Владимирский государственный университет

Е. А. ТРОИЦКАЯ Л. А. АРТЮШИНА

WEB-ПРОГРАММИРОВАНИЕ

Учебное пособие

Владимир 2022

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Е. А. ТРОИЦКАЯ Л. А. АРТЮШИНА

WEB-ПРОГРАММИРОВАНИЕ

Учебное пособие

Электронное издание

Владимир 2022

ISBN 978-5-9984-1602-6

УДК 004.438
ББК 32.973.4

Рецензенты:
Кандидат физико-математических наук
доцент кафедры прикладной математики
Московского государственнного технического университета
гражданской авиации
О. В. Крисько

Доктор технических наук, профессор
зав. кафедрой информационных систем и программной инженерии
Владимирского государственного универстета
имени Александра Григорьевича и Николая Григорьевича Столетовых
И. Е. Жигалов

Троицкая, Е. А. WEB-программирование [Электронный ресурс] : учеб. пособие / Е. А. Троицкая, Л. А. Артюшина ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир: Изд-во ВлГУ, 2022. – 344 с. – ISBN 978-5-9984-1602-6. – Электрон. дан. (4,02 Мб). – 1 электрон. опт. диск (CD-ROM). – Систем. требования: Intel от 1,3 ГГц ; Windows XP/7/8/10 ; Adobe Reader ; дисковод CD-ROM. – Загл. с титул. экрана.

Рассматриваются вопросы, связанные с инструментальными средствами создания web-сайтов, основными понятиями и принципами их функционирования. Даются общий обзор языка JavaScript и основы технологии программирования на PHP. Разработано в соответствии с программой дисциплины «WEB-программирование».

Предназначено для студентов бакалавриата очной и заочной форм обучения по направлению 38.03.05 «Бизнес-информатика», будет полезно при подготовке семинарских занятий, курсовых проектов, отчетов по практике.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС ВО.

Табл. 24. Ил. 19. Библиогр.: 10 назв.

ISBN 978-5-9984-1602-6

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цель преподавания дисциплины "Web-программирование" – освоение обучающимися практических приемов Web-конструирования и Web-программирования, а также закрепление знаний о принципах функционирования глобальной компьютерной сети Internet, знакомство с общими подходами к поиску и отбору информации в сети; обучение разработке Web-страниц на основе комплексного подхода, программированию в Internet на стороне клиента и сервера, способам маркетинга, рекламы и продвижения разработанных Internet-ресурсов.

Развитие заложенного в студентах научно-исследовательского и технологического компонентов мышления в области разработки и продвижения сайтов представляет собой основную задачу изучения материала данного курса.

Развитие средств вычислительной техники и телекоммуникаций предопределяет курс на расширение знаний и способов владения навыками работы со специальным программным и аппаратным обеспечением информационных сетей и средств телекоммуникаций.

Основными задачами преподавания дисциплины являются систематизация и обобщение знаний и информации о современных тенденциях языков Web-программирования и баз данных для хранения онлайн информации.

В результате изучения курса обучающийся должен овладеть навыками работы с компьютером как средством управления информацией, с информацией из различных источников, в том числе в глобальных компьютерных сетях.

Достижение этих целей предполагает изучение студентами следующих модулей:

Модуль 1. Введение в Web-конструирование

Модуль 2. Инструментальные средства создания web-сайтов

Модуль 3. Программирование на JavaScript

Модуль 4. Программирование на PHP. MySQL & PHP.

К каждому модулю разработаны планы практических и семинарских занятий, даются контрольные вопросы, тестовые задания, приведен библиографический список литературы, представлен глоссарий.

Модуль 1. ВВЕДЕНИЕ В WEB-КОНСТРУИРОВАНИЕ

Тема 1. Введение в Web-конструирование

Основные понятия, принципы функционирования

Глобальная сеть - это протяженная коммуникационная сеть связи, работа в которой обеспечивается с помощью телекоммуникационных компаний.

Основными ячейками глобальной сети являются ***локальные вычислительные сети***. При этом локальные сети могут входить как компоненты в состав региональной сети, региональные сети - в состав глобальной сети. Существуют также компьютеры, самостоятельно (непосредственно) подключенные к глобальной сети. Они называются ***хост-компьютерами.***

Глобальные сети, объединяя пользователей, расположенных в разных странах, на различных континентах, позволяют решить проблему объединения информационных ресурсов всего человечества и организации доступа к этим ресурсам.

В настоящее время в мире зарегистрировано более 200 глобальных компьютерных сетей, но крупнейшей из них является сеть Internet.

Глобальная компьютерная сеть Internet

Интернет (англ. **Internet,** от лат. *inter* - между и англ. *net* - сеть) - это глобальная компьютерная сеть, которая объединяет в единое целое множество компьютерных сетей и отдельных компьютеров, предоставляющих обширную информацию в общее пользование и не является коммерческой организацией.

Относительно строгое определение Интернета с технической точки зрения можно дать следующее:

Internet - это метасеть, состоящая из многих сетей, которые работают согласно протоколам семейства TCP/IP, объединены через шлюзы, используют единое адресное пространство и пространство имен. ***Метасеть*** - это сеть, в которой характер и топология сетевых связей различны и не имеют строго определенной структуры.

Потребности формирования единого мирового информационного пространства привели к объединению локальных, региональных и корпоративных сетей в глобальную компьютерную сеть Интернет.

Надежность функционирования глобальной сети обеспечивает большое количество каналов передачи информации с высокой пропускной способностью между локальными, региональными и корпоративными сетями. Например, российская региональная компьютерная сеть Рунет (RU) соединяется многочисленными каналами передачи информации с северо-американской (US), европейской (EU) и японской (JP) региональными сетями (рис. 1).

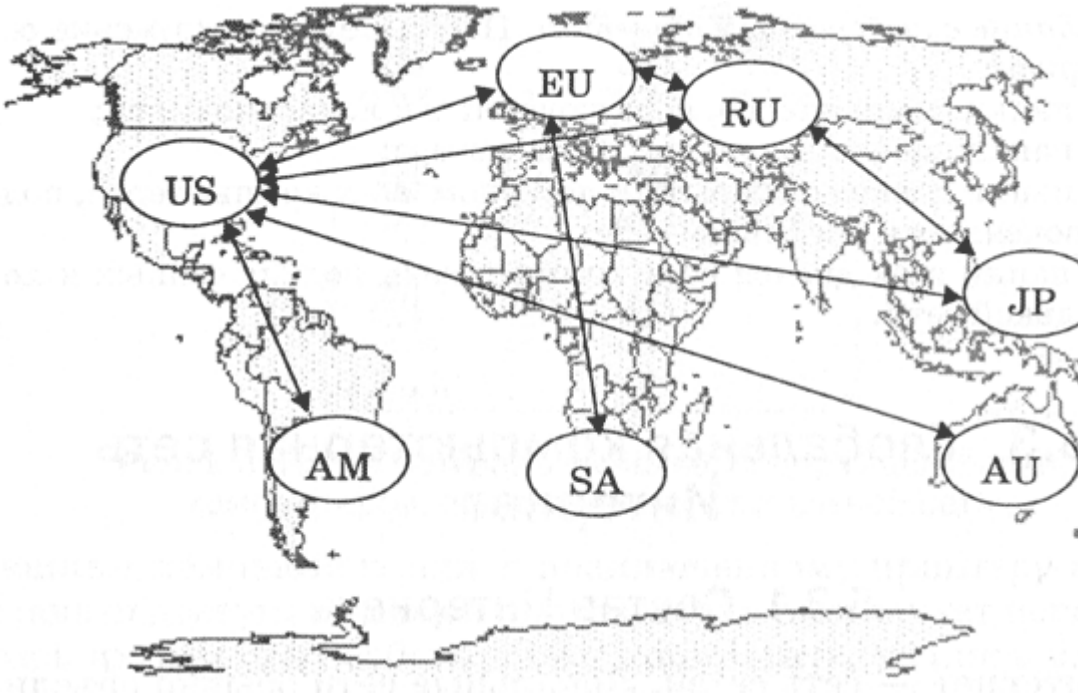


Рис. 1. Региональные компьютерные сети, объединенные в глобальную сеть Интернет

Подключение к Интернету. В каждой локальной, региональной или корпоративной сети имеется, по крайней мере, один компьютер (сервер Интернета), который имеет постоянное подключение к Интернету.

Для подключения локальных сетей чаще всего используются **оптоволоконные линии** связи. Однако в случаях подключения неудобно расположенных или удаленных компьютерных сетей, когда прокладка кабеля затруднена или невозможна, используются беспроводные линии связи. Если передающая и принимающая антенны находятся в пределах прямой видимости, то используются **радиоканалы**, в противном случае обмен информацией производится через **спутниковый канал** с использованием специальных антенн (рис. 2).

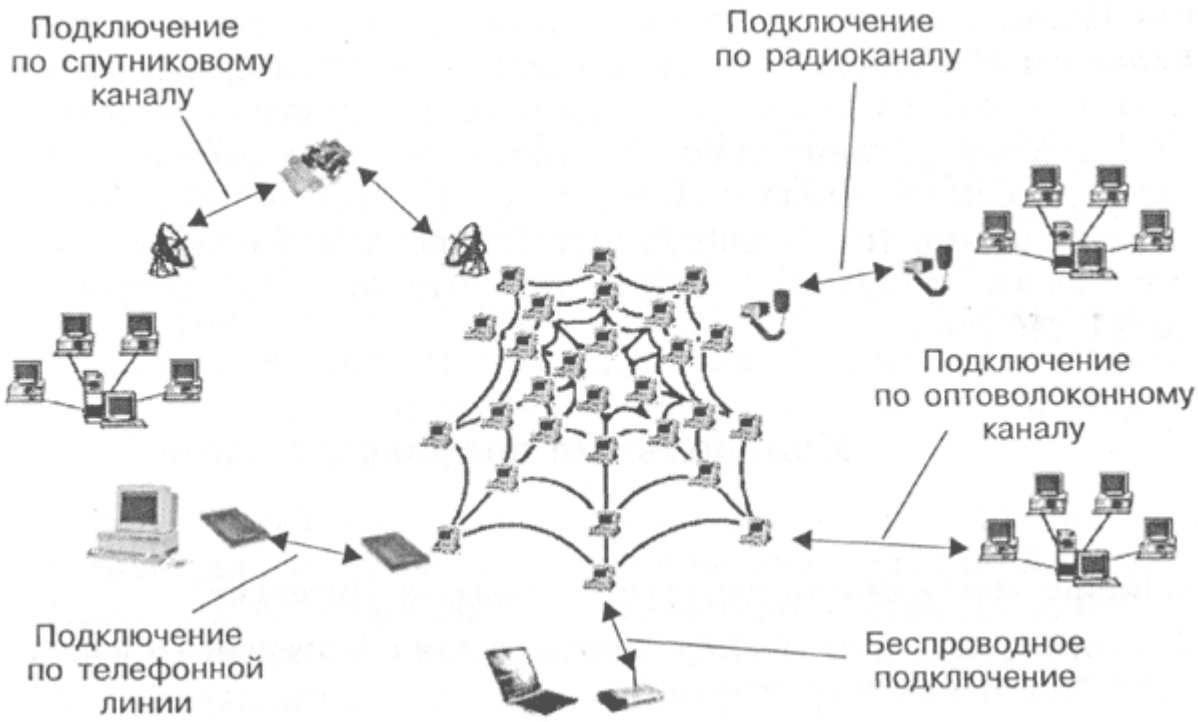


Рис. 2. Различные варианты подключения к глобальной компьютерной сети Интернет

Сотни миллионов компьютеров пользователей могут периодически подключаться к Интернету по **коммутируемым телефонным каналам** с помощью **провайдеров Интернета**. Провайдеры Интернета имеют высокоскоростные соединения своих серверов с Интернетом и поэтому могут предоставить Интернет-доступ по телефонным каналам одновременно сотням и тысячам пользователей.

Для соединения компьютера пользователя по телефонному каналу с сервером Интернет-провайдера к обоим компьютерам должны быть подключены модемы. Модемы обеспечивают передачу цифровых компьютерных данных по аналоговым телефонным каналам со скоростью до 56 Кбит/с.

Современные ADSL-технологии позволяют использовать обычные телефонные каналы для высокоскоростного (1 Мбит/с и выше) подключения к Интернету. Важно, что при этом телефонный номер остается свободным.

Пользователи портативных компьютеров могут подключаться к Интернету с использованием беспроводной технологии Wi-Fi. На вокзалах, в аэропортах и других общественных местах устанавливаются

точки доступа беспроводной связи, подключенные к Интернету. В радиусе 100 м портативный компьютер, оснащенный беспроводной связью, автоматически получает доступ в Интернет со скоростью до 11 Мбит/с.

В 1957 году в рамках Министерства обороны США выделилась отдельная структура — Агентство передовых исследовательских проектов (Advanced Research Projects Agency, DARPA). Основные работы DARPA были посвящены разработке метода соединений компьютеров друг с другом. Глобальная сеть Интернет начала развиваться на основе сети ARPAnet (Advanced Research Project Agency), созданной DARPA в 1969 году.

Эта сеть была предназначенная для связи различных научных центров, военных учреждений и оборонных предприятий. Для своего времени ARPAnet была передовой и необычайно устойчивой к внешним воздействиям закрытой системой. С ее помощью планировалось облегчить процесс общения многочисленных организаций, работающих на оборонную промышленность, а также создать практически не поддающиеся разрушению каналы связи. В частности, при создании ARPAnet предполагалось, что данная система продолжит функционировать и в условиях ядерного нападения.

