МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени Н.Э. БАУМАНА

Учебное пособие

Верстов В.А., Власов А.И.

Курс лекций

«Конструкторско-технологическая информатика»

МГТУ имени Н.Э. Баумана

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени Н.Э. БАУМАНА

Курс лекций

«Конструкторско-технологическая информатика»

Москва
МГТУ имени Н.Э. Баумана

2012

УДК 681.3.06(075.8)
ББК 32.973-018
И201

Верстов В.А., Власов А.И. «Конструкторско-технологическая информатикая» : учебное пособие – М.: МГТУ им. Н.Э. Баумана, 2012. – 135 с.: ил.

Рассмотрены вопросы проектирования информационных интернет систем. На конкретных практических примерах проиллюстрированы особенности разработки современных информационных интернет систем с использованием HTML, Java-Script, VbScript, Flash – технологий, технологий обработки растровой графики. Приведена справочная информация по базовым командам и синтаксические диаграммы, поясняющие особенности их использования. Основное внимание уделено вопросам проектирования информационных систем: компоновке, композиции, навигационным моделям, цветовым решениям и техническим вопросам реализации. Представленный в пособии материал является основой для разработки современных информационных интернет систем радиотехнических предприятий на основе архитектуры интернет/интранет.

Для студентов радиотехнических специальностей вузов, пособие может быть полезно разработчикам информационных систем.

Ил. 39. Табл. 5. Библиогр. 7 назв.

УДК 681.3.06(075.8)

© МГТУ им. Н.Э. Баумана, 2012

Содержание

1. ИСТОРИЯ РАЗВИТИЯ СЕТИ ИНТЕРНЕТ

На сегодня понятие Интернет уже прочно вошло в нашу жизнь. От обилия различных сервисов и возможностей, предоставляемых "всемирной паутиной" просто захватывает дух, а еще буквально несколько десятилетий назад трудно даже было не то, что представить о таких возможностях, но и прочитать о них в фантастической литературе. История развития и становления интернет технологий, тесно связанная с успехами в микроэлектронике и технологиях обработки данных позволяет проследить все основные этапы становления мировой электронной информационной инфраструктуры.

Появление многих полезных технических решений в гражданской сфере обусловлено успехами военных исследований и разработок, это относится и к интернет. В далеких 60-х годах, сразу же вслед за запуском первых отечественных спутников, военные ведомства, как у нас, так и в США начали воплощать в жизнь программы создания распределенных (разнесенных в пространстве), гетерогенных (разноплатформенных, как на программном, так и на аппаратном уровне), резервированных (дублированных с целью повышения надежности) сетей передачи данных. Используемые в то время телефонные, ВЧ и радиоканалы каналы передачи данных уже не могли удовлетворять потребностям эффективного и надежного управления войсками и встала необходимость в построении сетей передачи данных способных сохранять свою работоспособность при их частичном повреждении. В СССР основное внимание было нацелено на развитие спутниковых систем передачи данных, в США же первоначально, основные усилия были направлены на построение наземных сетей передачи данных. Проследим эволюцию технологий передачи данных по отдельным датам:

В **1958** в США Дуайтом Эзенхауэром было образовано агентство ARPA (Advanced Research Projects Agency of the U.S.Department of Defense) специализировавшееся в военно-космических исследованиях. В дальнейшем многие из программ данного агентства перешли в NASA.

В **1962** году ARPA открывает проект построение распределенной сети передачи данных. В это же время в ряде университетов США проводились исследовательские работы в области систем передачи данных, причем технологии, лежащие в основе сетей передачи данных, являются междисциплинарными и их развитие обусловлено успехами в различных смежных областях.

Впервые, судя по зарубежным источникам, идею построения глобальной сети, обеспечивающей доступ к данным и программам любому пользователю высказал сотрудник Массачусетсского технологического института (MIT) Дж.С.Ликлидер. Другой ученый из MIT, Леонард Клейнрок, в своей докторской диссертации развил и обобщил вопросы теории телекоммуникационных сетей. Айвен Сазерленд (MIT) создал первый прототип интерактивной графической программы Sketchpad (Блокнот).

В конце **1962** года Дж. Линкер становится руководителем отдела ARPA IPTO (ARPA Information Processing Techniques Office) и в 1963 году он приглашает Айвена Сазерленда, а в 1965 и другого ученого Боба Тейлора, принять участие в работе над проектом в ARPA, которые проводились при поддержке MIT и Калифорнийского университета (UCLA).

В **1963** году вышел в свет первый **универсальный стандарт кодирования**, который ставит в соответствие конкретным буквам, цифрам и другим символам конкретные цифровые коды. Этот стандарт получил название ASCII (American Standard Code for Information Interchange) (а на экзаменах некоторые студенты трактовали его как: "а-эс-це-два" ☺). Введение и этого стандарта явилось одним из важных этапов в развитии сетей передачи данных, так как позволило обеспечивать обмен информации между вычислительными устройствами различных производителей, т.е. был решен вопрос однообразия представления информации в гетерогенной среде.

Системы передачи и обработки информации представляют собой совокупность математического, алгоритмического, аппаратного, программного, информационного, лингвистического, организационного и других видов вспомогательных обеспечений. Работы по

разработке отдельных компонентов и видов обеспечений информационных систем велись многими компаниями и лишь те из них, кто сумел объединить все это в единую стройную систему добились успеха. Но обо всем по порядку.

В **1964** Ликлидер возвращается в MIT, чтобы в тесном сотрудничестве с Сазерлендом продолжить работы по созданию операционной системы, работающей в режиме разделения времени (т.е. создания **многозадачной операционной системы**).

С **1964** года начинается лавинообразный рост парка вычислительной техники. IBM выпускает свою новую машину IBM360, при этом устанавливается всемирных **стандарт байта - восьмибитного слова** (почему в байте - восемь бит я не понимаю до сих пор, будем считать это аксиомой), что автоматически вывело из игры машины с 12 и 36 битными словами. Развивались терминальные вычислительные системы (один мощный вычислительный центр - тысячи терминалов, подсоединяемым в основном по телефонным линиям). Компьютеры все больше подвержены тенденциям микроминиатюризации и внедряются в различные сферы деятельности.

В **1965** году DEC представляет свою вычислительную машину PDP-8, размеры которой позволяли установить ее на рабочем столе, не заставили себя долго ждать и пионерские по тем временам разработки фирм Xerox и Apple.

Параллельно с развитием компьютерной техники, практически одновременно в MIT, RAND Corporation и GBNPL (Great Britain National Physical Laboratory) проводились работы по совершенствованию систем надежной передачи информации. Была предложена идея **коммутации пакетов**, суть которой заключалась в том, что любая информация, передаваемая по сети, разбивается на несколько составных частей (пакетов), которые затем передаются независимо друг от друга от исходной точки до точки назначения различными путями (маршрутами). Впервые основные элементы данной теории были изложены в статье Пауля Бэрана "Передача данных в сетях", в дальнейшем они получили свое развитие в работах Дональда Девиса, Леонарда Клейнрока и других исследователей.

В **1965** году в рамках проекта при финансовой поддержке ARPA Ларри Робертс и Томас Марилл создают первую региональную вычислительную сеть WAN (Wide-Area Network). Данная сеть объединила вычислительную машину TX-2 (MIT) с Q-32 в Сант-Монике через выделенную телефонную линию. На созданной сети была экспериментально подтверждена гипотеза Клейнрока о том, что коммутация пакетов - один из самых эффективных способой связи между вычислительными узлами.

В **1966** году Айвен Сазерленд приглашает Боба Тейлора, который к тому времени работал в NASA принять участие в проекте по организации сети. ARPA в свою очередь в этом году открывает финансирование проекта RAND Corporation - JOSS (Johnniac Open Shop System), в рамках которого была создана система предоставлявшая пользователь доступ к вычислительным ресурсам с удаленных терминалов, которыми служили модифицированные электрические пишущие машинки.

В **1966** Тейлор заменяет на посту директора ARPA IPTO Сазерленда и приступает к реализации проекта **одновременной коммутации** распределенных вычислительных устройств. Сутью проекта стало объединение всех клиентов ARPA IPTO в одну сеть (надо отметить, что до этого доступ к вычислительным ресурсам осуществлялся по коммутируемым каналам связи типа "точка-точка"). Вскоре к реализации проекта подключается Ларри Робертс, перешедший в ARPA из MIT.

В **1967** году Джоном Ван Гином из Станфордского научно-исследовательского института (SRI) был значительно усовершенствован модем (МОДулятор-ДЕМодулятор), изобретенный в начале 60-х годов. Были предложены новые решения по повышению надежности распознавания битов информации на фоне шумовых помех телефонных линий.

В **1967** году Лари Робертс собрал научную конференцию в Анн-Арбор в штате Мичиган, на которую он пригласил основных разработчиков сетевого проекта. Многие из участников конференции узнали о работах друг друга и стали объединяться. На этой же конференции в своем докладе Ларри Роберт впервые употребил термин "ARPANET", а Уэсли Кларк высказал идею "IMP" (Interface Message Processors - устройство для управления

трафиком в сети), которая впоследствии нашла свое воплощение в современных маршрутизаторах.

В **1968** году начались работы по созданию IMP. ARPA заключила контракт на создание четырех IMP устройств для построения сети ARPANET с небольшой консалтинговой фирмой BBN (Bolt Beranek &Newman). В 1969 году контракт был выполнен и ARPANET, охватившая западное побережье США, стала реальностью.

Создав сетевую инфраструктуру, общесистемные программные средства пришло время разработчикам уделить внимание лингвистическому обеспечению (средства и языки разработки) информационных систем. И в первую очередь в этой связи следует отметить два языка: язык Си - как средство разработки общесистемного и прикладного программного обеспечения, и я зык HTML - приложение стандарта ISO 8879. Простой язык гипертекстовой разметки, используемый для создания платформенно - независимых документов.

Предшественник HTML зародился в 60-е годы, как концепция универсального языка разметки для разных типов компьютеров. В IBM группой разработчиков (Charles Goldfarb, Edward Mosher, and Raymond Lorie) был создан язык Generalized Markup Language (GML). Он содержал прототипы основных компонент иерархической структуры документа и определения типов документов.

В **1978** году Американский Национальный Институт Стандартизации (AM51) начал исследования в области разметки документов и представления их в международном стандарте.

В **1984** году Международная организация по стандартизации (ISO) активно включилась в разработку интернациональной версии стандарта обобщенного мета языка, позволяющего строить системы логической, структурной разметки любых разновидностей текстов.

В **1986** году был принят стандарт "ISO 8879: 1986 "Information processing - Text and office systems - Standard Generalized Markup Language (SGML)." С точки зрения стандарта, размеченный таким образом текст не несет информации о внешнем виде документа, а лишь обретает логическую структуру, определяющую подчинение и сочленение его составных частей. Благодаря такой концепции, текст, определенный данным образом мог интерпретироваться любой программой с любым средством ввода/вывода. То есть определения вида представления текста целиком возлагалось на клиента. Сам SGML не представлял собой готовую систему разметки текста, а определял лишь синтаксис записи элементов разметки - тегов - и их атрибутов, а также правила определения новых тегов и указания структурных отношений между ними.

Поэтому сам по себе SGML не приобрёл широкого распространения, пока Европейском институте физики (CERN) не озадачились созданием системы передачи гипертекстовой информации через Internet. Тогда SGML и был взят за основу для создания языка гипертекстовой разметки (HTML). Изначально HTML, как и положено SGML-приложению, разделял все особенности идеологии SGML то есть не содержал тэгов описывающих физическое представление документа, а лишь описывал его логическую структуру.

Предшественник HTML зародился в 60-е годы, как концепция универсального языка разметки для разных типов компьютеров. В IBM группой разработчиков (Charles Goldfarb, Edward Mosher, and Raymond Lorie) был создан язык Generalized Markup Language (GML). Он содержал прототипы основных компонент иерархической структуры документа и определения типов документов.

В **1978** году Американский Национальный Институт Стандартизации (AM51) начал исследования в области разметки документов и представления их в международном стандарте.

В **1984** году Международная организация по стандартизации (ISO) активно включилась в разработку интернациональной версии стандарта обобщенного мета языка, позволяющего строить системы логической, структурной разметки любых разновидностей текстов.

Так в **1986** году был принят стандарт "ISO 8879: 1986 "Information processing - Text and office systems - Standard Generalized Markup Language (SGML)." С точки зрения HTML, размеченный таким образом текст не несет информации о внешнем виде документа, а лишь обретает логическую структуру, определяющую подчинение и сочленение его составных частей. Благодаря такой концепции, текст, определенный данным образом мог

интерпретироваться любой программой с любым средством ввода/вывода. То есть определения вида представления текста целиком возлагалось на клиента. Сам SGML не представлял собой готовую систему разметки текста, а определял лишь синтаксис записи элементов разметки - тегов - и их атрибутов, а также правила определения новых тегов и указания структурных отношений между ними. Поэтому сам по себе SGML не приобрёл широкого распространения, пока Европейском институте физики (CERN) не озадачились созданием системы передачи гипертекстовой информации через Internet. Тогда SGML и был взят за основу для создания языка гипертекстовой разметки (HTML). Изначально HTML, как и положено SGML-приложению, разделял все особенности идеологии SGML то есть не содержал тэгов описывающих физическое представление документа, а лишь описывал его логическую структуру.

В **1989** году в CERN Tim Berners-Lee предложил концепцию World Wide Web (WWW), основой которой стал распределенный всемирно гипертекст. Вскоре в Национальном центре суперкомпьютерных приложений США (NCSA) был реализован первый графический броузер Mosaic.

В **1994** году началась подготовка спецификации HTML 2.0. В этом же году Консорциум W3C унаследует от CERN авторитет и дальнейшую поддержку и разработку стандартов HTML. Окончательный вариант HTML 2.0 принят только в 1995 году, когда уже велись разработки HTML 3.0, в котором появился существенный сдвиг в сторону введения в язык дескрипторов определяющих конкретную визуализацию текста на стороне пользователей. Что бы избежать противоречий с изначальной концепцией SGML в HTML вводится поддержка иерархических стилевых спецификаций (CSS).

Так, породив двойственность разметки с помощью нововведенных тэгов и стилей в 1996 вышел в свет стандарт HTML 3.2, а проект HTML 3.0 был заморожен. HTML 3.2 был принят в январе **1997** года. Нет сомнения, что CSS - почти идеальное средство избавить HTML от наследственных недостатков и перевести его развитие на принципиально новые рельсы. Тем досаднее то, как сложилась судьба этого замечательного изобретения. Поскольку спецификацию CSS увязали с другими нововведениями HTML 3, W3C долго не утверждал ее в качестве официального стандарта; задерживалось и доведение CSS до более или менее завершенного вида, при котором стала бы возможной ее реализация в коммерческих продуктах.

В апреле **1998** года выходят рекомендации W3C относительно HTML 4.0.

Гипертекстовый подход в общем случае, помимо связывания распределенных данных, осуществляют еще одну очень важную функцию. Он позволяет рассматривать информацию с нужной степенью детализации, что существенно упрощает анализ больших объемов данных. Можно быстро отобрать самое интересное, а затем изучить выбранный материал во всех подробностях.

В то же время, такому подходу присущи и определенные недостатки. Во-первых, Web-страницы статичны. При использовании протокола HTTP, на клиентскую систему передаются только пассивные данные, но не методы объектов. Из общих соображений очевидна ограниченность подобного подхода.

Во-вторых, он обладает весьма ограниченными интерактивными возможностями, которые сводятся к заполнению пользователем чисто текстовых форм с последующей отправкой на сервер. Такая форма общения вполне устраивала пользователей терминалов ЕС ЭВМ лет 15 назад. Сейчас ее явно недостаточно, особенно, чтобы претендовать на роль интерактивной системы, которая требует реального активного взаимодействия с пользователем.

Развитием языка HTML явился язык **DHTML** (динамический язык гипертекстовой разметки), который позволяет создавать динамические интерактивные страницы. Этот язык является броузерооориентированным (т.е. зависит от типа используемого броузера на стороне клиента), что накладывается определенные ограничения на его применение.

На сегодняшний день известны и широко применяются несколько основных технологии создания интерактивного взаимодействия с пользователем в Web. Первый путь заключается в включении JavaScript или VBScript - сценариев в тело Web-страниц, второй в создании активных серверных страниц, третий в использовании специальных технологий для доступа к

данным: PHP, XML, Стандартного Интерфейса Шлюза (Commom Gateway Interface) – CGI и наконец самый мощный инструмент, предоставляющий практически неограниченные возможности способ - применение технологии Java (ипользование Java-апплетов и сервлетов).

JavaScript представляет собой в чистом виде **интерпретируемый язык** (своего рода язык сценариев) который может быть интерпретирован стандартным Web - броузером. Разработан язык компанией Netscape Communication Corporation под первоначальным наименованием LiveScript. Главной целью языка JavaScript является обеспечение активного взаимодействия HTML-документов с пользователем. Этот язык не претендует на то, чтобы быть полномасштабным языком программирования, таким как Java и C++. Скорее, он является расширением языка HTML, облегчающим работу пользователя с конкретным броузером. С введением обработчиков событий разработчик получает возможность определять поведение HTML-документов в зависимости от действий пользователя. Важен тот факт, что JavaScript-программы действительно являются выполняемым содержимым документов: они физически находятся внутри HTML-документов, в отличии от Java-аплетов, которые существуют вне документов, их активизирующих. Сценарии на JavaScript позволяют реализовывать различные визуально-интерактивные ээфекты: создание и модификация элементов ввода/вывода и кнопок управления на странице, задания режимов обработки аудио эффектов, анимации и контроля за действиями пользователями и действиями самой страницы. Основной недостаток программ написанных на JavaScript – это их броузеро-ориентированность (т.е. зависимость от конкретного броузера). Для создания программ работающих на различных клиентских платформах часто приходится встраивать в тело документа специальные функции определения типа броузера и по результату их работы интерпретировать те, или иные элементы кода. Все это приводит к чрезмерному увеличению как кода самого документа, так и времени его обработки.

Сам язык изобрел Brendan Eich (компания Netscape) и назвал его JavaScript. Впервые новый язык был использован в броузере Netscape Navigator 2.0. После этого он стал использоваться во всех последующих браузерах от Netscape и во всех браузерах от Microsoft, начиная с Internet Explorer 3.0. Компания Microsoft по-своему развила идею, и дала своей версии языка более короткое название: JScript.

Далее, чтобы обеспечить совместимость версий языка независимых разработчиков, Генеральной Ассамблеей ECMA был создан стандарт. Этот стандарт основан на нескольких базовых технологиях, наиболее известными из которых являются упомянутые уже JavaScript (Netscape) и JScript (Microsoft).

Развитие этого Стандарта началось в ноябре **1996**. Первое издание Стандарта ECMA было принято Генеральнаой Ассамблеей ECMA в июне **1997**.

Данный ECMA Стандарт был представлен международной комиссии по стандартам ISO/IEC JTC 1 для принятия, и одобрен как международный эталон ISO/IEC 16262 в апреле 1998. Генеральная Ассамблея ECMA в июне 1998 одобрила второе издание ECMA-262, с сохранением всех требований ISO/IEC 16262.

Работа над языком еще не закончена. Технический комитет работает над существенными расширениями, включая механизмы для сценариев, которые будут созданы для использования в Internet, и более жесткой координацей с другими основными стандартами групп World Wide Web Консорциум и Wireless Application Protocol Форум.

VBScript – язык клиентских сценариев, реализованный компанией Микрософт в своих программных продуктах. Он представляет клон Microsoft Visual Basic и по своим функциональным возможностям близок к JavaScript. Его использование в интернете в качестве языка разработки сценариев для html страниц является нецелесообразным из-за ограничений по интерпретации отличными от Microsoft Internet Explorer (MIS) броузерами.

Технологии ADO (ActiveX Data Objects) предоставляют набор методов и объектов для доступа к базам данных. Именно используя объекты ADO серверные сценарии ASP обращаются к базе данных. Объекты ADO также могут вызываться не только из сценариев, но и из других объектов ActiveX, стандартных приложений Windows и т.п. Элементы управления ActiveX представляют собой динамические библиотеки работающие как на стороне клиента, так и на стороне сервера. Для создания элементов ActiveX в последнее время наряду с

библиотекой стандартных шаблонов (STL) широко используются библиотеки шаблонов ActiveX Template Library (ATL).

ActivX элементы можно использовать как на стороне сервера, так и на стороне клиента. Основным недостатком применение этих технологий – является проблемы безопасности. По определению ActivX элементы могут предоставить доступ к ресурсам локального компьютера. Для решения данной проблемы предлагается ряд различным способов, одним из которых является технология цифровых сертификатов, однако и она кроме того что стоит конкретных средств, не гарантирует от всех возможных проблем с безопасностью.

Активные серверные страницы (Active Server Pages) является базовой для создания приложений WEB на платформе Microsoft Internet Information Server. Данные приложения основаны на наборе текстовых файлов с расширением *.asp, в которых интегрированы сценарии Jscript (версия Java Script от MS) или VB Script осуществляющие обращение к базе данных. Эта технология основывается на интенсивном использовании COM объектов и серверных сценариев для реализации интерактивного взаимодействия в WEB. Платформой данной технологии является Microsoft Internet Information Server. ASP страницы позволяют в ряде случаев избежать необходимости программирования на Perl или C++ ограничившись HTML, Jscript или VB Script.

Технология функционирования активных серверных страниц ASP заключается в следующем: при обращении пользователя к странице ASP, WEB сервер интерпретируем реализованной в ней сценарий в соответствии с которым происходит обработка параметров, переданных этой страницы, и происходит ее модификация. Результат такой модификации отправляется пользователю в виде html страницы. Для работы ASP страниц с базами данных посредством использования методов технологии ADO. Бизнес-логика реализуется посредством использования библиотечных и дополнительных COM объектов.

В начале февраля **1998** г международная организация W3C утвердила спецификацию "Extensible Markup Language(XML) 1.0". Полные спецификации **XML** и связанных с ним языков доступны на официальной странице W3C - http://www.w3.org/.

XML (Extensible Markup Language) - это язык разметки, описывающий целый класс объектов данных, называемых XML- документами. Этот язык используется в качестве средства для описания грамматики других языков и контроля за правильностью составления документов. Т.е. сам по себе XML не содержит никаких тэгов, предназначенных для разметки, он просто определяет порядок их создания. Таким образом, если, например, мы считаем, что для обозначения элемента TITLE в документе необходимо использовать тэг <flower>;, то XML позволяет свободно использовать определяемый нами тэг и мы можем включать в документ фрагменты, подобные следующему: <flower>TITLE</flower>

Набор тэгов может быть легко расширен. Если, предположим, мы хотим также указать, что описание цветка должно по смыслу идти внутри описания оранжереи, в которой он цветет, то просто задаем новые тэги и выбираем порядок их следования:

```
<conservatory>
        <flower>TITLE</flower>
</conservatory>
```

Для дальнейших расширений надо внести следующие изменения:

```
<conservatory>
   <flower>TITLE1</flower>
   <flower>TITLE2</flower>
   <flower>TITLE3</flower>
</conservatory>
```

Как видно, сам процесс создания XML документа очень прост и требует лишь базовых знаний HTML и понимания тех задач, которые мы хотим выполнить, используя XML в качестве языка разметки. Таким образом, у разработчиков появляется уникальная возможность

определять собственные команды, позволяющие им наиболее эффективно определять данные, содержащиеся в документе. Автор документа создает его структуру, строит необходимые связи между элементами, используя те команды, которые удовлетворяют его требованиям и добивается такого типа разметки, которое необходимо ему для выполнения операций просмотра, поиска, анализа документа.

Еще одним из очевидных достоинств XML является возможность использования его в качестве универсального языка запросов к хранилищам информации. Сегодня в глубинах W3C находится на рассмотрении рабочий вариант стандарта XML-QL(или XQL), который, возможно, в будущем составит серьезную конкуренцию SQL. Кроме того, XML-документы могут выступать в качестве уникального способа хранения данных, который включает в себя одновременно средства для разбора информации и представления ее на стороне клиента. В этой области одним из перспективных направлений является интеграция Java и XML - технологий, позволяющая использовать мощь обеих технологий при построении машинно-независимых приложений, использующих, кроме того, универсальный формат данных при обмене информации.

XML позволяет также осуществлять контроль за корректностью данных, хранящихся в документах, производить проверки иерархических соотношений внутри документа и устанавливать единый стандарт на структуру документов, содержимым которых могут быть самые различные данные. Это означает, что его можно использовать при построении сложных информационных систем, в которых очень важным является вопрос обмена информацией между различными приложениями, работающими в одной системе. Создавая структуру механизма обмена информации в самом начале работы над проектом, менеджер может избавить себя в будущем от многих проблем, связанных с несовместимостью используемых различными компонентами системы форматов данных.

Также одним из достоинств XML является то, что программы-обработчики XML-документов не сложны и уже сегодня появились и свободно распространяются всевозможные программные продукты, предназначенные для работы с XML-документами.Было заявлено о его поддержке в последующих версиях Netscape Communicator, СУБД Oracle, DB-2, в приложениях MS-Office . Все это дает основания предполагать, что, скорее всего, в ближайшем будущем XML станет основным языком обмена информации для информационных систем, заменив собой, тем самым, HTML. На основе XML уже сегодня созданы такие известные специализированные языки разметки, как SMIL, CDF, MathML, XSL, и список рабочих проектов новых языков, находящихся на рассмотрении W3C, постоянно пополняется.

PHP был создан где-то осенью **1994** г. Расмусом Лердорфом. Самые первые версии не распространялись, а использовались у него на домашней странице - контролировали, кто просматривает его резюме. Модуль PHP появился в первые в виде простой небольшой cgi оболочка, созданной на Perl. Для реализации SQL запросов использовалась отдельная CGI-программа - Интерпретатор Форм (Form Interpreter - FI), которая анализировала запросы SQL и облегчала создание форм и таблиц, основанных на этих запросах, интеграция этих двух программ и явилось ядром создания современного PHP. Первоначальное название PHP - Персональные Инструментальные средства для домашней страницы позднее трансформировалось в Персональный комплект создания домашней страницы.

Первая версия, которая использовалась и другими людьми, появилась где-то в начале 1995 г. и называлась Personal Home Page Tools. Он включал простейший синтаксический анализатор, способный анализировать только небольшое количество специальных макросов, и ряд утилит, общепринятых в то время для персональных страниц. Гостевая книга, счетчик и пр. Синтаксический анализатор был переписан в середине 1995 г. и назван PHP/FI Version 2. "FI" пришло из другого пакета, написанного Расмусом для интерпретации данных форм html. Он объединил скрипты Personal Home Page с интерпретатором Form Interpreter и добавил поддержку mSQL - так появился PHP/FI. PHP/FI рос с достойной восхищения скоростью, и многие люди приложили свои усилия к дополнению и совершенствованию его кода.

PHP (официальное название "PHP: Hypertext Preprocessor (гипертекстовый процессор)" представляет собой встроенный в HTML язык сценариев, простейших PHP скрипт показан в

таблице.

```
<html>
    <head>
        <title>Определение версии PHP</title>
    </head>
    <body>

     <?php
        echo "Версия PHP";
        phpinfo();
     ?>

    </body>
</html>
```

PHP/FI – полнофакторная версия из этих двух пакетов, объединенных в одну систему. Сегодня эту технологию можно рассматривать как простой язык программирования, внедренный внутрь HTML файлов. Первоначальный сокращение PHP, хотя и прижился, но не соответствует действительности, PHP/FI сегодня используется больше для создания целых web серверов, чем для малых домашних страниц. Модуль устраняет потребность в многочисленных малых cgi программах на Perl, позволяя поместить простые скрипт-программы непосредственно в HTML файлы. Это увеличивает общую производительность web страниц посредством уменьшения непроизводительных затрат на запуск Perl-а. Пакет также упрощает управление большими web серверами, помещая все компоненты web страницы в одиночном файле html. Встроенная поддержка различных баз данных упрощает создание интерактивных форм взаимодействия с ними.

Обратите внимание на отличие от CGI-скрипта, написанного на другом языке типа Perl или C: вместо написания программы со множеством команд для вывода HTML-кода Вы просто включаете в HTML специальный код, который производит некоторые действия (в данном случае выводит некоторый текст). Код PHP располагается между специальными начальным и конечным тегами, позволяющими входить в режим PHP и выходить из него.

В отличие от клиентского Javascript PHP-скрипты выполняются на сервере. В этом случае клиент просто получит результаты выполнения скрипта, совершенно не зная о том, каким образом эти результаты получены. Можно даже сконфигурировать web-сервер так, чтобы он обрабатывал все HTML-файлы с помощью PHP, тогда пользователи не будут знать, каким образом генерируются страницы.

PHP/FI

PHP берёт свой исток в старом продукте, имевшем название PHP/FI. Последний был создан Rasmus'ом Lerdorf'ом в 1995 и представлял из себя набор Perl-скриптов для ведения статистики посещений его резюме. Этот список скриптов был назван 'Personal Homepages Tools'. Очень скоро потребовалась большая функциональность и Расмус пишет новую, намного более обширную версию на C, работающую с базами данных и позволяющую пользователям разработать простейшие web-приложения. Расмус решил выложить исходный код PHP/FI на всеобщее обозрение, исправление ошибок и дополнение.

PHP/FI (Personal Home Page / Forms Interperter) включал в себя базовую функциональность сегодняшнего PHP. Он имел переменные в стиле Perl, автоматическую интерпретацию форм и возможность встраиваться в html-код. Собственно синтаксис языка имел много общего с Perl, хотя и был намного проще и ограниченнее.

В **1997** выходит PHP/FI 2.0, вторая версия C-имплементации обозначила группу пользователей: несколько тысяч людей по всему миру, с примерно 50,000 доменами, что составляло примерно 1% всего числа Internet-доменов. Несмотря на то, что уже несколько людей занималось разработкой, PHP/FI 2.0 оставался крупным проектов одного человека.

Официально PHP/FI 2.0 вышел только в Ноябре **1997**, после проведения большей части своей жизни в бета-релизах. Вскоре после выхода, его заместили альфа-релизы PHP 3.0.

PHP 3.0 был первой версией PHP, похожей на то, что мы имеем сегодня. В **1997** Andi Gutmans и Zeev Suraski переписали код с начала: разработчики сочли PHP/FI 2.0 не пригодным для разработки коммерческих приложений, над которыми они работали для проекта Университета. Для совместной работы над PHP 3.0 с помощью базы разработчиков PHP/FI 2.0, PHP 3.0 был объявлен официальным продолжением PHP/FI и его разработка была закрыта.

Одной из сильнейших сторон PHP 3.0 была возможность расширения ядра. В последствии интерфейс написания расширений привлёк к PHP множество сторонних разработчиков, работающих над своими модулями, что дало PHP возможность работать с огромным количество баз данных, протоколов, поддерживать большое число API. Фактически, это и был главный ключ к успеху, но стоит добавить, что немаловажным шагом оказалась разработка нового, намного более мощного и полного синтаксиса с поддержкой ООП.

Абсолютно новый язык программирования получил новое имя. Разработчики отказались от дополнения о персональном использовании которое имелось в аббревиатуре PHP/FI. Язык был назван просто 'PHP' -- аббревиатура, содержащая рекурсивный акроним: 'PHP: Hypertext Preprocessor' (PHP: Процессор гипертекста).

К концу **1998,** PHP использовался десятками тысяч пользователей. Сотни тысяч web-сайтов сообщали о том, что они работаю с использованием языка. В то время PHP 3.0 использовался на серверах примерно 10% сети Интернет.

Официально PHP 3.0 вышел в Июне 1998, после проведения 9 месяце в стадии публичной отладки.

К зиме **1998**, практически сразу после официального выхода PHP 3.0, Andi Gutmans и Zeev Suraski начали переработку ядра PHP. В задачи входило улучшение производительности комплекса приложений и улучшение модульности базиса кода PHP. Расширения дали PHP 3.0 возможность успешно работать с набором баз данных и поддерживать большое количество различных API и протоколов, но PHP 3.0 не имел качественной поддержки модулей и приложения работали не эффективно.

В версии (PHP 4) для повышения производительности используется ядро скриптов Zend, добавлена поддержка еще большего числа библиотек и расширений третьих разработчиков, а PHP выполняется как "родной" серверный модуль на всех популярных web-серверах.

Новое ядро, названное 'Zend Engine' (от имён создателей: Zeev и Andi), удачно справлялось с поставленными задачами. PHP 4.0, основывающийся на Zend Engine и принёсший с собой набор дополнительных функций, официально вышел в Мае 2000, почти через два года после выхода своего предшественника PHP 3.0. В дополнение к улучшению производительности, PHP 4.0 имел ещё несколько ключевых нововведений, таких как поддержка сессий, буферизации вывода, новые способы приёма ввода пользователей и несколько новых конструкций синтаксиса.

Развитие PHP во многом определяется его ядром, Zend Engine. PHP5 будет основываться на новом Zend Engine 2.0.

CGI - это механизм для выбора, обработки и форматирования информации. Возможность взаимодействия, обеспечиваемая CGI, предоставляется во многих формах, но в основном это динамический доступ к информации, содержащейся в базах данных. Например многие узлы применяют CGI для того, чтобы пользователи могли запрашивать базы данных и получать ответы в виде динамически сформированных Web-страниц.

Имеются в виду узлы, предоставляющие доступ к базам данных, средствам поиска, и даже информационные системы, предающие сообщения в ответ на ввод пользователя. Все эти узлы используют CGI, чтобы принять ввод пользователя и передать его с сервера Web базе данных. База данных обрабатывает запрос и возвращает ответ серверу, который в свою очередь пересылает его опять броузеру для отображения. Без CGI база данных этого не смогла бы. Данный интерфейс можно считать посредником между броузером, сервером и любой информацией которая должна передаваться между ними.

В отличии от HTML, CGI не является языком описания документов. Собственно,

это и не язык вообще; это стандарт. Он просто определяет, как серверы Web передают информацию, используя приложения, исполняемые на сервере. Это способ расширения возможностей сервера Web без преобразования при этом его самого. Подобно тому как броузер Web обращается к вспомогательным приложениям для обработки информации, которую он не понимает, CGI предоставляет серверу Web возможность преложить работу на другие приложения, такие как базы данных и средства поиска.

При написании программы шлюза (которая может конвертировать ввод из одной системы в другую) CGI позволяет использовать почти любой язык программирования. Способность использовать при написании программы шлюза любой язык, даже язык сценариев, чрезвычайно важна. Самыми популярными языками являются shell, Perl, C и C++. Сценарием традиционно называют программу, которая выполняется с помощью интерпретатора, выполняющего каждую строку программы по мере ее считывания. Несмотря на возможность использовать традиционные языки программирования, такие как Pascal, Fortran и даже Basic, языки сценариев популярнее, когда дело касается программирования CGI. В результате программы CGI обычно называют сценариями CGI.

Последовательность действий при взаимодействии клиента с программой запущенной на Web-сервере можно сформулировать как следующая последовательность шагов:

1. Броузер принимает введенную пользователем информацию, как правило с помощью форы.
2. Броузер помещает введенную пользователем информацию в URL, указывающий имя и местоположение сценария CGI, который требуется ввести в действие.
3. Броузер подключается к серверу Web и запрашивает URL. Сервер определяет, что URL должен ввести в действие сценарий CGI, и запускает указанный сценарий.
4. Сценарий CGI выполняется, обрабатывая все передаваемые ему данные.
5. Сценарий CGI динамически формирует Web-страницу и возвращает результат серверу.
6. Сервер возвращает результат клиенту.
7. Броузер отображает результат пользователю

Это является упрощенной схемой взаимодействия между броузером, сервером и сценарием CGI.

Наибольшую популярность CGI - сценарии нашли при использовании в качестве обработчиков форм, средства доступа к базам данных, средства осуществления локального и глобального поиска, шлюзовых протоколов.

Рис. 1.1 - Обобщенная схема функционирования CGI

ISAPI – Если Вы приверженец решений от Microsoft и решились построить свой WEB сервер на платформе Microsoft Internet Informations Server, то тогда вместо CGI ☹ у Вас

появляется возможность использовать приложения ISAPI, которые реализуются в виде динамически загружаемых библиотек. Классифицируя приложения ISAPI можно выделить: расширения ISAPI и фильтры ISAPI.

Фильтры ISAPI предназначены для реализации управления потоком данных между броузером и WEB сервером на уровне протокола HTTP. Область применения: расширенные процедуры аутентификации и безопасности, статистическая обработка, преобразование данных на лету и т.п.

Расширения ISAPI обеспечивают аналогичные возможности, что и CGI программы. К их недостаткам можно отнести то, что расширения ISAPI загружаясь в адресное пространство сервера не представляют собой отдельный процесс, что влечет за собой соответствующие ограничения по их эффективному использованию, а аварийное их завершение часто приводит к остановке самого WEB сервера, а в ряде случаев и других сервисов. Другим существенным недостатком является необходимость очень тщательной отладки ISAPI расширений вследствие их тесной взаимосвязи с адресным пространством сервера.

Единственное, что можно отметить в положительном контексте это некоторый выигрыш в производительности при использовании решения IIS+MS SQL+Windows NT в локальной корпоративной сети, по сравнению с CGI на этой же платформе. Однако практически по всем параметрам такое решение проигрывает решениям на базе UNIX+Oracle под архитектуры Интернет/Интранет.

При реализации сложных и многофункциональных приложений целесообразно использовать в качестве инструментария **язык Java**. В узком смысле слова **Java** - это объектно-ориентированный язык, напоминающий C++, но более простой для освоения и использования. В более широком смысле Java - это целая технология программирования, изначально рассчитанная на интеграцию с Web-сервисом, то есть на использование в сетевой среде, Поскольку Web-навигаторы существуют практически для всех аппаратно-программных платформ, Java-среда должна быть как можно более мобильной, в идеале полностью независимой от платформы.

