет и встроить в Web-страницу, и, что особенно важно, при этом весь код пользовательского интерфейса можно оставить без изменения. Для этого нужно сделать следующее:

1. Создать HTML-страницу с соответствующим дескриптором APPLET, который указывает на аплет в виде байт-кода (файл с расширением .class), для загрузки кода аплета.

Синтаксис:

```
<applet code="Имя_файла.class"
width="целое_число" height="целое_число">
</applet>
```

2. Описать подкласс класса JApplet (в первой строке описания класса необходимо добавить фразу extends JApplet). Сделать этот класс открытым (public), иначе аплет будет невозможно загрузить.

3. Удалить из приложения метод main. Окно фрейма для приложения создавать не следует. Приложение будет отображаться внутри броузера.

4. Перенести все операторы инициализации из конструктора окна фрейма в метод аплета init. Создавать объект аплета явно не обязательно– броузер сам создаст его и вызовет метод init.

5. Удалить вызов метода setSize. В аплетах размеры задаются с помощью параметров WIDTH и HEIGHT в HTML-файле.

6. Удалить вызов метода setDefaultCloseOperation или addWindowListener. Аплет не сможет закрываться и прекратит работу вместе с броузером, поэтому не нужен никакой объект слушателя окна.

7. Если в приложении есть вызов метода setTitle, следует его удалить. Аплеты не имеют заголовков. (Разумеется, можно задать заголовок самой Web-страницы, используя дескриптор title языка HTML).

8. Не вызывать метод show. Аплет отображается на экране автоматически.

Жизненный цикл аплета

Основу для создания любого серьезного аплета составляют четыре метода из класса Applet: init, start, stop и destroy. Ниже приводится краткое описание, в котором указано, когда их следует вызывать и какой код они должны содержать.

– init

Этот метод используется для инициализации аплета. Он очень похож на конструктор, поскольку автоматически вызывается при первом запуске аплета. Обычно все действия этого метода внутри аплета сводятся к обработке значений param и добавлению компонентов пользовательского интерфейса. Несмотря на то что аплеты могут иметь конструктор по умолчанию, инициализацию следует выполнять с помощью метода init.

– start

Этот метод автоматически вызывается после метода init. Он также вызывается, если пользователь вернулся на страницу, содержащую аплет, после просмотра других страниц. Это значит, что, в отличие от метода init, start можно вызывать повторно. По этой причине код, который должен выполняться только один раз, следует помещать в метод init, а не start. Этот метод следует вызывать при повторном запуске потока для аплета, например, для возобновления анимации. Если аплет не выполняет никаких действий, которые нужно приостановить при переходе пользователя на другую страницу, метод start реализовывать не нужно (как и метод stop).

– stop

Этот метод автоматически вызывается, когда пользователь покидает страницу, на которой расположен аплет. Таким образом, в одном и том же аплете он может вызываться несколько раз. Метод stop предназначен для остановки процессов, на выполнение которых затрачивается слишком много времени. Это предотвращает замедление работы системы в те моменты, когда пользователь не работает с аплетом. Этот метод не следует вызывать явно. Если аплет не выполняет анимацию, не воспроизводит аудиофайлы и не осуществляет вычислений в потоке, метод stop обычно не нужен.

– destroy

Этот метод вызывается только при нормальном завершении работы броузера. Поскольку аплеты предназначены для работы на HTML-страницах, заботится об освобождении ресурсов после закрытия страницы, содержащей аплет, совершенно не обязательно. Если все же вы решили сделать это самостоятельно, то для освобождения ресурсов вам придется заместить метод destroy.

9 Публикация сайта

Web-хостинг

После того как вы закончите работу над Web-страницами, плод вашего напряженного труда следует разместить в Internet. При этом необходимо сделать все возможное, для того чтобы ваш Web-узел был найден пользователями Internet, причем, в ближайшее время. Для реализации этого плана, если у вас нет возможности приобрести собственный Web-сервер, что достаточно дорого, следует обратиться в организации, которые предоставляют услуги Web-хостинга хостеры).

Они предоставят место для вашей страницы сайта на своем сервере – машине, на которой установлены специальные программы, и которая постоянно подключена к сети. Собственно, Internet и состоит из множества таких машин объединенных между собой.

Одних только специальных программ и круглосуточного подключения компьютера к сети недостаточно. Дело в том, что Internet подобен большому городу, где у каждого сервера есть свой уникальный цифровой адрес (например, 197.84.789.10), и для того, чтобы вам выделили такой адрес, вам придется платить специальной организации, которая занимается их распределением.