В основу проекта были положены три базовые идеи:

— каждый узел сети соединен с другими, так что существует несколько различных путей от узла к узлу;

— все узлы и связи рассматриваются как ненадежные — существуют автоматически обновляемые таблицы перенаправления пакетов;

— пакет, предназначенный для несоседнего узла, отправляется на ближайший к нему, согласно таблице перенаправления пакетов, при недоступности этого узла — на следующий и т. д.

Эти идеи должны были обеспечить функционирование сети в случае разрушения любого числа ее компонентов. В принципе, сеть можно было считать работоспособной даже в случае, если будут функционировать всего два компьютера. Созданная по такому принципу система не имела централизованного узла управления и, следовательно, могла легко изменять конфигурацию без малейшего для себя ущерба.

Первоначально сеть состояла из 17 мини-компьютеров. Память каждого имела объем 12 Кб. В апреле 1971 года к сети было подключено 15 узлов. В 1975 году сеть ARPAnet включала уже 63 узла.

В середине 1972 года среди пользователей сети стало распространяться мнение, что передать письмо по компьютерной сети намного быстрее и дешевле, чем традиционным методом. Так начала зарождаться электронная почта — сервис, без которого сегодня невозможно представить Интернет.

Вскоре появляется программа UUCP (Unix-to-Unix Copy). Это привело к созданию следующего сервиса— USEnet (сетевые новости). Именно так первоначально называлась сеть, позволяющая пользователю войти в компьютер, где размещалась информация, и выбрать оттуда все интересующие его материалы. Уже на начальном этапе развития количество пользователей сети USEnet ежегодно утраивалось.

Достаточно быстро архитектура и принципы сети ARPAnet перестали удовлетворять выдвинутым требованиям. Возникла необходимость создания универсального протокола передачи данных.

В 1974 году Internet Network Working Group (INWG), созданная DARPA, разработала универсальный протокол передачи данных и объединения сетей Transmission Control Protocol/Internet Protocol (TCP/IP), являющийся основой функционирования Интернет. В 1983 году DARPA обязала использовать на всех компьютерах ARPAnet протокол TCP/IP, на базе которого Министерство обороны США разделило сеть на две части: отдельно для военных целей — MI Lnet и научных исследований — сеть ARPAnet.

Первоначально сеть была ориентирована только на пересылку файлов и неформатированного текста. Однако для работы многих пользователей была необходима инфраструктура, позволяющая работать в более удобном режиме. В частности, обмениваться результатами исследований через сеть Интернет в виде привычного для научных работников отформатированного и иллюстрированного текста, включающего ссылки на другие публикации. В 1989 году в Европейской лаборатории физики элементарных частиц (CERN, Швейцарии, Женева) была разработана технология гипертекстовых документов World Wide Web, позволяющая получать доступ к любой информации, находящейся в сети на компьютерах по всему миру. Так было положено начало Всемирной Информационной Паутине, которая к настоящему

времени «оплела» своими сетями практически весь компьютерный мир и сделала Интернет доступным и привлекательным для миллионов пользователей.

В 1990 году сеть ARPAnet перестала существовать, и на ее месте возникла сеть Интернет.

Основные особенности сети Интернет:

- универсальность концепции, независящей от внутреннего устройства объединяемых сетей и типов аппаратного и программного обеспечения;

- максимальная надежность связи при заведомо низком качестве коммуникаций, средств связи и оборудования;

- возможность передачи больших объемов информации. Быстрое расширение сети привело к проблемам диапазонов, не предусмотренным в исходном проекте, и заставило разработчиков найти технологии для управления большими распределенными ресурсами.

В первоначальном проекте имена и адреса всех компьютеров, присоединенных к Интернет, хранились в одном файле, который редактировался вручную и затем распространялся по всей сети Интернет. Но уже в середине 1980 года стало ясно, что центральная база данных неэффективна. Во-первых, запросы на обновление файла скоро должны были превысить возможности людей, обрабатывавших их. Во-вторых, даже если существовал корректный центральный файл, не хватало пропускной способности сети, чтобы позволить либо частое распределение его по всем местам, либо оперативный доступ к нему из каждого места.

Были разработаны новые протоколы, и стала использоваться система имен по всей объединенной сети Интернет, которая позволяла любому пользователю автоматически определять адрес удаленной машины по ее имени. Известный как доменная система имен (DNS), этот механизм основывается на машинах, называемых серверами имен, отвечающих на запросы об именах. Нет одной машины, содержащей всю базу данных об именах. Вместо этого данные распределены по нескольким машинам, которые используют протоколы TCP/IP для связи между собой при ответе на запросы.

Таким образом на сегодняшний день сеть Интернет представляет собой объединение огромного числа различных компьютерных сетей практически по всему миру.

Для того чтобы в процессе обмена информацией компьютеры могли найти друг друга, в Интернете существует единая система адресации, основанная на использовании Интернет-адресов.

Все компьютеры, включенные во всемирную сеть, работают в автоматическом режиме, то есть без участия людей, но чтобы было можно однозначно обозначить любой компьютер в Интернете, применяется специальная система адресов, называемая ***IP-адресами.*** Хотя нет центра управления Интернетом, но есть специальные организации, занимающиеся проверкой и выдачей адресов (например, информационный центр Интернета - InterNIC).

Каждый компьютер, подключенный к Интернету, имеет свой уникальный двоичный 32-битовый ***Интернет-адрес.***

Существует формула, которая связывает между собой количество возможных информационных сообщений N и количество информации I, которое несет полученное сообщение:

$N = 2^I$.

Интернет-адрес несет количество информации I = 32 бита, тогда общее количество N различных Интернет-адресов равно:

$N = 2^I = 2^{32} = 4\ 294\ 967\ 296$

Интернет-адрес длиной 32 бита позволяет подключить к Интернету более 4 миллиардов компьютеров.

По новой технологии "Умный дом" к Интернету подключены не только компьютеры, но и бытовые приборы (холодильники, стиральные машины и др.) и аудио- и видеотехника, которыми можно будет управлять дистанционно. В этом случае четырех миллиардов Интернет-адресов может оказаться недостаточно и придется перейти на более длинный Интернет-адрес.

Для удобства восприятия двоичный 32-битовый Интернет-адрес можно разбить на четыре части по 8 битов и каждую часть представить в десятичной форме. Десятичный Интернет-адрес состоит из четырех чисел в диапазоне от 0 до 255, разделенных точками (например, 213.171.37.202) (табл. 1).

Таблица 1. Интернет-адрес в двоичной и десятичной форме

Двоичный	11010101	10101011	00100101	11001010
Десятичный	213	171	37	202

Все серверы Интернета имеют постоянные Интернет-адреса. Однако провайдеры Интернета часто предоставляют пользователям доступ в Интернет не с постоянным, а с временным Интернет-адресом. Интернет-адрес может меняться при каждом подключении к Интернету, но в процессе сеанса остается неизменным и пользователь может его определить.

Доменная система имен. Человеку запомнить числовой адрес нелегко, поэтому для удобства пользователей Интернета была введена доменная система имен, которая ставит в соответствие числовому Интернет-адресу компьютера уникальное доменное имя.

Компьютеры при пересылке информации используют цифровые адреса, а пользователи при работе с Интернетом используют в основном имена, поскольку адреса, образованные из слов, запомнить гораздо проще. В Интернете применяется так **называемая доменная (или многоуровневая) система имен (DNS).**

DNS (*Domain Name System* - доменная система имен) - это база данных, обеспечивающая преобразование доменных имен компьютеров, подключенных к Интернет, в числовые IP-адреса. После ввода пользователем доменного имени компьютер обращается к серверам DNS, в результате чего происходит автоматическое преобразование доменного имени в цифровой адрес.

Домен (*domain*) - это отдельный уровень в многоуровневой системе имен Интернета, несущий определенную информационную нагрузку. Под понятием ***домен*** можно понимать совокупность компьютеров в составе сети, объединенных каким-либо общим признаком (например, находящихся в одном государстве, принадлежащих одной фирме и т.п.). Доменная система имен в Интернете использует принцип последовательных уточнений. **Домен верхнего уровня располагается в имени правее, а домен нижнего уровня левее.** В имени может быть любое число доменов, но чаще всего используются имена с количеством доменов от трех до пяти. Домены состоят из поддоменов (subdomain), имена которых разделяются точками. Часто домен 1-ого уровня - указывает на страну, 2-ого уровня - на город, 3-го уровня - на компанию (организацию); если имя города отсутствует, имя компании становится доменом 2 уровня.

Домены верхнего уровня существуют двух типов: географические и административные. Каждой стране мира выделен свой географический домен, обозначаемый двухбуквенным кодом. Например, России принадлежит географический домен ru, в котором российские организации и граждане имеют право зарегистрировать домен второго уровня.

Административные домены обозначаются тремя или более буквами и предназначены для регистрации доменов второго уровня организациями различных типов (табл. 2).

Таблица 2. Некоторые имена доменов верхнего уровня

Административные	Тип организации	Географические	Страна
com, biz	Коммерческая	ca	Канада
edu	Образовательная	de	Германия
net	Коммуникационная	JP	Япония
org, pro	Некоммерческая	ru	Россия
name	Персональная	it	Италия
museum	Музей	uk	Великобритания

Так, компания Microsoft зарегистрировала домен второго уровня *Microsoft* в административном домене верхнего уровня *com*, а Московский институт открытого образования - домен второго уровня *metodist* в географическом домене верхнего уровня *ru* (рис. 3).

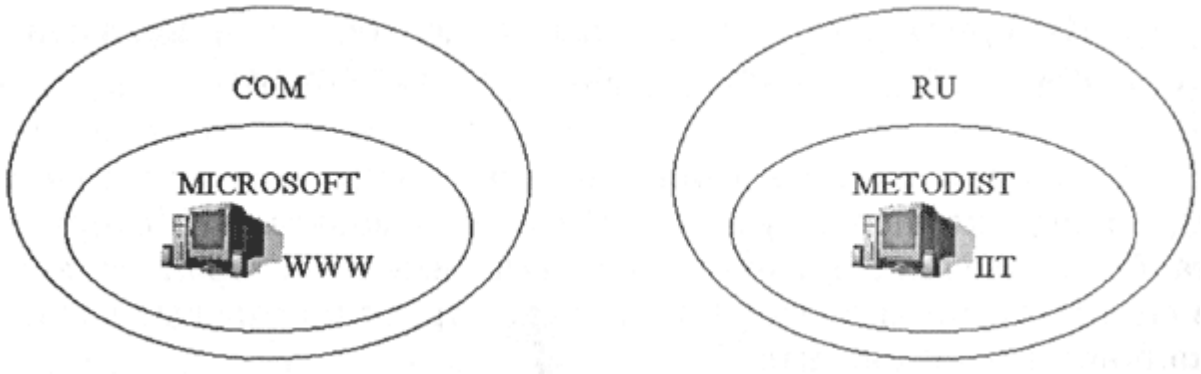


Рис. 3 Доменная система имен

Для доменов нижних уровней можно использовать любые адреса, но для доменов самого верхнего уровня существует соглашение.

В системе адресов Internet приняты домены, представленные географическими регионами ***(географические домены).*** Они имеют имя, состоящее из двух букв. Например, *by* — Беларусь, *ru* — Россия, *ua* - Украина, *us* — США, *de* — Германия, *fr* - Франция, *pl* - Польша, *uk* - Великобритания, *jp* - Япония и др.

Исторически сложилось так, что в США было не принято указывать название страны, а использовались обозначения, определяемые типом организации-владельца адреса, так называемые ***тематические домены.*** Например, *edu* - учебные заведения, *gov* -правительственные учреждения, *com* - коммерческие организации, *mil* - военные организации, *net* - организации, управляющие сетями, *org* - прочие организации.

Достаточно часто самое левое имя в адресе обозначает тип информации, на который указывает данный адрес. Например, *www.microsoft.com,* указывает на использование WWW.

При работе в Internet используется не просто доменный адрес, а ***универсальный указатель ресурса*** (*URL*).

URL (*Uniform Resource Locator*) — это адрес любого ресурса в Интернете с указанием того, с помощью какого протокола следует к нему обратиться, какую программу запустить на сервере и какой конкретно файл следует открыть.

В общем виде формат URL можно представить следующим образом:

протокол://сетевой адрес компьютера/путь/имя файла, где ***протокол*** (метод доступа) может иметь одно из следующих значений:

http - файл на WWW-сервере,

ftp - файл на FTP-сервере,

gopher - файл на Gopher-сервере,

news - группа новостей телеконференции Usenet,

telnet — доступ к ресурсам другого компьютера в режиме удаленного терминала и пр.);

сетевой адрес компьютера указывает доменный (или IP) адрес компьютера, содержащего данный ресурс в сети Интернет.

Доменное имя сервера Интернета состоит из последовательности (справа налево) имен домена верхнего уровня, домена второго

уровня и собственно имени компьютера. Так, основной сервер компании Microsoft имеет имя www.microsoft.com, а сервер института имеет имя iit.metodist.ru.

Каждый компьютер, подключенный к Интернету, имеет Интернет-адрес, однако он может не иметь доменного имени. Доменные имена имеют серверы Интернета, но доменного имени обычно не имеют компьютеры, подключающиеся к Интернету по телефонным линиям.