С целью решения перечисленных проблем были приняты, помимо интеграции с Web-навигатором, два других важнейших постулата.

1. Была специфицирована виртуальная Java-машина (JVM), на которой должны выполняться (интерпретироваться) Java-программы. Определены архитектура, представление элементов данных и система команд Java-машины. Исходные Java-тексты транслируются в коды этой машины. Тем самым, при появлении новой аппаратно-программной платформы в импортировании будет нуждаться только Java-машина; все программы, написанные на Java, пойдут без изменений.
2. Определено, что при редактировании внешних связей Java-программы и при работе Web-навигатора прозрачным для пользователя образом может осуществляться поиск необходимых объектов не только на локальной машине, но и на других компьютерах, доступных по сети (в частности, на WWW-сервере). Найденные объекты загружаются, а их методы выполняются затем на машине пользователя.

Несомненно, между двумя сформулированными положениями существует тесная связь. В компилируемой среде трудно абстрагироваться от аппаратных особенностей компьютера, как трудно (хотя и можно) реализовать прозрачную динамическую загрузку по сети. С другой стороны, прием объектов извне требует повышенной осторожности при работе с ними, а, значит, и со всеми Java-программами. Принимать необходимые меры безопасности проще всего в интерпретируемой, а не компилируемой среде. Вообще, мобильность, динамизм и безопасность - спутники интерпретатора, а не компилятора.

Принятые решения делают Java-среду идеальным средством разработки интерактивных клиентских компонентов (аплетов) Web-систем. Особо отметим прозрачную для пользователя динамическую загрузку объектов по сети. Из этого вытекает такое важнейшее достоинство, как нулевая стоимость администрирования клиентских систем, написанных на Java. Достаточно обновить версию объекта на сервере, после чего клиент автоматически получит именно ее, а не старый вариант. Без этого реальная работа с развитой сетевой инфраструктурой практически

невозможна.

Распространенная история Java (http://java.sun.com/nav/whatis/storyofjava.html) связывается с личностью Патрика Нотона. Этот человек устроился на работу в Sun в 1988 г. и занимался стыковкой ОС X11 и сетевой оконной системы NeWS, которую ее автор, известный компьютерный специалист Джеймс Гослинг, специально переделывал для Sun. Нотону приходилось поддерживать сотни несовместимых программных интерфейсов данных систем, и, устав от непродуктивной деятельности, он задумал перейти в компанию NeXT. Скотт Мак-Нили, исполнительный директор Sun, упросил Патрика перед уходом подробно описать недостатки организации труда в компании. Билл Джой, ведущий научный руководитель и один из основателей Sun, вместе с Гослингом внимательно изучил большое критическое письмо и пообещал исправить все, что возможно. Вскоре после интенсивного обсуждения с Нотоном стратегических направлений развития Sun было решено начать экспериментальный проект Green (создание ручного компьютера Star 7). В проект первоначально вошли Гослинг, Нотон и сотрудник Sun Майк Шеридан. В рамках Green создавались оригинальная ОС GreenOS (версия Unix для компьютеров с ОЗУ 1 Мб) и новый объектно-ориентированный язык Oak, прообраз современной Java.

Но даже авторы технологии Oak долго не понимали ее реального потенциала. Так, в 1995 г. Гослинг назвал багом оказавшееся весьма эффективным представление в Java-пакетах иерархического пространства имен. После полосы неудач, связанных с отсутствием покупателей на созданную технологию, в 1995 г. Билл Джой, очарованный бурным ростом Сети, решил выпустить в свободном варианте элементы технологии Java, в которую к тому времени был переименован Oak. Он угадал: за полгода Java была лицензирована всеми ведущими корпорациями, а сегодня по праву считается одной из наиболее распространенных и перспективных информационных технологий.

Однако в официальной истории ни слова не говорится о концепциях, заложенных в Java в первые годы деятельности Green, хотя ее авторы, конечно, были прекрасно знакомы с множеством существовавших языков и систем программирования. Может показаться, что участники группы Green создавали Java с нуля, равно как и идея самой Green возникла на пустом месте. Но это не так.

С 1989 г. фирма Bridge Systems, основанная Томом Стенбаухом (бывшим сотрудником IBM, проектировщиком PDP 11 и специалистом по Smalltalk), готовила по контракту с исследовательской лабораторией Sun систему, от которой заказчик требовал сочетания индустриальной мощи Smalltalk и выразительного синтаксиса C++. Ей было дано предварительное название Smalltalk Minus Minus, и Стенбаух вкладывал в нее опыт разработки виртуальных машин (ВМ) Smalltalk для RISC-процессоров. Его ежемесячные отчеты постоянно читали Билл Джой и Джеймс Гослинг.

В 1991 г. прототип системы был куплен Sun вместе с правами на любое использование заложенных в нее решений и идей. И впоследствии никаких официальных контактов Bridge Systems с группой Green не возникало. Поставленный прототип, во многом послуживший основой создаваемой Oak, уже тогда обладал большинством отличий современной версии Java от C++. Правда, в ходе работ инженеры Sun решили, что семантика C++ препятствует реализации ряда передовых требований, и решили взять за основу идеи Objective C (прежде всего средства описания интерфейсов классов).

ВМ Oak использовала компилятор, который готовил промежуточный код для каждого объектного метода и затем переводил его в код целевой платформы (фактически это был прообраз компиляции на лету). Автором этой идеи был Питер Дейч (создавший также интерпретаторы GhostScript/GNU PostScript), разрабатывавший ВМ Smalltalk в лаборатории Xerox PARK. В период с 1982-го по 1984-й он подготовил версию Portable Smalltalk – первую ВМ с компиляцией на лету (эту технологию он называл динамической трансляцией). ВМ была написана на ассемблере Motorola 68000 для рабочих станции Sun 1 и выполнялась без ОС на "голом" оборудовании. Вскоре последовала версия для Sun 2/SunOS – она вошла в пакет Object Works 2.3. В 1986 г. Дейч приступил к реализации на Си новой портативной ВМ High Performance Smalltalk (известной также как ParkPlace Smalltalk-80), включенной затем в

ObjectWorks 2.5.

Производительность ВМ Smalltalk оказалась для сложного объектного языка столь высокой, что ею всерьез заинтересовалась IBM, уделявшая особое внимание Smalltalk-программированию. Свою первую ВМ IBM Smalltalk она подготовила на базе ВМ, разработанной фирмой OTI в начале 80-х годов, и затем использовала ее в своих наработках (а возможно, использует и сегодня). Так, машина IBM Java по внутренней организации была очень похожа на IBM Smalltalk. Надо отметить, что собственный опыт инженеров IBM в создании ВМ также очень богат. Достаточно вспомнить рассчитанные на выполнение в виртуальном окружении язык IBM BAL 360/370 или абстрактный ассемблер для машин AS/400. Правда, эти виртуальные среды были непереносимы на другое оборудование.

Идеи ВМ Smalltalk оказали существенное воздействие на развитие архитектуры процессоров Sun SPARC, которая представляла собой коммерческую реализацию технологии SOAR (SmalltalkOnARisc), подготовленной исследовательским сообществом Smalltalk-разработчиков. Инженеры Sun рассчитывали на значительный эффект от аппаратной оптимизации ВМ Smaltalk, но в SPARK-процессорах удалость достичь "только" двух-четырехкратного выигрыша в быстродействии в сравнении с программными эмуляторами. Такой результат, как отмечала Адель Голдберг ("мать" Smalltalk, много сделавшая для его популяризации, автор одной из популярнейших книг), был сочтен недостаточным для того, чтобы привлекать ресурсы к созданию специализированных ВМ для этого языка, способных использовать аппаратное преимущество. Проще и выгоднее было применять кросс-платформную технологию компиляции на лету, нежели ориентироваться на жесткую привязку к оборудованию. Поэтому группа Green с повышенным интересом относилась ко всем виртуальным технологиям, и здесь наработки Bridge Systems оказались весьма кстати.

С 1987 г. Sun вела еще один проект – создание виртуальной системы программирования Self (сегодня сложно сказать, пересекался ли он с Bridge Systems), и его достижения также активно использовались в проекте Green. Система Self (http://www.sun.com/research/self/) развивалась под девизом "Self – это нечто похожее на Smalltalk, только немного больше". ВМ, положенная в ее основу, была создана фирмой Anamorphic, выпускавшей быстрые коммерческие оболочки Smalltalk. Реализованные в них идеи впоследствии вошли в технологию ускорения Java-программ HotSpotJava.

В языке Self нет выделенного понятия классов. Разработчик имеет дело только с прототипами и может не описывать класс, если для работы нужен единственный объект. Не делает Self и различия между полями класса и методами. Программисту доступен универсальный слот (данная идея воплощена в JavaScript), способный содержать данные или код. Объекты в ходе выполнения программы могут непрерывно изменять свою структуру, поэтому в Self нет также понятия состояния объектов. Создание приложения ведется, основываясь только на средствах взаимодействия объектов, что устраняет традиционный для массовых языков программирования разрыв между методами и данными. Self предоставляет разработчику универсальную графическую среду манипулирования объектами (Morphic User Interface), рассчитанную на быстрое прототипирование и рефакторинг. Эта система развивается до сих пор. Последняя версия 4.1.6 вышла в сентябре 2002 г.

Self реализует идеи, выдвинутые Аланом Кеем, автором Smalltalk, который всегда стремился уйти от формального заучивания и механического использования операторов программирования и призывал к чисто объектному мышлению при создании ПО. Он полагал, что программист должен прежде всего стремиться к организации эффективного обмена сообщениями между объектами, а не заботиться о формализации их внутренних интерфейсов. К сожалению, эта идея оказалась слишком сложной для большинства разработчиков и осталась невостребованной. Но, возможно, наметившееся в последнее время удешевление и массовое распространение систем проектирования, моделирования и рефакторинга вдохнет жизнь в идеи Кея, избавит создателей программ от рутинных аспектов объектных концепций и позволит в полной мере использовать их потенциально неограниченную мощь.

2. ВВЕДЕНИЕ В ИНТЕРНЕТ ТЕХНОЛОГИИ

Одним из самых действенных инструментов Интернета является WWW, причем необходимо обратить внимание на генетическую связь данной подсистемы с информационно-поисковыми системами и глобальными сетями. По существу, WWW представляет собой результатов применения возможностей доступа к территориально распределенной информации для создания глобальных гипертекстовых мультимедийных информационно-поисковых систем (ИПС). Возможности доступа к территориально распределенной информации обеспечивает для WWW всемирная сеть Интернет или локальная сеть интранет. Глобальная WWW система развивается в основном как хранилище слабоструктурированной, разноплановой информации и тем отличается от баз данных, где информация структурирована и взаимосвязана. Для ликвидации данного серьезного объективного недостатка при реализации WWW системы создаваемой ИПОС предполагается использовать базы данных, что позволит четко структурировать информацию по разделам и тематическим рубрикам и обеспечит единую взаимосвязанную среду.

Широкое распространение WWW система получила из-за относительной простоты опубликования информационных материалов и удобства доступа к информационным ресурсам.

Указанные достоинства привели к широкому использованию WWW технологий не только в сети Интернет, но и в интранет сетях. В данных сетях WWW играет роль прикладной службы компьютерной сети. Сама сеть обеспечивает передачу данных между компьютерами согласно используемым протоколам и системой адресации (в Интернет используется протокол TCP/IP, который обеспечивает для WWW транспортный уровень передачи).

2.1. Архитектуры клиент/сервер и Интернет/интранет

Как и ряд других служб Интернета , WWW использует архитектуру клиент сервер, т.е. в каждом сеансе программные средства клиента и сервера совместно решают конкретную прикладную задачу, выполняя информационное обслуживание пользователей (рис. 2.1).

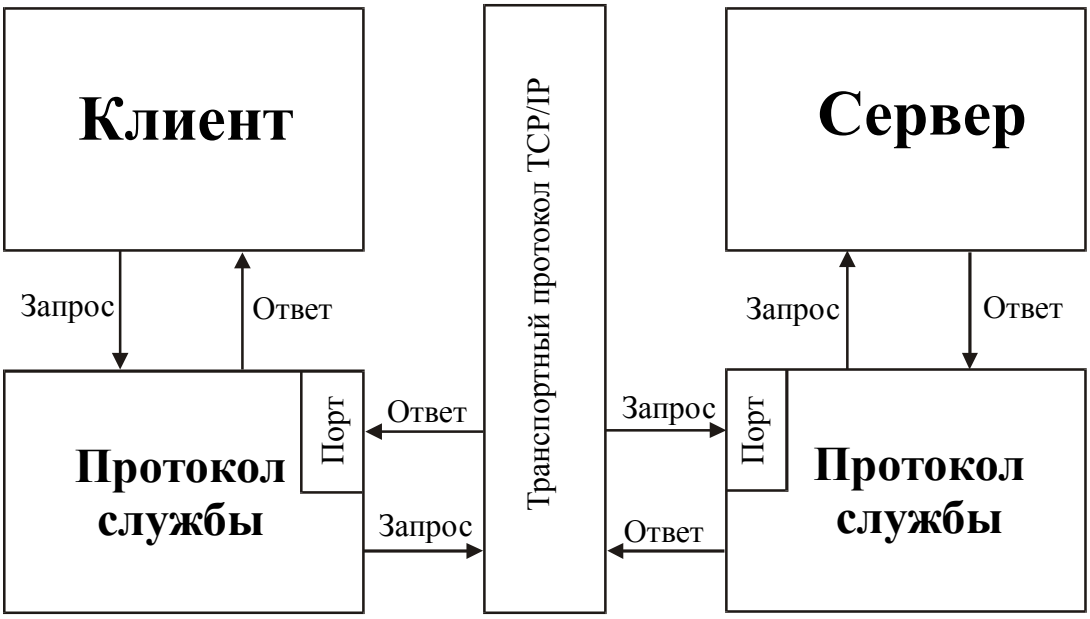

Рис.2.1. Схема функциональных взаимосвязей в архитектуре клиент-сервер.

При построении информационных систем, наряду с архитектурой клиент/сервер, все большее использование находят архитектура Интернет/интранет. В чем же преимущества и недостатки использования каждой из данных архитектур и когда их применение оправдано?

Одной из самых распространенных на сегодня архитектур построения корпоративных информационных систем является архитектура КЛИЕНТ-СЕРВЕР. В реализованной по данной

архитектуре сети клиенту предоставлен широкий спектр приложений и инструментов разработки, которые ориентированы на максимальное использование вычислительных возможностей клиентских рабочих мест, используя ресурсы сервера в основном для хранения и обмена документами, а также для выхода во внешнюю среду. Для тех программных систем, которые имеют разделение на клиентскую и серверную части, применение данной архитектуры позволяет лучше защитить серверную часть приложений, при этом предоставляя возможность приложениям либо непосредственно адресоваться к другим серверных приложениям, либо маршрутизировать запросы к ним.

Однако в данном случае частые обращения клиента к серверу снижают производительность работы сети, кроме этого приходится решать вопросы безопасной работы в сети, так как приложения и данные распределены между различными клиентами. Распределенный характер построения системы обуславливает сложность ее настройки и сопровождения. Чем сложнее структура сети, построенной по архитектуре КЛИЕНТ-СЕРВЕР, тем выше вероятность отказа любого из ее компонентов.

Рис.2.2. Архитектура Клиент-сервер

В последнее время все большее развитие получает архитектура Интернет/Интранет. В основе реализации корпоративных информационных систем на базе данной архитектуры лежит принцип «открытой архитектуры», что во многом определяет независимость реализации корпоративной системы от конкретного производителя. Все программное обеспечение таких систем реализуется в виде апплетов (программ написанных на языке JAVA).

Архитектура Internet - Intranet

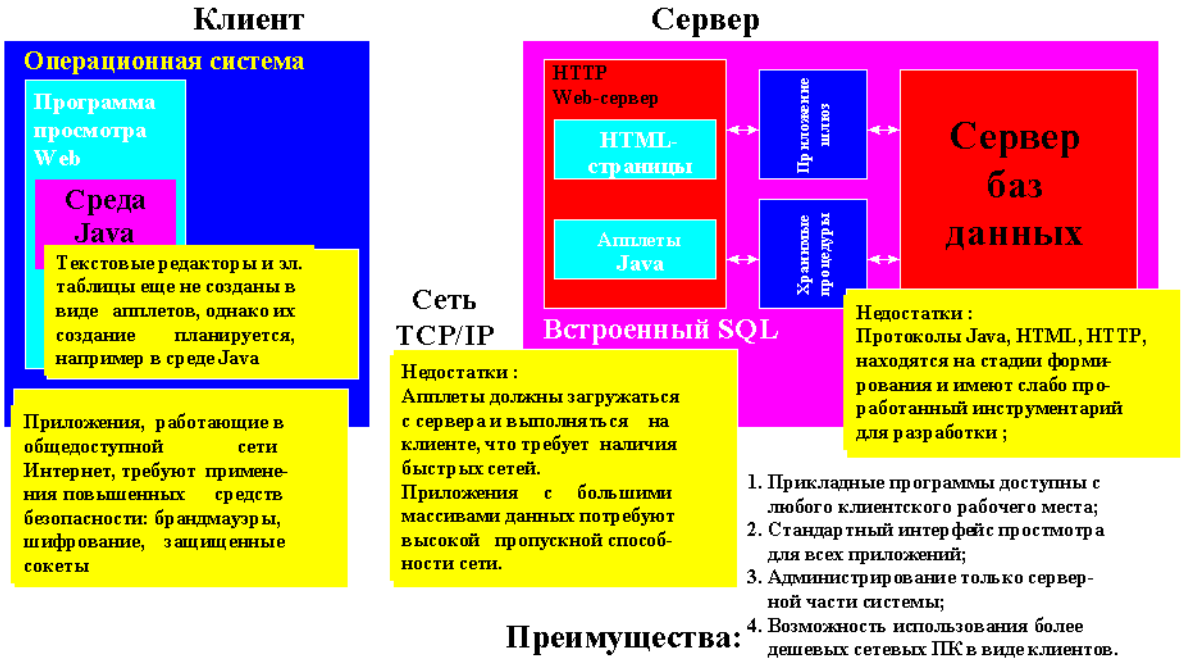

Рис.2.3. Архитектура Интернет-Интранет

Основными экономическими преимуществами данной архитектуры являются:

- относительно низкие затраты на внедрение и эксплуатацию;
- высокая способность к интеграции существующих гетерогенных информационных ресурсов корпораций;
- повышение уровня эффективности использования оборудования (сохранение инвестиций). Прикладные программные средства доступны с любого рабочего места, имеющего соответствующие права доступа;
- минимальный состав программно-технических средств на клиентском рабочем месте (теоретически необходима лишь программа просмотра – броузер и общесистемное ПО); минимальные затраты на настройку и сопровождение клиентских рабочих мест, что позволяет реализовывать системы с тысячами пользователей (причем многие из которых могут работать за удаленными терминалами). Основными сложностями при реализации корпоративных систем на базе данной архитектуры являются:
- отсутствие многих популярных приложений и средств разработки реализованных в виде JAVA апплетов;
- относительное высокое время компиляции апплетов на клиентских местах (временно); вопросы безопасной работы в сети.

Реализация Intranet сетей стала возможной с формированием концепции полнофункционального Intranet (FSI-Full Service Intranet), которая стала реальной альтернативой тиким ОС, как Novell NetWare, Windows NT и т.п. FSI представляет собой корпоративную сеть, реализованную на базе протокола TCP/IP. В основе FSI лежит ядро состоящее из пяти равноправных сервисных частей:

- управление ресурсами;
- управление электронной почтой;
- управление файлами;
- управление печатью;
- общесетевое управление.

Взаимодействие всех частей позволяет обеспечить применение стандартных технологий Intranet для построения эффективных, полнофункциональных, всеобъемлющих

информационных инфраструктур, предназначенных для совместного использования информации, средств коммуникаций и приложений.

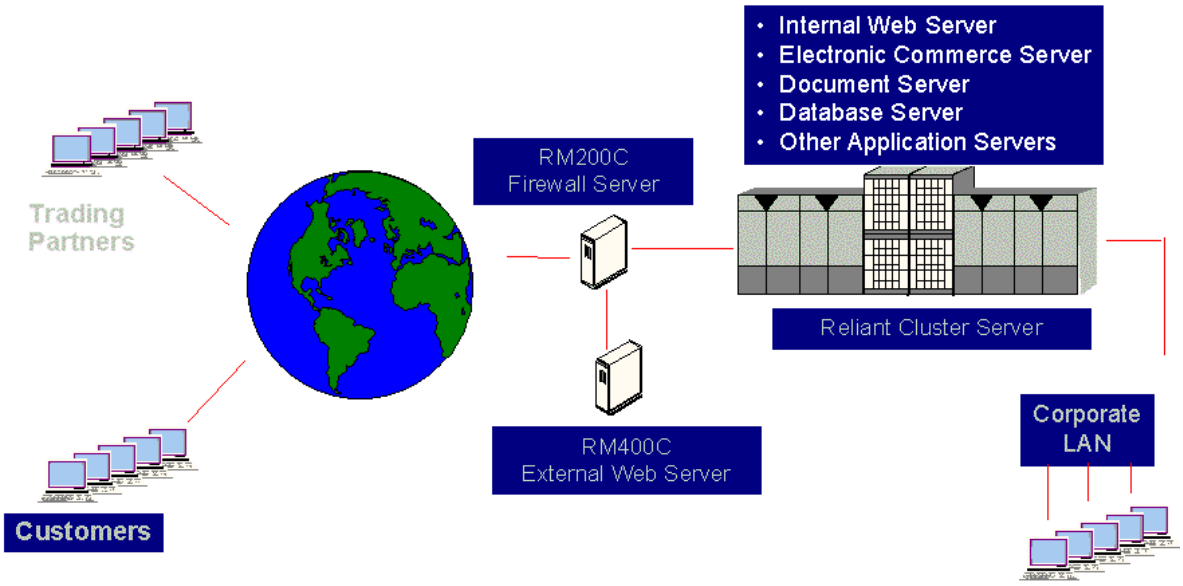

Рис.2.4. Типовая структура сети по архитектуре Интернет/Интранет.

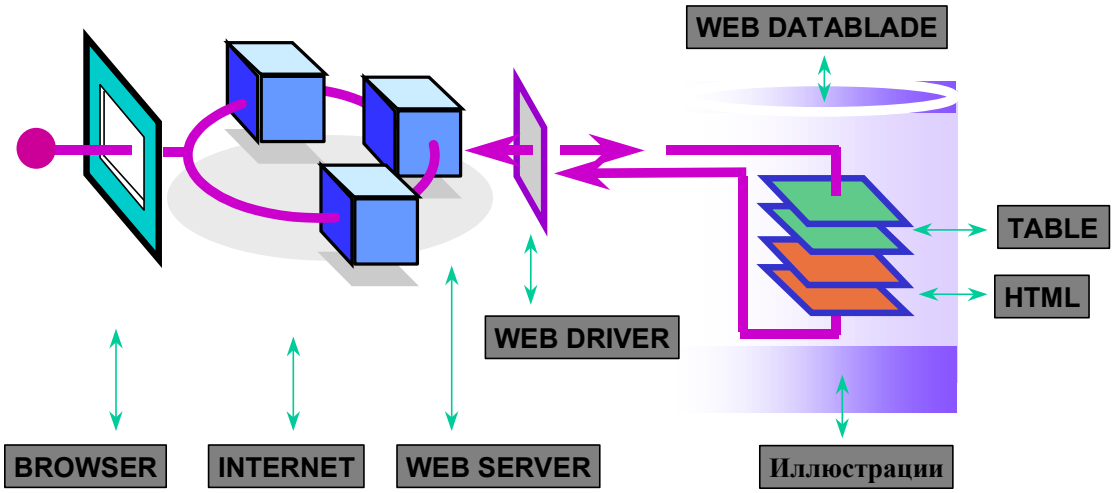

Рис.2.5. Функциональная инфраструктура потоков данных в сети Интернет/Интранет.

За передачу гипертекстовой информации отвечает протокол HTTP - (Hypertext Transfer Protocol - протокол пересылки гипертекста) - это набор правил по которому клиенты и сервера WWW осуществляют обмен информацией между собой. Он представляет собой ядро WEB. Перед тем как перейти к рассмотрению принципов работы HTTP познакомимся с одним из важнейших понятий сетевых технологий - транзакцией.

Транзакция - это набор операций которые переводят базу данных из одного корректного (допустимого) состояния в другое.

В протоколе HTTP все транзакции имеют один общий формат, каждый запрос клиента и ответ сервера состоит из трех частей: строки запроса (ответа), раздела заголовка и тела.

Алгоритм инициирования клиентом транзакции следующий:

Устанавливается связь клиента с сервером по назначенному номеру порта.

Посылается команду (метод) на запрос документа с указанием адреса документа и номера версии HTTP, например: GET /contents.htm HTTP/2.0. (GET - метод запроса документа , в данном случае contents.htm).

Клиент посылает вспомогательную информацию о настройках клиентской системы (конфигурация , данные о форматах обрабатываемых документов. Форма представления данной информации - построчная. Например, приведенный ниже заголовок содержит имя клиента, номер версии и предпочтительные типы обрабатываемых документов.

User-Agent: Mozilla/2.02 (Win95; I) IP: 194.130.120.190 Accept: */*, image/gif, image/jpeg .

Послав запрос и заголовки , клиент может отправить и дополнительные данные. Эти данные используются главным образом CGI программы , которые применяют метод POST.

Эти данные также могут использоваться и Netscape Novigator при размещении отредактированной страницы обратно на сервер. Данная информация о клиенте может быть получена в любое время, практически любым пользователем сети Интернет, необходимо для этого написать лишь не сложный CGI скрипт. Наиболее интересным представляется здесь IP адрес по которому даже в случае динамического назначения IP адресов провайдером возможно очень легко идентифицировать клиента, также интересна и информация о используемых операционных системах. Это еще раз подчеркивает тот факт, что перемещаясь по просторам Интернета надо помнить, что о вас все известно.

Алгоритм ответа сервера на запрос клиента следующий:

Сначала сервером формируется строка состояния, которая состоит из трех полей: версия HTTP, код состояния и описание. Код состояния - это трехзначное число, обозначающее результат обработки сервером запроса клиента. Описание ,следующее за кодом состояния, представляет собой просто текст, поясняющих данный код состояния, например: HTTP:/2.0 200 OK - т.е. сервер для ответа использует протокол HTTP 2.0. Код состояния 200 означает, что запрос клиента был успешным и затребованные данные будут переданы после заголовков.

После строки состояния сервер передает клиенту информацию заголовка, содержащую данные о самом сервере и затребованном дрокументе, например: Date: Fri, 2 Sep 1998 10:10:23 GMT Server: NSCA/1.5.2 Last-modified: Mon, 1 Sep 1998 11:11:11 GMT Content-type: text/html Content-length: 4068 завершается заголовок пустой строкой.

Если запрос обработан успешно, то посылаются затребованные данные. Это может быть как копия файла. так и результат выполнения CGI программы. Если же удовлетворить запрос клиента не удается, то сервер формирует информацию с указанием причин невозможности обработки запроса пользователя, которая и передается клиенту в понятной форме.

Транзакции и соединения с сервером взаимосвязаны. В более старом протоколе HTTP 1.0 при завершении транзакции связь с сервером разрывалась, в HTTP 1.1 она сохраняется до тех пор пока соединение явно не закроется пользователем. HTTP не сохраняет информация по транзакциям, поэтому при следующем запросе приходится начинать все заново.

Функциональная структура сети Интернет.

Основное предназначение Интернета – это среда (инфраструктура) для обмена информацией (рис.2.6). Информация, передаваемая по Интернету, разбивается на пакеты, которые по пути к адресату проходят через большое число промежуточных узлов в сети. Связь между абонентами, компьютеры которых имеют различные операционные системы, осуществляется по стандартному протоколу TCP-IP.

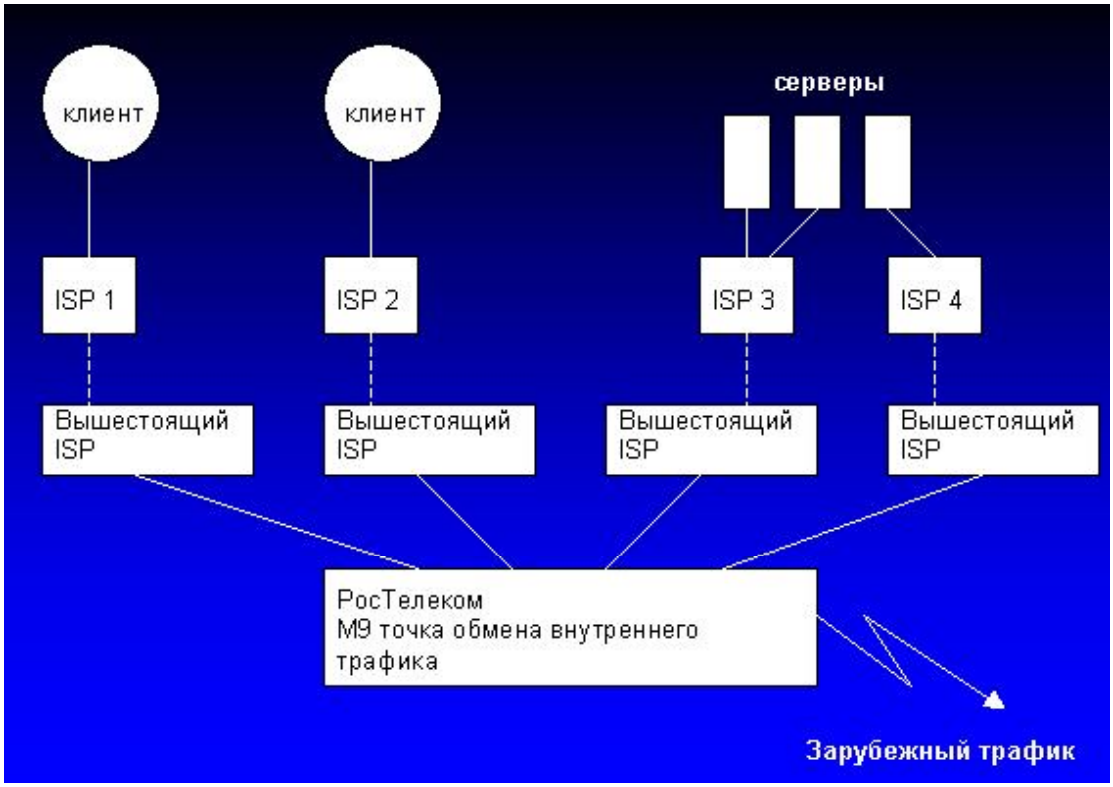

Рис.2.6. Функциональная структура сети Интернет.

Интернет представляет собой объединенную в глобальную сеть компьютеры, которые способны обмениваться между собой различной информацией. Некоторые из данных компьютеров активно работают в сети постоянно, другие только некоторое время. Работа Интернета схожа с функционированием международной телефонной системы – отсутствует единый владелец или управляющий, но все объединены и работают как в одной большой сети.

В основе сети Интернет лежат региональные сети. Управлением региональной сетью осуществляет специальная организация, которая обеспечивает функционирование региональной точки входа в Интернет – NAP (Network Access Point). Непосредственное подключение пользователей к сети осуществляют провайдеры услуг Интернет (ISP, Intrenet Service Provider). Если вы планируете выходить в сеть Интернет через локальную интранет сеть своего предприятия, то Вам нет необходимости входить в контакт с провайдером, достаточно лишь убедиться в том, что вашей рабочей станции в сети предприятия предоставлен доступ для выхода во внешнюю сеть. Если же подключаетесь к Интернет самостоятельно, например, посредством телефонных линий, то тогда Вам придется обратиться к одному из провайдеров услуг Интернет, расположенных в вашем регионе.

Под интерактивными средствами предоставления и управления информационными ресурсами мы будем понимать весь комплекс технологических возможностей предоставляемых сетями Интернет/интранет. К числу таких основных технологий (сервисов Интернета) относятся:

- WWW – всемирная паутина гипертекстовых ресурсов;
- E-MAIL – электронная почта
- FTP - средства организации файловых архивов и доступа к ним
- NFS - сетевая файловая система
- IRC - диалоговые средства реального времени
- NEWS – служба новостей
- PUSH – технологии для принудительной доставки информации.
- JAVA апплеты – средства выполнения приложений.

2.2. Сервисы и службы сети Интернет

2.2.1. WWW и гипермедийное представление информации.

WWW - это информационный сервис сети Internet (структурированное представление информации пользователям).World Wide Web построена на технологии, в основу которой положено понятие гипертекста.

За последние несколько лет объем гипертекста в Internet увеличился до огромных размеров. Служба World Wide Web (сокращенно WWW или просто Web) имеет графический, удобный для поиска документов интерфейс. Эти документы, а также связи между ними, образуют пространство Web. Файлы или страницы Web связаны между собой. Страницы Web могут содержать текст, рисунки, видео и звукозаписи - все, что угодно. Такие страницы могут находиться на компьютере, расположенном в любом месте земного шара. При подключении к Web вы получаете доступ ко всемирной информационной службе. Гиперссылки являются текстом или графическими объектами, содержащими адреса Web. Если установить указатель на гиперссылку и нажать кнопку мыши, произойдет переход к странице, размещенной на другом узле Web. Каждая страница, включая основную страницу узла Web, имеет свой адрес универсального указателя ресурсов URL (Uniform Resource Locator). Адрес URL состоит из имени компьютера, содержащего необходимую страницу, и полного пути к этой странице.

Серьезная проблема - недостаток инструментальных средств для построения структуры ссылок. Большинство существующих гипертекстовых документов скрупулезно создавались вручную. Редакторы гипертекста только разрабатываются; но каждая уважающая себя фирма имеет WWW- сервер и представляет свою информацию в виде гипертекстов с рисунками, файлами, фильмами и т.д. Хотя WWW и Gopher имеют много сходства, но WWW построена на гипертекстовых документах и структурирована ссылками между страницами гипертекста, в то время как в Gopher вся информация основывается только на индивидуальных ресурсах и серверах.

WWW - очень гибкая система. Например, она позволяет читать материалы телеконференций USENET. Если вы читаете новости, то, вероятно, отметили, что каждая статья содержит ссылки на другие сообщения. Программа просмотра преобразует материалы, полученные по электронной почте, в гипертекст, превращая перекрестные ссылки в гипертекстовые. Следовательно, Вы можете перемещаться между исходными статьями, статьями, продолжающими обсуждение по перекрестным ссылкам (материалами, на которые в тексте имеются ссылки), используя установленные связи. Наконец, WWW устраняет преграду между вашими данными и "общедоступными данными". Если вы установите WWW-сервер и соответствующий редактор гипертекста, то сможете интегрировать в WWW свои персональные заметки.

Итак WWW является информационной системой, основанной на гипертекстовых структурах. Она интегрирует документы различных форматов и традиционные услуги Интернет, такие как Telnet, FTP, Gopher, E-mail, WAIS и News.

2.2.2. Электронная почта (E-mail)

Получив возможность поиска и обработки информации, размещенных на WEB серверах (сайтах) перед Вами рано или поздно встает вопрос о возможности получать или посылать информацию в любую точку земного шара, и тогда для решения этого вопроса Вы обращаете свой взор на внедрение коммуникационных технологий. Для многих пользователей знакомство со средствами коммуникаций начинается с электронной почты (e-mail) говорить о преимуществах которой перед обычной, для человека, который постоянно пользуется ею, получая информацию ежедневно со всего мира, нет необходимости.

Один раз попробовав работать с электронной почтой никакой пользователь уже ни за что не откажется от ее услуг. Оградить этого человека от пользования электронной почтой - это значит поставить его в положение, когда он не сможет нормально работать, а может быть и

вообще не сможет работать. Электронная почта дает огромное преимущество даже перед телефоном, потому что пользуясь ею любой вопрос решается намного быстрей и проще, кроме всего прочего и намного дешевле в случае нахождения абонента за многие тысячи километров. Что же нужно для реализации электронной почты? Ответ на этот вопрос достаточно прост: минимум-это наличие персонального компьютера с соответствующим программным обеспечением, модема и телефонной линии (или подключения посредством ЛВС).

2.2.3. FTP доступ к файловым архивам

FTP архивы являются одним из основных информационных ресурсов Internet. Фактически это распределенный депозитарий текстов, программ, фильмов, фотографий, аудио записей и прочей информации, хранящейся в виде файлов на различных компьютерах во всем мире. Вся эта информация разделена на три категории:

- защищенная информация, режим доступа к которой определяется ее владельцами и разрешается по специальному соглашению с потребителем. К этому виду ресурсов относятся коммерческие архивы (например коммерческие версии программ в архивах ftp.microsoft.com или ftp.bsdi.com), закрытые национальные и международные некоммерческие ресурсы (например работы по международным проектам, частная некоммерческая информация со специальными режимами доступа (частные благотворительные фонды, например)
- информационные ресурсы ограниченного использования, к которым относятся, например, программы класса shareware (Trumpet Winsock, Atis Mail, Netscape и т.п.). В данный класс могут входить ресурсы ограниченного времени использования (текущая версия Netscape перестанет работать в июне, если только кто-то не сломает защиту) или ограниченного времени действия, т.е. пользователь может использовать текущую версию на свой страх и риск, но никто не будет ему оказывать поддержку (здесь имеется довольно близкое пересечение со свободно распространяемыми)
- свободно распространяемые информационные ресурсы или freeware, если речь идет о программном обеспечении. К этим ресурсам относится все, что можно свободно получить по сети без специальной регистрации. Это может быть документация, программы или что-либо еще. Наиболее известными свободно распространяемыми программами являются программы проекта GNU Free Software Foundation. Следует отметить, что свободно распространяемое программное обеспечение не имеет сертификата качества, но как правило его разработчики открыты для обмена опытом. Если у Вас есть время для "ковыряния" чужих программ, и Вы получаете удовольствие от общения по переписке, то из freeware можно собрать все необходимое для любого рода компьютерной деятельности.