Итак, вам для нашего сайта нужен – **хостинг**. Его можно получить и бесплатно. Однако, вы должны понимать, что пользуясь бесплатным хостингои ваши возможности будут сильно ограничены: вы не сможете пользоваться скриптами (установить свою гостевую книгу, голосование, форум, чат и прочие скрипты), вас могут обязать вешать на вашей

странице рекламу (баннеры), вы ограничены местом (как правило не больше 3-10 мегабайт вам отводится под вашу страницу), и бесплатные хостеры не несут перед вами обязательств, и вы можете лишиться его в любой момент.

Однако, если ваша страничка – домашняя, и вы пока что не создаете серьезный проект, то бесплатный хостинг это то, что вам нужно на первых порах. Бесплатный хостинг предоставляют Narod.ru, By.ru, Boom.ru и многие другие, вы легко найдете их с помощью любой поисковой системы, достаточно ввести в строке поиска “бесплатный хостинг”.

Как зарегистрировать место под страницу, что такое домен, как завести уникальное имя для сайта.

Допустим, вы решили зарегистрироваться на Narod.ru. Ищем на Narod.ru ссылку на раздел Регистрация (Создать страницу, Зарегистрировать сайт - это может называться по-разному), читаем внимательно правила и соглашение с пользователем, заполняем внимательно предложенную форму (анкету).

После регистрации адрес вашего сайта будет, допустим, vasya.narod.ru – это называется домен третьего уровня.

Что такое **домен**: когда-то разработчики решили, что цифровой адрес не удобен для пользователей интернет, что гораздо удобнее набрать vasya.ru, а не 197.84.789.10, но это было удобнее для людей, а не для машин. И тогда решили следующее: пусть будет так – человек набирает буквенный адрес, а машина сверяет его с некой базой, и смотрит какой цифровой адрес соответствует буквенному, и отправляет человека туда. И назвали буквенный адрес доменом. Каждой стране был присвоен адрес из 2-3 букв – например, .ru – был присвоен России, и назвали такой адрес – доменом первого уровня, и дали его каждой стране.

Соответственно, теперь каждая организация могла купить в зоне какой-либо страны себе домен – например, firma.ru – это домен второго уровня. На своем домене второго уровня каждая фирма может создать сколько угодно доменов третьего уровня, что и делают бесплатные хостеры.

Что делать, если вы хотите уникальное имя для своего сайта, и не хотите сидеть на бесплатном хостинге? Сначала зарегистрировать свой домен. Регистрацией доменов в России занимается РосНИИРОС, вы можете найти информацию о них и зарезервировать домен для регистрации на сайте http://nic.ru/.

Иногда, платные хостеры предлагают вам следующую услугу – регистрацию домена за вас. Но дело в том, что регистрируют домен не на вас, а на себя. Это делается, чтобы вы потом не могли уйти от них к другому хостеру, если вам не понравится их обслуживание, ведь в этом

случае вы потеряете уникальное имя для своего сайта, т.к. оно зарегистрировано не на вас, и жаловаться вы можете в таком случае только на свою глупость. Поэтому, лучше регистрировать домен самостоятельно, а не через кого-нибудь. В год домен обойдется вам около 20 у.е., что не так дорого.

О платных хотерах

Какие преимущества вам дают платные хостеры: все по-разному, у них существуют различные тарифы и услуги, о которых вы можете узнать на сайте каждого хостера и подобрать себе подходящего. Одно из самых главных преимуществ для фирмы или проекта – это свое имя, согласитесь, проект, который имеет свое уникальное имя, который завтра никуда не исчезнет, вызывает больше доверия и уважения, т.е. firma.narod.ru явно проигрывает просто firma.ru.

Чтобы разместить страницу на сайте платного хостера не достаточно одного лишь заполнения формы, обычно нужно лично ваше присутствие для заключения контракта, более того, раз хостер платный, то его услуги придется сначала оплатить. Совет: прежде, чем выбрать платного хостера, походите по сайтам разных хостеров и почитайте внимательно, что они предлагают, сравните цены. Лучше всего выбирать хостера, который имеет офис в вашем родном городе, потому что при возникновении проблем с ними будет легче контактироваться, и лично приехать для выяснения обстоятельств, если возникнет такая надобность.

Список хостеров платных и бесплатных можно посмотреть тут: http://yaca.yandex.ru/yca/ungrp/cat/Computers/Internet/Hosting/.

Как закачать ваши файлы (картинки, html-документы и др.) на ваш сайт. Программы для закачки.