Структура Internet. Физически структуру Интернета составляют компьютеры самых разных типов. Те из них, которые подключены постоянно и участвуют в передаче данных между другими участниками сети, обеспечивая пользователей определенными услугами, называют ***серверами.*** Несмотря на то, что многие из серверов не совместимы программно, вся система функционирует надежно благодаря тому, что каждый сервер использует стандартный протокол передачи данных *Transmission Control Protocol/Internet Protocol (TCP/IP)* ***(протокол*** — это совокупность правил и соглашений, позволяющих связываться между собой компьютерам разных типов, работающих в разных операционных системах). **Протокол TCP/IP** на самом деле не один протокол, а два. Протокол **TCP** (*Transmission Control Protocol* - протокол управления передачей) отвечает за то, как информация «разбивается» на пакеты и как потом собирается в полный документ, а протокол **IP**(*Internet Protocol* — Межсетевой протокол) отвечает за то, как эти пакеты передаются в сети и как они достигают адресата.

Протокол IP (Internet Protocol – межсетевой протокол) является главным протоколом семейства, он реализует распространение информации в IP-сети и выполняется на третьем (сетевом) уровне модели ВОС. Протокол IP обеспечивает дейтаграммную доставку пакетов, его основная задача – маршрутизация пакетов. Он не отвечает за надежность доставки информации, за ее целостность, за сохранение порядка потока пакетов. Сети, в которых используется протокол IP, называются IP-сетями. Они работают в основном по аналоговым каналам (т. е. для подключения компьютера к сети требуется IP-модем) и являются сетями с коммутацией пакетов. Пакет здесь называется дейтаграммой.

Высокоуровневый **протокол TCP** (Transmission Control Protocol – протокол управления передачей) работает на транспортном уровне и

частично на сеансовом уровне модели ВОС. Это протокол с установлением логического соединения между отправителем и получателем. Он обеспечивает сеансовую связь между двумя узлами с гарантированной доставкой информации, осуществляет контроль целостности передаваемой информации, сохраняет порядок потока пакетов. Протокол TCP делит поток байт на сегменты и передает их сетевому уровню. На приемной стороне этот протокол снова собирает сегменты в непрерывный поток байт.

Для компьютеров протокол TCP/IP – это как правила разговора для людей. Он принят в качестве официального стандарта в сети Internet, т. е. сетевая технология TCP/IP де-факто стала технологией всемирной сети.

Протокол TCP/IP основывается на концепции одноранговых сетей. Все рабочие станции, соединенные при помощи этого протокола, имеют одинаковый статус. Однако любая из них, располагая соответствующими средствами, может временно выполнять дополнительные функции, связанные, например, с управлением ресурсами сети. Ключевую часть протокола составляет схема маршрутизации пакетов, основанная на уникальных адресах сети Internet. Каждая рабочая станция, входящая в состав локальной или глобальной сети, имеет уникальный адрес, который включает две части, определяющие адрес сети и адрес станции внутри сети. Такая схема позволяет передавать сообщения как внутри данной сети, так и во внешние сети. Часть протокола TCP/IP, отвечающая за распознавание адреса, называется IRP (протокол распознавания адреса).

Многоуровневая структура протоколов TCP/IP. Семейство протоколов (или стек протоколов) TCP/IP имеет четыре ярко выраженных уровня:

I – прикладной уровень;

II – транспортный (основной) уровень;

III – сетевой уровень (уровень межсетевого взаимодействия);

IV – канальный уровень (уровень сетевых интерфейсов).

Каждый уровень выполняет свои функции по решению основной задачи – организации надежной и эффективной работы составной сети, т. е. совокупности нескольких сетей, построенных на основе раз-

ных сетевых технологий и соединенных между собой маршрутизаторами. Протокол на более высоком уровне при своей работе использует сервисы, предоставляемые протоколами более низкого уровня.

С помощью многоуровневой модели стека TCP/IP проблема перемещения информации между взаимодействующими компьютерами через среду сети разбивается на более мелкие и более легко разрешимые проблемы. Каждый уровень относительно автономен, т. е. его функции можно представить независимо от других уровней. Многоуровневая модель данной АС исключает прямую связь между соответствующими уровнями модели другой АС. Эта связь осуществляется через услуги, предоставляемые данному уровню модели со стороны смежных уровней.

Каждый протокол стека TCP/IP оперирует со своей единицей передаваемых данных, названия которых либо стандартизированы, либо определяются традиционно. От прикладного уровня к транспортному данные поступают в виде потока. На транспортном уровне протоколами TCP и UDP из потока нарезаются на сегменты или дейтаграммы. Под дейтаграммой понимается единица данных, которыми оперируют протоколы без установления соединений (дейтаграммные протоколы). К ним относится и протокол IP сетевого уровня. На сетевом уровне протоколом IP дейтаграмма преобразуется в пакет. Наконец, на канальном уровне пакеты преобразуются в кадры (фреймы). Кадрами принято называть единицы данных, на основе которых IP-пакеты переносятся через подсети составной сети. Каждая единица данных состоит из заголовка и собственно данных. По мере перемещения данных сверху вниз на каждом уровне добавляется свой заголовок, т. е. каждый пакет более высокого уровня вкладывается в «конверт» протокола нижнего уровня (это напоминает вложенные друг в друга матрешки). На приемной стороне, где данные перемещаются снизу вверх, происходят обратные процессы: на каждом последующем уровне пакет освобождается от заголовка предыдущего уровня, так что к пользователю поступают только собственно данные. Таким образом, протокол – это система правил для работы с данными определенного формата, а формат данных определяется их заголовком (именно заголовок пакета данных определяет способ его обработки сетевым программным обеспечением).

Стандарты TCP/IP опубликованы в серии документов, названных Request for Comment (RFC). Документы RFC описывают внутреннюю работу сети Internet. Некоторые RFC описывают сетевые сервисы или протоколы и их реализацию, в то время как другие обобщают условия применения. Стандарты TCP/IP всегда публикуются в виде документов RFC, но не все RFC определяют стандарты.

Стек был разработан по инициативе Министерства обороны США (Department of Defence, DoD) более 20 лет назад для связи экспериментальной сети ARPAnet с другими сателлитными сетями как набор общих протоколов для разнородной вычислительной среды. Сеть ARPA поддерживала разработчиков и исследователей в военных областях. В сети ARPA связь между двумя компьютерами осуществлялась с использованием протокола Internet Protocol (IP), который и по сей день является одним из основных в стеке TCP/IP и фигурирует в названии стека.

Большой вклад в развитие стека TCP/IP внес университет Беркли, реализовав протоколы стека в своей версии ОС UNIX. Широкое распространение ОС UNIX привело и к широкому распространению протокола IP и других протоколов стека. На этом же стеке работает всемирная информационная сеть Internet, чье подразделение Internet Engineering Task Force (IETF) вносит основной вклад в совершенствование стандартов стека, публикуемых в форме спецификаций RFC.

Итак, лидирующая роль стека TCP/IP объясняется следующими его свойствами:

- Это наиболее завершенный стандартный и в то же время популярный стек сетевых протоколов, имеющий многолетнюю историю.
- Почти все большие сети передают основную часть своего трафика с помощью протокола TCP/IP.
- Это метод получения доступа к сети Internet.
- Этот стек служит основой для создания intranet- корпоративной сети, использующей транспортные услуги Internet и гипертекстовую технологию WWW, разработанную в Internet.
- Все современные операционные системы поддерживают стек TCP/IP.

- Это гибкая технология для соединения разнородных систем как на уровне транспортных подсистем, так и на уровне прикладных сервисов.
- Это устойчивая масштабируемая межплатформенная среда для приложений клиент-сервер.

Так как стек TCP/IP был разработан до появления модели взаимодействия открытых систем ISO/OSI, то, хотя он также имеет многоуровневую структуру, соответствие уровней стека TCP/IP уровням модели OSI достаточно условно.

Структура протоколов TCP/IP приведена на рисунке 4. Протоколы TCP/IP делятся на 4 уровня.

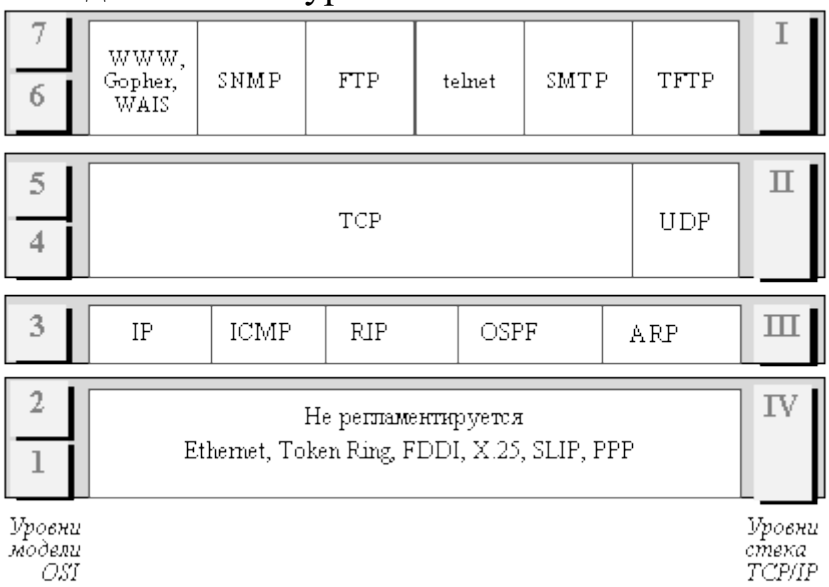


Рис. 4. Стек TCP/IP

Самый нижний (***уровень IV***) соответствует физическому и канальному уровням модели OSI. Этот уровень в протоколах TCP/IP не регламентируется, но поддерживает все популярные стандарты физического и канального уровня: для локальных сетей это Ethernet, Token Ring, FDDI, Fast Ethernet, 100VG-AnyLAN, для глобальных сетей - протоколы соединений "точка-точка" SLIP и PPP, протоколы территориальных сетей с коммутацией пакетов X.25, frame relay. Разработана также специальная спецификация, определяющая использование технологии ATM в качестве транспорта канального уровня. Обычно при появлении новой технологии локальных или глобальных сетей она быстро включается в стек TCP/IP за счет разработки соответствующего RFC, определяющего метод инкапсуляции пакетов IP в ее кадры.

Следующий уровень (***уровень III***) - это уровень межсетевого взаимодействия, который занимается передачей пакетов с использованием различных транспортных технологий локальных сетей, территориальных сетей, линий специальной связи и т. п.

В качестве *основного протокола сетевого уровня* (в терминах модели OSI) в стеке используется протокол **IP**, который изначально проектировался как протокол передачи пакетов в составных сетях, состоящих из большого количества локальных сетей, объединенных как локальными, так и глобальными связями. Поэтому протокол IP хорошо работает в сетях со сложной топологией, рационально используя наличие в них подсистем и экономно расходуя пропускную способность низкоскоростных линий связи. Протокол IP является дейтаграммным протоколом, то есть он не гарантирует доставку пакетов до узла назначения, но старается это сделать.

К *уровню межсетевого взаимодействия* относятся и все протоколы, связанные с составлением и модификацией таблиц маршрутизации, такие как протоколы сбора маршрутной информации **RIP** (Routing Internet Protocol) и **OSPF** (Open Shortest Path First), а также протокол межсетевых управляющих сообщений **ICMP** (Internet Control Message Protocol). Последний протокол предназначен для обмена информацией об ошибках между маршрутизаторами сети и узлом - источником пакета. С помощью специальных пакетов ICMP сообщается о невозможности доставки пакета, о превышении времени жизни или продолжительности сборки пакета из фрагментов, об аномальных величинах параметров, об изменении маршрута пересылки и типа обслуживания, о состоянии системы и т.п.

Следующий уровень (***уровень II***) называется *основным.* На этом уровне функционируют протокол управления передачей **TCP** (Transmission Control Protocol) и протокол дейтаграмм пользователя **UDP** (User Datagram Protocol). Протокол TCP обеспечивает надежную передачу сообщений между удаленными прикладными процессами за счет образования виртуальных соединений. Протокол UDP обеспечивает передачу прикладных пакетов дейтаграммным способом, как и IP, и выполняет только функции связующего звена между сетевым протоколом и многочисленными прикладными процессами.

Верхний уровень (***уровень I***) *называется прикладным.* За долгие годы использования в сетях различных стран и организаций стек

TCP/IP накопил большое количество протоколов и сервисов прикладного уровня. К ним относятся такие широко используемые протоколы, как протокол копирования файлов FTP, протокол эмуляции терминала telnet, почтовый протокол SMTP, используемый в электронной почте сети Internet, гипертекстовые сервисы доступа к удаленной информации, такие как WWW и многие другие. Остановимся несколько подробнее на некоторых из них.

Протокол *пересылки файлов* ***FTP*** (File Transfer Protocol) реализует удаленный доступ к файлу. Для того, чтобы обеспечить надежную передачу, FTP использует в качестве транспорта протокол с установлением соединений - TCP. Кроме пересылки файлов протокол FTP предлагает и другие услуги. Так, пользователю предоставляется возможность интерактивной работы с удаленной машиной, например, он может распечатать содержимое ее каталогов. Наконец, FTP выполняет аутентификацию пользователей. Прежде, чем получить доступ к файлу, в соответствии с протоколом пользователи должны сообщить свое имя и пароль. Для доступа к публичным каталогам FTP-архивов Internet парольная аутентификация не требуется, и ее обходят за счет использования для такого доступа предопределенного имени пользователя Anonymous.