Из выше перечисленных ресурсов наиболее интересными, по понятным причинам, являются две последние категории, которые как правило оформлены в виде FTP архивов.

Технология FTP была разработана в рамках проекта ARPA и была предназначена для обмена большими объемами информации между машинами с различной архитектурой. Главным в проекте было обеспечение надежной передачи, и поэтому с современной точки зрения FTP кажется перегруженным излишними редко используемыми возможностями. Стержень технологии составляет FTP протокол.

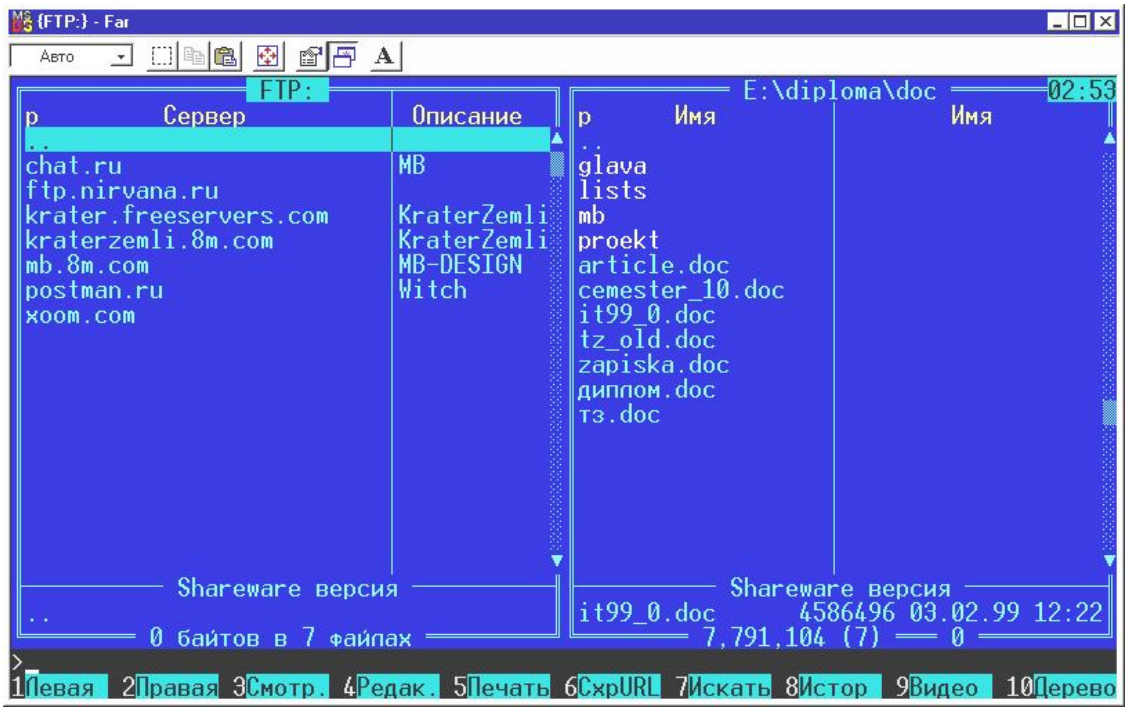

Рис.2.7. Пример использования FTP клиента.

Средством доступа к FTP архивам могут быть включены, как в файловые менеджеры типа Far, Windows Commander, так и в специализированных FTP клиентах (Cute-FTP).

2.2.4. Технология принудительной доставки информации (push - технологии)

Основные особенности push-систем.

В современных информационных системах, основанных на WWW-технологиях, получение информации се конечным потребителем в настоящее время осуществляется преимущественно с использованием способов, базирующихся на предварительном поиске источника информации и последующем ее перемещении на рабочую станцию пользователя. Такая технология обмена информацией получила название pull-технологии (в переводе — "тяни"). Для этой технологии характерны три основных недостатка:

• поиск информации в Web-пространстве сильно перегружает сети неэффективным графиком;

• WWW-сервер в данном случае выступает в роли пассивного источника информации, работающего с низким коэффициентом посещаемости отдельных его разделов, которые недостаточно хорошо организованы и создают ощущение дискомфорта у пользователей;

• для поиска нужной информации требуется соответствующее время.

Указанные недостатки послужили причиной создания новой технологии доставки и распространения информации в Web-пространстве, основанной на механизме публикации и подписки и получившей название push-технологии (в переводе — "толкай"). Ее также называют технологией "принудительной доставки" ("принудительного распространения") или технологией "программируемых Web-каналов". Она используется как в глобальной сети Internet, так и в корпоративных intranet-сетях и предназначена для принудительной доставки с помощью Web-каналов разнообразной информации (в том числе, различной информации образовательного назначения). Распространение сложных составных документов — именно та область, в которой push-технология наиболее эффективна. Web-каналы — это превосходный транспорт для распространения информации, которая изменяется еженедельно, ежедневно, ежечасно или даже еще более динамично. Образовательные Web-каналы дистанционного обучения, которые принимают новые уроки каждый день, — это лишь один частный пример, наглядно демонстрирующий возможности применения push-технологии. С помощью push-систем можно осуществлять рассылку различного рода документов, объявлений, пересылку в фоновом режиме больших по объему файлов, автоматическое обновление программного обеспечения, автоматическую доставку любых часто обновляемых материалов и данных.

Технология принудительной доставки позволяет реализовать на сервере различные дисциплины доставки информации:

- одновременная рассылка информации тысячам пользователей по единственному централизованному запросу;
- дифференцированная рассылка информации по группам пользователей;
- рассылка информации отдельному пользователю.

Пользователи получают необходимую информацию, предварительно устанавливая соответствующие параметры клиентской части при подписке па канал; посылать запрос на эту информацию не требуется. При этом WWW-сервер становится активным источником информации, "принудительно распространяя" ее по сети и освобождая пользователя от необходимости поиска и "вытаскивания" нужной информации.

Формы отображения информации, предоставляемые клиентским программным обеспечением (ПО) push-систем, могут быть разнообразными. Вывод информации может осуществляться:

- на персональную Web-страницу;
- в отдельное окно push-приложения;
- информация может быть передана клиенту электронной почты.

Оригинальный способ выдачи новостей в виде экранной заставки — "хранителя экрана" — реализован в push-продуктах, созданных фирмой Point-Cast **(www.pointcast.com),** например, в PointCast Network.

Еще одной примечательной особенностью push-тсхпологии является возможность вывода распространяемой информации не только на экран пользовательской рабочей станции, но также и па другие периферийные устройства вывода и отображения информации, например принтеры, факсы, пейджеры.

Кроме разнообразных форм отображения распространяемой информации push-приложения позволяют быстро и легко устанавливать пользовательские предпочтения и менять тематику новостей.

Программное обеспечение поддержки push-технологии. Анализ основных архитектурных принципов, механизмов и функциональных характеристик push-систем показал, что этот класс WWW-систем имеет по сравнению с традиционными pull-системами целый ряд преимуществ. Основными из них являются:

- целенаправленная, программируемая на основе выбранных критериев доставка информации пользователю;
- оперативность;
- возможность просмотра информации в режиме off-line и вывода ее не только на экран, но и на различные периферийные устройства (принтер, факс и пейджер).

Описанные свойства push-систем способствуют переориентации многих поставщиков ПО для WWW-систем с традиционных технологий, основанных на поиске и "вытягивании" необходимой информации, на системы с использованием технологии организации Web-каналов, которая обеспечивает прямую доставку информации и удобную единую среду для настольных систем и позволяет полностью сфокусировать внимание пользователя непосредственно на данных, а не на рабочей среде. Перспективность технологии принудительной доставки была оценена также и лидерами Internet/in-tranet-технологий — фирмами Microsoft и Netscape Communications, которые интегрировали механизмы создания и использования Web-каналов в своих новых WWW-браузерах.

Так как на сегодняшний день коммерческие версии push-систем стоят достаточно дорого для большинства российских образовательных организаций, то для реализации конкретных проектов Web-каналов образовательного назначения рекомендуется использовать свободно распространяемые бесплатные push-продукты (например, Point-Cast Intranet Broadcast Solution 2.1) или бесплатные оценочные бета-версии таких систем (например, Castanet 2.1 Public Beta), а в качестве клиентской части — бесплатные WWW-браузеры Microsoft Internet Explorer и Netscape Communicator, позволяющие работать с такими Web-каналами.

2.2.5. Chat системы

Чат системы позволят Вам принимать участие в любом количестве дискуссий, как по служебным вопросам, так и по личным интересам. Вы можете отправлять сообщения непосредственно интересующему Вас лично или всем собеседникам сразу. Чат системы работают в on-line режиме, т.е. все пользователи чата должны находится в сети Интернет. Существуют несколько видов чатов: WEB-чаты, основанные на технологии CGI и протоколе HTTP, используется обычный браузер (наиболее медленный чат), IRC-чатов, основанных на специальных программных средствах т.н. IRC-клиент, (самый быстрый чат) и JAVA-чаты, созданные на основе JAVA-технологий, (долго загружается).

2.2.6. Новостные системы (NEWS)

Новостные системы позволят Вам принимать участие в любом количестве дискуссий, как по служебным вопросам, так и по личным интересам. Вы можете подписаться на материалы интересующих вас дискуссий ограничив материалы узкой направленности, можете получать все материалы, а можете просматривать эти материалы один или два раза в месяц.

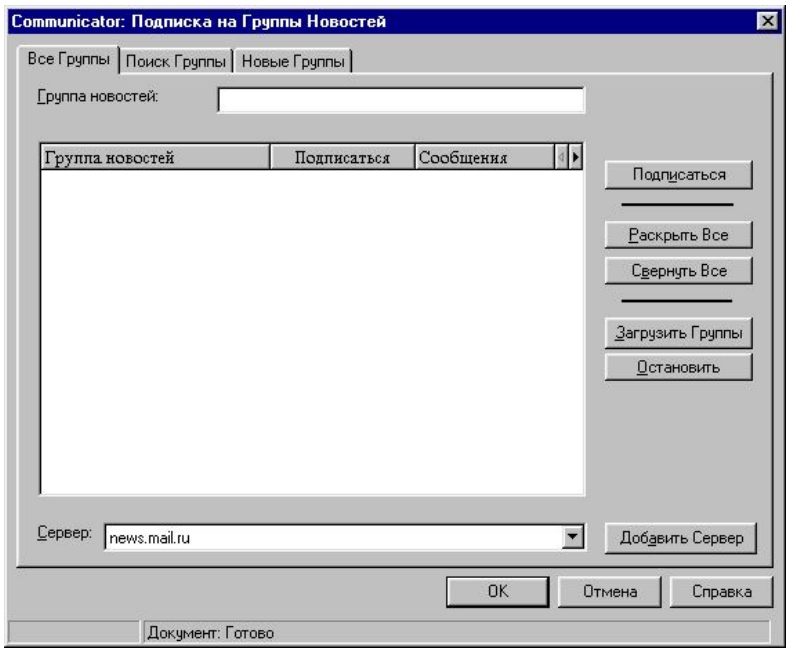

Рис.2.8. Подписка на группы новостей.

Новостные системы - это Internet эквивалент электронной доски объявлений (BBS), работающих в CompuServe или на частных коммутируемых линиях. Программа чтения телеконференций представляет материалы этих дискуссий в упорядоченном виде, отделяет статьи, которые вы уже видели, и показывает только новые, которые поступили после окончания последнего сеанса. После показа списка существующих статей по данной теме, вы можете выбирать и читать то, что Вас интересует. Если вы забыли, где вы видели этот материал, то можете найти эту статью по данным об авторе, теме или по указанному контексту.

2.2.7. Поисковые системы

Пользователям Internet уже хорошо известны названия таких сервисов и информационных служб, как Lycos, AltaVista, Yahoo, OpenText, InfoSeek, а без услуг этих систем сегодня практически нельзя найти что-либо полезное в море информационных ресурсов Сети. Но что собой представляют эти сервисы изнутри, как они устроены, почему результат поиска в терабайтных массивах информации осуществляется достаточно быстро и как устроено ранжирование документов при выдаче - все это обычно остается за кадром. Тем не менее без

правильного планирования стратегии поиска, знакомства с основными положениями теории ИПС (Информационно-Поисковых Систем), насчитывающей уже двадцатилетнюю историю, трудно эффективно использовать даже такие скорострельные сервисы, как AltaVista или Lycos.

На сегодняшний день существует целый ряд различных информационно- поисковых систем. Хорошо известны названия таких сервисов и информационных служб, как Lycos (www.lycos.com), AltaVista (www.altavista.digital.com),Yahoo(www.yahoo.com),InfoSeek (www.infoseek.com), Rambler (www.rambler.ru), Yandex (www.yandex.ru) и т.д., без услуг которых сегодня практически невозможно найти что-то необходимое в терабайтном море информационных ресурсов Сети.

3. ОСНОВЫ ЯЗЫКА HTML

HTML - приложение стандарта ISO 8879. Простой язык гипертекстовой разметки, используемый для создания платформенно - независимых документов. HTML 3.2 был принят в январе 1997 года. В апреле 1998 года выходят рекомендации W3C относительно HTML 4.0.
Итак что же такое HTML:

- HTML разработан специально для WWW (сервис сети интернет предоставляющий доступ пользователям к документам по протоколу http) и является независимым от платформ.
- HTML - открытый стандарт. То есть не является собственностью какой-либо фирмы. Использование языка - бесплатное.
- HTML доступен для людей не программистского склада. Несмотря на схожесть с языками программирования это именно язык разметки (хотя мы его в данном пособии будем изучать как язык программирования, а точнее проектирования информационных систем).
- HTML поддерживает мультимедиа.

Понятие Гипертекст (hypertext) - особый вид текстового документа, при просмотре которого в соответствующей программе - броузере, имеет активные области (гиперссылки) по которым осуществляется переход к объектам, на которые указывает ссылка. Это может быть как часть этого же документа, так и отдельный документ или объект мультимедиа, для просмотра которого будет запущена отдельная программа.

Гипермедиа (hypermedia) - воспроизведение видео- или аудиозаписи непосредственно на странице гипертекста, благодаря модульной технологии (plug-ins).

3.1. Структура HTML-документа

HTML (HyperText Markup Language) - язык гипертекстовой разметки документов, подчиняется ряду правил определяющих его синтаксис и семантику. **Синтаксис** языка представляет собой набор правил, определяющих допустимые конструкции языка, его форму. **Семантика** языка представляет собой набор правил, определяющих смысл синтаксически корректных конструкций языка, его содержание [2]. Язык HTML, представляет из себя набор дескрипторов с атрибутами.

Дескриптор - основной элемент кодирования, принятый в стандарте HTML, состоит из открывающегося < > и закрывающегося </ > тэгов - такие дескрипторы именуются контейнерами (container). В ряде случаев закрывающийся тэг может отсутствовать - одиночные дескрипторы.

Под действие контейнерного дескриптора попадает содержимое контейнера.

<дескриптор> содержание контейнера </дескриптор>

Атрибут - элемент тэга определяющий, в зависимости от значения аргументов атрибута, дополнительные параметры.

<дескриптор атрибут=”значение атрибута”>содержание контейнера</дескриптор>

Значения атрибута, как правило, являются константами, определенными при разработке документа, основные типы которых представлены в таблице 1.

Таблица.3.1 Типы констант

Условное обозначение	Наименование
n	Целочисленные константы
char	Строковые константы, состоящие из символов (см. приложение 1 – символы и их обозначения в HTML).
color	Задание цвета (приложение 2).
URL	Уникальный идентификатор ресурса (см. приложение 3).
FILENAME	Имя файла, это может быть как имя файла на локальном диске (относительное задание пути и наименование), так и его URL адрес (абсолютное задание пути и наименование документа). **!!Внимание: Большинство серверов в интернете работают на ОС UNIX клона и имена файлов, указанные в различных регистрах – это не одно и тоже. Пример, index.htm и INDEX.HTM – ЭТО ДВЕ БОЛЬШИЕ РАЗНИЦЫ!!**
%%%%	Зарезервированные слова

Следовательно, для того, чтобы произвести какие-либо действия с элементом тестового документа необходимо:

- Воспользовавшись любым текстовым редактором, разместить данный элемент текста в «контейнер».
- Разместить «контейнер» внутри контейнерного (или после одиночного) дескриптора.
- Определить дополнительные параметры разметки назначив значения атрибутам тега.
- Сохранить документ.
- Проверить результат разработки в броузере.
- Для размещения созданного документа в сети интернет его еще придется опубликовать на соответствующем WEB сервере с помощью, как правило, FTP доступа.

Вот в принципе и весь алгоритм действий, который необходимо выполнить при создании гипертекстового документа и его публикации в сети интернет/интранет.

Прежде чем перейти к рассмотрению основных синтаксических конструкций для разметки гипертекстовых документов (т.е. описание алфавита и правил построения различных конструкций языка из символов алфавита и более простых конструкций языка) введем понятие синтаксической диаграммы.

Синтаксическая диаграмма – это графическое представление правил построения конструкций языка в наглядной форме. При этом принято, символы алфавита изображать блоками в овальных рамках, названия конструкций – в прямоугольных, а правила построения конструкций в виде линий со стрелками (часто овальные и прямоугольные рамки для упрощения опускают). Также выделяют осевую линию синтаксической конструкции, на которой размещаются все обязательные символы алфавита и конструктивные элементы. Все необязательные элементы и символы размещаются ниже или выше осевой линии конструкции. Если конструкция имеет значительную длину, то осевая линия переносится соответственно на следующую строку. Существуют и альтернативные формы отображения синтаксиса языка, например форма Бэкуса-Наура (БНФ) и т.п. [2].

Рассмотрим суть использования синтаксической диаграммы для представления синтаксиса языка HTML на простейшем примере. Так, например, стандарт определяет, что документ должен начинаться тэгом **<html>** и заканчиваться тэгом **</html**>. Сам документ, с всевозможными включениями (например, JavaScript или VBScript), располагается внутри этих тэгов. Синтаксическая диаграмма, описывающая эту конструкцию представлена на рис.3.1.

Рис.3.1 Синтаксическая диаграмма конструкции HTML документа.

В дальнейшем обозначение символов алфавита в овальных рамках будем опускать.

Гипертекстовый документ принято разделять на две части: «заголовок» и «тело» (рис.3.2.).

Рис.3.2 Структура HTML документа.

В заголовке располагается различная служебная информация, а в теле – элементы (конструкции) документа. Заголовок располагается внутри контейнерного дескриптора <head>, а тело внутри контейнерного дескриптора <body>. Синтаксическая диаграмма структуры html документа представлена на рис.3.3

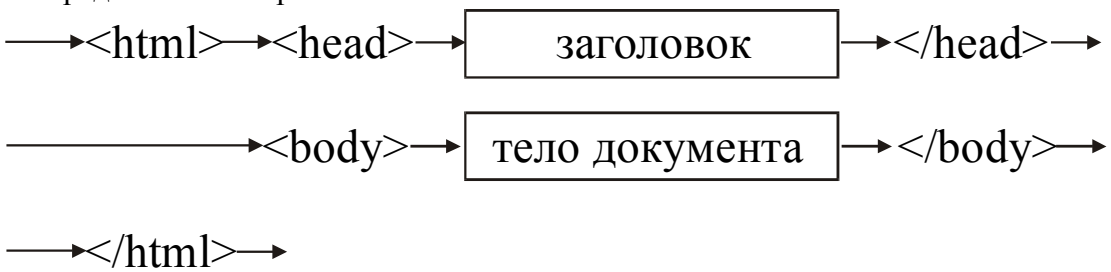

Рис.3.3. Синтаксическая диаграмма структуры HTML документа.

Перед тем, как перейти непосредственно к рассмотрению синтаксиса и семантики гипертекстовой разметки, акцентируем внимание на таком понятии, как **структурное программирование**.

Под **структурным программированием** понимается такой способ представления исходного кода программы (в не зависимости от языка программирования), при котором отдельные функционально законченные блоки программы объединяются в «структурные модули», первый и последний оператор которых располагаются на одинаковом расстоянии от левой стороны поля документа исходного кода программы. Все остальные элементы данного модуля располагаются со сдвигом влево, причем взаимозависимые элементы принято располагать с одинаковым отступом, элементы-«родители» левее, элементы-«потомки» - правее.

Например, представим синтаксическую диаграмму (рис.3.3) согласно правилам структурного программирования:

```
<html>
   <head>
     <title>Заголовок</title>
  </head>
  <body>
     Тело документа
  </body>
</html>
```

Чтобы отразить название создаваемого HTML документа в заглавную часть введен дескриптор <title>. Название будет отображаться в верхней статусной строке броузера. Конструкция «Тело документа» будет отображена в виде текста в рабочем поле броузера.

Задание 1: Разработать html страницу согласно вышеприведенному примеру с именем index.htm (имя index.htm или иное – это имя страниц на которое настроен используемый веб сервер для загрузки по умолчанию, т.е. когда явно не указана страница к которой обращаются по URL адресу, а указан только каталог, например, http://iu4.bmstu.ru/sport/). Последовательность действий представлена алгоритмом на странице 25. Разработанный документ опубликовать в сети интернет на одном из серверов, который предоставляет услуги по бесплатному web- хостингу: www.chat.ru, www.narod.ru, www.nm.ru и т.п.

Мы сделали наши первые шаги по изучению структуры HTML документов, но перед тем, как более детально познакомиться с синтаксисом и семантикой гипертекстовой разметки документов, мы должны ответить на вопрос, а что же мы проектируем, зачем, какие задачи будет решать создаваемая нами информационная система.

3.2. Конструкторско-технологическое проектирование сайта

Что же является предметом разработки сайта, что должно быть размещено в «теле» создаваемого HTML документа. Попробует ответить на эти вопросы.

Во-первых, термин “WEB design” - это WEB проектирование, т.е. разработка конструкции и технологии, где **конструирование** – описание конструкции (расположения элементов, их функциональных возможностях, требования к внешнему виду), **технология** – описание последовательности действий и применяемых технических средств (языков программирования, стандартов, пакетов разработки) для получения требуемой конструкции. Т.е. мы говорим о конструкторско-технологическом проектировании информационных систем.

Последовательность процесса проектирования сайта можно представить следующей схемой:

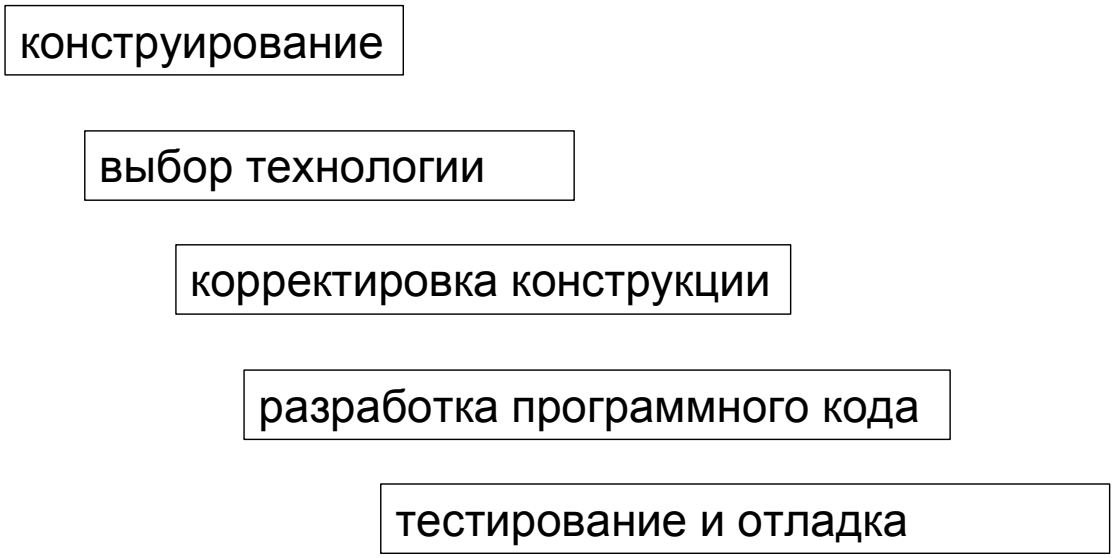

Рис. 3.4 Последовательность проектирования сайта.

Разрабатываемые сайты можно классифицировать следующим образом:

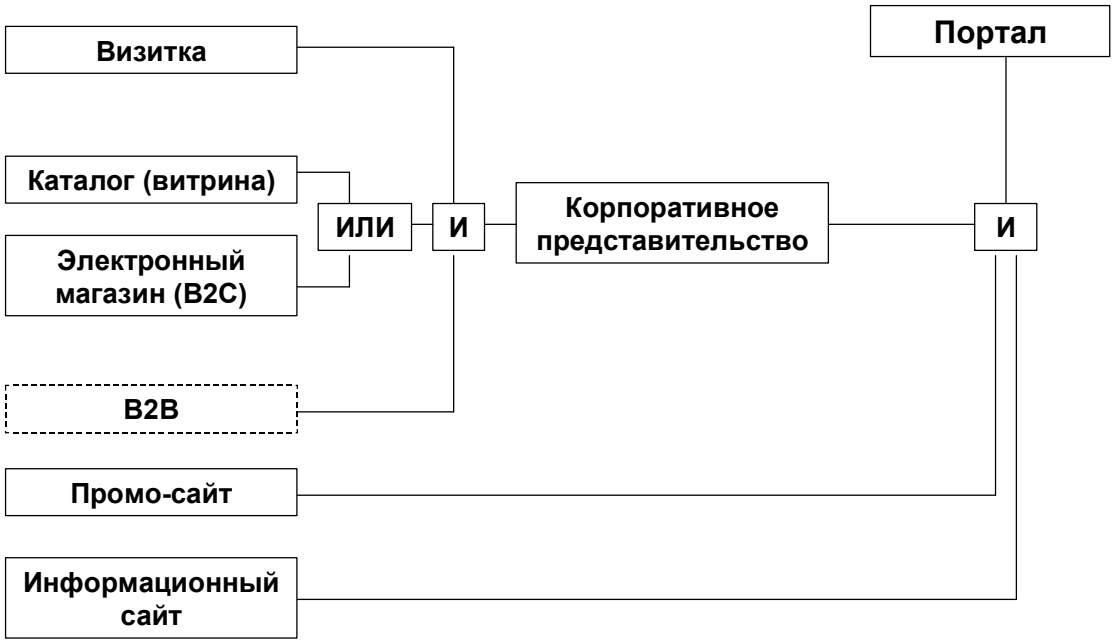

Рис. 3.5 Классификация сайтов

На конструкцию сайта влияют целый набор факторо, основными из которых являются: решаемые задачи, информаицонное содержание, технологические возможноти, корпоративный стиль, пожелания заказчика и т.п.

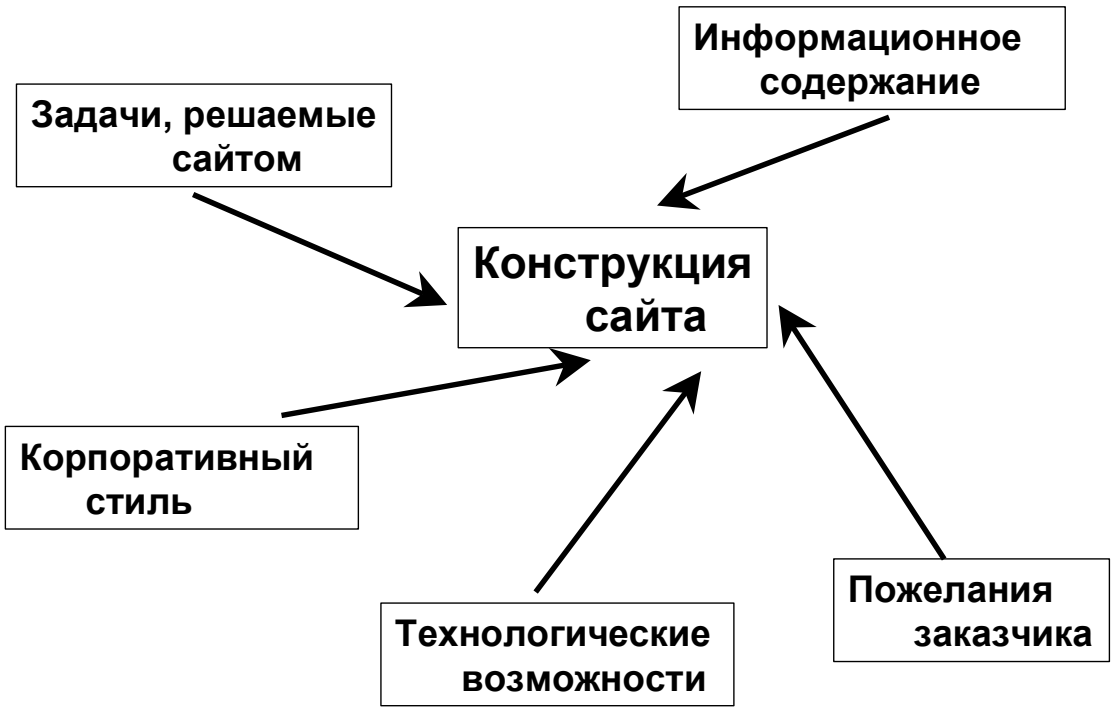

Рис. 3.6 Факторы вляющие на конструкцию сайта.

Элементами конструкторской проработки являются:
— Дизайн (цвето-тоновая композиция)
— Расположение элементов (форма-композиция)
— Представление информации и типовые конструкции сайтов
— Выбор технологий и типовых модулей

В зависимости от решаемых конструкторских и технологчиеских задач можно выделить следующие этапы проектирвоания сайта:
— Определение целей, задач, аудитории сайта
— Аудит проекта
— Определение информационного наполнения сайта, обсуждение идей. Технические, организационно-экономические подходы к созданию и обслуживанию
— Проектирование концепции системы и разработка бизнес-логики
— Проектирование иерархической структуры, навигационной модели, формулировка названия рубрик
— Подбор информационного материала
— Написание скриптов (программ) интерактивного взаимодействия с посетителям
— Запуск в опытную эксплуатацию
— Отладка и доработка проекта
— Запуск в рабочую эксплуатацию

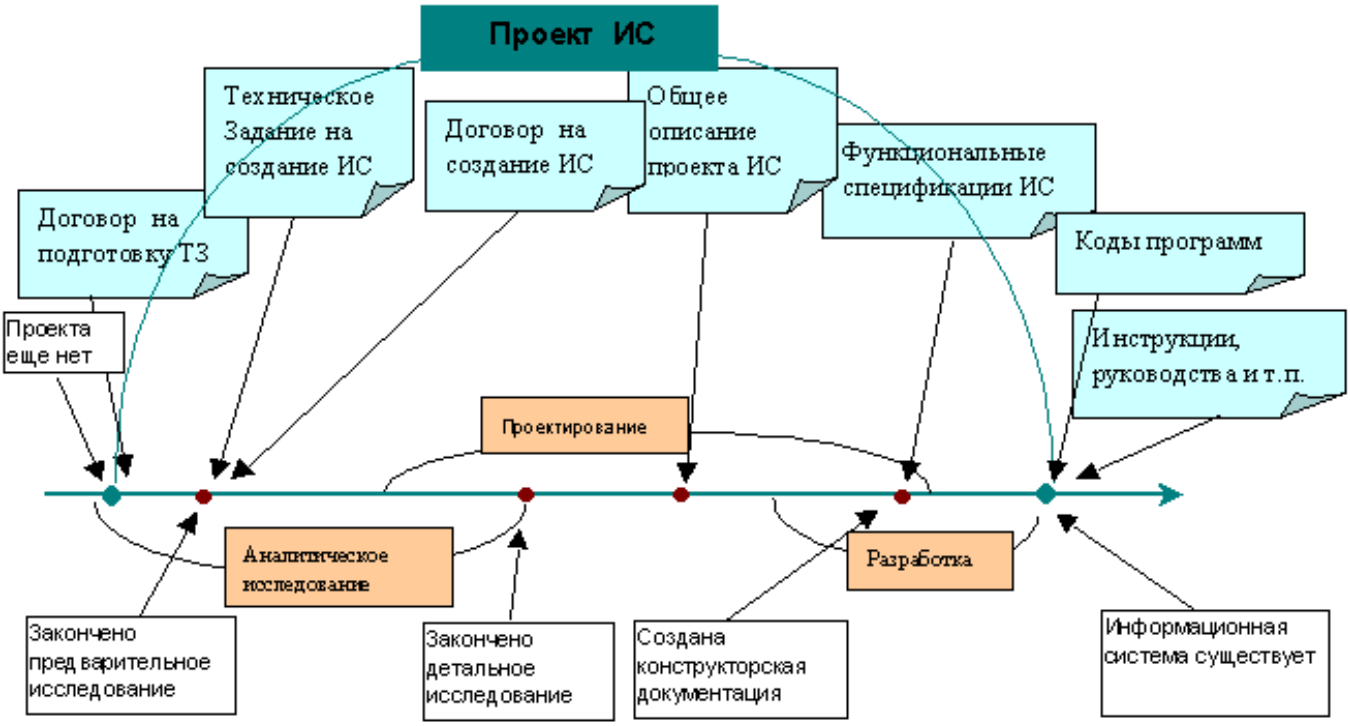

Рис. 3.7 Жизненный цикл абстрактного проекта информационной системы

Принципы проектирования сайтов

Вот несколько основополагающих принципов, которые должны быть приняты за основу при проектировании сайтов:

1. Проектирование для пользователей

Важнейшим условием хорошей работы проектировщика сайта является абсолютное желание угодить пользователям, для которых и делается сайт. При этом любые технологии, которые планируется применить при создании сайта, должны приниматься с учетом упрощения работы пользователей.

2. Ориентация на широкий круг пользователей

Цель любого бизнеса - зарабатывание денег. Цель любого сайта - работать на целевую аудиторию. Сайт должен отвечать потребностям максимального количества пользователей. Он должен быть доступен и удобен, информативен и функционален для всех, кто им постоянно пользуется или вполне может быть использован в ближайшем будущем.

3. Поддержка пользователей

Для нормального функционирования сайта необходима обратная связь с пользователями. Так же, как и в обычном офисе: посетители должны иметь возможность не только получить достаточно информации со страниц сайта (в офисе для этого используются стенды и рекламные проспекты), но и связаться напрямую с сотрудниками компании по интересующим вопросам. Чем более доступны и удобны функциональные способы взаимоотношений сотрудников компании и клиентов, тем более эффективным становится использование сайта в работе компании. Пользователям должно быть достаточно комфортно на сайте. Для этого необходимо внедрять различные средства информационной поддержки пользователей: публиковать на сайте справочные материалы по работе с относительно сложными процессами, публиковать ответы на часто задаваемые вопросы и др.

4. Применение проверенных временем методов

Прежде всего это касается технологий. И вот почему. Для того чтобы «угодить» как можно большему количеству пользователей, интернет-разработчикам приходится ориентироваться на самые распространенные технологии. Любое нововведение в Интернет должно пройти достаточно жесткое тестирование на пригодность для большинства

пользователей. В противном случае сайт, использующий «эксклюзивные» технологии, может оказаться малопосещаемым.

5. Коллективная работа

Еще совсем недавно считалось, что разработкой и сопровождением сайта может заниматься один человек. Называли такого человека веб-мастер. Он должен был совмещать множество функций. Вот некоторые из них: системный администратор, администратор баз данных, программист, дизайнер, HTML-кодер (верстальщик), журналист, маркетолог, менеджер веб-проекта и многое другое. Однако при правильном планировании сайта необходимо понимать, что вряд ли найдется человек, который способен одинаково хорошо выполнять все эти функции. Понятно, что в чем-то он обязательно окажется профаном. А выход очень прост. Вместе с внедрением сайта следует создать новые бизнес-процессы, в которых сайт был бы инструментом работы. Например, подумать о том, кто мог бы готовить и публиковать новости и пресс-релизы, а равно и общаться через сайт с представителями СМИ и общественностью, кто мог бы взять на себя обязанности обрабатывать поступающие через сайт заказы на услуги и продукты, консультировать потенциальных покупателей, следить за аккуратной публикацией прайс-листов, кто смог бы продумать и осуществить комплексную рекламную компанию для сайта и т.д. В результате окажется, что добрая половина коллектива компании будет так или иначе вносить свою лепту в ежедневное поддержание сайта, а сайт станет удобным и нужным инструментом в определенных бизнес-процессах.

6. Отсутствие излишеств

К сожалению, Интернет - не та среда, в которой можно закрывать глаза на излишества. Слабые каналы связи, высокая стоимость доступа, обилие конкурентной информации, сложность ориентации в огромном количестве информации, трудность чтения текста с экрана монитора - вот некоторые из проблем, с которыми постоянно сталкиваются пользователи Интернета. При проектировании сайта необходимо их учитывать.

Список наиболее часто встречаемых излишеств, от которых стоит отказаться ради удобства пользователей, таков:

• Сложные, трудно запоминаемые названия сайтов.
• Использование нестандартных технологий на стороне клиента, избыточная нагрузка на клиента (в данном контексте «клиент» - это браузер, которым пользуются посетители сайта).
• Излишнее использование графики и анимации.
• Перегруз страниц информацией.
• Излишнее количество пунктов в меню (более 6-8).
• Слишком много «воды» в статьях. Статьи плохо структурированы.
• Использование на сайте излишних промежуточных страниц, не несущих существенной информационной нагрузки.
• Нерациональное использование верхней части страниц (много пустого пространства или несущественной информации).