Теперь, когда вы зарегистрировались, то вам надо закачать файлы со своей страницей на сайт хостера. Хостеры часто позволяют вам сделать это через их веб-страницу. Для этого надо авторизоваться (войти) в центр управления своей страницей на сайте хостера, введя свои логин и пароль в форме на первой странице. Дальше вам все будет ясно, как правило, центр управления вашей страницей имеет интуитивно понятный пользователю интерфейс, и вы без труда разберетесь куда нажимать и что делать, чтобы закачать на свой сайт ваши документы и картинки. Если у вас возникнут какие-то вопросы, то на сайте, предоставившем вам место под страничку, как правило, есть раздел Помощь, советую вам ознакомиться с ним, прежде, чем вы начнете предпринимать какие-то действия по размещению своей странички, это поможет вам избежать многих ошибок и лишних действий.

Вы можете закачивать файлы на свою страницу не только при помощи броузера, авторизовавшись на сайте, предоставившим вам хостинг. Вы можете использовать специальную программу, это гораздо удобнее. Но для этого, вам надо узнать, дает ли ваш хостер пользователю, т.е. вам, возможность доступа к своей страничке через ***FTP*** (это такой протокол, основное назначение которого пересылать файлы, соответственно специальная программа, при помощи которой вы будете закачивать файлы на свой сайт, используя FTP, называется ***FTP-клиент***). Узнать это можно в разделе Помощь на сайте вашего хостера. Я лично предпочитаю программу CuteFTP, поэтому опишу, что и куда заполнять, ориентируясь на этот Ftp-клиент, в других ftp-клиентах действия будут аналогичными. Итак, откроем CuteFTP (допустим, вы его уже скачали).

В меню выбираем пунт FTP -> File Manager (или нажимаем кнопку F4, чтобы вызвать Менеджер файлов)

- В появившемся окошке нажимаем на кнопку Add site
- Во вновь появившемся окошке заполняем следующее:
  - Site Label - любое название (например, Мой сайт, каждый раз потом, вызывая File Manager, вы будете видеть список сайтов, название для каждого будет такое, какое вы зададите в site label)
  - Host address - например, ftp.narod.ru (это вы должны посмотреть в разделе Помощь, сайта предоставившего вам хостинг)
  - User Id - имя пользователя (логин)
  - Password - ваш пароль
  - Login Type - Normal
  - Transfer Type - Auto-detect
- Нажимаете на кнопку Ок
- Теперь, чтобы подсоединиться к вашему сайту (и заодно проверить правильность набранных параметров: Пароля Логина, и Имени сайта), нажмите на Мой сайт в File Manager.

Если вы не можете соединиться со своим сайтом, значит вы неправильно набрали пароль, логин или host address. Редактировать (изменять) информацию о соединении с вашим сайтом, вы можете, нажав кнопку Edit site в Site Manager, удалить - кнопка Delete site.

Также в этой статье я раскажу вам как закачивать файлы на свой сайт при помощи программы FAR, если вы умеете в ней работать, то может быть вам будет легче пользоваться ей: -

Откройте Far- В командной строке наберите

ftp://ваш_логин:ваш_пароль@имя_сайта (например: ftp://vasiliy:moyparol@vasya.ru)

- нажмите кнопку Enter

Far, соединится с вашим сайтом, список файлов которого отобразится на одной из панелек файл-менеджера Far, переключившись на другую