В стеке TCP/IP протокол FTP предлагает наиболее широкий набор услуг для работы с файлами, однако он является и самым сложным для программирования. Приложения, которым не требуются все возможности FTP, могут использовать другой, более экономичный протокол - простейший протокол пересылки файлов **TFTP** (Trivial File Transfer Protocol). Этот протокол реализует только передачу файлов, причем в качестве транспорта используется более простой, чем TCP, протокол без установления соединения - UDP.

Протокол **telnet** обеспечивает передачу потока байтов между процессами, а также между процессом и терминалом. Наиболее часто этот протокол используется для эмуляции терминала удаленного компьютера. При использовании сервиса telnet пользователь фактически управляет удаленным компьютером так же, как и локальный пользователь, поэтому такой вид доступа требует хорошей защиты. Поэтому серверы telnet всегда используют как минимум аутентификацию по паролю, а иногда и более мощные средства защиты, например, систему Kerberos.

Протокол **SNMP** (Simple Network Management Protocol) используется для организации сетевого управления. Изначально протокол SNMP был разработан для удаленного контроля и управления маршрутизаторами Internet, которые традиционно часто называют также шлюзами. С ростом популярности протокол SNMP стали применять и для управления любым коммуникационным оборудованием - концентраторами, мостами, сетевыми адаптерами и т.д. и т.п. Проблема управления в протоколе SNMP разделяется на две задачи.

Первая задача связана с передачей информации. Протоколы передачи управляющей информации определяют процедуру взаимодействия SNMP-агента, работающего в управляемом оборудовании, и SNMP-монитора, работающего на компьютере администратора, который часто называют также консолью управления. Протоколы передачи определяют форматы сообщений, которыми обмениваются агенты и монитор.

Вторая задача связана с контролируемыми переменными, характеризующими состояние управляемого устройства. Стандарты регламентируют, какие данные должны сохраняться и накапливаться в устройствах, имена этих данных и синтаксис этих имен. В стандарте SNMP определена спецификация информационной базы данных управления сетью. Эта спецификация, известная как база данных MIB (Management Information Base), определяет те элементы данных, которые управляемое устройство должно сохранять, и допустимые операции над ними.

Согласно протоколу ***TCP/IP все данные, передающиеся по сети, «разбиваются» на небольшие блоки и «вкладываются» в пакеты***. Каждый пакет кроме данных, вложенных в него, имеет заголовок, содержащий адрес отправителя, адрес получателя, и прочую информацию, необходимую для правильной сборки пакетов в пункте назначения. Пакеты переходят с одного сервера на другой и далее пересылаются на следующий сервер, находящийся «ближе» к адресату. Если пакет передан неудачно, передача повторяется. При этом от клиентов к серверам идут запросы, разбитые на пакеты, а от серверов к клиентам - затребованные данные.

При выходе из строя любой части всемирной сети, пакеты с информацией автоматически пойдут в обход пораженного участка. Можно перерезать все трансатлантические кабели между Европой и

Америкой. Не получив подтверждения о доставке пакетов, серверы автоматически повторят передачу через спутниковые каналы связи или по сетям радиорелейных станций.

Семейство протоколов TCP/IP работает на любых моделях компьютеров, произведенных различными производителями компьютерной техники и работающих под управлением различных операционных систем. С помощью протоколов TCP/IP можно объединить практически любые компьютеры. И что самое удивительное, сегодняшние реализации протокола TCP/IP очень далеки от того, как он задумывался исходно.

В конце 60-х годов начался исследовательский проект, финансируемый правительством США, по разработке сети пакетной коммутации, а в 90-х годах результаты этих исследований превратились в наиболее широко используемую форму сетевого взаимодействия между компьютерами. В настоящее время это действительно открытая система, а именно, семейство протоколов и большое количество бесплатных реализаций (либо достаточно дешевых). Они составляют основу того, что в настоящее время называется словом Internet.

Рассмотрим состав и **основные функции протоколов каждого уровня стека TCP/IP**.

Прикладной уровень объединяет все службы, предоставляемые пользовательским приложениям. Он идентифицирует и устанавливает наличие предполагаемых партнеров для связи, синхронизирует совместно работающие прикладные программы, определяет наличие ресурсов для реализации взаимодействия с другими узлами сети, обеспечивает читабельность информации на прикладном уровне другой абонентской системы, устанавливает соглашение по процедурам устранения ошибок и управления целостностью информации, устанавливает и завершает сеансы взаимодействия между прикладными задачами, управляет этими сеансами, синхронизирует диалог между взаимодействующими узлами сети и управляет обменом информацией между ними.

Протоколы прикладного уровня занимаются деталями конкретного приложения и не участвуют в реализации способов передачи данных по сети. Выполнение этих протоколов осуществляется программными средствами, построенными в архитектуре клиент-сервер.

Комплект протоколов прикладного уровня включает в себя большое число протоколов. Он постоянно расширяется за счет присоединения к старым, прошедшим многолетнюю эксплуатацию сетевым службам типа Telnet, FTR, SNMP, сравнительно новых служб таких, как гипертекстовая система WWW, предоставляющая для работы со своим сервисом высокоуровневый протокол HTTP.

Транспортный уровень предоставляет услуги по транспортировке данных, решая вопросы надежной и достоверной передачи данных через сеть. Он реализует механизмы установки, поддержания и закрытия соединения, а также механизмы обнаружения и устранения ошибок в передаваемых данных, управления информационным потоком.

На этом уровне функционирует протокол управления передачей TCP, о котором сказано выше, и протокол дейтаграмм пользователя UDP (User Data gram Protocol). Протокол UDP работает без установления соединения, т. е. обеспечивает передачу пакетов дейтаграммным способом. Его функции существенно проще, чем у протокола TCP: он только отправляет пакеты без какого-либо дополнительного сервиса. Протокол TCP обеспечивает полный сервис транспортного уровня – надежность, достоверность и контроль соединения.

Сетевой уровень является стержнем всей архитектуры стека TCP/IP. Он обеспечивает передачу (через составную сеть) пакетов дейтаграммным способом, используя наиболее рациональный в данный момент маршрут. На этом уровне работает основной протокол стека – межсетевой протокол IP. Он хорошо работает в составных сетях со сложной топологией, используя в них протоколы подсистем и экономно расходуя пропускную способность низкоскоростных линий связи. Протокол IP отличается от других сетевых протоколов способностью выполнять динамическую фрагментацию пакетов при передаче их между различными сетями. Это свойство во многом способствовало той доминирующей позиции, которую занял протокол IP в сложных составных сетях.

К сетевому уровню относятся и все протоколы, обеспечивающие маршрутизацию пакетов. Это протоколы сбора маршрутной информации RIP и OSPF, необходимой для составления и модификации

таблиц маршрутизации, протокол межсетевых управляющих сообщений ISMP, предназначенный для обмена информацией об ошибках между маршрутизаторами сети и узлом-источником пакета, и др.

Канальный уровень (уровень сетевых интерфейсов, уровень сопряжения с физической средой) обеспечивает передачу данных по физическому каналу. Протоколы этого уровня должны обеспечивать интеграцию в составную сеть других сетей независимо от того, какая внутренняя технология передачи данных в них используется. Для каждой технологии должны быть собственные интерфейсные средства, например, протоколы инкапсуляции IP-пакетов в кадры локальных технологий. В связи с этим канальный уровень нельзя определить раз и навсегда, этот уровень в протоколах TCP/IP не регламентируется, но поддерживает все популярные стандарты физического и канального уровней ЛКС.

Протоколы канального уровня тесно связаны с физической (аппаратной) средой, в которой они работают (Ethernet, Token Ring, FDDI, ISDN и др.). В стеке TCP/IP нет протоколов этого уровня, за счет чего и достигается аппаратная независимость семейства TCP/IP.

Ниже канального уровня расположен только аппаратный уровень, главные функции которого состоят в определении электротехнических, механических, функциональных и процедурных характеристик активизации, поддержания и деактивизации физического канала между взаимодействующими конечными узлами.

Многоуровневую схему протоколов TCP/IP можно представить в виде дерева: канальный уровень уподобляется корню этого дерева (подобно тому, как корни дерева состоят из множества отростков, так и конкретные физические реализации сети весьма разнообразны), ствол дерева – это сетевой уровень (уровень IP), толстые сучья дерева – это уровень TCP, ветви кроны – протоколы прикладного уровня и, наконец, листья кроны – пользовательские приложения, работающие с протоколами верхнего уровня.

В настоящее время существует три основных **способа доступа к Интернету:**

1. ***Прямое соединение (выделенное соединение)*** — это соединение, при котором частное лицо или компания подключается к магистральным каналам (backbone) Интернета через выделенную машину, называемую шлюзом. *Шлюз* - это специализированный компьютер,

обеспечивающий внешнюю связь одной сети с другой сетью, использующей иной протокол передачи данных.

2. ***Соединение через чужой шлюз*** - это доступ в Интернет через шлюз какой- либо организации или учреждения, обычно с использованием для соединения модема.

3. ***Сеансовое (коммутируемое) соединение*** - это соединение через сервис-провайдеров. *Коммутация* – это установление связи между устройствами путем создания временных соединений.

Сервис-провайдеры - это компании со шлюзами в Интернете, которые они предоставляют другим компаниям или частным лицам за абонентскую плату.

В настоящее время для рядовых пользователей самым популярным способом подключения к глобальной компьютерной сети является подключение с помощью телефонной линии. Так как при наборе телефонного номера для установки связи двух абонентов на АТС происходит переключение (коммутация) линии связи, то телефонные линии часто называют коммутируемыми. Для подключения компьютеров к линиям связи необходимо использовать специальные электронные устройства - модемы (факс-модемы).

Модем (от слов модулятор и демодулятор) – это устройство ввода-вывода информации, обеспечивающее модуляцию и демодуляцию сигналов, преобразуя, таким образом, цифровые сигналы ПК в звуковые сигналы и обратно для передачи их по телефонным линиям связи. **Факс-модем** – это модем, совмещенный с факсимильным аппаратом – устройством, предназначенным для обмена символьной, графической, видео- и аудиоинформацией (т.е. факсимильными сообщениями) по телефонным каналам связи.

Единицей измерения скорости передачи информации является ***1 бит/сек***.

Трафик - это объем информации, передаваемый по сети за определенный промежуток времени и измеряемый в битах.

1.1.2 Каталоги ресурсов. Поисковые системы.

С точки зрения пользователя Интернет можно рассматривать как мощное глобальное средство обмена информацией. Одним из распространенных и перспективных сервисов Интернет является сервис прямого доступа Word Wide Web — WWW, представляющий собой си-

стему документов, включающих текстовую и графическую информацию, размещенных на узлах Интернет и связанных между собой гиперссылками.

Классификация источников информации в Интернете может проводиться по разным основаниям. По способам представления информации могут быть выделены следующие виды:

- *web-страницы* — наиболее распространенный и используемый из информационных ресурсов. Этот ресурс представляет собой страницы гипертекста. Страницы наряду с текстовой могут содержать графическую, звуковую, видеоинформацию;

- *файловые серверы* представляют собой реализацию в Интернете традиционного способа представления информации;

- *телеконференции* могут являться важным источником информации. Они разбиваются на группы (рубрики) по тематике. Участвующие в телеконференциях могут написать свое сообщение или послать комментарии на чужое сообщение;

- *базы данных* могут быть доступны через сеть Интернет. В них часто содержатся, кроме текстовой, также и другие виды информации.

Информационные ресурсы также могут быть разделены по языковому признаку, в сети Интернет представлены практически все основные языки, однако главным языком в силу исторически сложившихся традиций является английский. Ряд сайтов представляет информацию на нескольких языках.

В сети имеет место классификация и по территориальному признаку. Ряд сайтов предоставляет свою информацию для потребителей определенного региона, хотя доступ к сайту возможен и из любой точки сети.

Наиболее важным аспектом классификации информационных ресурсов сети Интернет является содержание информации. Деловая информация, необходимая в предпринимательской деятельности, по этому критерию может быть разделена на следующие группы.

1. *Сведения о фирмах, организациях.* Эта группа сведений существенно различается по своему наполнению для различных организаций. Различия определяются степенью освоения организацией возможностей Интернета по продвижению продукции или услуг. Различают три типа серверов данной группы (категории):

• серверы присутствия в Интернете. Эти серверы могут быть разделены на рекламные и информирующие серверы. Рекламный сервер обычно содержит одну или несколько страниц. Информирующий сервер содержит более подробную информацию о фирме и производимой ею продукции или оказываемых услугах;

• информационные серверы. Целью этих серверов является предоставление различного рода информации потребителям. Серверы данной группы ведут информационно-аналитические агентства и другие структуры, в том числе государственные, чья деятельность связана с предоставлением различного рода информации потребителям;

• интерактивные магазины. Серверы этой группы обеспечивают продажи товаров посредством Интернета. При этом могут быть реализованы в электронном виде следующие функции:

— предоставление клиенту необходимой информации о товаре или услуге;

— оформление заказа;

—оплата заказа (при использовании онлайновых платежных систем);

—отправка полученного товара, если товаром является информация.

2. *Сведения о состоянии мировой экономики и экономики отдельных стран.* Данная информация представлена достаточно широко в профессиональных базах крупнейших информационно-аналитических агентств мира. Серверы этих агентств входят в состав информационных ресурсов сети Интернет. Однако сама информация, как правило, платная. Информация о состоянии национальной экономики обычно размещается на серверах государственных структур, отвечающих за государственную поддержку экономики, государственных статистических органов, различных экономических институтов.