Анализ целей и задач сайта

С чего начинаются сайты? Советую начать с банального вопроса: а нужен ли он вообще? Если нет, то читать дальше нет никакого смысла. Ну а если все-таки нужен, то попробуйте проделать следующее:

1. Определение назначения сайта

Как говорилось уже не раз, сайт делается для пользователей. Иного смысла у сайта быть не может. Поэтому первое, что требуется определить, это состав пользователей и принципов организации их работы на сайте. Насколько четко вы сможете ответить на вопрос, кто должен стать пользователем сайта, настолько четко сможете представить себе будущий сайт. Для этого нужно представить себе и потребности пользователей, и те процессы, в которые они могут быть вовлечены на сайте, а также уровень исполнения и полноту информации, которая потребуется им для принятия определенных решений.

2. Определение и организация информационных тем

Проанализировав предлагаемый состав аудитории сайта, а также процессы, которые будут организованы на сайте, можно определить его информационную модель. Для этого следует определить информационные темы, предназначенные для каждой группы пользователей, а затем разбить их по категориям. После этого очень важно построить информационную иерархию сайта с указанием как вертикальных, так и горизонтальных связей между отдельными информационными темами. Полученную структуру необходимо проверить на соответствие тем процессам, которые будут реализованы на сайте таким образом, чтобы каждый конкретный процесс точно вписывался в информационную структуру.

Таблица 3.2 Типовые модули сайта.

Типовые информационные модули сайта	Типовые сервисные модули
О компании Контакты (координаты) Достижения (реализованные проекты) Портфолио заказчиков Каталог продукции Прайс-лист Ссылки (партнеры, клиенты, вендоры, дистрибьюторы и т.п.) Новости	CMS (content management system) Новости Форум \ гостевая книга \ чат Счетчик Подписка на новости, рассылки Каталог товаров и систем поддержки продаж CRM (Customer relationship management) и т.п.

Предлагаемая схема анализа применима практически для любых корпоративных и информационных сайтов. Она позволит избежать двусмысленности в определении назначения и общей функциональности сайта и тем самым получить исходные данные для разработчиков.

Задание 2: Сформулировать цели и задачи сайта, определить целевую группу пользователей, сформулировать требования к конструкции и технологиям сайта, типовым модулям и их функциональности.
Результат: Техническое задание на информационную систему (сайт) по ГОСТ 34. 601-90, ГОСТ 34.602-90.

Таблица 3.3 Пример постановки задачи для разработки сайта

Задачи	Результат выполнения
1. Определите целевую аудиторию сайта Составьте список целевых групп пользователей сайта Сгруппируйте близкие по целям нахождения на сайте группы пользователей	• Клиенты (постоянные клиенты, потенциальные клиенты) • Партнеры (имеющиеся партнеры, потенциальные партнеры) • Специалисты (специалисты неконкурирующих компаний, начинающие специ- алисты, эксперты) • Представители СМИ
2. Определите процессы, которые будут происходить на сайте Составьте список процессов, которые могут быть реализованы при помощи сайта.	• Получение клиентом информации о предлагаемых товарах • Оформление клиентом предварительного заказа на покупку товара • Получение клиентом, партнером, представителем СМИ корпоративных новостей • Обсуждение специалистами профессиональных тем • Опросы клиентов и партнеров • Оформление потенциальным партнером предложения о сотрудничестве

3. Определите информационные темы сайта Используя полученную в двух первых пунктах информацию, составьте список информационных тем сайта. Разбейте темы по иерархии	• О компании - Общая информация о компании - Новости компании • Площадка продаж (электронная витрина) - Информация о товарах - Форма предварительного заказа • Информация для партнеров - Информация для потенциальных партнеров - Форма предложения о сотрудничестве - Информация для существующих партнеров • Информация для специалистов - Информационные статьи - Форум • Контактные данные
4. Определите концепцию дизайна сайта Исходя из корпоративных требований к дизайну и его целевого назначения определите наиболее важные требования к дизайну	• Логотип в верхней части страниц • Светлый фон, корпоративные цвета: синий, желтый • Образ динамично развивающейся компании со сложившимися традициями, уделяющей внимание качеству обслуживания клиентов и четкой работе с партнерами. Мы законодатели в своем секторе рынка. Каждый серьезный специалист хочет работать именно у нас. • Копирайт в нижнем колонтитуле

Завершающим этапом является проектирование информационного наполнения сайта. Планируя сайт, невозможно не уделить внимания его информационному наполнению - контенту (анг. content - содержимое).

Однако после определения информационной и навигационной структур сайта можно приступить к определению конкретного информационного наполнения отдельных страниц сайта и их последовательностей. Для этого достаточно совместно с человеком, ответственным за подготовку информационного наполнения, определить, какая информация будет расположена на каждой странице сайта (текст, изображения, анимация и т.д.). Все полученные данные должны быть перенесены в таблицу, где напротив каждой задачи должны стоять исполнитель и срок исполнения. Кроме того, совместно с веб-дизайнером (тем специалистом, кто будет делать графический дизайн сайта) необходимо определить количество и разметку шаблонов страниц, которые могут быть использованы в различных разделах сайта. Например, «Главная страница», «Главная страница раздела», «Одностраничная статья», «Многостраничная статья» и т.д.

Полученная информация определит полный объем работ для веб-дизайнера и HTML-кодера, поможет объективно оценить время, необходимое для исполнения проекта.

Итак, для реализации веб-сайта уже имеется следующее:

• определены целевые пользователи;
• определена информационная структура;
• определены объемы информационного наполнения;
• определены количество и специфика шаблонов страниц.

На этом этапе для статического сайта постановка задачи заканчивается. Но для динамического сайта осталось определить функциональность, т.е. сделать опись необходимых функций программной части сайта, составить «Техническое задание» (ТЗ).

Задание 3: Провести оценку бизнес логики сайта на моделях «вариантов использования»

Таблица 3.4 Пример спецификации элементов бизнес логики.

Параметры функции	Характеристики
Название функционала:	обратная связь с посетителями сайта
Назначение:	сбор администратором сайта информационных сообщений от посетителей сайта.
Функции посетителей:	заполнение и отправка формы сообщения, в котором обязательными полями являются: «Имя», «Контактный e-mail» и «Сообщение»;
TRUE	после неудачной отправки сообщения - получение сообщения о неудачной отправке сообщения с приглашением заполнить форму повторно;
FALSE	после успешной отправки сообщения - получение сообщения об успешной отправке сообщения;
RELOAD	при попытке отправить форму с незаполненными обязательными для заполнения полями - получение сообщения о том, что не заполнено обязательное для заполнения поле с указанием его наименования.

Для проверки качества исполнения сайта можно использовать следующий шаблон:

Таблица 3.5 Оценка дизайна и навигации сайта

Наименование характеристики	Значение
Понятно ли с первого взгляда, чему посвящен сайт?	
Соответствует ли дизайн сайта его назначению? Соответствует ли дизайн сайта корпоративным требованиям?	
Экономится ли место в верхней части страницы?	
Имеются ли на странице лишние, громоздкие, бесполезные элементы? Будет ли хуже, если от них избавиться или уменьшить?	
Сможет ли пользователь, просматривая любую страницу сайта, ответить на следующие вопросы:	
На какой странице он находится?	
Куда теперь можно переместиться?	
Можно ли вернуться на это место?	
Как вернуться назад? (для внутренних страниц сайта)	
Как связаться с администрацией сайта?	
Понятно ли пользователям с различным уровнем подготовки, как пользоваться функциональной частью сайта (заполнять формы, взаимодействовать с администрацией сайта и др.)?	

Следующими этапами реализации проекта сайта обычно являются изготовление **шаблонов** страниц и создание на их основе действующей модели (прототипа) сайта, а для этого нам надо будет более необходимо познакомиться с синтаксисом и семантикой языка разметки.

3.3. Синтаксис и семантика HTML

3.3.1. Задание структуры HTML документа.

Как было отмечено в разделе 3.1. любой HTML документ должен начинаться тегом <html> и завершаться тегом </html>. При этом сам документ, как было показано в предыдущем разделе, размещается внутри этих тэгов. Синтаксическая диаграмма представлена на рис.3.1.

Спецификация дескриптора <HTML>		
Дескриптор	**Назначение**	
<html></html>	Начало/конец HTML – документа	
	Атрибут	**Значение**
	version	Обозначение HTML стандарта

Примеры:

```
version="-//IETF//DTD HTML 3.0//EN"
version="-//W3C//DTDHTML 3.2//EN"
```

Дескрипторы **<html>** и **</html>** являются необязательными, но, тем не менее, указываются для идентификации исходного кода.

Однако можно указать стандарт и с помощью дескриптора **<!doctype>,** располагающегося перед дескриптором **<html>**.

Дескриптор	**Назначение**
<!doctype>	обозначение HTML стандарта

Примеры:

```
<!doctype html public "-//IETF//DTD HTML 3.0//EN">
<!doctype html public "-//W3C//DTD HTML 3.2//EN">
```

Комментарии

Хорошим тоном программирования является включение в текст исходного кода **комментариев** и использованием **принципа мнемонического наименования** элементов программы (функций, объектов и т.п.). Т.е., например, функция ввода для модуля управления может быть названа: control_input и т.п. Что касается второго постулата по наименованию элементов конструкций языка, то для HTML документов он тоже полностью выполняется. Сложнее дело обстоит с комментариями, которые, как правило, в HTML документах отсутствуют, так как все таки накладно загружать по каналам передачи данных дополнительную информацию, необходимую скорее разработчику, чем пользователю. В основном комментарии присутствуют в конструкциях HTML документов, разработанных на языках сценариев, а также для служебных целей, с которыми мы познакомимся позднее.

Спецификация дескрипторов задания комментариев	
Дескриптор	**Назначение**
<!- ->	Комментарий в HTML-документе
<comment> </comment>	Начало и конец комментария [IE]

Пример:

```
<!-- Этот текст не будет виден при просмотре документа в броузере ->
```

Заголовок HTML-документа

В заголовке определяются глобальные установки, такие как - имя документа, адрес, используемые шрифты, описание связей с другими документами.

Спецификация дескриптора заголовка HTML документа	
Дескриптор	**Назначение**
<head> </head>	Заголовок HTML-документа

Дескрипторы, используемые в блоке заголовка документа:

Спецификация дескрипторов заголовка HTML документа	
Дескриптор	**Назначение**
<title>	Название (имя) документа
<base>	Базовый URL-адрес данного документа
<basefont>	Базовый размер символов
<isindex>	Организация простого поиска
<link>	Связь с другими документами
<meta>	Передача мета-информации
<script>	Указание на то, что внутри контейнера размещается скрипт на языке сценариев.

При отсутствии **<title>** в заголовке окна броузера будет просто указываться название файла. Название страницы должно быть содержательным и в то же время кратким. Слишком длинные названия не будут отображаться целиком.

Также в заголовочной части HTML документов рекомендуется располагать функции, разработанные на языках сценариев, например Java-script. Это обеспечит их интерпретацию в первую очередь, так как «разборка» HTML документа осуществляется средствами броузера сверху вниз. Для указания броузеру на то, что далее следует программная часть написанная на языке сценариев осуществляется с помощью конструкции:

```
<SCRIPT LANGUAGE="JavaScript1.2" TYPE="text/javascript">
  <!--
  текст программы на java-script
  //-->
</SCRIPT>
```

Текст самой программы на java-script рекомендуется заключать в контейнер дескриптора комментарий, что позволит в случае, если броузер не поддерживает дескриптор <SCRIPT>, считать текст скрипта - комментарием и не отображать его.

Семантическая диаграмма заголовка HTML документа представлена на рис.3.8.

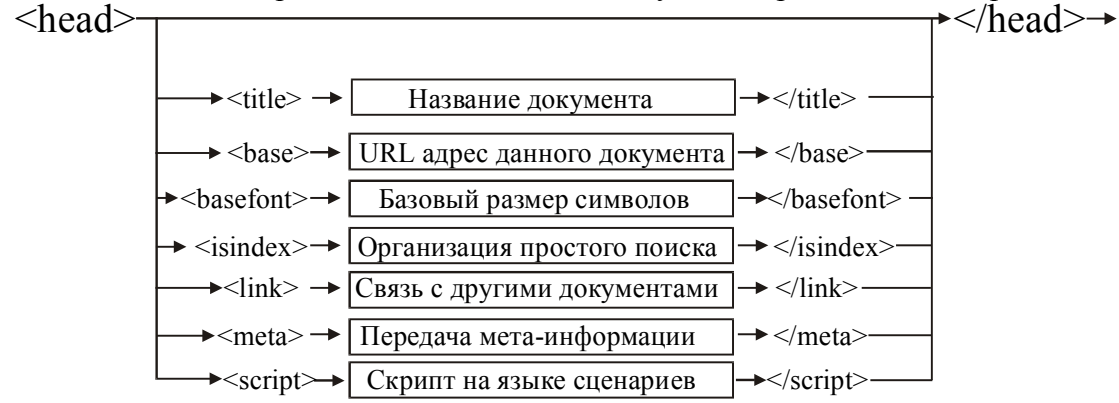

Рис.3.8. Семантическая диаграмма заголовка HTML документа.

Теги meta несут в себе большое функциональное значение, в них можно указывать различную метаинформацию, например: кодировку документа, ключевые слова для поиска, аннотацию документа и т.п. Вся эта информация призвана помочь броузеру или поисковому серверу правильно обработать данный документ.

С помощью дескриптора **<meta>** можно добавлять к странице информацию невидимую для пользователя, но несущую дополнительные сведения об авторе, программном обеспечении, используемой кодировке и т.д. Использование атрибутов этого дескриптора несколько сложнее форматов большинства дескрипторов HTML. Так, например, в одном тэге **<meta>** нельзя использовать атрибуты **name и http-equiv,** в то время как самих тэгов **<meta>** может быть несколько.

Автоматическое обновление

Концепция автоматического обновления предусматривает помещение в заголовок документа дескриптора **<meta>** с параметрами, предписывающими что и как обновлять. То есть атрибут **http-equiv** всегда имеет значение **"refresh", a content** содержит число, определяющее время в секундах через которое надо производить обновление страницы, и LJRL в случае если надо осуществлять автоматический переход на другую страницу. В последнем случае используя этот способ можно организовать последовательный автоматический показ нескольких страниц, указав в заголовке каждой переход на следующую.

Пример:

```
<meta http-equiv="refresh" content=10>
<meta http-equiv="refresh" content="5;url=nextpage.htm">
```

Наибольший интерес представляет возможность влиять на индексацию документа поисковыми системами с использованием мета информации. Раньше, например, была возможность, указывая распространенные в поисковых запросах ключевые слова, повышать рейтинг документа в поисковых системах. Однако, в последнее время, большинство поисковых систем внедряет интеллектуальные методы поиска информации, и если не игнорирует ключевые слова, то, по крайней мере, учитывает только те из них, которые встречаются в тексте самого документа. Использование аргумента **"keywords"** атрибута **name** в дескрипторе **<meta>** позволяет с помощью **content** перечислить ключевые слова, которые будут автоматически извлечены поисковым серверам и помещены в его тематический каталог (поисковый сервер сам, случайным образом выбирая адреса, сканирует сеть и составляет базы данных соответствия адресов и мета-информации). То есть, если в запросе к серверу встретятся какие либо из ключевых слов, перечисленных в заголовке вашей станицы, сервер выдаст в ответ URL всех тех страниц, в заголовках которых эти слова встречаются, среди которых будет и ваша страница.

Дополнительные данные в HTTP заголовке

Сервера HTTP могут использовать название свойства, указываемое атрибутом HTTP-EQUIV, для создания в HTTP-ответе особого заголовка в стиле RFC 822. К примеру, такая возможность может использоваться промежуточных сетевых кэшах при определении момента, когда возникает необходимость в получении новой копии соответствующего документа.

Пример:

```
<HEAD>
    <meta name="author" content="iu4.bmstu.ru">
    <meta name="generator" content="Mozilla/4.08 [en] (Win95;I)
    [Netsacpe]">
    <meta name="Description"
        content="Кафедра Конструирование и производство электронно-
        вычислительных и телекоммуникационных систем: История кафедры
        и МГТУ им. Н.Э. Баумана, Дистанционное компьютерное
        образование, Компьютерная подготовка, сети, телекоммуникации,
        нейронные сети и нейрокомпьютеры, вычислительная техника">
    <meta name="Keywords"
        content="МГТУ, дистанционное, компьютерное, обучение,
        подготовка , Linux , Сети, сетевые технологии,
        микропроцессоры, ЭВМ, нейро, нейрокомпьютеры, курсовик,
        реферат, диплом">
    <title>
        Department of Designing and  Technology of Computers and
        Telecommunications Systems (IU-4)
    </title>

    <!-- функция определения версии броузера
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
             NS4 = (document.layers);
             IE4 = (document.all);
            ver4 = (NS4 || IE4);
            IE5 = (IE4 && navigator.appVersion.indexOf("5.")!=-1);
            isMac = (navigator.appVersion.indexOf("Mac") != -1);
            isMenu = (NS4 || (IE4 && !isMac) || (IE5 && isMac));
        function popUp(){return};
        function popDown(){return};
        if (!ver4) event=null;
    //-->
    </SCRIPT>
</HEAD>
```

Задание 4: Доработать HTML документ из задания 1, включив в него необходимые дескрипторы заголовка HTML документа. Разработанный HTML документ опубликовать в сети интернет, заменив ранее созданный файл index.htm, новой версией документа.

Тело HTML документа

Для задания «тела» HTML документа используется дескриптор **<BODY>.** Семантическая диаграмма дескриптора <BODY> представлена на рисунке 3.9.

Спецификация дескриптора <BODY>		
Дескриптор	**Назначение**	
<body></body>	Тело HTML – документа	
	Атрибут	**Значение**
	bgcolor=color	Цвет фона документа
	bgproperties=fixed	Фиксирование фоновой картинки в IE.
	background=filename	Указание имени графического файла, который будет использован в виде фона документа.
	link=color	Цвет гиперссылки
	alink=color	Цвет активизированной гиперссылки
	vlink=color	Цвет посещенных гиперссылок
	text=color	Цвет текста документа по умолчанию
	topmargin=n	Отступ от верхнего края
	leftmargin=n	Отступ от левого края
	marginheight=n	Отступ между таблицами по высоте
	marginwidth=n	Отступ между таблицами по ширине

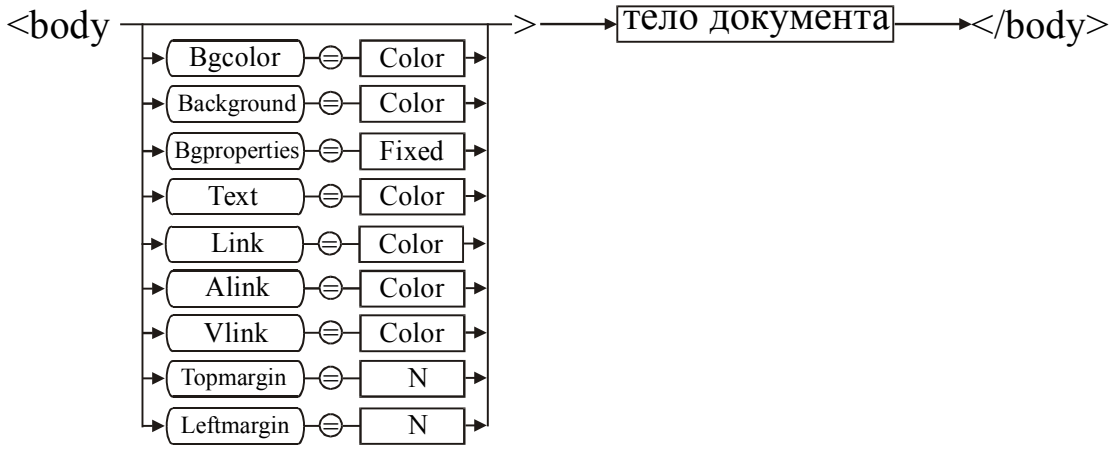

Рис.3.9. Семантическая диаграмма дескриптора <BODY>.

Пример:

```
<body background="fon.gif" bgcolor=#b0c9c0 text="#000000"
    link="#000000" vlink="#000000" alink="#000000"
    marginheight=0 marginwidth=0
    leftmargin=0 topmargin=0>

     тело HTML документа

</body>
```

Данный дескриптор определяет в качестве фона документа цвет с кодом #b0c9c0 (приложение 2), в качестве фонового рисунка используется графический файл fon.gif, текст в документе по умолчанию черный (#000000), цвета ссылок также черные, отступы табличных конструкций по высоте и ширине равны нулю, отступы слева и сверху также равны нулю.

Задание5: Доработать HTML документ из задания 2, включив в него необходимые атрибуты дескриптора <BODY>. Разработанный HTML документ опубликовать в сети интернет, заменив ранее созданный файл index.htm, новой версией документа.

3.3.2. Форматирование текста

При просмотре HTML документов лишние пробелы и пустые строки, имеющиеся в исходном коде, игнорируются. Поэтому, для размещения тексту в рабочем поле документа необходимо оформлять его отдельными частями (заголовками, абзацами и т.п.), иначе весь текст будет отображаться в одну строку, ширина которой будет определяться размерами окна броузера.

3.3.2.1. Заголовки

Стандарт HTML 4.0, как и стандарт HTML 3.2, поддерживает шесть уровней заголовков. Заголовки применяются для создания иерархической структуры документа с главами и подглавами.

Спецификация дескриптора определения заголовка <Hn>		
Дескриптор	**Назначение**	
<Hn></Hn>	Заголовок уровня n, где = 1 … 6	
	Атрибут	**Значение**
	align={left, right, center}	Выравнивание заголовка по горизонтали

3.3.2.2. Управление размещением текста на странице.

Спецификация дескрипторов выравнивания текста		
Дескриптор	**Назначение**	
** **	Перенос на новую строку	
<nobr></nobr>	Неразрывная строка	
<wbr>	Место в котором броузеру можно осуществлять разрыв текста (практически не используется).	
** **	Текстовый абзац	
	Атрибут	**Значение**
	align={left, right, center}	Выравнивание заголовка по горизонтали

*Атрибут align, возможно, не будет включен в последующие версии.

Для выравнивания абзаца по центру кроме атрибута align можно пользоваться дескриптором ** .**

Если необходимо все-таки воспроизвести все пробелы и пустые строки как они есть в исходном тексте надо заключить его в контейнер **<pre> </pre>**. Дескрипторы абзацев внутри такого контейнера игнорируются.

Спецификация дескрипторов преформатирования		
Дескриптор	**Назначение**	
<pre> </pre>	Предварительно отформатированный текст	
	Атрибут	**Значение**
	Width=n	ширина (в пикселях)
<plaintext> </plaintext>	Обычный текст (практически не используется).	

Задание 6: В документ из задания 3, вставить шесть уровней подзаголовков и шесть абзацев текста с различным выравниванием. Разработанный HTML документ опубликовать в сети интернет, заменив ранее созданный файл index.htm, новой версией документа.

3.3.2.3. Форматирование текста с помощью стилей

Физические стили

Под физическими стилями оформления текста в HTML документе понимаются такие стили, которые определяют вид отображения текста, вне зависимости от настроек броузера.

Спецификация дескрипторов физических стилей	
Дескриптор	**Назначение**
** **	Полужирный шрифт
** **	Курсив
<tt> </tt>	Моноширинный шрифт (пишущая машинка)
<u> </u>	Подчеркнутый шрифт
****	Подстрочный текст (нижний индекс)
****	Надстрочный текст (верхний индекс)
<s> </s> или <strike></strike>	Перечеркнутый шрифт
<big>	Увеличенный шрифт
<small>	Уменьшенный шрифт
<blink>	Мерцающий ("моргающий") шрифт (Netscape)

Дескриптор **s/strike**, возможно, не будет поддерживаться в последующих версиях.

Логические стили

Под логическими стилями оформления текста в HTML документе понимаются такие стили, которые определяют отображение текста в зависимости от настроек броузера.

Спецификация дескрипторов логических стилей	
Дескриптор	**Назначение**
** **	Выделенный текст
** **	Сильно выделенный текст
<code> </code>	Фрагмент HTML-кода (моноширинный шрифт)
<listing> </listing>	Фрагмент кода (сохраняет первоначальное форматирование) *Устаревший.* Аналог <pre width=132>
<dfn> </dfn>	Определение
<samp> </samp>	Пример (аналог фрагмента кода)
<xmp> </xmp>	Пример. *Устаревший.* Аналог <pre width=80>
<kbd> </kbd>	Название клавиши клавиатуры
<var> </var>	Переменная или значение
<acronym> </acronym> или <abbr> </abbr>	Аббревиатура (акроним) и её расшифровка

Задание 7: Разработать HTML документ с именем text.htm в котором в столбик разместить содержание полей «назначение» из спецификаций физических и логических стилей, причем к соответствующему «назначению» должны быть применены соответствующие дескрипторы. Разработанный HTML документ опубликовать в сети интернет.

Адреса авторов

Для выделения адреса контактного лица (автора) применяется контейнер **<address>,** содержимое которого в большинстве броузеров отображается курсивом. В нем обычно размещается адрес электронной почты и т.п. контактная информация, размещается в конце страницы.

Спецификация дескриптора указания адреса автора	
Дескриптор	**Назначение**
<address> </address>	Адрес Web-мастера/дизайнера/автора

Дескриптор не делает ссылки, а только выделяет текст определенным стилем.

Цитаты

Согласно стандарту HTML 4.0 дескриптор **<blockquote>** предназначен для длинных цитат, дескриптор **<q>** для коротких. Для выделения цитат курсивом **-<cite>**

Спецификация дескрипторов указания цитат	
Дескриптор	**Назначение**
<blockquote> </blockquote>	Длинная цитата
<q> </q>	Короткая цитата
<cite> </cite>	Цитата, выделенная курсивом

При использовании **<blockquote>** текст разрывается и абзац с цитатой приобретает больший левый и правый отступ, относительно основного текста.

Задание 8: В созданный в примере 5 документ - text.htm, включить указание на адрес разработчика и цитаты. Разработанный HTML документ опубликовать в сети интернет.

3.3.2.4. Форматирование шрифтов

Текст в броузере отображается шрифтом по умолчанию. В свою очередь, шрифт по умолчанию может устанавливаться броузеру с помощью различных конструкций.

Используя дескриптор ****, для фрагмента текста можно дополнительно определить цвет и размер шрифта, а некоторых случаях даже гарнитуру. Размер шрифта можно задавать в относительно размера шрифта по умолчанию (-4 ... +4) (относительно **<basefont>,** значение которого по умолчанию 3) и абсолютных (1 ... 7).

Спецификация дескриптора форматирования шрифта		
Дескриптор	**Назначение**	
** **	Форматирование шрифта для фрагмента текста в контейнере	
	Атрибут	Значение
	Size=n	Размер
	Color=color	Цвет (см. приложение 2)
	Face=”FontName”	Гарнитура
	Lang=cod	Определение языка, код указан в приложении 4.

Задание 9: В созданном в примере 6 документе - text.htm выделить различные участки текста различными цветами. Разработанный HTML документ опубликовать в сети интернет.

Пример:

```
<html><!-- Задание типа документа - т.е. документ html -->
  <head><!-- Заголовок страницы -->
        <title>Пример 5-7</title>
  </head>

<!-- Тело страницы -->
  <body     bgcolor="#FFFFFF" text="#000000" link="#FF0000"
          alink="#FF00FF" vlink="#0000FF"
          marginwidth="0" topmargin="0">

  <center><h1>Страница №5</h1></center>

  <hr color="#FF0000" size=2>

  <center>
   <h1><b>
     <font color="#0000FF">ПРИ</font>
     <font color="#FFFF00">МЕР</font>
     <font color="#00FF00">ОФОР</font>
     <font color="#FF8000">МЛЕ</font>
     <font color="#00ff80">НИЯ</font>
     <font color="#00FFFF">ТЕКС</font>
     <font color="#FF0080">ТА</font>
     <font color="#800080">!</font>
   </h1></b>
  </center>

  <hr color="#FF0000" size=2>

     <font color="#000000"><b>Жирный текст</b></FONT><br>
     <font color="#FF0000"><i>Наклонный текст</i></FONT><br>
     <font color="#00FF00"><u>Подчеркнутый текст</u></FONT><br>
     <font color="#0000FF" size=+1>Текст размера +1</FONT><br>
     <font color="#FF0080" size=+2>Текст размера +2</FONT><br>
     <font color="#000080" size=-1>Текст размера -1</FONT><br>
     <font color="#800000" size=-2>Текст размера -2</FONT><br>

  </body>
</html>
```

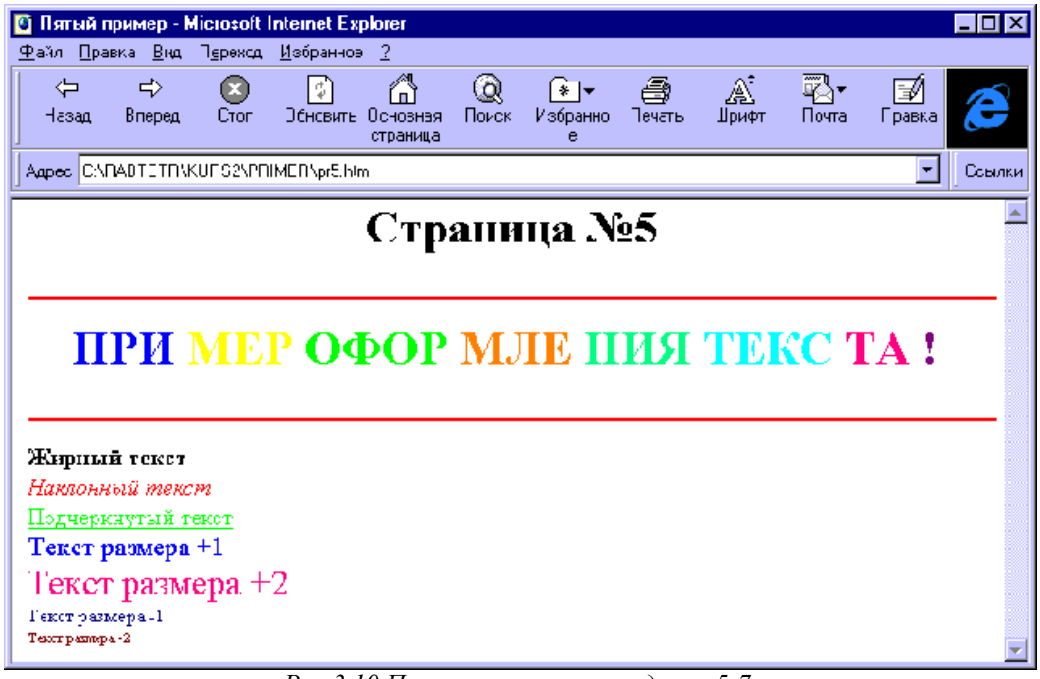

Рис 3.10 Пример выполнения задания 5-7.

Дополнительные атрибуты:

Атрибут	Значение
Dir	Направление прочтения текста: ltr - слева направо rtl - справа налево (поддерживается IE).
title	Присвоение названия элементу.
id	Присвоение имени элементу, что позволяет делать ссылки на данные элементы, например, при использовании JavaScript или VBScript. Значение должно быть уникальным в пределах документа

В отличии от **<id>** <**title**>необязательно должно быть уникальным. Результат использования зависит от броузера (обычно всплывающая подсказка).

Таблица 3.6 Задание основных цветов

белый	FFFFFF	темно-синий	000080
черный	000000	темно-бирюзовый	008080
красный	FF0000	темно-зеленый	008000
желтый	FFFF00	темно-желтый	808000
зеленый	00FF00	темно-красный	800000
синий	0000FF	темно-серый	808080
темно-серый	404040	светло-серый	C0C0C0
оранжевый	FF8000	лиловый	FF00FF
средне-серый	A0A0A4	бирюзовый	00FFFF
салатовый	00FF80	темно-лиловый	800080
розовый	FF0080		

3.3.3 Организация гиперсвязей, понятие навигационных моделей

Ссылка (гиперсвязь) в HTML-документе представляет собой URL-адрес необходимого Internet ресурса, записанный в соответствии с существующими стандартами Internet-адресации.

3.3.3.1. Организация гиперсвязей

URL (Uniform Resource Locator) - унифицированный указатель ресурсов. Состоит из двух основных частей: **Тип протокола (protocol) и Собственно адрес (destination)**

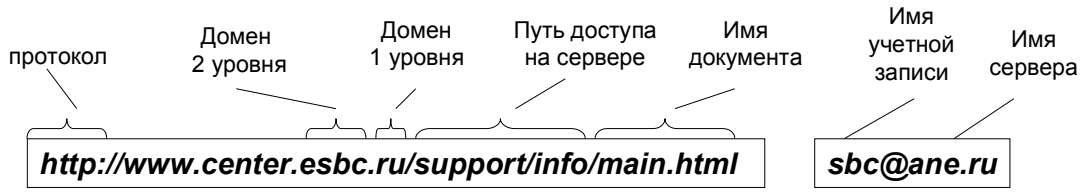

Рис. Адресация интернет ресурсов

Адрес в свою очередь состоит: имя домена n-го уровня (про домены было подробно рассмотрено в разделе 1); каталог (путь); имя файла.

Спецификация типов протоколов интернета	
Название протокола	**Описание**
File	Dummy-протокол (файлы на локальном диске)
Ftp	Передача файлов
http	Всемирная паутина
mailto pop3 imap smtp	Электронная почта: Протокол получения электронной почты. Протокол получения электронной почты с возможностью предварительной обработки «заголовков» почтовых сообщений. Протокол передачи электронной почты.
gopher	Меню-ориентированные ресурсы
news	Группы новостей
nntp	Телеконференции Usenet
telnet	Удаленный доступ

Классификация ссылок:
Все гипертекстовые ссылки можно объединить в следующие группы:

- **Внутренние ссылки (internal links)** - ссылки на объекты в пределах одного документа.
- **Внешние ссылки (internal links)** ссылки - ссылки на другие Web-серверы.
- **Относительные (relative links) или локальные (local links) ссылки** – ссылки на другие Web-страницы (или службы Internet), расположенные на одном сервере со страницей содержащей ссылки.

Для задания гипертекстовой ссылки используется дескриптор **<a>.**

Спецификация дескриптора создания гипертекстовой ссылки		
Дескриптор	**Назначение**	
<a>	Гиперссылка.	
	href=URL	URL целевого документа
	methods = char	Метод обработки связи
	name=char	Локальное имя ссылки
	rel={см. спецификацию}	Связь к целевому документу
	rev={см спецификацию}	Связь от целевого документа
	target=name	Имя целевого фрейма (т.е. фрейма в котором будет открыт указанный в URL
	title=char	Название ссылки

Иногда, бывает необходимо отображения связи между документами без создания реальной ссылки, для этого используется дескриптор <link></link>.

Спецификация дескриптора создания гипертекстовой ссылки		
Дескриптор	**Назначение**	
<link></link>	Гиперссылка.	
	href=URL	URL целевого документа
	methods = char	Метод обработки связи
	name=char	Локальное имя ссылки
	rel={см. спецификацию}	Связь к целевому документу
	rev={см спецификацию}	Связь от целевого документа
	target=name	Имя целевого фрейма (т.е. фрейма в котором будет открыт указанный в URL ресурс).
	title=char	Название ссылки

Спецификация основных значений атрибутов **rel** и **rev**	
Тип связи	**Обычная интерпретация: связь с...**
alternate	Другая версия документа
alternate (с атрибутом lang)	Тот же документ на другом языке
alternate (с атрибутом media)	Версия, подготовленная для другого носителя
bookmark	Закладка в документе
contents	Оглавление
copyright	Информация об авторских правах на данный документ
glossary	Толковый словарь терминов для данного документа
help	Справочный документ
Index	Алфавитный указатель данного документа
next	Документ, следующий сразу за данным документом

previous	Документ, находящийся непосредственно перед данным документом
start	Первый документ в группе
stylesheet	Внешний лист стилей

Внутренние ссылки задаются тегом Пункт 1, вызывающий тег имеет конструкцию: Перейти к пункту 1.

Внешние ссылки задаются тегом Кафедра ИУ-4 с указанием полного URL адреса ресурса или тегом Вернуться назад с указанием имени файла, размещенном на том же сервере, где и расположен файл с ссылкой.

Варианты задания ссылок:

- 1 - открыть файл index.htm расположенный в каталоге primer, расположенном в том же каталоге, что и файл с сылкой, текущего сервера (или локального каталога клиента).
- 1 - открыть файл index.htm расположенный в каталоге более высокого уровня, чем файл где расположена ссылка текущего сервера.
- 1 - открыть файл index.htm расположенный в каталоге более высокого уровня (на два порядка выше), чем файл где расположена ссылка текущего сервера.
- 1 - указание полного URL адреса ресурса.
- vlasov@chat.ru - запуск программы отправки почтового сообщения по e-mail: vlasov@chat.ru.
- 1 - Загрузить архив (вообще то файл может быть любой).
- - Отображает рисунок с именем bmstu.gif, расположенный по указанному URL.
- <img src="ris1.jpg"> - Оформление изображения как ссылки.

Задание 10: С учетом, тегов разделителей, управления ссылками и отображением рисунков модифицируйте страницу из файла пример 5-7 добавив в нее ссылки на предыдущие страницы и вставив изображение. Разработанный HTML документ опубликовать в сети интернет.