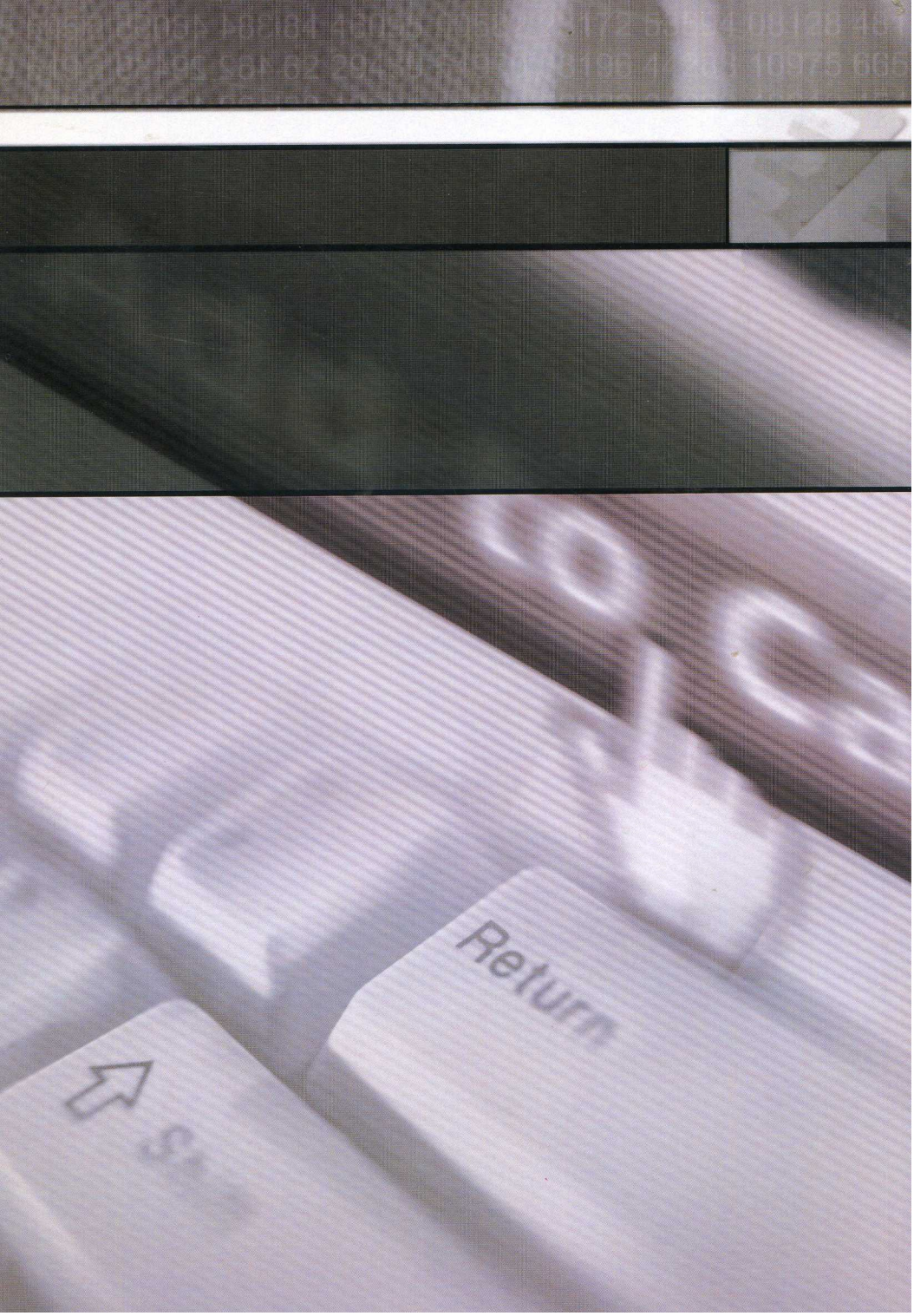
панельку кнопкой TAB, вы открываете там папку с файлами своей странички и перекачиваете их на свой сайт при помощи Copy (F5).

Напутствие.

На сайте вашего хостера обязательно должен существовать раздел “Помощь”, где вы найдете все необходимые данные для того, чтобы зайти на свой сайт, используя FTP, а также, возможно, ответы на некоторые свои вопросы. Если вы пользуетесь платным хостером – никогда не стесняйтесь обратиться в службу поддержки, они обязаны быстро отреагировать на ваш призыв о помощи и помочь вам. Это еще одно ваше преимущество перед пользователями бесплатных хостингов, служба поддержки которых, как правило, перегружена, и может ответить стандартными, ничего не объясняющими, письмами на вопрос пользователя.

Список использованной литературы

1. Храмцов П.Б., Брик С.А., Русак А.М., Сурин А.И.
   Основы web-технологий
   Интернет-университет информационных технологий - ИНТУИТ.ру, 2003
2. Матросов А.В. и др.
   HTML 4.0. В подлиннике
   BHV-Санкт-Петербург, 2002
3. Будилов В.А.
   Практические занятия по HTML. Краткий курс
   Наука и техника, 2001
4. Муссиано Ч., Кеннеди Б.
   HTML и XHTML. Подробное руководство
   2002
5. Штайнер Г.
   HTML/XML/CSS. Справочник
   Лаборатория Базовых Знаний, 2001
6. Айзекс С.
   Dynamic HTML
   BHV-Санкт-Петербург, 2001
7. Коржинский С.Н.
   Настольная книга Web-мастера: эффективное применение HTML, CSS и JavaScript
   КноРус, 2000
8. Уолтер Савитч. Язык Java. Курс программирования, 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2002.
9. Хорстманн К.С., Корнелл Г. Библиотека профессионала. Java 2. Том 1. Основы.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003.
10. Арнолд К., Гослинг Д., Холмс Д. Язык программирования Java. 3-е издание. - М.: Издательский дом «Вильямс», 2001.
11. Справочник по элементам разметки HTML 3.2
    http://www.citforum.ru/internet/html/elements.shtml
12. Краткая история HTML
    http://osp.admin.tomsk.ru/pcworld/1997/04/112.htm
13. Сайт World Wide Web Consortium (стандарты языка HTML)
    http://www.w3.org/MarkUp
14. Erik Wilde: Wilde's WWW, technical foundations of the World Wide Web. Springer 1998,ISBN:3-540-64285-4 [CSS1].
    http://wildesweb.com/
    Книга, охватывающая все информационные технологии World Wide Web.


Return