3. *Сведения о состоянии отраслевых рынков.* Анализ, отраслевых рынков осуществляют специализированные маркетинговые и консалтинговые агентства, а также маркетинговые службы фирм или организаций. Результаты этих исследований, используя Интернет, можно получить:

• из профессиональных баз крупнейших мировых информационных агентств, найдя сведения о технологиях доступа к этим базам на сайтах Интернета;

• в самих консалтинговых или маркетинговых агентствах, чьи сайты также представлены в Интернете;

• в многопрофильных и отраслевых журналах, регулярно публикующих обзоры рынков. Одни издания, например, многопрофильный журнал «Эксперт», представляют на сайтах оглавления номеров журналов. Другие, как, например, журнал «Профиль», размещают в открытом доступе публикуемые материалы.

4. *Деловые новости.* Подавляющее большинство мировых информационных агентств предоставляют потребителям доступ к профессиональным базам, содержащим деловые новости. Из зарубежных агентств крупнейшими поставщиками деловых новостей являются LEXIS-NEXIS, Dialog, Reuters. Среди отечественных агентств следует выделить:

• «Интегрум-Техно», предоставляющий доступ к материалам 250 центральных и крупнейших региональных газет, а также к зарубежным новостям;

• РИА «Новости» — государственное информационно-аналитическое агентство РФ;

• «ИТАР-ТАСС» — государственное информационное телеграфное агентство РФ;

• агентство «Интерфакс», входящее в состав международной информационной группы Interfax Information Services. Интернет предоставляет бесплатный доступ к ежедневной электронной интернет-газете «Gazeta.ni». Ряд крупнейших газет имеют в Интернете электронные версии. Доступ к некоторым из них платный, например, к электронным версиям печатных изданий Издательского дома «Коммерсантъ». К другим, например, к электронной версии газеты «Аргументы и факты», — бесплатный.

5. *Справочная информация* представлена в сети Интернет весьма широко. Это и списки web-сайтов компаний, отобранных по определенному принципу, и телефонно-адресный справочник «Желтые страницы» с возможностью поиска информации по названию фирмы и виду деятельности, и телефонные справочники городов Российской Федерации, стран СНГ и Балтии. Также в Интернете имеется информация о расписании движения поездов, авиарейсов, о погоде и многое другое.

Для того чтобы облегчить поиск в больших массивах текстовой информации, существуют информационно-поисковые системы, в которых документы описываются с помощью специальных поисковых языков. С помощью элементов этих же языков отписываются и запросы. Для отбора документов в ответ на запрос осуществляется сравнение поисковых образов запросов и поисковых образов документов, которое проводилось на одном и том же искусственном языке. Такой подход является вынужденным.

Из-за недостатков естественного языка, документальные информационные системы не дают ответа на вопрос потребителя, а выдают ему документы, в которых может содержаться ответ на его запрос, предоставляя потребителю самому выявить смысловое содержание этих документов. Вопросы оценки эффективности поиска информации в документальных информационных системах будут рассмотрены ниже.

Анализ содержимого профессиональных баз за последние 15 лет показывает неуклонный рост доли текстовой информации в общем объеме информации в профессиональных базах. Если в 1985 г. доля текстовой информации составляла 47 %, то в 2000 г. эта доля составляла уже 84 %. Представляется, что основная информация в Интернете также является текстовой. Эти обстоятельства позволяют сделать вывод о том, что подходы к оценке эффективности поиска в документальных системах в полной мере распространяются и на профессиональные базы, и на информационные ресурсы Интернета.

Информационные ресурсы Интернета и имеющиеся в среде Интернет поисковые средства обладают определенной спецификой, которая оказывает существенное влияние на эффективность поиска в этой среде.

Основными поисковыми средствами в Интернете являются поисковые системы и каталоги. Поисковые системы состоят из трех частей:

♦ *робот* — программа, которая посещает web-серверы, считывает и индексирует полностью или частично их содержимое и далее следует по ссылкам, найденным на сервере. Просмотры серверов осуществляются периодически, например, раз в месяц, раз в две недели;

♦ *индексные массивы и копии текстов* просмотренных страниц, хранящиеся в поисковой системе;

♦ *программа,* которая, просматривая в соответствии с запросом

пользователя индексные массивы, отбирает и выдает потребителю найденные документы.

В каталогах имеются иерархические тематические рубрики. Отнесение серверов к тем или иным рубрикам каталога осуществляется человеком. Пользователь ищет информацию в каталоге вручную, используя рубрики.

В связи с тем, что в средствах поиска в Интернете не используются информационно-поисковые языки, на которых могли бы быть описаны исходные документы и запросы, полнота поиска в Интернете с учетом указанных выше поисковых средств будет значительно ниже, чем в документальных системах, построенных на базе информационно-поисковых языков.

В 2000 г. специалисты компаний AltaVista, IBM и Compag исследовали ресурсы и гиперсвязи существующего информационного пространства WWW. Просмотрев с помощью поисковых средств AltaVista свыше 600 млн. web-страниц и 1,5 млрд. ссылок, размещенных на этих страницах, они пришли к выводу, что исследуемое пространство состоит из следующих компонентов:

- центральное ядро — тесно связанные между собой web-страницы, с каждой из которых можно попасть на любую другую (27 %);

- отправные страницы. В них могут быть ссылки, ведущие к ядру, но из ядра к отправным страницам попасть нельзя (22 %);

- оконечные web-страницы, к которым можно прийти по ссылкам из ядра, но к ядру от них попасть нельзя (22 %);

- полностью изолированные от центрального ядра страницы (22 %);

- web-страницы, не пересекающиеся с остальными ресурсами Интернета (7 %).

Исследования показали, что при увеличении общего объема информационных ресурсов Интернета установленные отношения компонентов остаются прежними. Проведенный анализ позволяет сделать вывод о том, что информационное пространство Интернета является достаточно сложным и неоднородным. К отдельным ресурсам Интернета поисковые машины не имеют доступа.

У каждой поисковой машины свой процент индексирования документов и своя стратегия выбора — какие из ресурсов индексировать.

Анализ доли документов, заиндексированных крупнейшими зарубежными поисковыми системами, от общего числа документов в Интернете, проведенный в 1999 г., показал, что доля заиндексированных документов у лидеров в этой области не превышает 30%. Следует отметить, что количество документов в Интернете значительно увеличивается с каждым годом, при этом доля просмотренных и заиндексированных документов уменьшается.

Информационные ресурсы Интернета делятся на «видимую» и «невидимую» части сайтов.

«Видимая» - часть сайтов — это та часть, которая обрабатывается поисковыми системами и индексируется. «Невидимая» — часть сайтов, которая не предназначена для обработки поисковыми системами. Американская фирма BrightPIanet разработала программное обеспечение по исследованию «невидимой» части сайтов. Полученные результаты показывают, что число документов «невидимой» части превышает более чем в 500 раз число документов, относящихся к «видимой» части.

Перечисленные особенности информационных ресурсов Интернета и поисковых средств позволяют сделать вывод о том, что эффективность поиска информации в Интернете существенно уступает эффективности поиска в документальных информационно-поисковых системах, использующих специальные информационно-поисковые языки, и эффективности поиска в профессиональных базах. Указанные обстоятельства определяют высокие требования к профессиональной подготовке пользователя, которая необходима для получения нужной информации из информационных ресурсов Интернета.

Организация информации в профессиональных базах отличается от организации информации в Интернете в первую очередь тем, что информация накапливается и постоянно обновляется в базах данных, которых в настоящее время свыше 13 тыс. В каждой базе собрана специфичная информация. Отбор достоверных источников и накопление информации ведут информационные агентства-генераторы. Они же поддерживают эти базы в актуальном состоянии, то есть обновляют. Получение информации из баз потребителями обеспечивают агентства-поставщики. Потребителю предоставляется язык запроса и документация, характеризующая базы данных, которая включает сле-

дующие сведения по каждой базе: название, отражающее вид информации, хранимой в базе, с какого времени ведется база, объем накопленной информации, период обновления, источники информации.

Указанные особенности обеспечивают высокие показатели по достоверности, полноте и точности предоставляемой информации.

Наилучшим вариантом работы с информационными ресурсами Интернета является вариант, когда пользователь знает адрес сайта и получает возможность ознакомиться с его содержимым. Адрес сайта может быть получен из различных справочников, например, желтых страниц Интернета, рекламных материалов и других источников. На сайт можно прийти по гиперссылкам, просматривая другие сайты

Опытный специалист в области информации всегда будет иметь список, каталог адресов самых важных для ведения бизнеса фирмы сайтов. В этом случае необходимо лишь отслеживать появление новых сайтов, информация в которых может представлять интерес.

Если пользователь исследует новую проблему в бизнесе, ищет информацию среди ресурсов, которые он еще не освоил, одним из основных методов является использование поисковых машин и каталогов.

В этом случае может быть предложена следующая технология подготовки и проведения поиска.

1. Определение общей направленности запроса, его содержания.

2. Определение географических регионов поиска. В первую очередь для практических задач ценность информационного ресурса может зависеть от его географического расположения.

3. Отбор поисковых машин. Осуществляется отбор и устанавливается последовательность использования поисковых машин в соответствии с убыванием ожидаемой эффективности поиска в каждой из машин. Качество выполнения этого этапа будет зависеть от опыта работы пользователя с поисковыми машинами.

4. Составление запросов к поисковым машинам. Это наиболее сложный этап. Для эффективного использования поисковых машин запрос составляется так, чтобы область поиска была сужена в максимальной степени. Предпочтение должно отдаваться не одному расширенному, а нескольким узким запросам. Необходимо смоделировать, представить себе, как может выглядеть искомая информация. По ключевым словам следует составить тезаурус. Для этого необходимо хорошее

знание языка, на котором работает пользователь, и специфических терминов предметной области.

5. Выполнение запроса и его уточнение. Составленный запрос передается на обработку. Анализ полученных результатов позволяет корректировать запрос, чаще всего с целью сужения области поиска.

Поиск деловой информации в Интернете — это творческий процесс, требующий глубоких знаний в области информатики, лингвистики, принципов построения информационных и поисковых ресурсов Интернета.

Вопросы для самоконтроля

1. Какие типы компьютерных сетей образуют Интернет?

2. Какие существуют способы подключения к Интернету и каковы их достоинства и недостатки?

3.Имеет ли каждый компьютер, подключенный к Интернету, Интернет-адрес? Доменное имя?

4. Как строится доменная система имен?

Каким образом производится доставка данных по указанному Интернет-адресу?

5. В каких целях при передаче файлов по компьютерным сетям производится их разбиение на Интернет-пакеты?

Тема 2. Язык гипертекстовой разметки страниц HTML

Общая структура документа, абзацы, цвета, ссылки. Фреймы. Разработка макета страницы списки, графика (графические форматы, графический объект как ссылка), формы

1.2.1 Общая структура документа, абзацы, цвета, ссылки

HyperText Markup Language(HTML) – язык разметки гипертекста – предназначен для написания гипертекстовых документов, публикуемых в World Wide Web.

Язык HTML был разработан британским учёным Тимом Бернерсом-Ли приблизительно в 1986—1991 годах в стенах ЦЕРНа в Женеве в Швейцарии. HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области вёрстки. HTML успешно справлялся с проблемой сложности SGML путём определения небольшого набора структурных и семантических элементов — дескрипторов. Дескрипторы также часто называют «тегами». С помощью HTML можно легко создать относительно простой, но красиво оформленный документ. Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста. Мультимедийные возможности были добавлены позже.

Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи. Например, тег <table> предназначен для создания в документах таблиц, но часто используется и для оформления размещения элементов на странице. С течением времени основная идея платформонезависимости языка HTML была принесена в жертву современным потребностям в мультимедийном и графическом оформлении.

С помощью HTML создаются Web-страницы, которые находятся в глобальной компьютерной сети Интернет. HTML – это не язык программирования в традиционном смысле, он является языком разметки. С помощью HTML текстовый документ разбивают на блоки смысловой информации (заголовки, параграфы, таблицы, рисунки и т.п.).

Гипертекстовый документ– это текстовый файл, имеющий специальные метки, называемые тегами, которые впоследствии опознаются браузером и используются им для отображения содержимого файла на экране компьютера. С помощью этих меток можно выделять заголовки документа, изменять цвет, размер и начертание букв, вставлять графические изображения и таблицы. Но основным преимуществом гипертекста перед обычным текстом является возможность добавления к содержимому документа **гиперссылок**– специальных конструкций языка HTML, которые позволяют щелчком мыши перейти к просмотру другого документа.

Особенности HTML-документа:

1. HTML-документ может содержать текст, графику, видео и звук.

2. В общем случае HTML-документ – это один или несколько текстовых файлов, имеющих расширение .htm или .html.

3. Создавать HTML-документ можно как с помощью специальных программ – редакторов HTML (например, FrontPage). Этот способ позволяет создавать документы для WWW без знания языка HTML. HTML-редакторы автоматизируют создание гипертекстовых документов, избавляют от рутинной работы. Однако их возможности ограничены, они сильно увеличивают размер получаемого файла и не всегда полученный с их помощью результат соответствует ожиданиям разработчика. Но, безусловно, этот способ незаменим для новичков в деле подготовки гипертекстовых документов. Альтернативой служит создание и разметка документа при помощи обычного редактора текста (NotePad, блокнота Windows). При этом способе в текст вручную вставляются команды языка HTML. Создавая документы таким способом, вы точно знаете, что делаете.

4. Для просмотра HTML-документов существуют специальные программы-**браузеры**. Они **интерпретируют** HTML-документы, т.е. переводят текст документа в Web-страницу, и отображают ее на экране

пользователя. Существует очень много различных браузеров, но наиболее распространенными браузерами являются Microsoft Internet Explorer, Netscape Navigator и Opera.