Пример:

```
<html>
  <head>
     <title>Пример 8</title>
  </head>

  <body background="fon.jpg" bgcolor="darkgreen"
        text="#000000" link="#ff0000"
        alink="#ff00ff" vlink="#0000ff"
        marginwidth="0" topmargin="0">

  <font>содержание</font><br>
     <a href="#p1">1.Пример размещения изображения</a><br>
     <a href="#p2">2.Оформление рисунка, как ссылки</a><br>
     <a href="#p3">3.Пример оформления ссылок</a><br>
```

```
   <p><!-- абзац -->
   <a name="p1"></a>
     <font>1.Пример размещения изображения</font><br>
     <img src="ris1.gif" alt="рисунок 1, реальный размер">
     <img src="ris1.gif" alt="рисунок 1 размером 200x200" width=200
height=200>
     <img src="ris1.gif" alt="рисунок 1 размером 100x100" width=100
height=100><br>
   <p>
   <a name="p2"></a>


     <font>2.Оформление рисунка как ссылки</font><br>
     <a href="pr1.htm">
          <img src="ris2.gif" alt="рисунок 2 - ссылка на первую
          страницу"><br>
     </a>

   <p>
   <a name="p3"></a>
     <font>3.Пример оформления ссылок</font><br>
     <font>
     <a href="pr1.htm">Ссылка на первую страницу</a><br>
     <a href="pr2.htm">Ссылка на вторую страницу</a><br>
     <a href="http://www.gazprom.ru">Ссылка на сервер ОАО
     Газпром</a><br>
     <a href="http://www.bmstu.ru/">Ссылка на сервер МГТУ им.
     Н.Э.Баумана</a><br>
     </font>

  </body>
</html>
```

3.3.3.2. Навигационные модели

Навигационная модель сайта строится на основе информационной иерархической структуры с учетом интересов целевых групп пользователей и включает в себя описание всех опорных страниц сайта и элементов навигации, которые реализуются посредством «гипертекстовых ссылок» (или иных связей (вызовов) между страницами сайта).

В первую очередь необходимо определить главное меню сайта и меню сервисов. В главное меню сайта включаются ссылки на главную страницу (main page, home page) и основные информационные разделы сайта. Т.е. такие разделы, которые относятся к информационному наполнению сайта. Количество таких разделов должно быть от трех до шести, иначе навигация по сайту будет затруднена. Если у вас имеется большее количество информационных разделов, то объедините схожие по тематике или по группам пользователей разделы в один. Например, новости, пресс-релизы и информацию для представителей СМИ можно объединить в раздел «Пресс-центр». Основной упор при составлении главного меню нужно делать на удобство для различных категорий пользователей: для тех, кому нужна тематически различная информация (например, клиенты и партнеры); для тех, кому нужна информация различного уровня сложности (начинающие и эксперты); а также для тех, кто посещает сайт впервые и для постоянных посетителей.

В меню сервисов включаются разделы, которые не относятся к информационному наполнению сайта, а являются вспомогательными сервисами. Например, поиск по сайту, карта сайта, контактные данные по общим вопросам, форма для обратной связи по общим вопросам и другое. Такие сервисы всегда должны находиться «под рукой» у любого посетителя и на любой странице сайта.

При разработке небольших сайтов имеет смысл визуально объединить главное меню и меню сервисов, но в крупных сайтах эти меню должны визуально различаться. После определения главного меню и меню сервисов имеет смысл начать разбор внутренней структуры и навигации разделов сайта.

Как известно, существует 4 типа веб-навигации:

1. **Плоская** - все страницы имеют общий набор ссылок. При этом с одной страницы можно попасть на любую другую. Применяется при построении верхнего уровня страниц сайта или крупных разделов сайта.

2. **Линейная** - каждая страница имеет ссылки только на следующую и предыдущую страницы. В основном применяется в многостраничных публикациях, где требуется просматривать страницы последовательно, или при последовательном заполнении форм.

3. **Иерархическая** - все последующие страницы ссылаются только на родительскую, но не пересекаются одна с другой. Используется в электронных каталогах, при построении сложных меню.

4. **Смешанная** - сочетает в различных вариантах предыдущие три типа навигации.

Используя все эти типы навигации, нужно помнить о том, что желательным максимальным уровнем вложенности для навигации по сайту являются 4-5 уровней. Это значит, что в идеале посетитель должен увидеть любую страницу сайта всего за 3-4 щелчка мышью. При построении структуры разделов сайта необходимо обращать внимание на ее прозрачность для понимания. Для этого достаточно показать структуру нескольким неспециалистам и попросить их дать свои рекомендации по совершенствованию структуры. После того как первый вариант навигационной модели сайта будет построен, подвергните ее проверке по приведенному ниже тесту:

Рис. 3.11 Навигационные модели сайта.

Задание 11: Разработать навигационную модель сайта и провести ее оценку.

Таблица 3.7 Оценка навигационной модели

Параметр	Значение
Какая информация имеется для создания сайта, и какая информация нужна пользователям?	
Как будет организована обратная связь с посетителями сайта?	
Какие возможности имеются для повышения посещаемости сайта?	
Как часто будут обновляться отдельные разделы сайта? Какими средствами будет обновляться сайт?	
Будет ли сайт удобен для каждой из целевых (тематических) групп пользователей?	
Будет ли сайт удобен для тех, кто заходит на него в первый раз? А для тех, кто заходит на него в двадцатый раз?	
Будет ли сайт удобен для людей с различным уровнем подготовки?	
Будет ли сайт удобен для людей из разных регионов, стран?	

3.3.4. Списки

Познакомившись с основами форматирования текста и организации гиперсвязей, перейдем к вопросу организации перечислений, например, содержание раздела, перечень элементов и т.п. Для отображения данной конструкции в HTML используется понятие списки – т.е. последовательное представление элементов текстовой конструкции, которые определяются двумя конструкциями: определяющей тип списка и вложенной конструкцией, определяющей тип элемента списка. Часто списки используются для отображения элементов в каталогах и меню, которые обычно кратки и не превышают одной строки.

Типы списков делятся на:

- **Упорядоченные** – элементу списка соответствует запись на новой строке, начинающаяся с маркера.
- **Нумерованные** – элементу списка соответствует абзац, начинающийся с номера.
- **Неупорядоченные маркированные** – элементу списка соответствует абзац, начинающийся с маркера.
- **Список определений -** элементу списка соответствует абзац определения, расположенный с отступом.
- **Список элементов меню** –выглядит в виде неупорядоченного списка.
- **Список элементов каталогов** – предназначен для построения иерархического дерева каталогов и файлов, выглядит, как неупорядоченный список.

Элементы списков:

- **Маркерные**
- **Нумерованные.**

Спецификация дескриптора задания маркерных элемента списков		
Дескриптор	**Назначение**	
	Элемент списка	
	Атрибут	**Значение**
	Type	тип вводимого символа элемента списка
	Value=number	текущий номер

Для данного дескриптора закрывающий тэг необязателен, атрибут value задает явным образом номер данного элемента; следующий будет n+1, в списках маркированных римскими цифрами или буквами n соответствующим образом конвертируется (например, 8 соответствует VIII или H).

Спецификация дескриптора задания нумерованного элемента списка		
Дескриптор	**Назначение**	
<o1> </o1>	Нумерованный список	
	Атрибут	**Значение**
	compact	измененный внешний вид
	start=number	начальный номер

type=cod	тип нумерации 1- арабские цифры (по умолчанию) i- строчные римские цифры I- прописные римские цифры a- строчные буквы латинского алфавита A- прописные буквы латинского алфавита

Для задания неупорядоченных маркированных списков используется дескриптор .

Спецификация дескриптора для задания неупорядоченных маркированных списков		
Дескриптор	**Назначение**	
** **	Неупорядоченный список	
	Атрибут	Значение
	Compact	Измененный внешний вид
	Type	Тип вводного символа Disk - Круг Square - Квадрат Circle - Окружность

Спецификация дескриптора задания списка определений.		
Дескриптор	Назначение	
<dl> </dl>	Список определений	
	Атрибут	Значение
	compact	измененный внешний вид

Спецификация дополнительных дескрипторов для списка определений	
Дескриптор	**Назначение**
<dt> </dt>	Определяемый термин в списке определений
<dd> </dd>	Текст определения

* закрывающий тэг необязателен

Спецификация дескриптора задания списка меню		
Дескриптор	**Назначение**	
<menu> </menu>	Список элементов меню	
	Атрибут	**Значение**
	compact	измененный внешний вид
	type	тип вводного символа [Netscape]

Спецификация дескриптора задания списка каталогов		
Дескриптор	**Назначение**	
<dir> </dir>	Каталог	
	Атрибут	**Значение**
	compact	измененный внешний вид
	Type	тип вводного символа [Netscape]

Пример:

```
<html
  <head>
     <title>шестой пример</title>
   </head>
<body bgcolor="#ffffff" text="#000000" link="#ff0000"
      alink="#ff00ff" vlink="#0000ff" marginwidth="0" topmargin="0">

  <center><h1>страница №6</h1></center>

  <hr color="#ff0000" size=2>
  <center><h1><b>
     <font color="#0000ff">при</font>
     <font color="#ffff00">мер</font>
     <font color="#00ff00">офор</font>
     <font color="#ff8000">мле</font>
     <font color="#00ff80">ния</font>
     <font color="#00ffff">спис</font>
     <font color="#ff0080">ков</font>
     <font color="#800080">!</font>
  </h1></b></center>

  <hr color="#ff0000" size=2>

  <h2>неупорядоченный список</h2>
  <ul>
     <li><font color="#000000"><b>жирный текст</b></font><br>
     <li><font color="#ff0000"><i>наклонный текст</i></font><br>
     <li><font color="#00ff00"><u>подчеркнутый текст</u></font><br>
  </ul>

  <hr color="#ff0000" size=2>

  <h2>нумерованный список</h2>
  <ol>
     <li><font color="#000000"><b>жирный текст</b></font><br>
     <li><font color="#ff0000"><i>наклонный текст</i></font><br>
     <li><font color="#00ff00"><u>подчеркнутый текст</u></font><br>
  </ol>

  <hr color="#ff0000" size=2>
```

```
  <h2>меню</h2>
  <menu>
     <li><font color="#000000"><b>жирный текст</b></font><br>
     <li><font color="#ff0000"><i>наклонный текст</i></font><br>
     <li><font color="#00ff00"><u>подчеркнутый текст</u></font><br>
  </menu>

  <hr color="#ff0000" size=2>

  <h2>каталог</h2>
  <dir>
     <li><font color="#000000"><b>жирный текст</b></font><br>
     <li><font color="#ff0000"><i>наклонный текст</i></font><br>
     <li><font color="#00ff00"><u>подчеркнутый текст</u></font><br>
  </dir>

 </body>
</html>
```

Задание 12: С учетом тегов задания списков модифицируйте страницу из файла пример 5-8 изменив способ отображения тестовой информации. Разработанный HTML документ опубликовать в сети интернет.

3.3.5.Таблицы

Методы проектирования HTML документов, рассмотренные выше позволяли нам выстраивать новые элементы на странице сверху вниз, но такое размещение HTML конструкций является не эффективным (***вспомните курс черчения – правильный чертеж, это чертеж, на котором заполнено 80% рабочей площади формата***). Такое упрощенное правило справедливо и для HTML документов – необходимо с максимальной отдачей использовать весь размер рабочего поля документа.

Правила и способы размещения HTML конструкций на рабочем поле документа определяются **КОМПОЗИЦИЕЙ**.

Технологическим инструментом для решения композиционных замыслов являются – **ТАБЛИЦЫ**. Таблицы – это один из самых важных, наряду с гипертекстовыми ссылками, элементов HTML, без них не возможно спроектировать эффективную и эргономичную информационную систему.

Как известно, таблицы, состоят из строк и столбцов, схема описания таблиц средствами HTML представлена на рисунке.

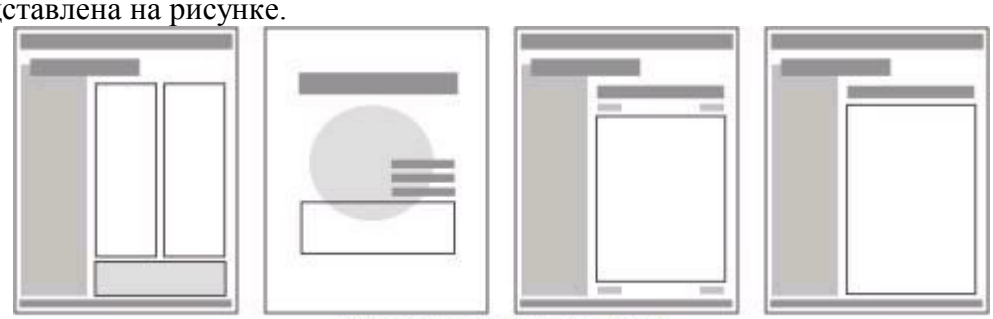

Рис. 3.13 Элементы HTML для представления таблиц.

Необходимо помнить, что в HTML для задания тела таблицы, описания строк и столбцов используются отдельные дескрипторы и если вы по каким-либо причинам забудете закрыть один из них, то в ряде броузеров , созданная конструкция или не будет отображаться вообще или будет отображаться некорректно.

Спецификация дескриптора тела таблицы		
Дескриптор	**Назначение**	
<table></table>	Задание тела таблицы	
	Атрибут	**Значение**
	Align={right|center|left}	Выравнивание по горизонтали
	Bgcolor=color	Цвет фона
	Border=N	Ширина рамки (border=0 – ее отсутствие).
	Bordercolor=color	Цвет рамки
	Bordercolordark=color	Цвет затененной части рельефной рамки (IE)
	Bordercolorlight=color	Цвет освещенной части рельефной рамки (IE)
	Cellpadding=N	Расстояние между содержимым и рамкой ячеек таблицы
	Cellspacing=N	Расстояние между ячейками таблицы
	Width=N (%)	Ширина таблицы
	Height=N (%)	Высота таблицы
	Dir	Направление чтения текста
	Lang	Язык

Спецификация дескриптора задания строк таблицы		
Дескриптор	**Назначение**	
<tr></tr>	Задание тела таблицы	
	Атрибут	**Значение**
	Align={right|center|left}	Выравнивание по горизонтали
	valign={top|bottom}	Выравнивание по вертикали
	Bgcolor=color	Цвет фона
	Border=N	Ширина рамки (border=0 – ее отсутствие).
	Bordercolor=color	Цвет рамки
	Bordercolordark=color	Цвет затененной части рельефной рамки (IE)
	Bordercolorlight=color	Цвет освещенной части рельефной рамки (IE)
	Width=N (%)	Ширина строки таблицы
	Height=N (%)	Высота строки таблицы

Спецификация дескриптора задания столбцов таблицы		
Дескриптор	**Назначение**	
<td></td>	Задание тела таблицы	
	Атрибут	**Значение**
	Align={right|center|left}	Выравнивание по горизонтали
	valign={top|bottom}	Выравнивание по вертикали
	Bgcolor=color	Цвет фона
	Border=N	Ширина рамки (border=0 – ее отсутствие).
	Bordercolor=color	Цвет рамки
	Bordercolordark=color	Цвет затененной части рельефной рамки (IE)
	Bordercolorlight=color	Цвет освещенной части рельефной рамки (IE)
	Width=N (%)	Ширина строки таблицы
	Height=N (%)	Высота строки таблицы
	Colspan=N	Ширина ячейки, выраженная в столбцах
	Nowrap	Выключение разрыва строк
	Rowspan=N	Высота ячейки, выраженная в строках

Спецификация дескриптора задания названия таблицы		
Дескриптор	**Назначение**	
<caption> </caption>	Элемент списка/меню в форме.	
	Атрибут	**Значение**
	Align={right|center|left}	Выравнивание по горизонтали
	valign={top|bottom}	Выравнивание по вертикали

Задание 13: С учетом тегов описания таблиц провести компоновку примеров 7-8, модифицировав страницы по критерию максимального увеличения плотность компоновки и размещения информации.

Пример:

```
<html>
  <head>
     <title>пример</title>
  </head>

<body bgcolor="#ffffff" text="#000000" link="#ff0000"
      alink="#ff00ff" vlink="#0000ff"
      marginwidth="0" topmargin="0">

<!--создаем таблицу с размером во весь экран броузера с прозрачными
границами-->
  <table width=100%>
   <tr><!-- определяем первую строку таблицы-->
     <td width=40%><!-- определяем первый столбец таблицы -->
      <center> <img src="gerb.gif" alt="рисунок">
      </center>
     </td>
     <td align="center" width=60%>
        <!-- определяем второй столбец таблицы-->
        <h1>страница №6</h1>
     </td>
   </tr>
  </table>

<!--создаем таблицу с размером во весь экран броузера с прозрачными
границами-->
  <table width=100% border=3>
   <tr><!-- определяем первую строку таблицы-->
     <td bgcolor="#80ff00" width=50>
      <h2>неупорядоченный список</h2>
      <ul>
       <li><font color="#000000"><b>жирный текст</b></font>
       <li><font color="#0000ff"><i>наклонный текст</i></font>
       <li><font color="#ff0000"><u>подчеркнутый текст</u></font>
      </ul>
     </td>
     <td bgcolor="#ff0080" width=50%>
      <h2>нумерованный список</h2>
      <ol>
       <li><font color="#000000"><b>жирный текст</b></font>
       <li><font color="#0000ff"><i>наклонный текст</i></font>
       <li><font color="#00ff00"><u>подчеркнутый текст</u></font>
      </ol>
     </td>
   </tr>
```

```
   <tr><!-- определяем вторую строку таблицы-->
     <td bgcolor="#00ffff" width=50%>
      <h2>меню</h2>
      <menu>
       <li><font color="#000000"><b>жирный текст</b></font>
       <li><font color="#0000ff"><i>наклонный текст</i></font>
       <li><font color="#ff0000"><u>подчеркнутый текст</u></font>
      </menu>
     </td>
     <td bgcolor="#00ff80" width=50%>
      <h2>каталог</h2>
      <dir>
       <li><font color="#000000"><b>жирный текст</b></font>
       <li><font color="#0000ff"><i>наклонный текст</i></font>
       <li><font color="#ff0000"><u>подчеркнутый текст</u></font>
      </dir>
     </td>
    </tr>
   </table>
 </body>
</html>
```

Таблица 3.8 Основа современного метода табличной компоновки HTML документов

единство, сплоченность			Дробление, разделение, фрагментация
равновесие, баланс			Неустойчивость, неуравновешенность
симметрия			ассиметрия
регулярность			иррегулярность
предсказуемость			спонтанность
активность			Статичность, застой
Острота, тонкость, утонченность			Цельность, основательность
нейтральность			Акцентированность
плоскостность			Глубина
Простота, умеренность			Сложность, запутанность (навороченность)

Задание 13: Реализовать шаблоны для проектируемой информационной системы с использованием табличного метода компоновки в зависимости от выбранного композиционного решения, представленных в таблице.

3.3.6.Формы

Формы превращают Web-страницы в интерактивную среду. Представляют собой простейший способ организации внутри HTML-документа обратной связи между пользователем и сервером. Упрощенно формы можно понимать как набор кнопок, флажков, полей ввода, передаваемых сценарию, в качестве входной информации для обработки. Обработка, принятой сервером информации, ничего общего с HTML не имеет и может выполняться самыми разными средствами.

Спецификация дескриптора задания форм		
Дескриптор	**Назначение**	
<form> </form>	Формуляр	
	Атрибут	**Значение**
	Action	URL-адрес для отправки заполненного формуляра
	Enctype	кодирование передаваемых данных
	Method	способ передачи формуляра

Типы внутренних конструкций в формах:
- Поля ввода объектов (типы объектов определяются значением атрибута type).
- Поля ввода многострочных текстов.
- Выпадающие меню.
- Поля списков.

3.3.6.1. Поля ввода.

Спецификация дескриптора задания форм		
Дескриптор	**Назначение**	
<input> </input>	Поле для ввода строки	
	Атрибут	**Значение**
	Action=URL	URL адрес для отправки заполненной формы
	Align=	
	Bottom	выравнивает нижний край кнопки по базовой линии строки
	Left	выравнивает кнопку-иллюстрацию по левому краю текста
	Middle	центрирует кнопку-иллюстрацию в текстовой строке
	Right	выравнивает кнопку-иллюстрацию по правому краю текста
	top	выравнивает верхний край кнопки-иллюстрации по верхней линии строки текста
	Checked	Установленный флажок или выбранное положение переключателя
	Maxlenght	Максимальная длина вводимых текстов
	Name	Название элемента ввода/управления
	Src	Источник графического файла картинки кнопки

Type =	Тип элемента управления /ввода
checkbox	флажки, независимы друг от друга, их можно установить/сбросить в любой комбинации
file	поле ввода для имени файла, рядом отображается кнопка, "Пролистать/Browse" открывающая стандартное диалоговое окно выбора файла
image	пользовательская командная кнопка вместо стандартных, получаемых с помощью type=submit или type=reset
hidden	параметры передаваемые на сервер, которые не могут быть изменены пользователем
password	текстовое поле, вводимые данные отображаются "звездочками"
radio	селекторные кнопки (переключатели), из группы можно выбрать только одну
reset	командная кнопка, возвращает формуляр к исходному состоянию; данные не
submit	пересылаются командная кнопка, отправляет на сервер всё внесенное в формуляр
text	однострочное текстовое поле
Value	Установленное по умолчанию значение

3.3.6.2. Поля ввода многострочных текстов

Спецификация дескриптора ввода многострочного текстового поля		
Дескриптор	**Назначение**	
<textarea> </textarea>	Многострочное поле для ввода текста в форме	
	Атрибут	**Значение**
	Cols=N	Количество символов /столбцов в поле ввода
	Name=char	Имя поля ввода
	Rows=N	Количество строк поля ввода
	Wrap=	
	off	верстка не выполняется (сервер получает текст одним куском)
	physical	автоматическая верстка, с переносом строк по мере достижения правого края (сервер получает текст с разрывами строк)
	virtual	автоматическая верстка, с переносом строк по мере достижения правого края (сервер получает текст одной строкой - без разрывов)

3.3.6.3. Формирование выпадающих меню и полей списков

Спецификация дескриптора меню или поля списка в форме		
Дескриптор	**Назначение**	
<select> </select>	Меню или поле списка в форме	
	Атрибут	**Значение**
	Multiple	Возможность выбора нескольких опций
	Name=char	Имя элемента
	Size=N	Количество одновременно отображаемых

Спецификация дескриптора задания элемента списка/меню в форме		
Дескриптор	**Назначение**	
<option> </option>	Элемент списка/меню в форме.	
	Атрибут	**Значение**
	Selection	Выбран по умолчанию
	Value	Параметры элемента

Задание 14: Реализовать форму для регистрации пользователей на сайте для подписки на рассылку новостей, параметры формы отправлять по e-mail используя «отрытые обработчики» (представляемые службами бесплатного хостинга).

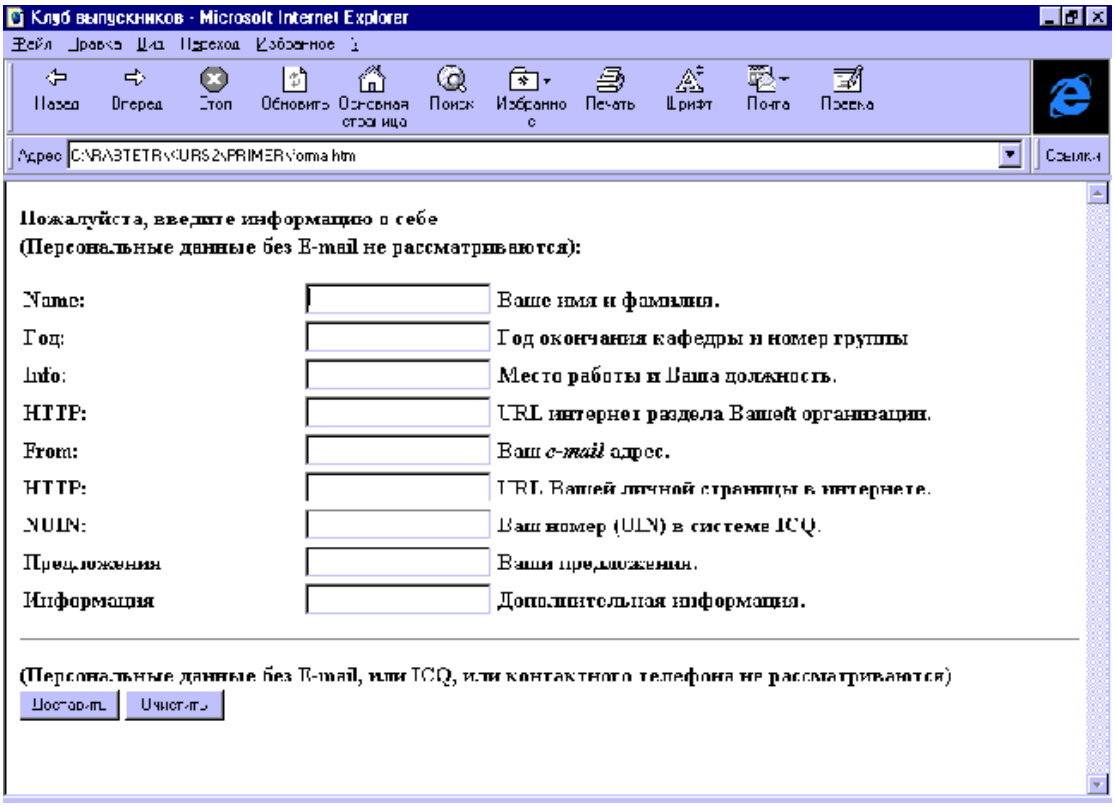

Рис. 3.13 Форма регистрации пользователя.

Пример:

```
<html>
  <head>
     <title>Регистрация пользователя</title>
  </head>

  <body background="TEXTURE.GIF" bgcolor="#FFFFFF">

     <form ENCTYPE="x-www-form-encoded" method="POST"
          action="http://www.cityline.ru/cgi-bin/formmail.ru"
          METHOD="POST">

     <input type="hidden"
          name="recipient"
          value="neurnews@cityline.ru">
     <input type="hidden"
          name="subject"
          value="klub IU4">
     <input type="hidden" name="redirect"
          value="http://iu4.bmstu.ru/">
     <input type="hidden"
          name="env_report"
          value="REMOTE_HOST,HTTP_USER_AGENT">
     <input type="hidden"
          name="print_config"
          value="email,subject">
     <input type="hidden"
          name="return_link_url"
          value="http://iu4.bmstu.ru/”>
     <input type="hidden"
          name="missing_fields_redirect"
          value="http://iu4.bmstu.ru/">

     <p><strong><font>
       Пожалуйста, введите информацию о себе
       <br>(Персональные данные без E-mail не рассматриваются):
     </font></strong></p>

     <table width="100%">
      <tr>
       <td><strong>Name: </strong></td>
       <td>
          <strong><input NAME="Name" SIZE="23">
          Ваше имя и фамилия.<br>
          </strong>
       </td>
      </tr>
      <tr>
       <td><strong>Год: </strong></td>
       <td><strong><input NAME="GOD EDN" SIZE="23">ъ
                Год окончания кафедры и номер группы<br>
          </strong></td>
      </tr>
```

```
      <tr>
       <td><strong>Info: </strong></td>
       <td><strong><input NAME="RABOTA" SIZE="23">
          Место работы и Ваша должность.<br>
          </strong>
       </td>
      </tr>
      <tr>
       <td><strong>HTTP: </strong></td>
       <td><strong><input NAME="URLORG" SIZE="23">
          URL интернет раздела Вашей организации.<br>
          </strong>
       </td>
      </tr>
      <tr>
       <td><strong>From: </strong></td>
       <td><strong><input NAME="E-mail" SIZE="23">
          Ваш <i>e-mail</i> адрес.<br>
          </strong>
       </td>
      </tr>
******************остальной скрипт аналогично примеру***************
  </table>
  <hr>
  <strong>
     (Персональные данные без E-mail, или ICQ,
     или контактного телефона не рассматриваются)<br>

     <input TYPE="submit" VALUE="Доставить">
     <input TYPE="reset" VALUE="Очистить">
  </strong>
</form>

</body>
</html>
```

3.3.7 Фреймы

Лет десять назад информационных ресурсов, использующих фреймовые модели компоновки было достаточно много, но со временем на смену фреймовым моделям компоновки приходит более эффективный табличный метод. Однако, так как стандарт HTML все еще поддерживает фреймы, то и на них мы не много остановимся, хотя сразу хотим заметить, что не рекомендуем их использовать при разработке реальных систем.

Технология фреймов позволяет разделить окно броузера на произвольное число подокон, каждое из которых отображает отдельную HTML-страницу. Установив связи между этими подокнами можно создать общую композицию из отдельных логических единиц, «удобную», с точки зрения навигации, но совершенно не оптимальную с точки зрения поисковых систем и времени загрузки документов (так как приходится обращаться не к одной, а к нескольким страницам).

Чтобы получить страницу с фреймами, нужно, во-первых, создать документ определяющий общую структуру страницы (Frameset document), а во-вторых те документа, которые будут помещаться в окна фреймов.

Спецификация дескриптора задания тела фрейма		
Дескриптор	**Назначение**	
<frameset> </frameset>	Определение структуры фреймов	
	Атрибут	**Значение**
	Cols=N	количество и ширина столбцов
	Rows=N	количество и высота строк
	Frameborder={YES|NO}	показать/скрыть границы между фреймами.
	Border=N	ширина границы в пикселах
	Bordercolor=color	цвет границы

* Контейнет **<frameset>** полностью заменяет в структурном документе пару **<body> </body>.** Дескриптор **<body>** можно использовать внутри элемента (находящегося внутри контейнера **<noframes>)** для указания того, что покажет броузер не поддерживающий фреймы.

В отдельном дескрипторе **<frameset>** может использоваться только один из атрибутов- **rows** или **cols.** Поэтому, чтобы создать строки внутри столбцов или наоборот надо использовать вложенные <frameset>.

Спецификация дескриптора задания конструкции для броузеров не поддерживающих фреймы	
Дескриптор	**Назначение**
<noframes> </noframes>	Определение конструкции для броузеров, не поддерживающих фреймы

Спецификация дескриптора задания тела фрейма		
Дескриптор	**Назначение**	
<frame>	Определение фрейма	
	Атрибут	**Значение**
	Marginheight=N	Отступ от верхнего/нижнего края
	Marginwidth=N	Отступ от правого/левого края
	Name=char	Название фрейма
	Noresize	Запрет изменения размеров фрейма пользователей
	Scrolling={YES|NO|AUTO}	Показать/убрать полосы прокрутки
	Src=URL	Исходный URL страницы для данного фрейма
	Border=N	Ширина границы в пикселах
	Bordercolor=color	Цвет границы
	Frameborder={YES|NO}	Показать/скрыть границы между фреймами

Пример:

```
<frameset cols="200,*">
   <frameset rows="50%,50%">
      <frame src="col1ln1.htm" name="logo">
      <frame src="col1ln2.htm" name="list">
   </frameset>
   <frameset rows="20%,*">
          <frame src="col2ln1.htm" name="head">
          <frame src="col2ln2.htm" name="main">
   </frameset>
   <noframes>
        Для просмотра страницы ваш броузер должен поддерживать
фреймы.
   </noframes>
</frameset>
```

Если нигде отдельно не оговорено ссылки со страницы, помещенной в фрейм, будут отображаться в том же фрейме. Суть навигационной модели состоит в том, что когда вы выбираете что-то из списка в одном фрейме, а подробное описание этого пункта открывается в другом. Что бы этого добиться, надо присвоить всем фреймам имена, а при организации ссылок указывать в качестве **target** указывать имя целевого фрейма.

Аргументы атрибута **target -** зарезервированные имена	
Аргумент	Значение
_blank	документ определенный в URL данной ссылки загружается в новом окне броузера
_self	используется для отмены target определенного в <base>, документ загружается в текущее окно
_parent	документ загружается в текущее родительское окно, которое описано в определении <frameset> непосредственно перед текущим (если родительских нет, то эквивалентно _self)

_top	документ загружается в самый верхний фрейм текущего окна броузера

При построении списка, каждый элемент которого ссылается *на* другой фрейм, желательно атрибут **target** указывать на в каждом дескрипторе <a>, а один раз в дескрипторе **<base>** контейнера **<head>.**

Плавающие фреймы

Начиная с версии HTML 3.2. появились новые возможности компоновки фреймовых конструкций, примером которых являются плавающие фреймы.

Спецификация дескриптора задания плавающего фрейма		
Дескриптор	**Назначение**	
<frame>	Определение фрейма	
	Атрибут	**Значение**
	Width=N	Ширина в пикселах
	Height=N	Высота в пикселах
	Marginheight=N	Отступ от верхнего/нижнего края
	Marginwidth=N	Отступ от правого/левого края
	Name=char	Название фрейма
	Align={left|center|right}	Выравнивание
	Scrolling={YES|NO}	Показать/убрать полосы прокрутки
	Src=URL	Исходный URL страницы для данного фрейма
	Frameborder={YES|NO}	Показать/скрыть границы между фреймами

3.3.8 Таблицы стилей

Таблицы стилей были введены в стандарт HTML 4.0 для более точного форматирования WEB документов и поддержки их стилистического единообразия. С их помощью можно определять гарнитуры и размеры шрифтов, расстояние между символами, и т.д.

Таблицы стилей основаны на рекомендациях разработанных W3C CSS (Cascading Style Sheets) CSS1 и CSS2, представляют из себя особый код, помещаемый в начале HTML-файла или в отдельном текстовом файле, содержащий некоторую информацию о стилях.

Пример:

```
<head>
  <title>Название документа</title>
  <styletype="Tun MIME">
          HTML__дескриптор.класс{свойство1: значение1;
                                 свойствоn: значениеn}
        <span> {специальные атрибуты форматирования}
   </style>
</head>
```

Таким образом, определив стили, далее в HTML-документе, включая в дескрипторы HTML атрибут **class** вызываем, описанный выше, стиль для придания данному элементу определенных стилем свойств.

Пример:

```
<html>
<head>
  <title>Пример использования стилей</table>
  <style type="text/css">
      S.green_caps {color: green; font-style: small-caps}
      S.yellow_helv {font-family: Helvetica, sans-serif; color:
yellow}
  </style>
</head>
<body>
      <s class="green_caps">Красный заголовок</p>
      <ol class="yellow_helv">
           <li>Желтый элемент 1
           <li>Желтый элемент 2
      </ol>
   </body>
</html>
```

**Использование дескриптора **

Дескриптор **** позволяет применить к определенным элементам атрибуты форматирования текста. Для этого необходимо определить стиль в заголовке документа **<head>,** а затем применить его к конкретному элементу в **<body>.**

Дескриптор	Назначение
	Применение заранее определенного стиля

Пример:

```
<html>
<head>
    <title>Пример использования дескриптора span</title>
    <style type ="text/css">
         span {font: 62 pt Courier; color: red}
    </style>
</head>
<body>
        <p><span>Пример использования дескриптора span</span>
                  в документе</P>
</body>
</html>
```

Спецификация свойств стилей согласно сертификации CSS		
Свойство	Значение	Пример
font-family	Название шрифта	Arial, Serif, Symbol
font-size	Число/величина в процентах	12pt,+l, 120%
font-weight	Число/уровень	+1, light, medium, extra bold
font-style	Название стиля	italic, normal
font	Сочетание всех возможностей стилей в применении к шрифту	12 pt Serif medium small caps
color	Слово/шестнадцатеричное число	red, green, blue, FFOOFF
background	Цвет/оттенок/имя файла	bgpaper.gif, red, black/white
word-spacing	Число + единицы измерения	Ipt, 4em, lin
text-spacing	Число + единицы измерения	3pt,0.1em,+l
text-decoration	Слово	underline, line through, box, blink
vertical-align	Слово/величина в процентах	baseline, sup, sub, middle, 50%
text-align	Слово	left, right, center, justify
text-indent	Число/величина в процентах	lin,5%3em
margin	Число	0.5in, 2em
list-style	Слово/URL	disc, circle, square, lower alpha
white-space	Pre/Normal	pre, normal

Замена стилей

Можно в **<head>** определить только пустой дескриптор <style> , без присвоения какого либо значения, а в элементах **<body>** использовать атрибут **style** для создания замен стилей документа.

Пример:

```
<html>
<head>
      <title>Замена стилей</title>
      <style></style>
</head>
<body>
     <p style="text-align: сеп1ег">Абзац по центру</p>
      <em style="color: red">Красный абзац</ет>
     <ol style="list-style: lower roman">
         <li> Каждый элемент списка
         <li> нумеруется
         <li> строчными римскими цифрами
     </o1>
</body>
</html>
```

**Использование дескриптора **

В таблицах стилей используется дескриптор разделения частей документа, с помощью которого можно присваивать различные атрибуты конкретным частям заключенным в контейнер . В контейнер можно добавлять другие дескрипторы, за исключение <head> и <body>, а также дескрипторов связанных с фреймами (кроме <iframe>).

Использование дескриптора <link>

Для придания единого стиля всем страницам сайта можно не определять стили каждый раз в заголовке документа, а один раз, определив их в отдельном документе и разместив на остальных страницах связь со страницей в которой содержится контейнер **<style>.**

Пример:

<link title="mystyle" rel=stylesheet href "style, htm" type="text/css">

***** Значение атрибута **title** должно быть заголовком страницы, на которой с помощью дескриптора <style> определен лист стилей.

Пример:

```
Файл: index.htm
<html>
<head>
    <title>Страница использующая внешний файл стилей</title>
    <link href="common.css" rel=stylesheet>
</head>
</html>
```

Файл common.css

```
BODY. back {background: yellow}
H1 {font-style: bold; color: red; text-align: center}
H2.ital {font-style: italic}
H3 {text-align: center}
p.dblue {color: darkblue}
```

4. Эргодизайн информационных систем

Кому-то может показаться странным размещение раздела, посвященного вопросам эргономики проектирования информационных систем после рассмотрения вопросов связанных с графикой. У многих понятия цветовых решений, композиции и компоновки связано с графикой, хотя при проектировании информационных систем это далеко не так. Известны два противоположных мнения об эргономике информационных, в том числе и интернет, систем. Одно из них основывается на преобладании графических решений и конструкций на сайтах, другое совершенно не приемлет графику. Каждый из подходов имеет свои плюсы и минусы, возможны и компромиссные решения, но об этом позже. По нашему мнению, решение о том просматривать сайт с графикой или без нее, должно оставаться за пользователем, т.е. управляться настройками броузера. Но при этом, как с графикой, так и без нее, информационная система должна не терять своих эргономических характеристик и соответственно, цветовые решения, компоновка и композиция должны быть проработаны для обоих возможных вариантов работы с ней пользователя.

В данном разделе мы рассмотрим вопросы проектирования сайта с использованием табличного метода компоновки и отсутствия какой либо графики. В дальнейшем к вопросам эргономичного дизайна мы еще вернемся при рассмотрении методов работы с растровой и векторной графикой.

4.1. Формы взаимодействия с информационными системами

4.1.1. Зрительное восприятие.

Зрительное восприятие обеспечивает возможность воспринимать форму, цвет, яркость и движение. Установлено, что 80% информации человек получает с помощью органов зрения. Глаз среднего человека способен воспринимать электромагнитные излучения длиной волны 300 … 700 нм.

Способность человека воспринимать информацию зрением характеризуется чувствительностью, полем зрения обоих глаз, остротой зрения, аккомодацией, адаптацией, конвергенцией, цветовым восприятием, стробоскопичностью и стереоскопичностью.

Чувствительность глаза зависит от уровня освещенности и яркости, диапазон оценки которой лежит в пределах от 10^{-4} до 10^{8} кд/м^2. Оптимальная яркость фона, обеспечивающая наибольшую разрешающую способность зрения, составляет 10^{4} кд/м^2.

Увеличение углов обзора происходит при поворотах и наклонах головы и туловища. Концентрация внимания снижает угол эффективной видимости до 30 градусов в горизонтальной и вертикальной плоскостях.

Острота зрения или разрешающая способность - свойство глаза обнаруживать малые объекты и различать тонкие детали. Это свойство зрения сильно меняется в зависимости от вида объекта, спектрального состава распределения энергии светового излучения, освещенности фона, контраста между объектом и фоном, продолжительности действия зрительных стимулов и некоторых других факторов. Для движущихся объектов острота зрения зависит от скорости их движения. Остроту зрения выражают в единицах, обратных углу зрения. Порог восприятия минимальных движений в угловых единицах составляет: на свету при наличии неподвижных объектов - 0’21’’ в секунду, в темноте без неподвижных предметов - 1’15’’ … 1’55’’ в секунду.

Аккомодация - процесс фокусировки хрусталика глаза на близкие или далекие предметы. С возрастом хрусталик глаза теряет свою эластичность. Объекты, расположенные на расстоянии 6 м и далее от наблюдателя, находятся для глаза в оптической бесконечности и фокусировка на эти объекты не требует аккомодации.

Адаптация - изменение чувствительности глаза в зависимости от воздействия на него раздражителей. Приспособление глаза к темноте называют темновой адаптацией. При переходе из светлого помещения в темное, через час пребывания в темноте чувствительность глаза

увеличивается в десятки тысяч раз. Для приспособления глаза к темноте требуется приблизительно 20 минут.

Конвергенция - нацеливание глаз на одну точку с помощью совместного действия глазных мышц и хрусталика. Среднее время, необходимое для нацеливания глаз и их фокусировки на новую точку, смещенную на некоторое расстояние, приблизительно лежит в пределах 165 мс. При чтении с печатного листа это время составляет 20 мс, а при чтении с экрана монитора около 40 мс.

Цветовое восприятие глаза заключается в его способности различать цвета по цветовому тону, насыщенности и контрастности с фоном. Нормальное цветовое зрение называют *трихроматическим* (трехцветным), так как любой из 160 различаемых человеком цветовых тонов можно получить в виде смеси трех базовых цветов: красного, синего, зеленого (вспомните, что современные мониторы воспроизводят 16, 256, 16К, 24К и 32К цветов). Диаграмма, поясняющая особенности восприятия цвета

На способность распознавать цвета большое влияние оказывает *контрастность* – разность яркостей объекта и фона. Различают контрастность прямую (объект темнее фона) и обратную (объект ярче фона). Численно контрастность можно оценить по формуле:
$K=[(B_ф - B_0)/B_ф\ 100\%$ *при* $B_ф > B_0$.
где $B_ф$, B_0 - соответственно яркости фона и объекта.

Рекомендуемая зона величины контрастности от 65 до 95%. Оптимальная контрастность от 85 до 100%. Между яркостью и освещенностью существует следующая зависимость: $B = r E / p$, где B - яркость, кд/м^2, r - коэффициент отражения поверхности, E - освещенность, лк.

Стробоскопичность - свойство зрения, обусловленное задержкой в восприятии информации. Критическая частота мельканий (частота кадров), которую еще способен различать глаз, зависит от яркости. Важным для глаза является также соотношение между светлой и темной фазами. Если информация поступает чаще, чем ее порции становятся различимыми, то отдельные ее фрагменты могут не восприниматься. Сильное мерцание изображения утомляет глаза.

Стереоскопичность - свойство зрения, обусловленное возможностью восприятия двух различных изображений, которое в свою очередь определено только одним световым раздражением. Стереоскопичность имеет “порог глубины”, который соответствует бинокулярному параллаксу 5 угл. с. Радиус стереоскопического зрения 1350…26000 м. Данные параметры важно учитывать при разработке электронных 3D систем “виртуальной реальности”.

4.1.2. Обработка информации пользователем

Восприятие информации пользователь осуществляет с помощью доступных ему анализаторов (зрительного, слухового и т.п. ☺), проводит обнаружение объекта восприятия; выделение в объекте отдельных признаков, ознакомление с выделенными признаками и распознавание объекта восприятия. Важным параметром, характеризующим способность человека воспринимать поступающую информацию, является его возможная скорость переработки информации (бит в секунду). Максимальная пропускная способность человека не превышает 40 бит/с, а номинальная пропускная способность составляет 2-6 бит/с (для сравнения, средняя пропускная способность телевизионного канала 3 10^4 бит/с, а сетевого, например ATM, несколько Гигабит).

Пропускная способность человека-оператора связана с темпом (скоростью) поступления информации от вычислительной системы. Низкий темп поступления информации проявляется в падении активности человека-оператора. Высокий темп, наоборот, приводит к резкому росту ошибок и отказу человека оператора от выполнения задачи. Пропускная способность человека оператора зависит также от условий работы и от того, насколько полно соответствуют эти условия психофизиологическим и антропометрическим характеристикам.

При оценке информации и принятии решений время анализа и принятия решений складывается из целого ряда субъективных характеристик, таких как: личностные характеристики “умственных” возможностей человека-оператора, параметры памяти, его опыт и навыки, которые практически не поддаются количественным оценкам.

Таблица 3.9 Характеристики умственной деятельности человека и усредненные оценки

<table>
<tr><td colspan="3">Долговременная память (Long-Term Memory - LTM):</td></tr>
<tr><td colspan="3">d_{LTM} = X,
m_{LTM} = X,
K_{LTM} = семантический</td></tr>
<tr><td colspan="3">Рабочая память (WORKING MEMORY)</td></tr>
<tr><td>Обработка визуальной инф. Visual Image Store (VIS)</td><td>Обработка звуковой инф.
Auditory Image Story (AIS)</td><td>m_{WM} = 3 (2.5 - 4.1) цепочек,
m_{WM} = 7 (5 - 9) цепочек,</td></tr>
<tr><td>d_{VIS} = 200 (70-1000) мс,
m_{VIS} = 17 (7-17) знаков,
K_{VIS} = физический</td><td>d_{AIS} = 1500 (900-3500) мс,
m_{AIS} = 5 (4.4 - 6.2) знаков,
K_{AIS} = физический</td><td>d_{WM} = 7 (5-226) с.
d_{WM} (1 цепочка) = 73 (73-226) с
d_{WM} (1 цепочки = 7 (5-34) с.
K_{WM} = Акустическая или визуальная</td></tr>
<tr><td colspan="2">Нейрообработка t_P = 100 (50-200) мс.</td><td>Познавательная обработка (приобретение знаний):
t_P = 70 (25-170) мс.</td></tr>
<tr><td colspan="2"></td><td>Моторная реакция:
t_P = 100 (30-100) мс.</td></tr>
<tr><td colspan="3">d = время стирания информации; X - индивидуальный показатель; m = емкость памяти; K = характер исполнительного органа.</td></tr>
</table>

Реакция на информацию определяется в основном длительность моторного действия (например, переход к другому ресурсу, прокрутка страницы и т.п.), которая труднее, чем латентный период поддается измерению, так как зависит от многих случайных факторов (места нахождения человека в момент приема информации, его позы, степени усталости и т.п., формы пульта управления, расположения органов управления, уровня образования и опыта работы человека-оператора и т.д.). Большинство данных параметров носят субъективный характер. Их трудно оценить численно, поэтому длительность моторного действия определяется статистически на моделях или макетах тех или иных устройств с участием человека. Время выполнения типовых операций строго регламентировано и определяется соответствующими стандартами, данные в которых устанавливаются путем экспериментальных измерений с дальнейшей статистической обработкой и усреднением.

Задание: Вы поручили своему сотруднику провести обзор интернет ресурсов, например, по автомобильной промышленности. Оцените, какое время он должен потратить, например, для составления обзора по 5 основным производителям.

Многие интерактивные интернет системы основы на использование баз данных, при их проектировании необходимо учитывать также и производительность используемых СУБД. Так, в обычном разговоре люди ожидают ответа (даже если это кивок или невнятное бормотание) около 2 с, и они ждут того же при работе с компьютером. Например, очень показателен пример с организацией дистанционного обучения посредством электронной почты: если в течение дня слушатель, отправивший свой вопрос преподавателю, не получает ответа, то он практически теряет интерес к данной проблеме.

Так как на представления пользователя оказывает сильное влияние его предшествующий опыт работы с системой, важным является преемственный и предсказуемый отклик системы. Пользователи могут обнаружить весьма малые изменения во времени ответа, хотя обычно малые отклонения их не беспокоят. При работе с системой, которая обычно дает ответ в

течение 2 с, ответ задерживается до 10 с, некоторые пользователи начнут нажимать клавиши, чтобы проверить, не вышла ли система из строя.

Диалог с информационной системой накладывает сходные требования, которые во многом определяются уровнем нервно-психологической нагрузки на человека-оператора. Опытный пользователь обычно вводит сложный запрос, включающий несколько связанных подсказок и ответов и завершающийся паузой. Таким образом, если для простого запроса или после завершения сложного запроса допустим ответ поступает через 10 с, то в рамках сложного запроса время реакции системы в 10 с на вопросы совершенно не приемлемо.

Характеристики нервно-психологической нагрузки человека-оператора по вниманию				
№	Интенсивность нагрузки	Количество одновременно наблюдаемых объектов, шт.	Время сосредоточенного наблюдения относительно продолжительности смены, %	Частота сигналов, $ч^{-1}$
1	Легкая	до 5	до 25	до 75
2	Средняя	5-10	25 - 50	75 - 175
3	Тяжелая	10 … 25	50-75	175 - 300
4	Очень тяжелая	больше 25	более 75	более 300
Характеристики нервно-психологической нагрузки человека-оператора по анализаторным функциям.				
№	Интенсивность нагрузки	Размеры объекта наблюдения, мм (учитывается и при установке разрешения на мониторе)	Время пассивного наблюдения относительно продолжительности смены в %	Отношение звукового сигнала и шума , дБ
1	Легкая	до 1	до 75	18 … 15
2	Средняя	1- 0,3	75 - 90	15 … 6
3	Тяжелая	0,3 -0,15	990 - 95	6 … 0
4	Очень тяжелая	менее 0,15	более 95	0 … 5

Если время ответа значительно превышает 2 с, функции, которые человек хочет или может выполнять, изменяются. Человек должен изменить свой ритм работы, как вам пришлось бы изменить привычку набирать телефонный номер, если бы была необходима задержка между набираемыми цифрами.

4.1.3. Состояния пользователя при обработке информации

Мобилизационные возможности человека- оператора - это его способность перейти из состояния пассивного ожидания в состояние активного принятия решений. Мобилизационные возможности каждого человека во многом субъективны и зависят от большого числа факторов, но все же основным является интегральная экстенсивная напряженность деятельности (нагрузка), которая вызывает утомление и соответственно снижение мобилизационных возможностей. Помимо абсолютной величины нагрузки, на степени развития утомления сказывается еще ряд факторов, среди которых необходимо выделить следующие:

- статический или динамический характер нагрузки;
- интенсивность нагрузки, т.е. ее распределение во времени;
- постоянный или ритмический характер нагрузки.

Динамика работоспособности и динамика утомления являются неспецифическими проявлениями организма, общей реакцией на интенсивность и экстенсивность рабочей деятельности, в то время как состояние специфической напряженности зависит то структуры и содержания потока информации.

Состояние адекватной мобилизации - такое состояние оператора, которое является оптимальным или близким к оптимальному для данных условий работы человека, включенного в конкретную систему управления. Симптоматика и выраженность этого состояния зависит, прежде всего, от объема информации, ее плотности и экстенсивности, от семантической значимости информации, характера кодирования, наличия шума, требуемых программ реализации принятой информации и особенностей управляемой системы. Чем больше требуемое состояние отличается от состояния оперативного покоя, тем больше выражена активная мобилизация. Характерной чертой адекватной мобилизации является ее линейность, т.е. наличие прямой зависимости от субъективной трудности выполняемой работы.

Первым шагом диагностики, или прогнозирования этого состояния является количественный анализ информационной модели рабочего процесса для выяснения, какой элемент этой деятельности в первую очередь определяет степень адекватной мобилизации. В большинстве случаев оперативной точкой для суждения служит положение найденных характеристик на шкале предельных возможностей человека.

В *состоянии динамического рассогласования* нарушается основная закономерность предыдущей стадии - уровень работы по восприятию информации не соответствует ожидаемому физиологическому состоянию. О таком состоянии свидетельствуют большие сдвиги вегетативных реакций, появление дополнительных реакций, в частности потоотделения, расширение сосудов кожи, нарушение мышечного баланса и др. Это состояние чрезвычайно важно для оценки работы специалиста, поскольку оно сопровождается выраженными нарушениями работоспособности и появлением большого числа ошибок, лишними действиями, увеличением времени работы, вплоть до отказа от работы или ее прекращения.

В общем случае восприятие и переработка информации зависят как от психологических возможностей человека, так и нервно-психологической напряженности. При большой нервно-психологической напряженности работы особое внимание должно быть обращено на концентрацию внимания, должны быть сведены к минимуму отвлечения оператора и обеспечены наилучшие условия восприятия и переработки информации. При проектировании человеко-машинных систем необходимо учитывать и то, что на точность функционирования всей системы в целом большое влияние оказывает и квалификация оператора, причем влияние ошибок оператора на точность системы сильнее, чем влияние несовершенства конструкции.

Под психологическими характеристиками человека-оператора понимаются состояния, вызванные переживанием человеком, его отношение к внешнему миру и к самому себе и характеризуемые изменениями количественных и качественных параметров реакций на воздействия внешней среды. Психологическое (эмоциональное) состояние тесно связано с индивидуальной семантической значимостью поступающей к человеку информации и являются как бы коррекцией, вносимой человеком в ответ, определяемый только информационной структурой раздражителя.

В тех случаях, когда наступает динамическое рассогласование между объективной значимостью ситуации и ее субъективной оценкой, появляются связанные с этим отрицательные изменения в двигательных и психических функциях, наступает состояние эмоциональной напряженности. При этом наблюдается снижение устойчивости ряда психических функций. Момент перехода эмоционального напряжения в эмоциональную напряженность определяет так называемую эмоциональную устойчивость. Чем меньше эмоциональная устойчивость, тем скорее при меньших значениях эмоционального фактора развивается состояние эмоциональной напряженности. Эмоциональная устойчивость является показателем, очень тесно связанным с таким свойством личности, как уровень тревожности, она очень низка у лиц с высоким уровнем тревожности.

Наряду с этими фактора эмоциональная возбудимость определяет быстроту развития того или иного эмоционального состояния, т.е. это качество очень близкое к тому, которое

характеризует эмоциональную устойчивость. Развитие эмоциональных состояний обусловлены внешними и внутренними группами факторов.

К *внешним эмоциогенным факторам* относятся прежде всего так называемые экстремальные факторы, физические или информационные характеристики которых ведут к развитию крайней степени напряжения физиологических и психологических функций с полным исчерпанием всех физиологических резервов. Чем более выражена экстремальность фактора, тем выше вероятность появления выраженных степеней эмоциональных сдвигов. Характер этих сдвигов определяется видом реакции, развивающейся в результате воздействия. В случае формирования адекватной реакции, т.е. реакции, направленной на преодоление действий фактора или на поддержание необходимого уровня деятельности при продолжении действия экстремальности, как правило, наблюдается та или иная степень эмоционального напряжения.

Внутренние эмоциогенные факторы сами по себе не являются эмоциогенными, они лишь придают тому или иному внешнему фактору необходимую степень эмоциональности. К этим факторам относятся такие как характеристики нервной деятельности, темперамент, уровень тревожности, ригидность личности, и т.п. Они, как правило, определяют уровень реакции.

4.2. Композиция и компоновка интерфейсов информационных систем

4.2.1. Общие положения

К общим требованиям эргодизайна информационных систем (ИС) относят:

- выразительность — способность видом интерфейса (т.е. той части системы, которая непосредственно взаимодействует с пользователем) наглядным образом отображать качество, обеспечивая соответствующее эстетическое восприятие;
- оригинальность — совокупность своеобразных элементов формы и их отношений, дающих возможность отличить данную систему от ряда однотипных. Понятие оригинальности не исключает, а предполагает сохранение определенных признаков формы: национальных, отраслевых, фирменных;
- гармоничность — свойство формы системы быть органично согласованной с элементами формы, что достигается определенными соотношениями яркости, цвета, размеров и расположением различных элементов; требование гармоничности распространяется также на согласованность системы со средствами, которыми она воспроизводится для пользователя;
- требование стилевого единства предъявляют к признакам формы системы, которые отражают исторически сложившиеся социально - экономические и идейно - эстетические принципы, а также художественно-конструкторские методы и средства их воплощения;
- современность стиля — согласованность между общим стилем машины и уровнем развития стиля мира материальной культуры.
- социальность стиля: предельная общественная целесообразность, гуманность, демократизм, общий мажорный тон, чистота, ясность, изящество.

Основные средства создания «формы» информационной системы — организация пространства, масштаб и пропорции объемов, его силуэт и композиция - ритм его «масс» (для информационных систем масса может задаваться различными цветовыми решениями), плоскостей и опор, тектоническая структура, цвет и фактура использованного материала. Этими средствами художественное проектирование выявляет в облике системы ее назначение и принципы конструктивного решения и в то же время воплощает определенный комплекс идей и функциональных характеристик, которые связаны с назначением данной системы.

Одним из основных понятий художественного проектирования информационных систем является *композиция* интерфейса пользователя — построение целостного произведения, элементы которого находятся во взаимосвязи и гармоническим единстве, Основу композиции составляет объемно-пространственная структура интерфейса системы, отвечающая его

назначению и выражаемая в характере взаимосвязи элементов формы системы, взаимном расположении его частей, пропорциях, ритмическом строе элементов формы и т. п.

Композиционной организации формы изделия достигают путем установления взаимосвязи и соподчинения образующих ее элементов. При этом главные и подчиненные элементы, взаимно усиливая друг друга, образуют в целом единство. Если форма изделия неоднородна или размеры его элементов не равны, то композиционное единство формы возникает, если средствами композиции обеспечивается соподчинение одних элементов другим. Если в интерфейсе системы, например, можно выделить три части, то средняя часть из-за своего центрального положения подчиняет себе обе крайние части. Усилить это соподчинение можно путем увеличения средней части. Главный элемент выделяют из соподчиненных элементов размерами, особым расположением элементов навигации, цветом, характером членения.

Рис. 4.1 Варианты комопновочных схем.

4.3. Цветовые решения

Важное средство композиции – **цветовое решение.** Умело применяя те или иные цвета, можно создавать впечатление легкости и тяжести, холода и тепла, простора и тесноты, выступания и отступания элементов и узлов изделия. Изучение цветовых решений в приборо- и машиностроении посвящено большое число работ [8, 16, 25, 31, 33, 37, 38, 39, 40, 41], обобщим, некоторые, наиболее важные с точки зрения конструктора методы и принципы цветообразования.

Цвет необходим для выделения нужных элементов, узлов, деталей или частей изделия и прежде всего опасных в отношении травматизма. Цвет является средством эстетического воздействия, влияя на настроение, поднимая и понижая эмоциональный тонус, вызывая ощущение творческого подъема. Кроме того, цвет способствует образному выражению сущности изделия, обеспечивает его связь с окружающей средой. Для ЭВМ наиболее

целесообразно применение гармонизирующих оттенков одного и того же цвета (например, черный, темно-серый, светло-серый или песочный, бежевый, оливковый и т.п.). Это не исключает там, где это необходимо, контрастирующих цветов, например красного – для сигнализации аварийных режимов работы, зеленого – для сигнализации нормального режима работы и т.п.

В процессе изучения цвета был составлен "цветовой круг". Условно крайние точки этого круга это синий, желтый, красный и зеленый цвета. Выделяют три группы цветов по сочетанию - *родственные, родственно-контрастные* и *контрастные.*

Родственные цвета располагаются в одной четверти цветового круга и имеют хотя бы один общий цвет, например, желтый, оранжевый и желто-красный. Существуют четыре группы родственных цветов - желто-красные, красно-синие, сине-зеленые, зелено-желтые. При этом в сочетании не должно быть одновременно двух контрастных цветов.

Родственно-контрастные цвета располагаются в двух соседних четвертях цветового круга и имеют один общий (главный) цвет, два других составляющих цвета - взаимодополнительные. Существуют четыре группы родственно-контрастных цветов - желто-красные и красно-синие, красно-синие и сине-зеленые, сине-зеленые и зелено-желтые, зелено-желтые и желто-красные. Эти сочетания достаточно активны.

Контрастные цвета располагаются в противоположных четвертях цветового круга. Глаз сразу замечает такое сочетание, поэтому они применяются там, где необходимо привлечь внимание.

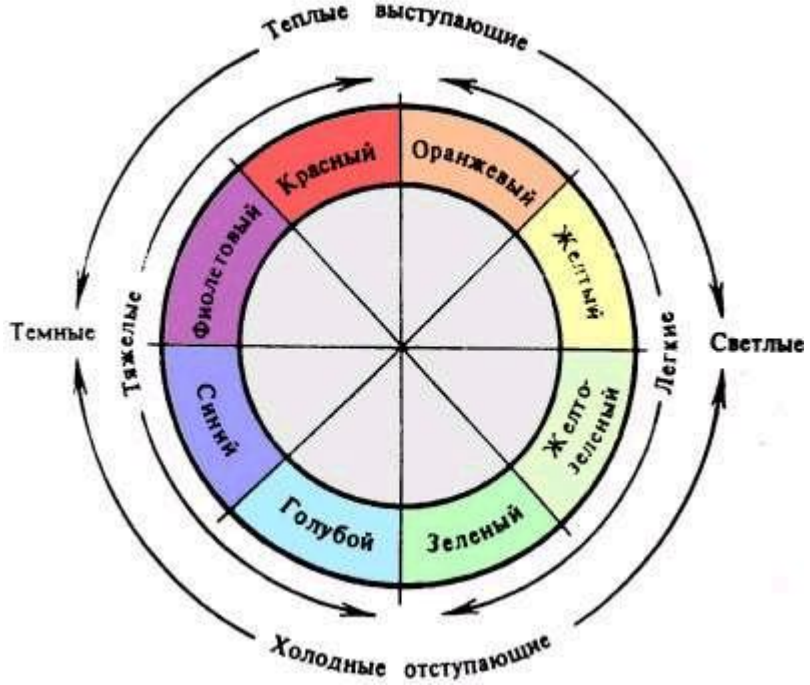

Рис. 4.2 Цветовой круг

Основным фактором обеспечения **цветовой гармонии** является равновесие **цветовых тонов**. Сочетать можно любые цвета по принципу контраста, родственного контраста или родства цветов. Два родственных цвета могут быть уравновешены равным количеством присутствующего в них одного из главных цветов, т.е. в одинаковой мере могут быть желтоваты, зеленоваты и т.д. Гармонизация родственных цветов возможна при условии ослабления насыщенности главных или других цветов либо уменьшения их активности путем затемнения. Например, чтобы родственные цвета (чисто желтый и желто-красный) стали гармоничными, необходимо ослабить насыщенность желтого смесью белого в таком количестве, сколько в желто-красном красного цвета. Гармоничные сочетания родственных

цветов неактивны, спокойны, мягки, особенно если цвета слабо насыщены и сближены по светлоте.

Гармоничные контрастные цвета расположены на концах диаметров цветового круга. Однако не все контрастные цвета гармоничны. Случаи трехцветной гармонии. К двум гармоничным родственно-контрастным цветам может быть прибавлен третий - главный цвет, их роднящий, ослабленной насыщенности. Цвета окажутся попарно родственно-контрастными и попарно дополнительными. Такие сочетания высокогармоничны и колористически богаты.

К двум гармоничным родственным цветам может быть добавлен один контрастный. Так, гармония образуется, если родственные зеленовато-желтый и лиственно-зеленый цвета дополнить красно-синим цветом, т.е. дополнительным промежуточному двух первых.

При проектировании цветового решения ВС (включая и интерфейсную часть прикладного программного обеспечения), следует руководствоваться следующими рекомендациями:

- Контрастные цвета от сочетания друг с другом становятся ярче, заметнее один на другом. Один и тот же красный цвет на зеленом фоне смотрится ярко и четко, на оранжевом приглушенно.
- Теплые цвета приближают предметы, холодные удаляют, при этом предмет, окрашенный в теплые цвета, кажется больше своих размеров, а холодные -меньше.
- На светлом фоне все цвета темнеют, на темном светлеют. Истинная светлота цвета может наблюдаться только на нейтральном фоне средней светлоты. В зависимости от фона ахроматические цвета приобретают кажущуюся цветность. Так, серое пятно на зеленом фоне приобретает розовый оттенок. Хроматические цвета в окружении цветов высокой насыщенности несколько меняют цветовой тон, например, желтый цвет на зеленом фоне становиться слегка оранжевым, а красный в окружении зеленого - более насыщенным.
- Чем больше цвета отличаются один от другого по светлоте, насыщенности и цветовому тону, тем менее они гармонируют друг с другом.
- Существует понятие краевого контраста т.е. равномерно окрашенная поверхность кажется у края светлее или темнее если она граничит с более темной или светлой поверхностью.

Цветовое решение ВС оказывает воздействие на человека-оператора и зависит от количества цвета, качества цвета, время воздействия, особенностей нервной системы, возраста, пола и других факторов.

Непосредственным физиологическим действием на весь организм человека объясняются явления, вызываемые красным и синим цветами, в особенности при максимальной их насыщенности. Красный цвет возбуждает нервную систему, вызывает учащение дыхания и пульса и активизирует работу мускульной системы. Синий цвет оказывает тормозящее действие на нервную систему. Красный, желтый, оранжевые цвета являются цветами экстраверсии, то есть импульса, обращенного наружу. Группа синего, фиолетового, зеленого напротив для пассивной интроверсии и импульсов обращенных внутрь.

Оранжевый и красный цвета, возбуждая попутно со зрительным и слуховой центр мозга, что вызывает кажущееся увеличение громкости шумов. Не лишено основания, что эти активные цвета часто называют "кричащими". Зеленый и синий, успокаивающие цвета, ослабляют возбуждение слухового центра, т.е. как бы ослабляют или компенсируют громкость шумов (табл.).

Табл. 4.1 Характеристики основных цветов.

Цвета	Возбуждающие	Угнетающие	Успокаивающие
Красный	+		
Оранжевый	+		
Желтый	+		
Зеленый			+
Голубой			+
Фиолетовый		+	

Темно-серый		+	
Черный		+	

Желто-коричневый цвет кажется сухим, зеленовато-синий - влажным, розовый - слащавым, красный - теплым, оранжевый - кричащим, фиолетовый - тяжелым, желтый - легким. Это действие цвета нельзя объяснить ассоциациями. Оно вызвано синестезией, т.е. возбуждением одного органа чувств при раздражении другого.

Табл. 4.2 Основные характеристики кажущегося воздействия цветов.

Цвета	"Вес"	"Температур	"Влажность"	"Громкость"
Белый	легкий			
Желтый	легкий	теплый	сухой	
Оранжевый		теплый	сухой	громкий
Красный	тяжелый	теплый	сухой	громкий
Фиолетовый	тяжелый			
Синий	тяжелый	холодный	влажный	тихий
Зеленый		прохладный	влажный	тихий
Голубой	легкий		влажный	тихий
Коричневый	тяжелый	теплый	влажный	
Черный	тяжелый		сухой	

Оптическое воздействие цвета обуславливает появление иллюзии или оптических явлений, изменяющих внешний вид предметов. Рассматривая оптические явления цвета, все цвета можно условно разделить на две группы: красные и синий, т.к. в основном цвета по своим оптическим свойствам будут тяготеть к какой-нибудь из этих групп. Исключение составляет зеленый цвет.

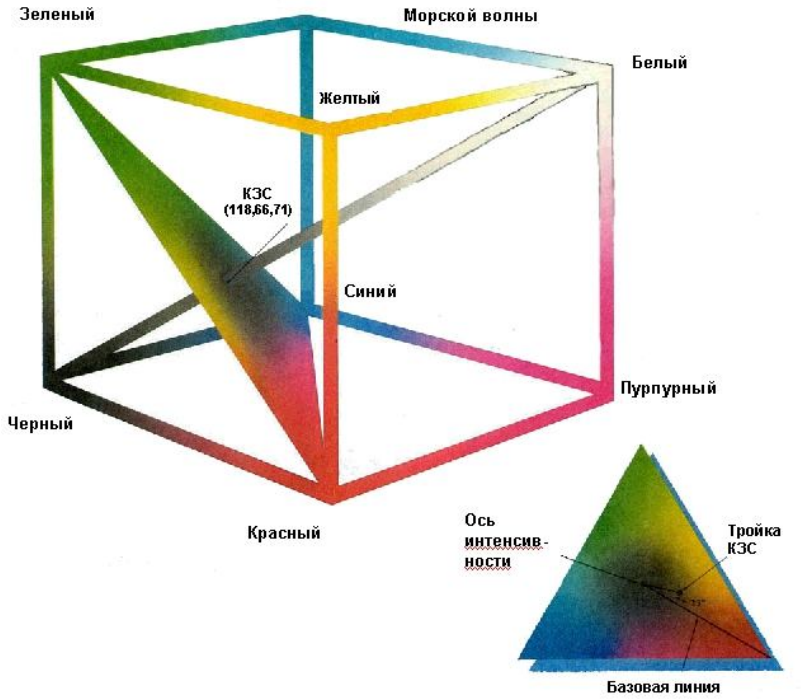

Рис. 4.3. Цветовой и световой колорит.

Светлые цвета, например белый или желтый создают эффект иррадации, они как бы распространяются на расположенные рядом с ними более темные цвета и уменьшают

окрашенные в эти цвета поверхности. Для примера, если через щель дощатой стены проникает луч света, то щель кажется шире, чем в действительности. Когда солнце светит сквозь ветви деревьев, ветви эти кажутся более тонкими, чем обычно.

Это явление играют существенную роль при конструировании шрифтов. В то время, как, например, буквы E и F сохраняют свою полную высоту, высота таких букв как O и G, несколько уменьшаются, еще больше уменьшаются из-за острых окончаний буквы A и V. Эти буквы кажутся ниже общей высоты строки. Чтобы они казались одинаковой высоты с остальными буквами строки, их уже при разметке выносят несколько вверх или вниз за приделы строки. Эффектом иррадации объясняется и различное впечатление от поверхностей, покрытых поперечными или продольными полосками.

Желтый цвет зрительно как бы приподнимает поверхность. Она кажется к тому же более обширной из за эффекта иррадации. Красный цвет приближается к нам, голубой, наоборот удаляется. Плоскости, окрашенные в темно-синий, фиолетовый и черный цвета, зрительно уменьшаются и устремляются книзу. Зеленый цвет - наиболее спокойный из всех цветов

Так же нужно отметить центробежное движение желтого цвета и центростремительное синего. Если сделать два круга равной величины и заполнить один желтым, а другой синим, то уже после короткой концентрации на них становиться заметным, что желтое лучеиспускает, приобретает движение из центра и почти осязаемо, приближается к человеку. Синее же развивает центростремительное движение (подобно стягивающей себя в свой домик улитке) и удаляется от человека.

Это воздействие увеличивается, если к нему добавить различие в светлоте и темноте, т.е. воздействие желтого увеличится при добавлении к нему белого цвета, синего - при утемнении его черным.

Эмоциональное воздействие цвета обуславливает воздействие на чувства переживаний, которые можно испытывать под влиянием того или иного цвета. Это влияние очень тесно связанно с оптическими свойствами цвета.

Абсолютно зеленое есть самый спокойный цвет. Он никуда не движется и не имеет ни призвука, ни радости, ни печали. Это постоянное отсутствие движения благотворно действует на утомленных людей, но может и прискучить со временем.

При введении в зеленый цвет желтого цвета он оживляется, становится более активным. При добавлении синего, наоборот, начинает звучать иначе, он делается более серьезным, вдумчивым.

С другой стороны, желтый цвет беспокоит человека, колет его, возбуждает. Сравненное с состоянием человеческой души, его можно было бы употребить как красочное выражение безумия, слепого бешенства (желтый цвет Достоевского).

Синее же склонно к углублению. Чем глубже, темнее становится синий цвет тем больше он зовет человека к бесконечному, будит в нем голод к чистоте и сверхчувственному. Очень темное синее дает элемент покоя. Доведенный до приделов черного синий цвет получает призвук печали. Становясь более светлым синий приобретает равнодушный характер и становится человеку далеким и безразличным, как голубое небо. И становясь светлее все более беззвучный, пока не дойдет до беззвучного покоя - станет белым.

Часто белый цвет определяется как "некраска". Он есть как бы символ мира где исчезают все краски, все материальные свойства. Поэтому и действует белый цвет на нашу психику как молчание. Но это молчание как бы не мертвое, а наоборот полное возможностей.
Черный цвет, наоборот, воздействует как нечто без возможностей, как мертвое пятно, как молчание без будущего.

Равновесии белого и черного рождает серое, естественно серый цвет не может дать ни движения, ни звука. Серое - беззвучно и бездвижно, но эта неподвижность другого характера чем у зеленого цвета, рожденного двумя активными цветами - желтым и синим. Поэтому серый цвет - это безутешная неподвижность.

Красный цвет, мы воспринимаем как характерно теплый цвет, воздействует внутренне как жизненный, живой, беспокойный цвет не имеющий, однако, легкомыслия желтого. В отличие от желтого красный цвет как бы пылает внутри себя. Но идеально красный цвет очень

сильно меняет своё влияние при изменении цвета. При добавлении в красный цвет черного возникает тупое, жесткое, не способное к движению коричневое. В более холодном оттенке красного пропадает активность пламени. Становясь оранжевым красное приобретает лучеиспускание желтого, но постоянно сохраняет серьезность

Фиолетовый цвет - это как бы охлажденный красный, поэтому он звучит несколько болезненно, как нечто погашенное и печальное.

Выбор предпочтительно (любимого) цвета человеком определяется его характером и зависит, также от социального фактора. На основании социологических исследований был получен следующий ряд цветов по мере уменьшения предпочтительности: голубой - фиолетовый - белый - розовый -пурпурный - красный - зеленый - желтый - оранжевый - коричневый - черный.

Психологическое воздействие на человека оказывают ни только отдельные цвета, но и цветовые сочетания. И здесь очень большое значение имеет расположение цветов в пространстве. Психофизиологическое воздействие цвета в значительной степени зависит от большей или меньшей насыщенности цвета, размера цветового пятна, расстояния и направления откуда воздействует цвет. Цвет расположенный по вертикали воспринимается легким, диагональ - динамика, горизонталь - устойчивость. Напряжение цвета внизу - композиция естественная и устойчивая. Вверху -неестественность положения, высокое давление. С какого либо краю -неустойчивость композиции.

Концентрация активного цвета в правом верхнем углу активизирует композицию, все увеличивается в размере рис. Напротив же в левом нижнем создает иллюзию пассивности и зрительное сжатие, движение назад. Цвет представленный кругом, увеличивает плоскость и создает движение вперед, впечатление усиливается, если это желтый, красный или оранжевый круг рис. Квадрат, окрашенный в холодные тона наоборот, создает впечатление вогнутости и сжатия рис.