5. Если при интерпретации HTML-документа браузер чего-то не понимает, то сообщения об ошибке не возникает, а это место в HTML-документе игнорируется и не отображается браузером.

HTML-документ состоит из элементов HTML.

Элемент HTML – это чаще всего два тега (открывающий и закрывающий) и часть документа между ними. Кроме того, существуют элементы HTML, состоящие только из одного тега.

Как уже отмечалось, HTML-документ содержит символьную информацию. Одна ее часть - собственно текст, т.е. данные, составляющие содержимое документа. Другая – **теги** (markup tags). **Тег –** в переводе с английского – ярлык, этикетка. Тег определяет тип выводимого элемента HTML (например, заголовок, таблица, рисунок и т.п.). Сам тег не отображается браузером. Тег представляет собой последовательность элементов:

·символ левой угловой скобки (<) – начало тега;

·необязательный символ слеша (/) – символ используется, чтобы обозначить закрывающий тег;

·имя тега;

·необязательные атрибуты в открывающем теге;

·символ правой угловой скобки (>)

Именно теги языка HTML определяют, в каком виде будет представлен текст, какие его компоненты будут исполнять роль гипертекстовых ссылок, какие графические или мультимедийные объекты должны быть включены в документ. Графическая и звуковая информация, включаемая в HTML-документ, хранится в отдельных файлах. Программы просмотра HTML-документов (браузеры) интерпретируют флаги разметки и располагают текст и графику на экране соответствующим образом

Атрибуты – необязательный набор параметров, определяющих дополнительные свойства элемента HTML (например, цвет или размер). Атрибут состоит:

·из имени атрибута;

·знака равенства (=);

·значения атрибута – строки символов, заключенной в кавычки

Прописные и строчные буквы при записи тегов не различаются. В большинстве случаев теги используются парами. Пара состоит из открывающего (start tag) и закрывающего (end tag) тегов. Синтаксис открывающего тега:

<имя_тега [атрибуты]>

Прямые скобки, используемые в описании синтаксиса, означают, что данный элемент может отсутствовать. Имя закрывающего тега отличается от имени открывающего лишь тем, что перед ним ставится наклонная черта:

</имя_тега>

Атрибуты тега записываются в следующем формате:

имя[="значение"]

Кавычки при задании значения аргумента не обязательны и могут быть опущены. Для некоторых атрибутов значение может не указываться. У закрывающего тега атрибутов не бывает.

Действие любого парного тега начинается с того места, где встретился открывающий тег и заканчивается при встрече соответствующего закрывающего тега. Часто пару, состоящую из открывающего и закрывающего тегов, называют **контейнером**, а часть текста, окаймленную открывающим и закрывающим тегом, –**элементом**.

Последовательность символов, составляющая текст, может состоять из пробелов, табуляций, символов перехода на новую строку, символов возврата каретки, букв, знаков препинания, цифр, и специальных символов (например, +, #, $, @), *за исключением* следующих четырех символов, имеющих в HTML специальный смысл: < (меньше), > (больше), & (амперсенд) и " (двойная кавычка). Если необходимо включить в текст какой-либо из этих символов, то следует закодировать его особой последовательностью символов.

Пример элемента HTML:

<H1 ALIGN= "CENTER">Глава 1</H1>

В этом примере:

<H1 ALIGN= "CENTER"> – открывающий тег

</H1> – закрывающий тег

H1 – имя тега

ALIGN= "CENTER" – атрибут

ALIGN – имя атрибута

"CENTER" – значение атрибута

Глава 1 – содержание элемента HTML

Правила создания HTML-документов:

1.Теги и атрибуты можно записывать в любом регистре, т.е. </H1> и </h1> – это одно и то же.

2.Несколько пробелов подряд, символы табуляции и перевода строки при интерпретации браузером заменяются на один пробел. Это позволяет писать хорошо структурированные исходные тексты файлов HTML.

3. Рекомендуется давать имена файлам HTML строчными английскими буквами. Длина имени – до восьми символов. В принципе, можно не придерживаться данной рекомендации, но тогда пользователи, работающие в операционных системах, отличных от Windows, не смогут воспользоваться вашими HTML-документами.

Вопросы для самоконтроля

1. Как называется параметр тега в языке HTML?
2. При помощи каких программ можно создавать веб-страницы на языке разметки гипертекста?
3. Как называется указание к оформлению разметки веб-страницы на языке HTML?
4. Укажите преимущества редакторов исходного кода перед текстовыми редакторами при создании веб-страниц?
5. Укажите недостатки визуальных HTML-редакторов.
6. Какими бывают теги для разметки html-страниц
7. Укажите разновидности html-редакторов.
8. При помощи какого языка пишется разметка веб-страниц?
9. Укажите преимущества визуальных HTML-редакторов, по сравнению с редакторами исходного кода при оформлении веб-страниц?
10. Как иначе называются визуальные html-редакторы?
11. Как называется параметр тега в языке HTML?

Модуль 2. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА СОЗДАНИЯ WEB-САЙТОВ

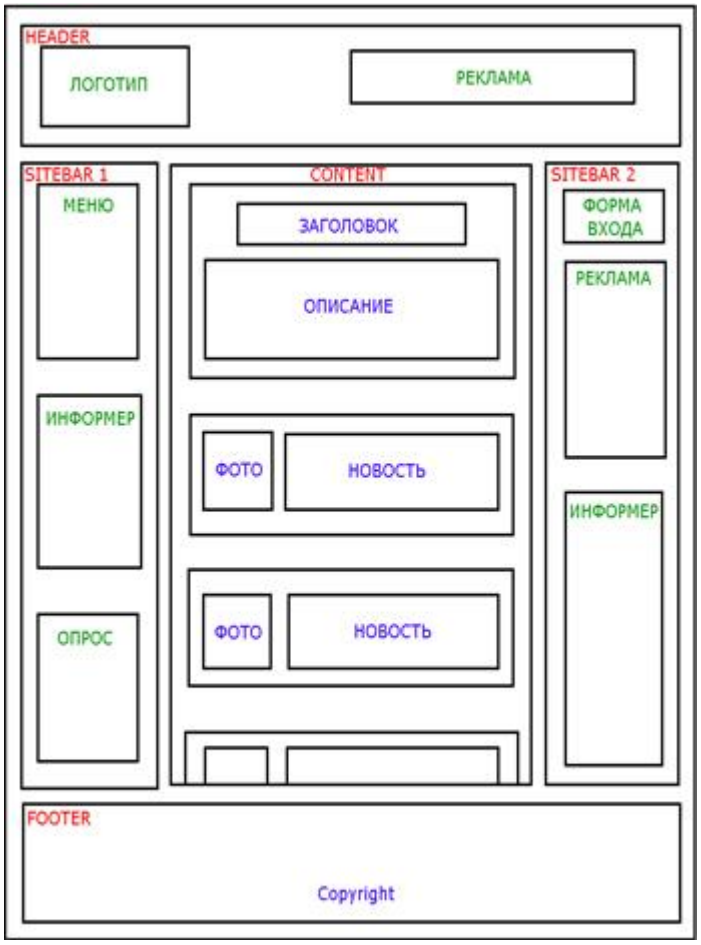
Тема 1. Дизайн сайтов

Структура сайтов. Виды меню. Требования, предъявляемые к сайтам. Дизайн сайтов. Сочетания цветов. Критерии оценки сайтов

2.1.1. Структура сайтов. Виды меню

Структура сайта – это логическая разметка и физическая связка страниц сайта, а также, расположение видимых элементов дизайна, обусловленная стандартами разработки сайтов. Разделяют внешнюю и внутреннюю структуру.

Внешняя структура включает в себя расположение видимых блоков на сайте (рис. 6) (шапка, сайтбары, футер, информеры, служебные формы и другие блоки). Разработка внешней структуры тесно связана с техническим созданием сайта.



Внутренняя структура включает в себя принадлежность материалов к определенным категориям, а категорий к разделам (другими словами – рубрикацию), а также ссылочную связку страниц. На некоторых источниках рубрикацию называют логической структурой. Именно о ней пойдет речь.

Итак, большинство веб-ресурсов построено по блочному принципу. Все просто – наверху шапка, слева - меню, справа - блок рекламы, посередине - информация, а внизу (в подвале сайта) написано - «Копирайт – все права защищены».

Структура сайта должна определяться еще на первых этапах создания проекта до начала разработки дизайна. И в будущем, по мере роста сайта, веб-мастера должны строго ее придерживаться. В этом случае можно быть уверенным, что веб-проект будет больше походить на ухоженный парк с проложенными повсюду тропинками, а не на лесную чащу, в которой посетители вынуждены будут пробираться к цели по настоящему бурелому. Однако для этого необходимо разобраться, какие вообще структуры сайта существуют и для каких проектов они больше всего подходят.

Линейная структура

Это самая простая структура сайта. Веб-страницы идут одна за другой, и пользователь должен просматривать их как слайд-шоу. В линейной структуре не существует разделения контента на уровни. Все страницы на таких сайтах равноправны, и их должен увидеть каждый посетитель. Несмотря на простоту реализации линейной структуры, недостатков у нее гораздо больше, нежели достоинств. А поэтому область ее применения четко ограничена. Она может использоваться на имиджевых сайтах-презентациях и в онлайновых учебных пособиях.

Реализация линейной структуры не представляет собой абсолютно никакой сложности. Самый простой вариант сайта — набор HTML-страниц, с каждой из которых есть ссылка на последующую предыдущую (естественно, исключение составляют крайние страницы). На каждой страницы обязательно должно быть какое-то заглавие и ссылка на первую страницу. Иначе посетители, попавшие в середину сайта, например, с поисковой системы, ничего не поймут и почти наверняка покинут проект разочарованными. Кроме того, очень полезно бывает показывать общее число страниц и как-то выделять номер той из них, на которой человек находится в данный момент. Иначе

просмотр проекта превратится для посетителей в путешествие в неизвестность.

Линейная структура с альтернативами и вариантами

Основой данной структуры является простое линейное размещение веб-страниц. Однако на сайтах, построенных по этому принципу, посетители могут проявить некоторую инициативу, облегчив для себя поиск нужной информации. Под альтернативами в данном случае понимается выбор между двумя ветками. Чаще всего подобная структура используется для сбора информации о посетителе. Примером здесь может служить процесс регистрации клиента на сайте какой-то фирмы, оказывающей определенные услуги. В этом случае все люди начинают работу со стартовой страницы. Однако потом частным лицам предлагается ввести одну информацию, а представителям коммерческих структур — другую. После этого и те, и другие попадают на одну и ту же страницу.

Линейная структура с альтернативами и вариантами удобна в том плане, что с одной стороны она позволяет веб-мастерам контролировать деятельность посетителей, направив их в определенное русло. Но с другой стороны и у последних есть возможность проявить некоторую инициативу, которая позволит им, во-первых, почувствовать свободу, а во-вторых, облегчить доступ к нужной именно им информации.

Линейная структура с ответвлениями

Это тоже контролируемая структура, которая напоминает дорогу с ответвляющимися от нее время от времени тупиковыми тропинками. То есть посетитель последовательно переходит с одной страницы на другую. Если информация, размещенная на какой-то из них его заинтересовала, и он хочет узнать подробности, то может перейти на ответвление, а потом вернуться обратно на основную «дорожку».

Главным преимуществом рассматриваемой структуры является то, что к ней легко перейти с обычного линейного размещения веб-страниц. Такое часто бывает, когда созданный однажды веб-сайт перестает удовлетворять возросшим требованиям, а глобальная переделка по тем или иным причинам невозможна. В этом случае веб-мастер может быстро и без всяких проблем расширить проект. И при этом сайт не превратится в «лесную чащу», а останется «ухоженным парком», по которому посетителям будет приятно гулять.

Древовидная структура

Древовидная структура — самый универсальный способ размещения веб-страниц. Она подходит для создания практически любых типов сайтов. Ее принцип, наверное, понятен всем. Как видно из рисунка 7 «структура сайта» – древовидное иерархическое представление всех элементов сайта, включая информационные разделы и их подразделы. Иными словами - это база веб-ресурса, которая впоследствии наполняется текстовой и графической информацией.

Пользователь при заходе на заглавную страницу оказывается перед выбором, куда идти дальше. После перехода в нужный раздел, он подбирает необходимый подраздел и т. п. У древовидной структуры очень много достоинств, мы даже не будем перечислять их все. Лучше рассмотрим ее главный недостаток (см. рис.7).

рис. 7 Пример логической структуры сайта

Речь идет о том, что в древовидной структуре очень сложно соблюдать баланс между глубиной и шириной. Если «дерево» вашего сайта будет расти только вглубь, то пользователям, чтобы дойти до какой-то информации, придется загрузить и просмотреть слишком много страниц. Естественно, это будет раздражать. Ну а если вы создадите очень широкую древовидную структуру, то посетители будут вынуждены каждый раз тратить очень много времени для выбора нужной им ветки. А это, естественно, тоже плохо. Таким образом, если вы решитесь использовать древовидную структуру сайта, то вам будет нужно

постоянно следить за ее разрастанием и придерживаться золотой середины.

Решетчатая структура

Эта структура уже на порядок сложнее всех рассмотренных ранее. В ней все страницы также размещаются в различных ветках. Но у пользователя есть возможность перемещаться по ним не только вертикально (вверх-вниз) но и горизонтально (то есть между ветками на разных уровнях). Используется решетка в основном только в каталогах. При этом перемещение между ветками на глубинных уровнях осуществляется с помощью отсылок на рубрики в других разделах.