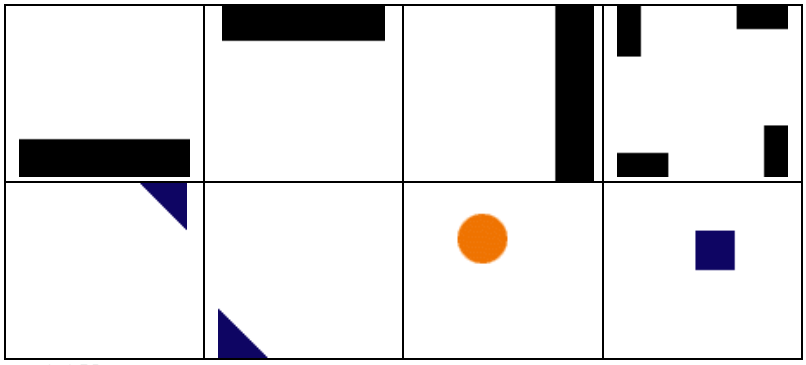

Рис. 4.4 Учет влияния расположения цветовых решений в пространстве.

Что касается воздействием цвета на расстоянии, то характерно что наибольшую различимость имеет сочетание черного цвета на желтом фоне. По мере ухудшения контрастности располагаются следующие сочетания:

- • желтый на черном,
- белый на синем,
- черный на оранжевом,
- оранжевый на черном
- черный на белом,
- • белый на красном,
- • красный на желтом,
- • зеленый на белом,
- • оранжевый на белом,
- красный на зеленом.

В любом случае разработчик должен в совершенстве владеть техникой выполнения проекта в цвете и знать особенности самого цвета.

Цвет – это одно из самых «субъективных» средств композиции, но одной лишь интуиции недостаточно, особенно при проектировании сложной машины или комплекса оборудования интерьера, насыщенного разнообразными приборами.

Цвет должен быть увязан с объемно-пространственной структурой объекта – это, пожалуй, одно из главных условий применения цвета в художественном конструировании. Форма лаконичная, геометрически просто и четко организованная плоскостями, не насыщенная тенями (таков, например, шлифовальный автомат или многие приборные комплексы) не будет мрачной при темно-синих (холодных) или темно-серых (от холодной до темной гаммы) цветах. С помощью цвета можно акцентировать нужные элементы формы или композиционно ослабить их, соподчинить и в известной мере объединить такие элементы структуры, которые не поддаются иным приемам соподчинения. Цвет позволяет скорректировать не слишком удачные пропорции, когда нет возможности изменить сами объемы. Особенно велика роль цвета для достижения образности формы изделия. Удачное цветовое решение помогает раскрыть сущность вещи, обострить или, напротив, сделать более нейтральным, если это нужно, характер формы. Контраст сложной структуры и простого объема можно усилить контрастом цвета и тона, а ньюанс в пластике сделать изысканно тонким введением легкого цветового ньюанса. Даже масштабность формы либо выявляется с помощью цвета, либо утрачивается при ошибках в выборе цвета и тона.

В ходе работы с цветом художнику-конструктору приходится учитывать еще одно обстоятельство. Речь идет о так называемом «одновременном контрасте», то есть об изменении восприятия цвета в зависимости от цветового окружения (контраст усиливается, когда один цвет выступает как пятно на фоне другого). «Изменение» цвета бывает очень сильным, и, если не предусмотреть этого обстоятельства, то в натуре цветовое окружение может сделать цвет зрительно совсем иным, чем это было задумано. Рассматривая эти явления, Н.Е. Кубасова составила таблицу изменения цвета в зависимости от изменения фона. Например, панель пульта управления – это цветовое пятно на фоне корпуса пульта.

Рис. 4.5 Цвето-тоновая композиция.

Контраст – средство композиции, позволяющее активнее выразить функциональные и конструктивные особенности элементов. Средства контрастирования – цветовые и размерные решения. Те элементы, которые необходимо выделить из общей совокупности, имеют обычно отдельную, неповторяющуюся форму и окрашиваются в контр. цвета. Контраст – противопоставление, борьба разных начал в композиции – всегда был одним из наиболее широко используемых средств в руках разработчика. На протяжении многовековой истории искусства тема контраста варьировалась, приобретая в зависимости от характера произведения, стиля эпохи и индивидуальности автора самые различные выражения. В полотнах великого Рембрандта драматизм действия выражается в борьбе яркого света и сгущенных до темноты теней. Спокойная, лаконичная до аскетизма горизонталь стен древнерусских кремлей прерывается, чтобы дать место могучей вертикали башни с пластически богатым верхом. На

использовании контраста как главного, основного в композиции начала построены многие памятники русского и мирового зодчества.

Сущность композиции, построенной на контрасте - в активности ее визуального воздействия. В отличии от ньюансных, контрастные отношения раскрываются сразу, и в зависимости от того, умело ли использован контраст, они вызывают и соответствующую реакцию восприятия.

Контраст активизирует любую форму, однако контрастность отнюдь не гарантия гармонии. Чтобы ее добиться, объективное контрастное начало нужно подчинить интересам композиции, сопроводить контраст ньюансом, найти необходимую меру контраста. Из всех видов контраста в технике больше всего композиционных возможностей таит в себе контраст открытой технической структуры (пространственно сложных сплетений элементов) и спокойных простых объемов. Противопоставляя эти два начала композиции, нельзя, однако, нарушать меру контраста. При слишком резком, жестком контрасте, когда отсутствует переход от одной части формы к другой, возникает опасность утраты композиционной связи элементов, а значит, и целостности формы. Такая форма может зрительно распадаться на две части.

Контраст сложной и простой структур в отличии от контраста цветотонового связан с сильным и глубоким насыщением формы светом и тенью. Поэтому он должен использоваться особенно точно и целенаправленно. Если ошибка в силе тона не может вызвать полной утраты целостности, то неточно найденный контраст структур способен целиком нарушить гармонию формы. Роль контраста в композиции различных промышленных изделий не одинакова. В одних случаях, обусловленный конструктивной основой, он является главным, ведущим средством организации формы, в других – скорее вспомогательным (например, цветотоновые контрасты лакокрасочных покрытий, хотя и они все же являются сильным средством).

Как средство композиции контраст имеет сильные и слабые стороны. Сила контраста в том, что построенная на его основе форма выразительна и надолго удерживается в памяти. Ведь противопоставление, борьба всегда есть своего рода «интрига», завязка всей композиции. Лишенная каких-либо контрастов форма становится вялой и неинтересной – она утрачивает важнейший из компонентов эмоциональной выразительности.

Парадоксально, но слабость контраста в его силе. В самом деле, любое сильнодействующее средство требует осторожности – его избыток разрушителен. Таков и контраст. Используя его в композиции конкретного промышленного изделия, нужно позаботиться, чтобы сила контраста не оказалась чрезмерной, то есть была соблюдена определенная *степень, или мера, контраста*. Для контрастирующих элементов можно построить целый ряд от наименьшей до наибольшей степени контраста. Взять, к примеру, контраст пятна и фона. Если фон абсолютной белизны, а пятно абсолютной черноты, контраст максимален. Но контрастным будет и отношение не совсем белой, а несколько тонированной (в разной степени) поверхности и не совсем черного, но темно-серого (различной силы) пятна. Для выразительности и целостности композиции конкретного изделия степень контраста имеет немалое значение. Например, когда *маленькое* темное пятно контрастирует с *большим* светлым фоном, степень контраста может быть предельно сильной. Таковы маленькие черные ручки и другие детали на белой или почти белой панели. При хорошей компоновке они не нарушают целостности, и контраст в этом случае может быть предельным. Но если все эти детали сильно увеличить, так что суммарно черное приблизится к площади белого, эффект контраста ослабеет, композиция может стать значительно менее выразительной и целостной, чем в первом случае. Это объясняется тем, что небольшие темные детали контрастируют с фоном не только по цвету и тону, но и по величине – малое противопоставляется большому. Во втором случае один из признаков контраста исчез, и равенство между черным и белым по существу сняло всю остроту противопоставления. В этом случае целесообразно несколько смягчить и цветовой контраст. И отнюдь не случайно многие приборостроительные фирмы изготавливают комплектующие детали не черного цвета, а серой, охристой или коричневой гаммы. При

насыщенных панелях приборов, когда площадь свободного фона уменьшается, такое смягчение контраста способствует достижению целостности.

Комментарии и советы

• Контрастные цвета от сочетания друг с другом становятся ярче, заметнее один на другом. Один и тот же красный цвет на зеленом фоне смотрится ярко и четко, на оранжевом приглушенно.

• Теплые цвета приближают предметы, холодные удаляют, при этом предмет, окрашенный в теплые цвета, кажется больше своих размеров, а холодные -меньше.

• На светлом фоне все цвета темнеют, на темном светлеют. Истинная светлота цвета может наблюдаться только на нейтральном фоне средней светлоты. В зависимости от фона ахроматические цвета приобретают кажущуюся цветность. Так, серое пятно на зеленом фоне приобретает розовый оттенок. Хроматические цвета в окружении цветов высокой насыщенности несколько меняют цветовой тон, например, желтый цвет на зеленом фоне становиться слегка оранжевым, а красный в окружении зеленого - более насыщенным.

• Чем больше цвета отличаются один от другого по светлоте, насыщенности и цветовому тону, тем менее они гармонируют друг с другом.

• Существует понятие краевого контраста т.е. равномерно окрашенная поверхность кажется у края светлее или темнее если она граничит с более темной или светлой поверхностью.

5. Введение в PHP

PHP - это система разработки скриптов, включающая в себя CGI-интерфейс, интерпретатор языка и набор функций для доступа к базам данных и различным объектам WWW. На данный момент PHP является наиболее удобным и мощным средством разработки приложений WWW и интерфейсов к БД в Интернет.

PHP - это интерпретируемый язык программирования, код которого встраивается непосредственно в HTML-страницы. При запросе пользователя web-сервер просматривает документ, выполняет найденные в нем PHP-инструкции, а результат их выполнения возвращает пользователю.

При этом статическая часть документа, написанная на языке HTML, фактически является шаблоном, а изменяемая часть формируется при исполнении PHP-инструкций. Для удаленного пользователя подобные документы ничем не отличаются от обычных статических HTML-документов, за исключением того, что в расширении имени файла для таких документов может стоять не htm или html, а phtml, PHP или PHP3.

На сегодняшний день PHP является одним из самых популярных средств создания скриптов серверной стороны. Это значит, что текст скрипта до конечной обработки броузером будет обработан на сервере интерпретатором PHP, что позволяет осуществлять динамическую генерацию HTML-кода, обрабатываемого броузером. То есть, говоря простыми словами, можно изменять вид конечной web-страницы, выдаваемый броузером в зависимости от входных данных.

В общем - на PHP можно сделать все, что можно сделать с помощью CGI-программ. Например: обрабатывать данные из форм, получать и посылать куки (cookies).

Кроме этого в PHP включена поддержка многих баз данных (data bases), что делает написание web-приложений с использованием БД совершенно простым. Вот неполный перечень поддерживаемых БД: AdabasD, Interbase, Dbase, mSQL, Unix DBMm, Oracle, Informix, Sybase и т. д. Вдобавок ко всему PHP понимает протоколы IMAP, SNMP, NNTP, POP3 и даже HTTP, а также имеет возможность работать с сокетами (sockets) и общаться по другим протоколам.

Разработчикам web-приложений нет необходимости говорить, что web-страницы - это не только текст и картинки. Достойный внимания сайт должен поддерживать некоторый уровень интерактивности с пользователем: поиск информации, продажа продуктов, конференции и т.п.

Традиционно все это реализовалось CGI-скриптами, написанными на Perl. Но CGI-скрипты очень плохо масштабируемы. Каждый новый вызов CGI требует от ядра порождения нового процесса, а это занимает процессорное время и тратит оперативную память. PHP предлагает другой вариант - он работает как часть web-сервера, и этим самым похож на ASP от Microsoft.

Синтаксис PHP очень похож на синтаксис C или Perl. Люди, знакомые с программированием, очень быстро смогут начать писать программы на PHP. В этом языке нет строгой типизации данных и нет необходимости в действиях по выделению/освобождению памяти. Программы, написанные на PHP, достаточно легкочитаемы. Написанный PHP-код легко зрительно прочитать и понять, в отличие от Perl программ.

5.1. Недостатки PHP

1. Основным недостатком PHP 3 является то, что по свой идеологии PHP изначально был ориентирован на написании небольших скриптов. Несмотря на то что движок несколько раз переписывался, PHP 3 непригоден для использования в сложных проектах - при обработке больших скриптов производительность системы резко падает (предчувствуя возмущение сторонников PHP 3, я скажу, что наличие такого недостатка подтверждает сам разработчик Зев Сураски). Однако этот недостаток ликвидирован в движке PHP 4, который, по словам того же разработчика, предназначен для работы в больших проектах.

2. PHP является интерпретируемым языком, и, вследствие этого, не может сравниться по скорости с компилируемым C. Однако при написании небольших программ (что, в общем-то, присуще проектам на PHP, когда весь проект состоит из многих небольших страниц с кодом) вступают в силу накладные расходы на загрузку в память и вызов CGI-программы, написанной на C.

3. Не такая большая база готовых модулей, как, например, CPAN у Perl. С этим ничего нельзя поделать - это дело времени. В PHP4 разработчики предусмотрели специальный репозиторий, аналогичный CPAM, и возможно очень скоро будет написано достаточное количество модулей для его наполнения.

4. Нет поддержки сессий (session), как, например, в ASP. В PHP 4 этот недостаток теперь устранен.

Дополнительную информацию по PHP можно получить на официальном сайте PHP расположен по адресу www.php.net. К достоинствам этого сайта можно отнести наличие наиболее полной и свежей информации о программном продукте, к основным недостаткам - отсутствие русскоязычных ресурсов.

Самым известным ресурсом рунета, посвященным PHP, является сайт phpclub.net. Здесь выложен довольно большой объем документации по PHP на русском и английском языке, имеется форум, коллекция бесплатных скриптов и ссылок на другие ресурсы подобной тематики, как в России, так и за рубежом. Также стоит упомянуть сайт www.webclub.ru - сайт клуба российских web-мастеров. Здесь, кроме информации по PHP, можно найти также множество другой полезной информации по web-дизайну и web-мастерингу.

5.2. Установка PHP

Итак, если вы решили установить PHP, то вы скорее всего ли устанавливаете и настраиваете web-сервер под управлением ОС UNIX - и тогда вам стоит прочитать первую часть этого раздела; либо ни хотите установить PHP на домашнем компьютере под управлением ОС семейства Windows, чтобы иметь возможность локальной отладки PHP скриптов - тогда вам стоит внимательно прочитать вторую часть этого раздела.

PHP под UNIX

Для инсталляции PHP под UNIX вам необходимо следующее:
- компилятор ANSI C (для компиляции исходных кодов);
- web-server

Первым делом вам нужно загрузить архив с исходными кодами. Их можно найти по адресам: www.php.net и phpclub.net. Далее необходимо провести компиляцию исходных кодов и конфигурирование PHP. Вследствие ограниченности объема этой книги, мы приведем лишь сценарий установки PHP 4.0.2. под web-сервер Apache 1.3.x.

1. Необходимо проверить, включена ли в Apache поддержка модулей:
httpd -1
Ответ должен выглядеть приблизительно так (в зависимости от конфигурации Apache):

```
Compiled-in modules:
http_core.c
mod_so.c
```

Строка mod_so.c показывает, что поддержка модулей включена, то есть установку можно продолжить.

2. Распаковка
gunzip php-4.0.2.tar. gz I tar xf -
cd php-4.0.2
./configure —with-mysql=<path> —with-apxs

Последняя строка запускает конфигурационный скрипт, предваряющий компиляцию. В нашем примере мы конфигурируем PHP с поддержкой MySQL; <path> - указывает путь к месту, где установлен MySQL. Вы можете добавить дополнительные модули PHP (например, GD, который мы советуем вам сразу установить). Чтобы это сделать, в конец последней строки нужно добавить строку:

—with—<имя модуля>=<путь к модулю>.

Список дополнительных модулей, включенных в дистрибутив PHP, можно найти в документации к дистрибутиву.
3. Компиляция
make
make install
4. Далее нужно произвести процедуру настройки Apache:
Скопируйте файл php. ini-dist в папку с конфигурационным файлом Apache -- httpd. conf (обычно его можно найти в папке /usr/local/httpd/conf). Переименуйте этот файл в php. ini. Далее добавьте в файл httpd. conf строку следующего содержания
AddType application/x-httpd-php .php .phtml .php3

Примечание

Если в системе ранее был установлен PHP 3, то необходимо удалить или перенести файл mod_php3.conf из каталога httpd/conf /addon-modules .
5. Перезапустите Apache.

PHP под Windows

Приведем пример установки PHP3 под Windows и Apache 1.3.x. Итак, прежде всего поговорим о каталоге, в котором у вас будут находиться файлы PHP. В дистрибутиве по умолчанию стоит такой:
f:/usr/local/php3

Если вы физически не можете или просто не хотите иметь такой каталог (хотя, если вы читали инструкцию по установке Apache, все должно быть в порядке), то вы вольны установить PHP в другой каталог, но тогда вам предстоит следующее: в файле php_iifsreg. inf из дистрибутива PHP найти ВСЕ строки "f: /usr/local/php3" (их там, кстати, 6 штук) и заменить их на тот каталог, где Вы предполагаете разместить PHP. Как обычно, укажем по порядку те действия по установке PHP, которые привели нас к положительному результату.

1. Создайте директорию f: /usr/local/php3. ') в которую будет установлен PHP.
2. Скачайте дистрибутив PHP, желательно в только директорию. Это саморазворачивающийся zip-архив, который будете запустить, чтобы разархивировать. По умолчанию он распакуется в текущую директорию, так что будьте внимательны.

3. Еще раз напоминаем: если вы решили установить PHP в другую директорию, вам необходимо вручную отредактировать файл php iifsreg.inf, чтобы заменить указанные в нем имена директорий на нужные вам (см. выше).

4. В файле php3.ini из дистрибутива есть закомментированные строки, выглядящие так:
extension=MMH_MOflynH.dll
Если вы хотите включить какой-нибудь модуль (по умолчанию уже включена поддержка GD и mSQL), раскомментируйте соответствующую строку (уберите точку с запятой).

6. Теперь в Проводнике Windows нажмите правой кнопкой мыши на файле php_iis_reg.inf и выберите в контекстном меню пункт "Установить" - этим вы автоматически добавите в Реестр некоторые установки, касающиеся PHP.

7. Скопируйте файл php3.ini в каталог с Windows (например, в C: Windows);

Настройка Apache
1. В файл конфигурации Apache conf /mime, types добавьте такую строку:
application/x-httpd-php3 phtml php3 php

3. Теперь откройте файл conf /httpd.conf и добавьте в его конец (но перед блоками виртуальных хостов, если они там есть) такие строки:

```
<Directory "f:/usr/local/php3">
Options ExecCGI
</Directory>
ScriptAlias "/ php_dir   /" "f:/usr/local/php3/"
Action application/x-httpd-php3 "/ php_dir   /php.exe"
```

Ну вот, пожалуй, и все. Если вы все сделали правильно, то PHP установлен. Проверьте его работоспособность с помощью простого скрипта, например такого:

5.3. Синтаксис PHP

Знакомство с PHP мы начнем с классического примера "Hello, World!"

```
<html>
  <head>
      <title>Пример</title>
  </head>
   <body>
     <?php echo "Hello,  world!";  ?>
  </body>
</html>
```

Как нетрудно заметить, данный скрипт состоит из двух частей: HTML-документа и встроенной в него PHP-инструкции "echo" выводящей строку "Hello, World!". Попробуем разобраться, как это все работает. Итак, вначале исходный текст скрипта поступает на вход интерпретатора PHP. Интерпретатор, обрабатывая файл, передает на выход текст из скрипта без изменений до момента встречи в тексте специального тэга (в нашем случае "<?php"). Весь текст после этого тэга и до закрывающего тэга " ?>" интерпретируется как php-скрипт, вывод которого производится в HTML-документ. Дальнейший текст также не интерпретируется. Таким образом после всех преобразований броузер получит следующий HTML-документ:

```
<html>
<head>
     <title>Пример</title>
</head>
     <body>Hello,  world!</body>
```

```
</html>
```

То есть, говоря простыми словами, вместо текста PHP-инструкций броузер получит их вывод. Кстати говоря, существуют также другие способы перехода из HTML в PHP.

А) <? Echo "простейший способ, но возможен конфликт при использовании XML "; ?>

Б) <?php echo («Наиболее часто используемый способ»); ?>

В) <?php3 echo("Используется, чтобы явно указывать на третью версию PHP"); ?>

Г) <script language="php">
echo ("используется для лучшей совместимости с HTML-редакторами") ;
</script>;

Д) <% echo("От PHP 3.0.4 можно факультативно применять ASP-тэги"); %>

Сами инструкции внутри скрипта разделяются как в C или Perl - точкой с запятой. Закрывающий тэг (?>) тоже подразумевает конец утверждения, поэтому следующие записи эквивалентны:

```
<php
echo "Это тест";
?>

<php echo "Это тест" ?>
```

Переменные

Типы переменных

Основными типами переменных, используемых в PHP являются:

- integer — целое
- double - число с дробной частью
- string - строковая переменная
- array - массив
- object - объектная переменная

Необходимо обратить внимания на два момента.

Во-первых, обычно тип данных явно не указывается, а определяется в зависимости от значения, присваиваемого переменной (переменная может вести себя по-разному, в отличие от типа данных, который определен для нее в данный момент). Если же необходимо указать тип переменной принудительно, используется инструкция cast или функция settype.

Во-вторых, в PHP нет символьного типа, базовым типом является строковый (что сближает PHP с языком Perl), по этой причине доступ к отдельному символу в возможен только через использование односимвольных строк.

Инициализация переменной

Для инициализации переменной в PHP вы просто присваиваете ей значение. Для большинства переменных это именно так; для массивов и объектных переменных, однако, может использоваться несколько иной механизм.

```
$MyNumber=l;
$MyString="Всем привет";
```

Инициализация Массивов

Массив может инициализироваться одним из двух способов: последовательным присвоением значений или посредством конструкции array []. При последовательном добавлении значений в массив вы просто записываете значения элементов массива, используя пустой индекс. Каждое последующее значение будет добавляться в качестве последнего элемента массива. Как в C и Perl, элементы массива нумеруются начиная с 0, а не с 1.

```
$array[] = "Нулевой элемент"; // $array[0] = "Нулевой элемент"
$array[] = "Первый элемент"; // $array[1] = "Первый элемент"
//и т. д.
```

Инициализация объектов

Инициализация объектов происходит почти также, как это делается в C. Сперва создается класс, в котором описываются свойства, далее можно создать объект - экземпляр этого класса.

```
class br {
function makebr () {
echo "<br>";
}
}
$brl = new br;
$brl -> makebr ();
```

Область видимости переменной

Областью переменной является контекст, внутри которого она определена. Главным образом, все переменные PHP имеют одну область. Однако, внутри функций определенных пользователем, представлена локальная область функции. Любая переменная, определенная внутри функции, по умолчанию ограничена локальной областью функции. Например:

```
$a = 1; /* глобальная область */
Function Test () {
echo $a; /* ссылка на переменную локальной области */
}
Test ();
```

Этот скрипт не выдаст что-либо на выходе, поскольку инструкция echo относится к локальной версии переменной $a, значение которой присваивается не внутри этой области. Вы можете заметить, что здесь имеется некоторое отличие от языка C в том, что глобальные переменные в C автоматически действуют и внутри функций, если только они не переписываются локальными определениями. Это может вызвать некоторые проблемы, т.к. по неосторожности можно изменить глобальную пе ременную. В PHP глобальные переменные должны быть продекларированы глобально внутри функции, если предполагается их использование в данной функции. Например:

```
$a = 1;
$Ь = 2;
Function Sum () {
global $a, $b;
$b = $a + $b;
}
Sum () ;
echo $b;
```

Вышеописанный скрипт выдаст значение "3". Поскольку $a и $b декларируются глобально внутри функции, ссылки на данные переменные трактуются как ссылки на их

глобальные версии. Нет ограничений на количество глобальных переменных, которыми можно манипулировать внутри функции.

Вторым способом доступа к переменным из глобальной области является использование специального определяемого PHP-массива $GLOBALS. При этом предыдущий пример может быть записан так:

```
$a = 1;
$Ь = 2;
Function Sum () {
$GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}
Sum () ;
echo $b;
```

Массив $GLOBALS является ассоциативным массивом, в котором имя глобальной переменной является ключом, а значение этой переменной является значением элемента массива.

Другой важной характеристикой от области определения переменной является статическая переменная. Статическая переменная существует только в локальной области функции, но она не теряет своего значения, когда программа, при исполнении, покидает эту область. Рассмотрим следующий пример:

```
Function Test () {
$a = 0;
echo $a;
$a++;
}
```

Эта функция совершенно бесполезна практически, поскольку каждый раз при ее вызове она устанавливает $a в 0 и выводит "О". Выражение $a++, которое увеличивает значение переменной, так же бесполезно, поскольку при выходе из функции переменная $a исчезает. Для придания дееспособности функции подсчета, которая не теряла бы нить текущего счета, переменная $a декларируется как статическая:

```
Function Test () {
static $a = 0;
echo $a;
$a++;
}
```

Теперь каждый раз при вызове функции Test () она будет выводить значение $a и увеличивать его. Статические переменные также весьма существенны, когда функции вызываются рекурсивно. Рекурсивные функции - это те, которые вызывают сами себя. Составлять рекурсивную функцию нужно внимательно, т.к. при неправильном написании можно сделать рекурсию неопределенной. Вы должны быть уверены в адекватности способа прекращения рекурсии. Следующая простая функция рекурсивно считает до 10:

```
Function Test () {
static $count = 0;
$count++;
echo $count;
if ($count < 10) {
Test ();
}
$count—;
}
```

Изменяемые переменные. Одной из наиболее интересных возможностей PHP является

возможность доступа к имени переменной. Проиллюстрируем, как это можно сделать. Проинициализируем обычную переменную:
$a = "hello";
Изменяемая переменная берет некое значение и обрабатывает его как имя переменной. В приведенном выше примере значение hello может быть использовано как имя переменной посредством применения двух записанных подряд знаков доллара, т.е.:
$$a = "world";
С этой точки зрения, две переменных определены и сохранены в символьном дереве PHP: $a с содержимым "hello" и $hello с содержимым "world". Так, инструкция:
echo "$a ${$a}";
осуществляет то же самое, что и инструкция:
echo "$a $hello";
а именно, обе они выводят: hello world.

Чтобы использовать изменяемые переменные с массивами, необходимо решить проблему неоднозначности. Иными словами, если вы пишете $$a [1], то синтаксическому анализатору необходимо знать, имеете ли вы в виду использовать $a[1] как переменную, или вы предполагаете $$a как переменную, а [1] как индекс этой переменной. Синтаксис для разрешения подобной неоднозначности такой: ${$a[1]} для первого случая и $ {$a} [1] для второго.

Переменные, переданные формой

Как уже говорилось выше, PHP позволяет обрабатывать переменные, переданные скрипту методами GET и POST.
То есть, если мы, например, запускаем скрипт test .php вот таким образом
test.php?a=l&b=5,
то внутри тела скрипта test. php будут уже проинициализированы переменные $a со значением 1 и $ь со значением 5.
Можно передавать данные из форм и по-другому.

```
<form action="foo.php3" method="post">
Name:  <input type="text" name="name"xbr>
<input type="submit">
</form>
```

При активизации формы PHP создаст переменную $name, значением которой будет то содержимое, которое было введено в поле Name: данной формы. PHP также воспринимает массивы в контексте переменных формы. Но это верно только для одномерных массивов. Вы можете, например, группировать взаимосвязанные переменные вместе или использовать это свойство для определения значений переменных при множественном выборе на входе:

```
 $foo = "O";     // $foo является строкой   (ASCII 48)
$foo++;          // $foo является строкой "1"   (ASCII 49)
$foo += 1;    // $foo сейчас является целым (2)
$foo = $foo + 1.3;    // $foo сейчас имеет тип double  (3.3)
$foo = 5 + "10 Little Piggies"; // $foo является цельм (15)
$foo = 5 + "10 Small Pigs";      // $foo является целым (15)
```

Если вы желаете изменить тип переменной, используйте функцию
settype().

Определение типов переменных

Поскольку PHP определяет типы переменных и преобразует их (в общем) по мере необходимости, не всегда очевидно, какой тип данная переменная имеет в какой-то отдельный момент. Поэтому PHP включает несколько функций, которые позволяют определить текущий

тип переменной. Это функции gettype(), is long(), is double(), is string(), is array (), и is object().

Приведение типа

Приведение типа работает в PHP во многом так же, как в C: название требуемого типа записывается в круглых скобках перед переменной, которая должна быть приведена к данному типу.

```
$foo = 10;      // $foo is an integer
$bar =  (double)   $foo;      // $bar is a double
```

Допускается следующее приведение типов:

- (int), (integer) - приведение к целому
-(real), (double), (float) - приведение к типу double
- (string) - приведение к строке
- (array) — приведение к массиву
- (object) - приведение к объектной переменной

Заметим, что табуляция и пробелы допускаются внутри круглых скобок, поэтому следующее функционально эквивалентно:

```
$foo  =  (int)  $bar;
$foo  =   (  int  )   $bar;
```

Преобразование строк

Когда строковая переменная оценивается как числовая, результирующее значение и тип переменной определяются следующим образом. Переменная string будет оценена как double, если она содержит любой из символов '.', 'e'.или '£'. Иначе она будет оценена как integer. Данное значение задается начальной частью строковой переменной. Если строка начинается с допустимых цифровых данных, то это значение и будет использоваться. Иначе будет значение 0 (ноль). Допустимые цифровые данные - это конструкция из факультативного символа, следующего за одной или несколькими цифрами (содержащими факультативно десятичную точку), обозначающего экспоненту. Экспонента может обозначаться символом *e' или 'E', который может следовать за одной или несколькими цифрами.

```
$foo = 1 + "10.5";       // $foo тип double   (11.5)
$foo -14 "-1.3e3";        // $foo тип double  (-1299)
$foo = 1 + "bob-1.3e3";  // $foo тип integer  (1)
$foo = 1 + "bob3";      // $foo тип integer  (1)
$foo = 1 + "10 Small Pigs";        // $foo тип integer (11)
$foo = 1 + "10 Little Piggies"; // $foo тип integer (11);
// строка содержит  'e'
```

Манипуляции с массивом

PHP поддерживает как скалярные, так и ассоциативные массивы. Фактически между ними нет разницы. Вы можете создать массив, используя функции list () или , array () или можно явно задать значение каждого элемента массива.

```
$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13;
```

Можно также создать массив просто, добавляя в него значения.

```
$a[] = "hello";  // $a[2]  == "hello"
$a[] = "world";  // $a[3]  == "world"
```

Массив может сортироваться функциями asort(), arsort(), ksort(), rsort(), sort (), uasort(), usort() и uksort() в зависимости от типа желаемой сортировки. Подсчет количества элементов массива осуществляется функцией count().
Перемещаться по массиву позволяют функции next () и prev(). Другим типовым способом перемещения по массиву является использование функции each ().

Константы

PHP определяет несколько констант и предоставляет механизм для определения ваших констант. Константы похожи на переменные, но они имеют слегка измененный синтаксис.

Предопределенные константы - это FILE и LINE , которые соответствуют имени файла и номеру строки, которая выполняется в настоящий момент.

```
<?php
function report_error($flie, $line, $message) {
echo "An error occured in $file on line $line: $mes-
sage.";
}
report_error( FILE , LINE , "Something went wrong!");
?>
```

Вы можете определить дополнительные константы с помощью функций define () и undefine ().

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
undefine ("CONSTANT");
?>
```

Выражения

Выражения - это краеугольный камень PHP. В PHP почти все является выражениями. Простейший и наиболее точный способ определить выражение - это "что-то, имеющее значение".

Простейший пример, приходящий на ум, это константы и переменные. Когда вы печатаете "$a = 5", вы присваиваете значение '5' переменной $a. '5', очевидно, имеет значение 5, иными словами '5' это выражение со значением 5 (в данном случае '5' - это целочисленная константа).

После этого присваивания вы считаете значением $a 5, так же, если вы напишете $Ь = $a, вы будете ожидать того же, как, если бы вы написали $Ь = 5. Другими словами, $a это также выражение со значением 5. Если все написано правильно, то именно так оно и случится. Несколько более сложные примеры выражений - это функции. К примеру, подумайте над следующей функцией :

```
function foo () {
return 5;
}
```

Подразумевая, что вы знакомы с концепциями функции (если нет, взгляните на часть, посвященную функциям), вы считаете, что $c = foo () это практически то же самое, что написать $ c = 5, и вы правы. Функции - это выражения с тем значением, которое они возвращают. Так как foo() возвращает 5, значение выражение 'food' - 5. Обычно функции подсчитывают возвращаемое значение, а не возвращают постоянное значение.

Конечно, значения в PHP не обязаны быть целыми и зачастую они не являются

таковыми. PHP поддерживает 3 скалярных типа значений: целое, число с плавающей точкой и строки (скалярные выражения вы не можете "разбить" на более маленькие части, как, к примеру, массивы).

PHP поддерживает 2 составных (нескалярных) типа: массивы и объекты. Каждое из таких значений может быть присвоено переменной или возвращено функцией.

PHP — это язык, ориентированный на выражения, практически все является выражениями. Обратимся к примеру, с которым мы уже разобрались, $a = 5'. Легко заметить, что тут задействованы два значения - значение целой константы 5 и значение $a, которое также становиться равным 5. На самом деле здесь присутствует еще одно значение, значение самого присваивания. Само присваивание становиться равным присваиваемому значению, в данном случае — 5. На практике это означает, что $a = 5', не обращая внимания на то, что оно равно выражению со значением 5. То есть, запись типа '$b = ($a = 5)' похожа на запись $ a = 5; $b = 5;' (точка с запятой отмечает конец выражения). Так как присваивания рассматриваются справа налево, вы также можете написать $b = $a = 5'.

Другой хороший пример направления выражения - это предварительный и последующий инкремент. В PHP, подобно С, есть 2 типа инкремента - предварительный и последующий. И предыдущий и последующий инкремент увеличивает значение переменной и воздействие на переменную идентично. Разница в значении выражения инкремента. Предыдущее увеличение, которое записывается как $variable', приравнивается увеличенной переменной (PHP увеличивает переменную до того, как прочитать ее значение). А если вы пишите '$variable++', то значение выражения приравнивается к неувеличенной переменной, а увеличение переменной происходит после.

Очень распространенный тип выражений - это выражения сравнения. Эти выражения имеют значение 0 или 1 (означает ложь или истину соответственно). PHP поддерживает > (больше, чем), >= (больше или равно), = (равно), < (меньше, чем) и <= (меньше или равно). Эти выражения в основном используются внутри условий, например оператора IF.

Последний пример выражений, с которыми мы разберемся, это совмещенные выражения оператор-присваивание. Вы уже знаете, что для того, чтобы увеличить значение $a на единицу, вы можете написать '$a++' или *++$a'. Но если следует увеличить значение больше, чем на единицу, к примеру - на 3? Вы можете написать $a++' несколько раз, но это не очень удобно и эффективно. Намного больше распространено написание '$a = $a + 3'. л$a + 3' вычисляется, как значение $ а плюс 3, а затем присваивается переменной $a, в результате чего значение $a увеличивается на 3. В PHP, так же как, и в ряде других языков типа С, вы можете записать это короче, что со временем бывает проще и быстрее также и для понимания. Добавление 3 к текущему значению $a может быть записано как '$a+=3'. Это значит следующее: 'возьми значение $a, добавь к нему 3 и присвой это обратно $a. Кроме того, что это быстрее и понятнее, такой тип выражений быстрее исполняется. Значение '$a+=3', как и значение обычного присваивания, это присвоенное значение. Заметьте, что оно не равно 3, а является общим значением $a и 3. Любой бинарный (имеющий 2 операнда. - Прим, пер.) оператор может быть записан таким методом, например: '$a-=5' (вычесть 5 из значения $a), '$Ь*=7' (умножить значение $ а на 7) и так далее.

Есть еще такое выражение, которое может показаться незнакомым, если вы не встречались с ним в других языках - условный оператор с тремя операндами :

$first? $second: $third

Если значение первого выражения истинно (не равно 0), то исполняется второе выражение и это является результатом данного условного выражения. Иначе исполняется третий оператор.

Этот пример должен помочь Вам лучше понять предварительное и последующее увеличение и вообще выражения

```
function double($i)  /* функция усваивания переменной */
{
return $1*2;
}
```

```
$Ь = $a = 5; /* присваиваем значения переменным $a
и $Ь */
$c = $a++; /* последующее увеличение, присваиваем $c
начальное значение $a (5)*/
$e = $d ++$b; /* предварительное увеличение,
присваиваем $d и $e увеличенное значение $Ь (6) */
/* тут и $d и $e равны 6 */
$f = double($d++) ; /* присвоить удвоенное значение $d
до его увеличения, то есть 2*6 = 12, переменной $£*/
$g = double(++$e); /* присвоить удвоенное значение $e
после его увеличения, то есть 2*7 = 14, переменной g */
$h = $g += 10; /* сначала увеличить значение $д на 10, что дает в результате 24, а затем
присвоить это значение переменной $h,. что также дает 24 */
```

В начале главы мы сказали, что объясним различные типы операторов и, как и было обещано, выражения могут быть операторами. Впрочем, не каждое выражение является оператором. В данном случае оператор имеет форму 'выражение";', то есть выражение, за которым следует точка с запятой. В '$Ь=$a=5;' $a=5 это правильное выражение, но само по себе оно не является оператором. А вот '$ь=$a=5;' является правильным оператором.

Еще одна вещь, которую нужно упомянуть, - это логические значения выражений. Во многих случаях, в основном в условных операторах и операторах циклов, вы заинтересованы не в конкретных значениях выражений, а только в том, являются ли их значения TRUE или FALSE (в PHP нет специального типа boolean). Логические значения вычисляются примерно так же, как и в языке Perl. Любое ненулевое целое значение - это TRUE, ноль - это FALSE. Обратите внимание на то, что отрицательные значения - это не ноль и поэтому они считаются равными TRUE. Пустая строка и строка '0' это FALSE; все остальные строки - TRUE. И насчет составных типов (массивы и объекты) - если значение такого типа не содержит элементов, то оно считается равным FALSE; иначе подразумевается TRUE.

PHP предоставляет полную и мощную реализацию выражений, и подробное ее описание выходит за пределы этого руководства. Приведенные выше примеры показали вам, что такое выражения и как вы можете построить реальные выражения.

Регулярные выражения

Как уже говорилось выше, в PHP (как и в Perl), в отличие от C, базовым типом данных является не символьный, а строковый. Это обусловлено спецификой решаемых на PHP задач. Скрипты на PHP в большинстве своем содержат части, предназначенные для обработки строк: это могут быть данные в форме, значения переменных окружения, данные, выбранные из таблиц баз данных, и т. д.

Для обработки строк в PHP был введен еще один чрезвычайно мощный и удобный инструмент - регулярные выражения. С их помощью можно проводить анализ и изменение строк на основе заданного шаблона. Шаблоном в PHP называется строка, состоящая как из символов, так и из модификаторов. Функции, работающие с регулярными выражениями ищут вхождение шаблона в обрабатываемую строку и производят над ней определенные операции - замену подстрок, разбивку строки на части и т. д.

Рассмотрим простые (далеко не все) правила формирования шаблона. Шаблон составляется из набора модификаторов, некоторые из которых приведены в нижеследующей таблице.

В PHP существует несколько функций для работы с регулярными выражениями: ereg (), ereg_replace(), eregi (), ereg_replacei() и split().

Функции с суффиксом i представляют из себя аналоги функций без этого суффикса, не чувствительные к регистру операндов.

Функция ereg является наиболее простой функцией - она проверяет вхождение шаблона в строку, то есть вызов <?php if (ereg ("шаблон", "строка для поиска шаблона"))

print "совпадение найдено" ?> выведет строку только при совпадении шаблона со строкой для поиска.

Функция ereg_replace заменяет все найденные шаблоны в строке поиска на новые подстроки .

<?php $str=ereg_replace("old","new","строка для замены old на new") ?>
В данном примере $str будет приравнен к строке "строка для замены new на new".

Функция split предусмотрена для разбития строки на части заданным в шаблоне разделителем. Функция возвращает массив подстрок

<?php $a=split("/","a/b/c") ?>
В примере получим массив $a с тремя элементами - "a","Ь","c".

Операторы

Арифметические операторы

$a + $Ь	Сложение	Сумма $a и $b. $a - $Ь.
$a-$Ь	Вычитание	Вычитает $Ь из $a.
$a*$Ь	Умножение	Произведение $a и $Ь.
$a/$Ь	Деление	Деление $a на $Ь.
$a % $Ь	Остаток деления	Остаток от деления $a на $Ь.

Оператор деления ("/") возвращает целую величину (результат целочисленного деления) если оба оператора - целые (или строка преобразованная в целое). Если каждый операнд является величиной с плавающей запятой, выполнится деление с плавающей запятой.

Операторы строк

В действительности есть только один оператор — оператор конкатенации (".").

```
$a = "Hello ";
$Ь = $a . "World!"; // теперь $b = "Hello, World!"
```

Операторы присваивания

Основным оператором присваивания является "=". Можно подумать что это - "равно" ("equal to"). Но это не так. В действительности это означает, что левый операнд получает значение выражения в правых (собирательное присваивание).

Значением выражения присваивания является присваиваемая величина. Так что если "$a = 3", то это 3. Это позволит вам делать некоторые мудреные вещи: $a = ($Ь = 4) +5; // теперь $a равно 9, а $Ь стало равным 4.

В дополнение к основным операторам присваивания есть еще "комбинационные операторы", для всех арифметических и строковых операторов, что позволяет вам использовать значение в выражении и затем устнавливать свое значение в результате этого выражения.

Например:

```
$a = 3; $a += 5; // теперь $a равно 8, как если бы
мы сказали: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // теперь $b равно "Hello There!", как если бы мы написали $b = $b. "There!";
```

Бинарные Операторы (Побитовые Логические Операторы)

Бинарные Операторы позволяют вам изменять биты в целых числах.

$a&$Ь И Будут установлены биты, которые были установлены и в $a и в $Ь.
$a=5; /* 0101 */
$Ь=12;/* 1100*/
$c=$a & $Ь; /* $c будет равно 4 (0100) */
$a| $Ь Или Будут установлены биты, установленные в $a или $Ь.
$a=5; /* 0101 */ $Ь=12; /* 1100 */ $c=$a|$Ь; /* $c
будет (1101)*/
~$a Не Будут установлены не_присутствующие в $a биты (реверс)
$a=5; /* 0101 */ ~ $a /* $a будет равно x (1010) */

Логические операторы

$a and $b И Истина, если истинны $a и $Ь.
$a or $b Или Истина, если истинны $a или $Ь.
$a xor $b Или Истина, если истинны $a или $Ь, но не оба.
!$a Не Истина, если не истинно $a.
$a && $b И Истина, если истинны и $a и $Ь.
$a || $b Или Истина, если истинны $a или $Ь.
Разница в двух различных вариантах операторов "and" и "or" – в различии приоритетов операций.

Операторы Сравнения

Операторы Сравнения, как и подразумевается в их имени, позволяют Вам сравнивать две величины.

$a == $b Равно Истина, если $a эквивалентно $Ь.
$a != $b Не равно Истина, если $a не эквивалентно $Ь.
$a<$b Меньше, чем Истина, если $a меньше чем $Ь.
$a>$b Больше, чем Истина, если $a больше $Ь.
$a <= $b Меньше или равно Истина, если $a меньше или равно $Ь.
$a >= $b Больше или равно Истина, если $a больше или равно $Ь.

Условный оператор IF

Структура IF - это одна из важнейших возможностей многих языков, включая PHP. Она позволяет организовать выполнение фрагментов кода по условию. Возможности PHP по использованию выражения IF похожи на C:

if (expr) statement

Как объяснялось в части про выражения, вычисляется логический результат "expr". Если expr равно TRUE, то PHP выполнит "statement", а если FALSE - проигнорирует.

Следующий пример выведет фразу 'a is bigger than b', если $a больше $b:
if ($a > $b)
print "a is bigger than b";

Зачастую вам требуется исполнить больше, чем одно выражение по условию. Конечно, не надо окружать каждое выражение конструкцией IF. Вместо этого вы можете сгруппировать несколько выражений в блок выражений. К примеру, следующий код не только выведет фразу, но и присвоит значение $a переменной $b:

if ($a > $b) { print "a is bigger than b"; $b = $a; }

Выражение IF может иметь неограниченную степень вложенности в другие выражения IF, что позволяет Вам эффективно использовать выполнение по условию различных частей программы.

ELSE

Иногда требуется исполнить одно выражение, если соблюдается какое-либо условие, и другое выражение в противном случае. Вот для это го применяется ELSE. ELSE расширяет возможности IF по части обработки вариантов выражения, когда оно равно FALSE. Данный пример выведет фразу 'a is bigger than b' если $a больше $b, и 'a is NOT bigger than b' в противном случае:

```
if ($a > $b)   {
print  "a  is  bigger  than  b";
}  else  {
print  "a  is  NOT bigger  than b";
}
```

Выражение ELSE выполняется только, если выражение IF равно FALSE, а если есть конструкции ELSEIF - то если и они также равны

FALSE
ELSEIF

ELSEIF, как и следует из его названия, является комбинацией IF и ELSE. ELSEIF, как и ELSE, позволяет выполнить выражение, если значение IF равно FALSE, но, в отличие от ELSE, оно выполнится только, если выражение ELSEIF равно TRUE. К примеру, следующий код выведет
'a is bigger than b' если $a>$b, 'a is equal to b', если $a=$b, и 'a is small-
er than b', если $a<$b:

```
if  ($a > $b)   {
print "a is bigger than b";
}  elseif  ($a == $b)   {
print "a is equal to b";
} else {
print "a is smaller than b";
}
```

Внутри одного выражения IF может быть несколько ELSEIF. Первое выражение ELSEIF (если таковые есть), которое будет равно TRUE, будет выполнено. В PHP 3 вы можете написать 'else if (два слова), что будет значить то же самое, что и 'elseif (одно слово).

Выражение ELSEIF будет выполнено только если выражение IF и все предыдущие ELSEIF равно FALSE, а данный ELSEIF равен TRUE.

Иной синтаксис для оператора IF: IFQ: — ENDIF;
PHP 3 предлагает иной путь для группирования операторов с оператором IF. Наиболее часто это используется, когда вы внедряете блоки HTML внутрь оператора IF, но вообще может использоваться где угодно.

Вместо использования фигурных скобок за "IF(выражение)" должно следовать двоеточие, одно или несколько выражений и завершающий ENDIF. Рассмотрите следующий пример:

```
 <?php if ($a==5):  ?> A = 5 <?php endif;  ?>
```

В этом примере блок "A = 5" внедрен внутрь выражения IF, альтернативным способом.

Блок HTML будет виден только, если На равно 5. Этот альтернативный синтаксис применим и к ELSE и ELSEIF (expr). Вот пример подобной структуры:

```
if  ($a == 5):
print  "a  equals  5";
print  "...";
elseif  ($a  ==  6):
print  "a  equals  6";
print  "!!!";
else:
print "a is neither 5 nor 6";
endif;
```

Циклы

Циклы WHILE

Цикл WHILE - простейший тип цикла в PHP 3. Он действует, как и его аналог в C. Основная форма оператора WHILE:

```
WHILE(expr)  statement
```

Смысл оператора WHILE прост. Он предписывает PHP выполнять вложенный(е) оператор(ы) до тех пор, пока expr равно TRUE. Значение выражения проверяется каждый раз при начале цикла, так что если значение выражения изменится внутри цикла, то он не прервется до конца текущей итерации (выполнение всего блока вложенных операторов – это одна итерация). Иногда, если значение expr равно FALSE с самого начала, цикл не выполняется ни разу.

Как и в IF, вы можете сгруппировать несколько операторов внутри фигурных скобок или использовать альтернативный синтаксис:

```
WHILE(expr):  выражения  ...  ENDWHILE;
```

Следующие примеры идентичны - оба выводят номера с 1 по 10:

```
/* example 1 */
$i = 1;
while ($i <= 10)   <
print $i++;     }
 /*  example  2  */
$i  =  1;
while  ($i  <=  10):
print  $i;
$i++;
endwhile;
```

Циклы DO..WHILE

Цикл DO..WHILE очень похож на WHILE, за исключением того, что значение логического выражения проверяется не до, а после окончания итерации. Основное отличие в том, что DO..WHILE гарантированно выполнится хотя бы один раз, что в случае WHILE не обязательно. Для циклов DO..WHILE существует только один вид синтаксиса:

```
$i = 0;
do {
print $i;
} while  ($i>0);
```

Этот цикл выполнится один раз, так как после окончания итерации будет проверено значение логического выражения, а оно равно FALSE ($i не больше 0), и выполнение цикла завершится. Опытные программисты на C, может быть, знакомы с иным использованием DO..WHILE, позволяющим прекратить исполнение блока операторов в середине путем

внедрения его в цикл DO..WHILE(0) и использования оператора BREAK. Следующий код демонстрирует такую возможность :

```
do {
if ($i < 5) {
print "i is not big enough";
break;
}
$i *= $factor;
if ($i < $minimum_limit) {
break;
}
print "i is ok";
...process i...
} while(0);
```

Не беспокойтесь, если вы не совсем поняли это. Удается программировать весьма мощные скрипты и без этой возможности.

Циклы FOR

Циклы FOR - наиболее мощные циклы в PHP. Они работают подобно их аналогам в C. Синтаксис цикла FOR:

```
FOR (exprl; expr2; exprЗ) statement
```

Первое выражение (exprl) безусловно вычисляется (выполняется) в начале цикла. В начале каждой итерации вычисляется expr2. Если оно равно TRUE, то цикл продолжается и выполняются вложенный(е) оператор(ы). Если оно равно FALSE, то цикл заканчивается. В конце каждой итерации вычисляется(исполняется) exprЗ. Каждое из этих выражений может быть пустым. Если expr2 пусто, то цикл продолжается бесконечно (PHP по умолчанию считает его равным TRUE, как и C). Это не так бесполезно, как могло бы показаться, так как зачастую вам требуется закончить выполнение цикла, используя оператор BREAK в сочетании с логическим условием вместо использования логического выражения в FOR.

Рассмотрим следующие примеры. Все они выводят номера с 1 по 10:

```
/* example 1 */
for ($i = 1; $i <= 10; $i++) {
print $i;
}
/* example 2 */
for ($i = l;;$i++) {
if ($i > 10) {
break;
}
print $i;
}
/* example 3 */
Si - 1;
for (;;) {
if ($i > 10) {
break;
}
print $i;
$i++;
I
/* example 4 */
for ($i = 1; $i <= 10; print $i, $i++) ;
```

Конечно, первый вариант кажется лучшим (или четвертый), но, как свидетельствует оный, возможность использования пустых выражений в и 11 кие FOR зачастую бывает полезной. PHP также поддерживает альтернативный синтаксис FOR:

```
FOR (exprl; expr2; expr3): выражение; ...; endfor;
```

BREAK и CONTINUE

BREAK

BREAK прерывает выполнение текущего цикла.

```
$i = 0;
while ($i < 10) {
if ($arr[$i] == "stop") {
break;
}
$i++;
}
```

CONTINUE

CONTINUE переходит на начало ближайшего цикла.

```
while (list($key,$value) = each($arr)) {
if ($key % 2) { // skip even members
continue;
}
do_something_odd ($value);
}
```

SWITCH

Оператор SWITCH похож на группу операторов IF с одинаковым выражением. Во многих случаях вам нужно сравнить переменную (или выражение) со многими различными значениями и выполнить различные фрагменты кода в зависимости от того, чему будет равно значение выражения. Это как раз то, для чего предназначается оператор SWITCH.

Следующие два примера это два различных пути для достижения одной цели, но в первом случае используется серия операторов IF, а во втором - оператор SWITCH.

```
/* example I */
if ($i == 0) {
print "i equals 0";
}
if ($i == 1) {
print "i equals 1";
}
if ($i =- 2) {
print "i equals 2";
}
 /* example 2 */
switch ($i) {
case 0:
print "i equals 0";
break;
case 1:
print "i equals 1";
break;
case 2:
print "i equals 2";
break;
}
```

Важно понять, как работает оператор SWITCH, чтобы избежать ошибок. SWITCH выполняет последовательно оператор за оператором. В начале код не исполняется. Только когда встречается оператор CASE с подходящим значением, PHP начинает выполнять программу. PHP продолжает выполнять операторы до конца блока SWITCH или пока не встретит оператор BREAK. Если вы не напишете BREAK в конце цикла операторов, то PHP продолжит выполнять операторы и следующего SWITCH'a. К примеру:

```
/* example 1 */
switch ($i) {
case 0:
print "i equals 0";
case 1:
print "i equals 1";
case 2:
print "i equals 2";
}
```

В этом случае, если $1 равно 0, то PHP выполнит все операторы |ч ml! Если $1 равно 1, то PHP выполнит последние два print. И только если $ i равно 2, вы получите ожидаемый результат и выведено будет только 'i equals 2'. Так что важно не забывать ставить BREAK (разве что в немнорых случаях вы специально не захотите их ставить для достижения определенной цели).

Специальный случай - это 'default case'. Этот оператор соответствует всем значениям, которые не удовлетворяют другим. К примеру:

```
/* example 4 */
switch ($i) {
case 0:
print "i equals 0";
break;
case 1:
print "i equals 1";
break;
case 2:
print "i equals 2";
break;
default:
print "i is not equal to 0, 1 or 2";
}
```

Другой заслуживающий упоминания факт - это то, что выражения в CASE могут быть любого скалярного типа, то есть целые числа или числа с плавающей запятой, а так же строки. Массивы и объекты не будут ошибкой, но в данном случае они не имеют значимости.

Функции

Функция может быть объявлена следующим образом:

```
function foo ($arg_l, $arg_2, ..., $arg_n) {
echo "Example function.\n";
return $retval;
}
```

Внутри функции может быть любой верный код PHP, даже объявление другой функции или класса. Функции должны быть определены перед тем, как на них ссылаться.

Возвращение результатов

Результаты возвращаются через необязательный оператор return. Возвращаемый результат может быть любого типа, включая списки и объекты.

```
function my_sqrt  ($num)   {
return $num * $num;
}
echo my_sqrt  (4);      // outputs   '16'.
```

Множественные результаты не могут быть возвращены в качестве результата, но вы можете реализовать это путем возврата списка:

```
 function fooO   {
return array  (O,  1,  2) ;
}
list   ($zero,  $one,  $two)  = food;
```

Аргументы

Информация может быть передана функции через список аргуменюв, которые являются разделенным запятыми списком переменных и/или констант.

РНРЗ поддерживает передачу аргументов по значению (по умолчанию), по ссылке и значения по умолчанию. Списки аргументов переменной длины не поддерживаются, но того же можно достичь, передавая массивы.

```
function takes_array($input)   {
echo "$input[0]  + $input[l]  = ",  $input[0]+$input[1] ;
}
```

Передача по ссылке

По умолчанию аргументы функции передаются по значению. Если им хотите в функции модифицировать аргументы, то можете передать их но ссылке. Если вы хотите, чтобы аргумент всегда передавался по ссылке, то • юдует поставить амперсанд (&) перед именем аргумента в объявлении функции:

```
function foo( &$bar )   {
$bar  .=  ' and something extra.';
}
$str =  'This is a string,   ';
foo  ($str);
echo  $str;     // выведет:  'This  is  a  string,  and
something  extra.'
```

Если вы хотите передать аргумент по ссылке в случае, когда по умолчанию такого не делается, то добавьте амперсанд перед именем аргумента в вызове функции:

```
function foo  ($bar)   {
$bar  .=  x and something extra.';
}
$str =  'This is a string,   ';
foo  ($str);
echo $str;       // выведет  'This is a string,
foo  (&$str);
echo  $str;       // выведет   'This  is  a  string,  and
something  extra.'
```

Значения по умолчанию

Функции могут определять значения по умолчанию для скалярных аргументов в стиле C++ как показано:

```
function makecoffee  ($type = "cappucino")   {
echo "Making a cup of $type.\n";
}
echo makecoffee  () ;
echo makecoffee   ("espresso");
```
Этот пример выведет следующее:
```
Making a cup of cappucino.
Making a cup of espresso.
```

Значение по умолчанию должно быть константой, а не переменной или, к примеру, членом класса.

Учтите, что, когда вы объявляете аргументы по умолчанию, они должны быть справа от всех не объявленных по умолчанию аргументов, в противном случае это не будет работать, как задумано.

К примеру:
```
function makeyogurt ($type = "acidophilus", $flavour)   {
return "Making a bowl of $type $flavour.\n";
}
echo makeyogurt  ("raspberry");      // не будет работать,
```

как ожидалось Этот пример выведет следующее:
```
Warning: Missing argument 2 in call to makeyogurt()   in
/usr/local/etc/httpd/htdocs/php3test/functest.html
on line 41
Making a bowl of raspberry .
```

А теперь сравните с этим:
```
function makeyogurt ($flavour,  $type = "acidophilus")   {
return "Making a bowl of $type $flavour.\n";
}
echo makeyogurt ("raspberry");      //а вот это работает
```

И выводит следующее:
```
Making a bowl of acidophilus raspberry.
```

Классы

Класс - это набор переменных и функций, работающих с этими переменными. Класс определяется следующим образом:

```
 <?php
class Cart {
var $items;     // Количество вещей в корзине
покупателя
// Добавить $num наименований типа $artnr в корзину
function add_item ($artnr,  $num)   {
$this->items[$artnr]   += $num;
}
// Убрать $num наименований    $artnr из корзины
function remove_item ($artnr,  $num)   {
if  ($this->items[$artnr]  > $num)   {
$this->items[$artnr]   -= $num;
return true;
} else {
return false;
}
}
I
?>
```

Это определения класса Cart, который состоит из связного списка наименований товаров в корзине и двух функций для добавления и удаления вещей из корзины. Классы - это типы, то есть заготовки для реальных переменных. Вы 1жны создавать переменные желаемого типа, используя оператор new:

```
$cart = new Cart;
$cart->add_item("10",  1);
```

Таким образом, мы создали объект $cart класса Cart. Функция i item () этого объекта вызывается для добавления одного товара но10 к корзине. Классы могут быть расширениями других классов. Расширенный юс обладает всеми переменными и функциями базового класса и тем, мы определите при расширении класса. Это делается посредством использования ключевого слова extends:

```
class Named_Cart extends Cart {
var $owner;
function set_owner  ($name)   {
$this->owner = $name;
}
}
```

6. Введение в использование MySQL

MySQL - компактный многопоточный сервер баз данных. MySQL характеризуется большой скоростью, устойчивостью и легкостью в использовании. MySQL был разработан компанией TcX для внутренних нужд, которые заключались в быстрой обработке очень больших баз данных. Компания утверждает, что использует MySQL с 1996 года на сервере с более чем 40 БД, которые содержат 10,000 таблиц, из которых более чем 500 имеют более 7 миллионов строк.

MySQL является идеальным решением для малых и средних приложений. Исходники сервера компилируются на множестве платформ. Наиболее полно возможности сервера проявляются на Unix-серверах, где есть поддержка многопоточности, что дает значительный прирост производительности.

MySQL-сервер является бесплатным для некоммерческого использования. Иначе необходимо приобретение лицензии, стоимость которой составляет 190 EUR.

6.1. Возможности MySQL

MySQL поддерживает язык запросов SQL в стандарте ANSI 92, и кроме этого имеет множество расширений к этому стандарту, которых нет ни в одной другой СУБД.

Краткий перечень возможностей MySQL.

1. Поддерживается неограниченное количество пользователей, одновременно работающих с базой данных.
2. Количество строк в таблицах может достигать 50 млн.
3. Быстрое выполнение команд. Возможно MySQL самый быстрый сервер из существующих.
4. Простая и эффективная система безопасности.

MySQL действительно очень быстрый сервер, но для достижения этого разработчикам пришлось пожертвовать некоторыми требованиями к реляционным СУБД. В MySQL отсутствуют:

5. Поддержка вложенных запросов, типа SELECT * FROM table1 WHERE id IN (SELECT id FROM table2). Утверждается, что такая возможность будет в версии 3.23.
6. Не реализована поддержка транзакций. Взамен предлагается использовать LOCK/UNLOCK TABLE.
7. Нет поддержки внешних (foreign) ключей.
8. Нет поддержки триггеров и хранимых процедур.
9. Нет поддержки представлений (VIEW). В версии 3.23 планируется возможность создавать представления.

Примеры использования PHP+MySQL

Работа с формами.

В этом примере показано как в PHP легко обрабатывать данные с HTML - форм.

Создадим простой HTML файл.

```
<HTML>
<HEAD>
<TITLE>Запрос информации</TITLE>
<BODY>
<CENTER>
Хотите больше знать о наших товарах?
<P>
<TABLE WIDTH = 400><TR><TD align = right>
<FORM ACTION="email.php3" METHOD="POST">
Ваше имя:<BR>
<INPUT TYPE="text" NAME="name" SIZE="20" MAXLENGTH="30">
<P>
Ваш email:<BR>
<INPUT TYPE="text" NAME="email" SIZE="20" MAXLENGTH="30">
<P>
Меня интересуют:
<SELECT NAME="preference">
<OPTION value = "Яблоки">Яблоки
<OPTION value = "Апельсины">Апельсины
</SELECT>
<P>
<INPUT TYPE="submit" VALUE="Отправить запрос!">
</FORM>
</TD></TR></TABLE></CENTER>
</BODY>
</HTML>
```

Назовем этот файл request.html. В нем мы указали, что данные формы будут обрабатываться файлом email.php3. Приведем его содержание:

```
<?
/* Этот скрипт получает переменные из request.html */
PRINT "<CENTER>";
PRINT "Привет, $name.";
PRINT "<BR><BR>";
PRINT "Спасибо за ваш интерес.<BR><BR>";
PRINT "Вас интересуют $preference. Информацию о них мы пошлем вам на email: $email.";
PRINT "</CENTER>";
?>
```

Литература

1. HTML 2.0 (RFC 1866)
2. Рихард Айзенменгер HTML 3.2/4.0 Справочник Пер. с нем. - М.: ЗАО “Издательство БИНОМ”, 1998. - 368 с.: ил.
3. Луиза Паттерсон и др. Использование HTML 4: Пер. с англ. - 3-е изд. - К.; М.; СПб.: Издат. дом “Вильямс” 1998. -384 с.: ил.
4. Брюс Морис HTML в действии Пер. с англ. - СПб.: “Питер”, 1997. - 256 с.: ил.
5. Бабушкин М. и др. Web-сервер в действии - СПб: “Питер”, 1997 - 272 с.: ил.
6. Ярмола Ю.А. Компьютерные шрифты – Спб: “BVH-Санкт-Петербург”, 1994 – 208 с.: ил.
7. Цвет в науке и технике М.: 1979
8. Дж. Мелони PHP 4 в действии – М.: “Лучшие Книги”, 2002 - 400 с.
9. Т. Ратшиллер, Т. Геркен PHP 4. Разработка Web-приложений - СПб: “Питер”, 2001 - 384 с.
10. С. Хьюгс, А. Змиевский PHP 4. Руководство разработчика - М.: “ДиаСофт”, 2001 - 384 с.
11. Игорь Григин PHP 4. Специальный справочник - СПб: “Питер”, 2002 - 672 с.
12. Поль Дюбуа MySQL - М.: “Вильямс”, 2004 - 1056 с.

Приложения

Приложение 1

Символы и их обозначения (escape-последовательности) в HTML

Название	Символ	&-ASCII	&-ИМЯ
Табулятор		&#009	
Перевод строки		&#010	
Возврат каретки		&#013	
Пробел		&#032	
Восклицательный знак	I	&#033	
Кавычка	U	&#034	"
Знак "решетка"	#	&#035	
Доллар	$	&#036	
Процент	%	&#037	
Амперсанд	&	&#038	&
Апостроф	c	&#039	
Скобка левая круглая	(	&#040	
Скобка правая круглая	)	&#041	
Звездочка	*	&#042	
Плюс	+	&#043	
Запятая	»	&#044	
Минус	-	&#045	
Точка	.	&#046	
Косой штрих	/	&#047	
Цифры	0-9	&#048-057	
Двоеточие		&#058	
Точка с запятой	»	&#059	
Меньше	<	&#060	&lt
Равно	=	&#061	
Больше	>	&#062	&gt
Знак вопроса	?	&#063	
Коммерческая эт ("обезьянка", "собачка")	@	&#064	
Прописные буквы	A-Z	&#065-090	
Скобка квадратная левая	[	&#091	
Косой штрих с наклоном вправо	\	&#092	
Скобка квадратная правая	]	&#093	
Степень	Л	&#094	
Знак подчеркивания		&#095	
Ударение	'	&#096	
Строчные буквы	a - z	&#097-122	

Фигурная скобка левая	{	&#123	
Вертикальный штрих	1	&#124	
Фигурная скобка правая	}	&#125	
Тильда	~	&#126	
Запятая	5	&#130	
Флорин	9	&#131	
Кавычка нижняя правая	95	&#132	
Многоточие	• * .	&#133	
Крест	t	&#134	
Двойной крест	t	&#135	
Крышка	л	&#136	
Промилле	%0	&#137	
Большая S с перевернутой "крышкой"	s	&#138	
Символ меньше	<	&#139	
Большое OE	H>	«fe#140	
Одинарная кавычка слева	t	&#145	
Одинарная кавычка справа	*1*	&#146	
Кавычка слева	1C	&#147	
Кавычка справа	55	&#148	
Крупная точка	•	&#149	
Короткий дефис	—	&#150	
Длинный дефис	——	&#151	
Тильда	~	&#152	
Торговый знак	TM	&#153	
Маленькая s перевернутой "крышкой"	Jb	&#154	
Символ больше	>	&#155	
Маленькое oe	H>	&#156	
Большой Y умляут	Ц	&#159	
Неразрывный пробел		Ml 60	
Перевернутый восклицательный знак	v_*	&#161	&ixcl;
Цент	У	*Ml 62*	¢
Фунт	J	&#163	£
Валюта	a	&#164	¤
Иена	r	*&Ш5*	¥
Прерванный штрих	i i	&#166	«febrvbar;
Параграф/Раздел	§	&#167	§
Точка умляута	Ё	&#168	¨
Копирайт	©	&#169	©
Порядковое числительное женского рода	e	&#170	ª
Кавычка "ёлочки" левая	u	&#171	«
Не/Нет	-	&#172	¬
Дефис	-	&#173	

Зарегистрированный торговый знак	®	&#174	®
Большая U с ударением	Щ	&#217	Ù
Большая U с ударением	ъ	&#218	Ú
Большая U с "крышкой"	ы	&#219	Û
Большая U умляут	ь	&#220	Ü
Большая Y с ударением	э	&#221	Ý
Большой Thorn	ю	&#222	Þ
Резкая "B"	я	&#223	ß
Маленькая a с ударением	а	&#224	à
Маленькая a с ударением	б	&#225	á
Маленькая a с "крышкой"	в	&#226	â
Маленькая a с тильдой	г	&#227	ã
Маленькая a умляут	д	&#228	ä
Маленькая a с кружком	е	&#229	å
Маленькое ae	ж	&#230	æ
Маленькая c-цедил	З	&#231	ç
Маленькая e с ударением	и	&#232	è
Маленькая e с ударением	w	&#233	é
Маленькая e с "крышкой"	К	&#234	<feecirc;
Маленькая e умляут	л	&#235	ë
Маленькая i с ударением	м	&#236	ì
Маленькая i с ударением	н	&#237	í
Маленькая i с "крышкой"	о	&#238	î
Маленькая i умляут	п	&#239	ï
Маленькое eth	р	&#240	ð
Маленькая n с тильдой	с	&#241	ñ
Маленькая o с ударением	т	&#242	ò
Маленькая o с ударением	У	&#243	«feoacute;
Маленькая o с "крышкой"	ф	&#244	«feocirc;
Маленькая o с тильдой	Х	&#245	õ
Маленькая o умляут	ц	&#246	ö
Знак деления	ч	&#247	÷
Маленькая o зачеркнутая	ш	&#248	ø
Маленькая и с ударением	Щ	&#249	ù
Маленькая и с ударением	ъ	&#250	ú
Маленькая и с "крышкой"	ы	&#251	û
Маленькая и умляут	ь	&#252	ü
Маленькая y с ударением	э	&#253	&y acute;
Маленькая thorn	ю	&#254	&þ
Маленькая y умляут	я	&#255	ÿ

Расширенная таблица цветов

Название	R (красный)	G (зеленый)	B (синий)
aliceblue	f0	ffi	ff
antiquewhite	fa	eb	d7
aqua	0	ff	ff
aquamarine	7f	ff	d4
azure	f0	ff	ff
beige	f5	f5	dc
bisque	ff	e4	c4
black	0	0	0
blanchedalmond	ff	eb	cd
blue	0	0	ff
bluevilet	8a	2b	e2
brown	a5	2a	2a
burlywood	de	b8	87
cadetblue	5f	9e	aO
chartreuse	7f	ff	0
chocolate	d2	69	le
coral	ff	7f	50
cornflowerblue	64	95	ed
cornsilk	ff	fB	dc
crimson	dc	14	3c
cyan	0	ff	ff
darkblue	0	0	8b
darkcyan	0	8b	8b
darkgoldenrod	b8	86	Ob
darkgrey	a9	a9	a9
darkgreen	0	64	0
darkhaki	bd	b7	6b
darkmagneta	8b	0	8b
darkolivengreen	55	6b	2f
darkorange	ff	8c	0
darkorchid	99	32	cc
darkred	8b	0	0
darksalmon	e9	96	7a
darkseagreen	8f	be	8f
darkslateblue	48	3d	8b
darkslategrey	2f	4f	4f

darkturquoise	0	ce	dl
darkviolet	94	0	d3
deeppink	ff	14	93
deepskyblue	0	bf	ff
dimgrey	le	90	ff
dodgerblue	le	90	ff
firebrick	b2	22	22
floralwhite	ff	fa	f0
forestgreen	22	8b	22
fuchsia	ff	0	ff
gainsboro	dc	dc	dc
ghostwhite	dc	dc	dc
gold	ff	d7	0
goldenrod	da	a5	20
gray	80	80	80
greeny el low	ad	ff	2f
honeydew	III	ff	fO
hotpink	ff	69	b4
indianred	cd	5c	5c
indigo	4b	0	82
ivory	ff	ff	fD
khaki	f0	e6	8c
lavender	e6	eб	fa
lavenderblush	ff	III	f5
lawngreen	7c	fc	0
lemonchiffon	ff	fa	cd
lightblue	ad	d8	eб
lightcoral	fO	80	80
lightcyan	eO	ff	ff
lightgoldenrodyel-low	fa	fa	d2
lightgreen	90	ee	90
lightgrey	d3	d3	d3
lightpink	ff	b6	cl
lightsalmon	ff	aO	7a
lightseagreen	20	b2	aa
lightkyblue	87	ce	fa
lightslategray	77	88	99
lightsteelblue	bO	c4	de
lightyellow	ff	ff	e0
lime	0	ff	0

limegreen	32	cd	32
linen	fa	fO	e6
magenta	ff	0	ff
maroon	80	0	0
mediumaquama-rine	66	cd	aa
mediumblue	0	0	cd
mediumorchid	ba	55	d3
mediumpuϕle	93	70	db
mediumseagreen	3c	b3	71
mediumslateblue	7b	68	ee
mediumspring-green	0	fa	9a
mediumturquois	48	dl	cc
mediumvioletred	c7	15	85
midnightblue	19	19	70
mintcream	f5	ff	fa
mistyrose	ff	e4	el
moccasin	ff	e4	b5
navajowhite	ff	de	ad
navy	0	0	80
oldlace	fd	f5	e6
olive	80	80	0
olivedrab	6b	8e	23
orange	ff	a5	0
orangered	ff	45	0
orchid	da	70	d6
palegoldenrod	ee	e8	aa
palegreen	98	fb	98
paleturquoise	at	ee	ee
palevioletred	db	70	93
papayawhip	ff	ef	d5
peachpuff	dd	da	b9
peru	cd	85	3f
pink	ff	c0	cb
plum	dd	a0	dd
powderblue	bO	e0	e6
purple	80	0	80
red	ff	0	0
rosybrown	be	8f	8f
royalblue	41	r/J	el
saddlebrown	8b	45	13

salmon	fa	80	72
sandybrown	f4	a4	60
seagreen	2e	8b	57
seashell	ff	f5	ee
sienna	a0	52	2d
silver	c0	c0	c0
skyblue	87	ce	eb
slateblue	6a	5a	cd
snow	ff	fa	fa
springgreen	0	ff	7f
steelblue	46	82	b4
tan	d2	b4	8c
teal	0	80	80
thistle	d8	bf	d8
tomato	ff	63	47
turquoise	40	e0	d0
violet	ee	82	ee
wheat	d5	de	Ь3
whitesmoke	f5	f5	£5
yellow	ff	ff	0
yellowgreen	9a	cd	32

Приложение 3

Коды представления в URL специальных символов

Символ	Описание	Представление в URL
5	Точка с запятой	%3Ь
/	Косая черта (слэш)	%2f
?	Знак вопроса	%3f
,	Двоеточие	%3a
@	Коммерческое эт	%40
=	Знак равенства	%3d
&	Логическое И (амперсанд)	%26
<	Знак меньше	%3c
>	Знак больше	%3e
a	Кавычка	%22
#	Решетка	%23
%	Проценты	%25
{	Открывающая фигурная скобка	%7Ь
}	Закрывающая фигурная скобка	%7d
1	Логическое ИЛИ	%7c
\	Обратная косая черта (бэкслэш)	%5c
Л	Степень	%5e
~	Тильда	%7e
[	Открывающая квадратная скобка	%5Ь
]	Закрывающая квадратная скобка	%5d
i	Апостроф	%60

Приложение 4

Коды указания на вид языка

(Более подробно см. RFC 1766, IS0639, IS03166)

Код	**Язык**
cz	Чешский
de	Немецкий
el	Греческий
en	Английский
en-US	Английский США
es	Испанский
fr	Французский
gd	Шотландский гаэльский
he или iw	Иврит
hi	Хинди
hu	Венгерский
it	Итальянский
ja	Японский
pt	Португальский
ru	Русский
ur	Урду
zh	Китайский

Глоссарий терминов

ANSI (American National Standards Institute) - Американский Национальный Институт Стандартизации.
SGML (Standard Generalized Markup Language) -Обобщенный метаязык разметки документов
NCSA (National Center for Supercomputer Applications) - Национальный центр суперкомпьютерных приложений США.
IETF (Internet Engineering Task Force) - Международная комиссия по стандартам в Internet.
ISO (International Standard Organization) -Международная организация по стандартизации.
W3C (World Wide Web Consortium) -Консорциум всемирной паутины - некоммерческая организация занимающаяся разработкой и реализацией стандартов HTML и WWW.