Использование решетчатой структуры в других проектах нецелесообразно. Во-первых, она относительно сложна в реализации. Во-вторых, обращаться с «решеткой» нужно с очень большой осторожностью. Иначе наш с вами «парк» можно очень быстро превратиться в непроходимую «чащу», в которой посетители будут вынуждены долго блуждать в поисках нужной им информации.

Итак, как мы видим, существует целый ряд различных структур сайта. Мы рассмотрели только самые основные из них. Между тем, у них есть различные вариации. Какую структуру выбрать для своего проекта — решает веб-мастер.

Перед тем как создать свой сайт нужно составить четкую структуру, представив его в виде книги, у которой есть оглавление. Вот именно оглавление – это и есть навигация по сайту. Названия основных разделов, расположенные на главной странице должны нести общую информацию о содержимом, а в подразделе – детализировать ее. Таким образом, пользователь, двигаясь по сайту, будет целенаправленно перемещаться от общего к частному. Люди (в прочем, как и поисковые роботы) любят последовательность, поэтому сделайте упор на создании логической цепочки. Например, если Ваш сайт посвящен домашним животным, то названия разделов можно использовать такие: «попугаи», «рыбки», «кошки», «собаки», а подразделы соответственно, должны рассказывать конкретно о каждом виде. Например: «уход за собаками», «питание собак», «дрессировка собак» и. т.д. Далее каждый подраздел можно разделить на несколько рубрик второго уровня. Т.е. «уход за собаками» подразумевает страницы «стрижка собак», «гигиена собак» и. т.п.

Помните, что хорошая и прозрачная структура, легкая навигация – является признаком хорошего тона. Нужно придумать названия разделам и подразделам, кратко охарактеризовать их (указать какая информации будет в них размещена). Причем подобные мероприятия нужно провести для формирования сайта любого типа: визитки, интернет-магазина или крупного развлекательно-информационного портала. Не забывайте, что глубина вложенности страниц не должна превышать правила - *«3 клика от главной страницы»,* но и бросаться в другую крайность, перегружая главную страницу несколькими десятками ссылок, ведущими непонятно куда, тоже не стоит, иначе сайт будет выглядеть пестро и непрезентабельно.

Заголовки для разделов и подразделов выбирайте короткие, простые, с применением ключевых слов, по которым будет продвигаться интернет-ресурс.

Навигация сайта – «компас» пользователя

Понятная навигация сайта нужно больше всего пользователям, чем самому владельцу сайта. Наверно, каждый знает, где у него что лежит в доме, даже если на рабочем столе царит бардак, а в ящиках комода – полная неразбериха. Сайт – в отличие от жилища – это общественное место, куда мы приглашаем гостей, поэтому нужно содержать его в порядке.

Навигационное меню является отражением структуры сайта, которое видит пользователь, попадая на ресурс, поэтому оно полностью (в идеале) соответствует структуре сайта. Разрабатывая систему навигации, представьте себя рядовым пользователем, а лучше попросите протестировать сайт своих знакомых, которые оценят его со стороны.

Посетитель должен всегда иметь перед глазами путеводную нить, коей является кнопка возврата в предыдущий раздел (страницу). Также на каждой странице сайта должна быть размещена ссылка – «вернуться на главную», если Вы осуществляете продажу товаров или услуг, то обязательно ссылка типа «заказать», «купить», «забронировать» и. т.п. Неплохо бы также иметь ссылку на форму обратной связи, если сайт предполагает взаимодействие с пользователем. Все эти ссылки и кнопки не являются заменой меню, а только дополняют его, для повышения комфорта использования сайта. Таким образом, посетитель должен представлять, где он находится и куда ему двигаться далее. Интересно, что при помощи грамотно расставленных ссылок

можно не только упростить навигацию, но и подтолкнуть пользователя к совершению определенного (нужного нам) действия. Например, оформить заказ, купить товар, скачать фильм и т.п.

Структура навигации - обзор распространенных систем навигации.

Существует несколько типов навигации, меню, выпадающее меню (вертушка), поиск, перебор страниц, пиктографическая система, индексы. Расскажем подробно, чтобы можно было определится и выбрать то что нужно.

Меню

Меню – наиболее распространенная навигационная система, она похожа на оглавление к книге. Меню состоит из коротких текстовых активных ссылок на определенные страницы сайта. Отметим, что меню должно быть видно с любой страницы ресурса тогда пользователь сможет воспользоваться им в любой момент. Составление полного меню, благодаря которому будет осуществляться быстрый переход на искомые страницы, для больших сайтов (свыше 50 страниц) практически невыполнимая задача. Тогда на помощь приходит раскрывающееся меню, т.е. когда человек кликает на какую-либо ссылку, открывается несколько дополнительных ссылок, характеризующих подразделы. Можно размещать несколько меню: одно под шапкой два других по бокам – справа и слева. В том случае, если сайт небольшой, то лучше ограничиться формированием меню в правой колонке сайта, т.к. это очень удобно и привычно для пользователя. В общем, справа лучше располагать самые важные ссылки на разделы, поскольку данная зона наиболее активна на сайте (на этот угол посетитель чаще всего смотрит, наверное, приноравливаясь нажать на крестик «закрыть» в правом верхнем углу браузера).

Помните о цветовом восприятии меню – не стоит делать так, чтобы ссылки сливались с общим фоном - они должны выделяться, но, тем не менее, гармонировать с дизайном сайта. Также выделяйте другим цветом те ссылки, по которым пользователь уже однажды совершал переход.

При формировании меню не забывайте выносить в названия разделов ключевые слова, по которым будет продвигаться ресурс, однако соблюдайте баланс, ведь за стремлением все подряд оптимизировать для роботов, можно потерять реальных посетителей - людей.

Поисковая форма на сайте.

Для крупных сайтов, где просто физически невозможно уместить все наименования разделов в меню, лучше всего добавить поисковую форму. При этом есть возможность установить специальный поисковый скрипт, либо занести на свой сайт поиск, любезно предоставленный поисковой системой (например, Яндекс или Google). Преимущества «поиска» в том, том, что благодаря нему пользователь находит несколько текстов, в которых встречалось искомое слово, поэтому имеет возможность наиболее полно ознакомиться с интересующей информацией.

Недостаток поиска в том, что часто происходит некачественная сортировка и в ответ на запрос выдается куча документов, содержащих искомое слово (или словосочетание) в ином контексте - это мешает правильной фильтрации информации и затрудняет ее изучение. Также необходимо обратить внимание на корректность работы поиска на сайте. Если поиск постоянно выдает ошибку или сообщает пользователю «искомая комбинация слов нигде не встречается», хотя на самом деле это далеко не так, то стоит задуматься, либо о переустановке системы, либо вообще о ликвидации такого неудобного помощника в навигации.

Карта сайта.

Карта сайта, несмотря на то, что является вспомогательным звеном в навигации, необходима для каждого веб-ресурса. Что представляет сбой карта сайта? Как и следует из названия – это модель, схематическая иерархичная структура ресурса, размещенная на одной странице, которая наглядно показывает пользователю все его разделы и подразделы. Чаще всего карту сайта выполняют в виде дерева, «стволом» которого является главная страница, а «ветвями» – разделы. Иногда применяют укороченные варианты, отображающие лишь основные вехи в структуре сайта. Ошибкой некоторых сайтостроителей является маскировка карты сайта, обычно ссылку на нее не так-то просто найти на главной странице. Считается - сделал карту сайта для робота, он и так ее найдет, пользователь же, такой чепухой вряд ли заинтересуется. На самом деле это не так. Растерявшемуся пользователю проще всего зайти по ссылке с главной страницы, разглядеть карту сайта и пройти по нужной ссылке, чем рыться в поиске или действовать «методом тыка».

Если рассматривать карту сайта не только с точки зрения удобства пользователя, но и как инструмент для продвижения сайта, то нужно отметить следующее. Карта сайта или файл sitemap.xml - обязательный спутник каждого веб-ресурса, именно благодаря этому файлу поисковые роботы находят Ваш сайт и индексируют его страницы. Такая карта сайта размещается на отдельном файле, который носит название sitemap.xml - он создается сугубо для поисковых машин. Иногда карта сайта помещается в текстовом файле и называется sitemap.txt. В файлах хранится не что иное, как информация о структуре веб-ресурса, там же находятся данные о частоте обновления страниц и дате обновлений, указания о важности той или иной страницы сайта. Такой файл сообщает поисковым системам (Яндексу, Гуглу) о том, что появились новые веб-страницы и их нужно индексировать, иными словами, такая карта сайта в виде файла - послание роботам на понятном им языке.

Конечно, это не означает, что сайт, лишенный подобного файла никогда не проиндексируется ботами, просто на это понадобится больше времени. Обычно поисковая машина переходит по внутренним ссылкам со страницы на страницу и, в конце концов, происходит полная индексации, однако, если сайт часто видоизменяется, имеет развернутую динамичную структуру, то роботу будет затруднительно понять структуру ресурса. В итоге поисковая машина пропускает часть страниц, либо неверно определяет их важность, что плохо сказывается на процессе продвижения сайта.

Для того чтобы создать карты сайта в файлах есть специальные программы-помощники, которые располагаются на сторонних ресурсах, они очень удобны, однако после значительных изменений на сайте нужно снова формировать структуру и перезаливать ресурс на хостинг. Избавить от подобного «мартышкиного труда» помогают специальные плагины, коих великое множество - для каждого движка сайта разработаны свои, их можно скачать, установить и формировать карту сайта с их помощью. В итоге, поисковые машины будут всегда в курсе свежей и полной информации о Вашем ресурсе.

Выпадающее меню.

Выпадающее меню, становится очень популярной системой навигации, даже не столько популярной, сколько модной. Его плюсом является эффектность, «красивость» и экономия места на сайте, а минусом – низкая наглядность и эффективность. Пользователь должен

для начала навести курсор на это меню и только после данного действия «вывалится» стопка ссылок.

Перебор страниц.

Перебор страниц - удобная и очень простая (для пользователя) система навигации. Эта та самая «путеводная нить», которая держит юзера «на коротком поводке» и не дает уйти на другой сайт. Под перебором страниц понимается наличие на каждой страничке активных ссылок (вперед, наверх назад и .т.п.). Таким образом, достигается главная задача - вовлечение пользователя в процесс самозабвенного серфинга по сайту. Впрочем, те же цели преследуют внутренние ссылки, грамотно расставленные в тексте, они стимулируют к переходу на другие страницы, показывают, где находится искомая информация. Недостаток такой системы - трудоемкость формирования и поддерживания для веб-ресурсов, наполненных разнородной информацией и содержащих большое число страниц.

Пиктограммы.

Пиктограммы – довольно интересная навигационная система, которая, по идее, должна быть вспомогательной. Дело в том, что графические символы не всегда одинаково истолковываются пользователем, а то, что непонятно – пугает и отталкивает. Конечно, символ «конверт» - привычно заменяет надпись «контакты» или «обратная связь», значок «генеалогического древа» - говорит нам, что за этой кнопкой таится заветная карта сайта, а схематично изображенный «домик», отсылает нас на главную страницу. Против данных символов нечего сказать, но опять-таки они рассчитаны на пользователя мало-мальски знакомого с всемирной сетью Интернет. Тот же, кто только начинает свой путь исследования «всемирной паутины» может быть ошарашен подобными графическими примочками. Если уж Вам так хочется поразить всех своей неординарностью, то постарайтесь сделать так, чтобы при наведении курсора мыши на определенный символ, рядом появлялась надпись, характеризующая его.

Индексы.

Индексы являются также формой дополнительного поиска по сайту. В отличие от карты сайта они позволяют осуществить быстрый доступ к тем или иным страницам и разделам веб-ресурса, с использованием алфавитного ранжирования. Благодаря индексам, можно найти что-то конкретное, известное, Например, пользователь знает название

песни или компьютерной игры, и, пользуясь алфавитным указателем, открывает тот или иной перечень, находит информацию касательно искомого предмета. По сути, индексы напоминают алфавитный указатель в книге, который отправляет читателя на ту или иную страницу. Сам по себе алфавитный указатель не заменяет оглавления, так и на сайте индекс не является полноценной заменой меню или карты сайта – это просто полезное дополнение.

2.1.2 Требования, предъявляемые к сайтам.

Требования, предъявляемые к сайтам прежде всего определяются целями и задачами его создания.

Определение целей и задач – это первый этап создания качественного веб-ресурса. Правильная постановка целей является непростой работой для оптимизатора.

Основная роль этого этапа:

- точно определить тематику и роль проекта;
- узнать на какие вопросы отвечает сайт;
- найти «уникальную жилку» в сравнении с сайтами конкурентов;
- мысленно «прорисовать» портреты и сценарии поведения пользователей;
- подготовить сайт для правильной организации будущей структуры;
- выбрать тактику дальнейшей работы по созданию и продвижению сайта;

Чтобы определить цели и задачи сайта, необходимо пошагово выполнить все нижеперечисленные пункты. В любом случае, определение основной роли проекта позволяет объективно оценить его идею и будущий успех по монетизации. Разберем данный подход на примере разработки коммерческого сайта.

ШАГ 1. Составление списка вопросов.

Политика поисковых систем проста – дать наилучший ответ на запрос пользователя. Сложность заключается в том, что никто не знает, какой именно ответ, по мнению поисковой системы (ПС), будет наилучшим.

Перед тем как давать «наилучшие» ответы, нужно определить вопросы, на которые сможет ответить сайт. Причем, чем больше будет вопросов, тем лучше.

Необходимо сформулировать перечень вопросов, на которые смогу дать ответы в контенте своего сайта. Вопросы пишу быстро, особо не задумываясь.

Тут главное разнообразность вопросов и исключение смешивания тем. Иначе говоря, если вы определились, что сайт будет рекламировать услуги по ремонту, то не включайте сюда вопросы о продаже или аренде.

Для правильного выполнения данного шага необходимо составлять вопросы, на которые можно дать четкое утверждение (риторические вопросы не нужны). Возможно, что для поиска подробного ответа придется собирать информацию с разных источников, потом перерабатывать ее и формулировать по-своему и со своими дополнениями.

Некоторые плюсы данного подхода для дальнейшего продвижения в поисковых системах:

- Вы уже частично выполните работу по подбору ключевых слов.
- Вы сможете использовать вопросы в заголовках своих статей.
- Этот шаг поможет в будущем правильно организовать структуру сайта.

Не нужно составлять подобие семантического ядра, сейчас ваша задача определить, на какие вопросы отвечает ресурс, кому он будет интересен, и чем заинтересовать посетителей. Для этого необходимо составить не менее 20 вопросов. Этот шаг очень важен для сайта любой тематики.

ШАГ 2. Определение целей по шаблонной заготовке (для коммерческих тематик).

Если вы выполните все пункты для определения своего собственного бизнеса, то вполне возможно откроете для себя много новых идей для его развития. Скопируйте все написанное в блокнот и под каждым пунктом дайте расширенный ответ.

Определение компании:

1. Чем она занимается?
2. Какая ценовая категория услуг?
3. Есть ли наличие торговых марок?
4. Существует ли продающая идея в других видах рекламы?

Определение целевой аудитории:

1. Социальный уровень (профессии, классы, доходы).
2. Возраст и пол.
3. Географическое положение.

Назначение и цели сайта:

12. Смысл этого сайта?
13. Перечислите краткосрочные цели.
14. Назовите долгосрочные цели.
15. Каково его общее назначение?
16. Для чего посетитель придет в первый раз?
17. Зачем ему возвращаться?
18. Сайт должен продавать товар или демонстрировать услугу?

Анализ сайтов конкурентов:

1. Выявить основных конкурентов с помощью поисковых систем.
2. Провести визуальный анализ (дизайн, удобство навигации).
3. Проанализировать информационное наполнение.
4. Выявить «что есть у них, чего нет у нас» и наоборот.

ШАГ 3. Прорисовка сценариев посещений.

Сценарий пользователя прорисовывается от начала посещения до конца. Вам необходимо мысленно «обрисовать» тип, характер, социальный статус посетителя. Представить для него различные цели и предположить, как он будет вести себя при посещении сайта.

Подумайте, какие действия он может совершать во время визита, какие страницы станут для него целевыми, куда он последует дальше. Чем его заинтересовать, чтобы как можно дольше задержать на сайте и побудить вернуться еще раз.

Тут пригодится хорошая фантазия. Нужно искусственно создавать себе проблемы (связанные с тематикой вашего сайта) и пытаться найти их решение через интернет. Анализируйте каждый свой клик по ссылкам, учитывайте время, потраченное на поиск решения, проводите визуальный анализ сайтов и качества представления контента. Обращайте внимание на все мелкие детали и выявленные ошибки, при этом

думайте как простой рядовой посетитель и смотрите на сайт его глазами.

ШАГ 4. Вывод.

Теперь необходимо обобщить все вышеперечисленные пункты и НАПИСАТЬ окончательное заключение с четко определенными целями.

Абсолютно не важно, какова тематика сайта и на чем вы будете зарабатывать. Главное определить ценность идеи и пользу для посетителей. Выявите основные проблемы тематики в интернете, предложите пути их решения. Опишите целевую аудиторию и решите, какие вопросы будут самыми главными. Придумайте что-то уникальное и неповторимое. Чем ваш сайт будет отличаться от других подобных веб-ресурсов.

В процессе написания заключения, вы подсознательно будете опираться на все вышеперечисленные ответы. Не поленитесь написать вывод на 2-3к знаков. Его вы не будете использовать в контенте сайта, но сможете получить четкое представление своих дальнейших действий.

2.1.3. Дизайн сайтов. Сочетания цветов. Критерии оценки сайтов.

Дизайн сайта (веб-дизайн) [от англ. design — проектировать, чертить, задумать, а также проект, план, рисунок] — проектирование внешнего вида, структуры и способа функционирования веб-сайта, а также результат этого проектирования.

Идеальный сайт должен быть **простым**, **удобным**, **понятным**, **красивым** и самое главное — **продающим** (и не обязательно продающим только товары, сайт может «продавать» идеи)

16 критериев оценки дизайна сайта:

1. Производит ли дизайн сайта правильное впечатление на посетителей?

Если вы хотите купить дешёвые вещи, то вы будете ходит по вещевым рынкам, а не по бутикам. Или если вы хотите купить качественную косметику, то врядли вы будете искать её в ларьках в подземном переходе.

Также и с сайтами — если вы продаёте недорогие товары, то шикарный красочный дизайн будет отпугивать посетителей. А если вы, например, предоставляете качественные и дорогие услуги, то простенький сайт сделанный знакомым студентом может заставить сомневаться в реальности вашего предложения.

Дизайн сайта **должен соответствовать** вашему бизнесу!

2. Следует ли сайт фирменному стилю компании?

Представьте сайт Билайна выполненный в красно-жёлтых цветах. Согласитесь — это полный бред!

Дизайн сайта должен **придерживаться** фирменного стиля компании!

3. Соответствует ли сайт рекламной стратегии компании?

Например, Volvo позиционирует свои автомобили, как самые безопасные и строит свои рекламные кампании в соответствии с этим. Сайт Volvo также должен подчеркивать «безопасность».

4. Помогает ли дизайн сайта представлять товары в наилучшем виде?

Красивый дизайн — это очень хорошо, но **красиво представить товары** более важная задача.

Если у вас в дизайне присутствуют яркие и контрастные цвета, то фотографии в таком обрамлении будут казаться более блёклыми, чем есть на самом деле. И наоборот, если дизайн выполнен в пастельных (предпочтительно серых) тонах, то фотографии **будут казаться более яркими** и насыщенными.

Посмотрите на сайт apple.com — всё внимание на иллюстрациях и фотографиях, и они выглядят очень ярко и привлекательно.

5. Легко ли посетителям вашего сайта найти нужную им информацию?

Если у вас на сайте 5 страниц, то проблем с размещением информации как правило не возникает — вы просто ставите меню с названиями этих страниц и всё.

А если у вас 100 страниц или 10.000 страниц?

В случае с большим сайтом возникает необходимость продумать структуру сайта, названия разделов и подразделов так, чтобы посетителям **было понятно где искать** нужную им информацию.

6. Легко ли на сайте воспринимается информация?

Мелкий шрифт красивее смотрится, но его труднее читать. Фоновое изображение под текстом может сделать сайт не только более красивым, но и более сложным для чтения текста.

Посмотрите, легко ли воспринимается информация, которую вы хотите сообщить посетителям своего сайта?

7. Легко ли воспринимаются анонсы и рекламные блоки?

Если переборщить со спецпредложениями, то сайт становится похожим на новогоднюю ёлку.

Когда на сайте становится слишком много источников привлечения внимания (анонсы, спецпредложения, реклама), человек **перестаёт их воспринимать**.

8. Понятны ли графические символы, которые использованы в дизайне?

Часто на сайтах используют различные иконки для того чтобы облегчить восприятие информации. Используйте **общепринятые символы**. Если вы замените корзину в интернет-магазине иконкой сумки, то не все ваши посетители поймут это.

9. Легко ли найти телефон и другую контактую информацию?

Когда у посетителя сайта возникнет желание связаться с вами, очень важно чтобы ваши контакты были легко им находимы.

Довольно часто бывает так, что телефоны и e-mail находятся только на странице «Контакты».

Желательно, чтобы **телефоны** были и в начале и внизу страницы.

10. Понятен ли порядок заказа и покупки товара в интернет-магазине?

Одним из барьеров во время покупки в интернет-магазине является неуверенность покупателя в том, что он правильно сделает заказ и получит именно то, заказывает.

Нужно чтобы на сайте было максимально доходчиво объяснено, как делать заказ и каким образом покупатель получит приобретённый им товар.

Основные моменты, на которые стоит **обратить внимание**: виды оплаты, способы и сроки доставки, гарантии и условия обмена (возврата) бракованного товара.

11. Выдержан ли дизайн сайта в единой стилистике?

Если у сайта слишком отличается дизайн для разных страниц, то у посетителей может возникнуть ощущение, что они перешли на другой сайт.

Как правило все страницы сайта имеют общие элементы (логотип, меню, цвета, шрифты)

12. Сочетаются ли цвета в дизайне?

Существует целая наука сочетания различных цветов. На сайте kuler.adobe.com вы можете посмотреть различные схемы сочетания цветов.

13. Сочетаются ли шрифты на сайте?

Шрифты на сайте должны сочетаться друг с другом. Неправильно подобранные шрифты создают ощущение неаккуратности и дилетанства.

14. Качественно ли отрисованы элементы дизайна?

Думаю, что вы встречали сайты, где линии нечёткие, с зазубринами, где тени такие, что возникает ощущение грязи, где фотографии нерезкие и блёклые. Такие сайты сразу вызывают ощущение «непрофесиональности» их создателя.

Простые по дизайну, но аккуратно выполненные сайты смотрятся всегда лучше, чем навороченные, но некачественно отрисованные.

15. Присутствуют ли в дизайне оригинальные графические элементы?

Оригинальная графика создаёт чувство индивидуальности и эксклюзивности. Это могут быть как просто иконки, так и различные графические элементы.

16. Насколько тщательно проработаны детали?

В качественном дизайне есть такие мелочи, которые больше чувствуются, чем осознаются. Это могут быть тщательно прорисованные блики, еле заметные фоновые текстуры, попиксельная прорисовка букв в логотипе и заголовках.

Цвета и их значения

Рассмотрим цветовой круг.

Немного истории: *цветовой круг изобрел Исаак Ньютон. Обосновав теорию света и цветов в 1666г. Именно она легла в основу становления и развития современной оптики, малой и составной частью которой является web-дизайн. Ньютон при помощи трёхгранной стеклянной призмы разложил белый свет на семь цветов (в спектр), тем самым доказав его сложность (явление дисперсии), открыл хроматическую аберрацию.*

Цветовой круг является непременным атрибутом многих дизайнеров и художников по всему миру. Это идеальное доказательства теории, что гениальное всегда просто. Круг позволяет вам выбрать цвета, которые гармонировали бы вместе. Он состоит из 6 основных цветов: красный, оранжевый, желтый, зеленый, синий, фиолетовый и дополнительных цветов. Чтобы найти правильную цветовою схему, необходимо использовать любые два цвета друг напротив друга, любые три цвета на равном расстоянии при формировании треугольника или любой из четырех цветов, образующих прямоугольник (две пары цвета

друг напротив друга). Цветовые схемы остаются правильными независимо от угла поворота.

Основные цвета

Есть три основных цвета: красный (# ff0000 в HTML или # F00 в CSS), желтый (# FFFF00 в HTML или # ff0 в CSS) и голубой (# 0000FF в HTML или # 00f в CSS). Нельзя их получить путем смешивания других цветов. Дополнительные цвета могут быть сформированы путем объединения этих трех цветов.

Составные цвета

Есть также три основных: оранжевый (# ff9900 в HTML или # F90 в CSS), зеленый (# 00FF00 в HTML или # 0f0 в CSS) и фиолетовый (# FF00FF в HTML или # f0f в CSS). Вы можете получить их путем смешивания красного и желтого(оранжевого), желтого и синего (зеленый) и синего и красного (фиолетовый).

Третичные цвета

Чтобы получить один из третичных цветов, необходимо смешать один основной цвет и один вторичный цвет. Возможности для третичные цветов безграничны.

Дополнительные цвета

Дополнительные цвета расположены прямо напротив друг друга на цветовом круге: красный и зеленый, синий и оранжевый, фиолетовый и желтый. В сочетании друг с другом, они составляют разительный контраст. Такие сочетания, как правило, используются для выделения некоторых элементов на web-сайте.

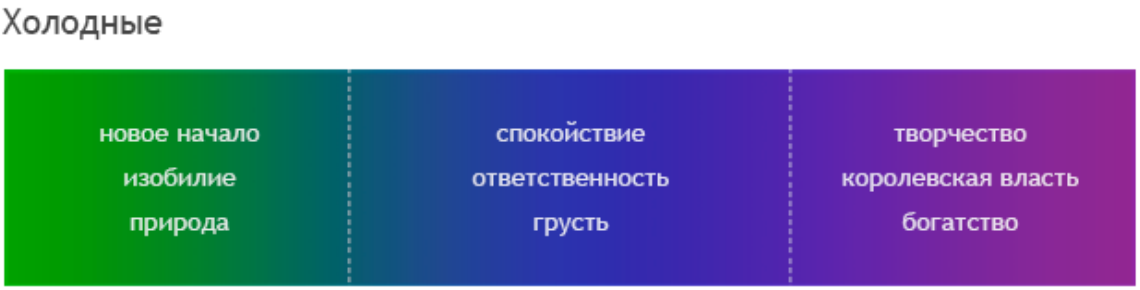
Аналогичные цвета

Эти цвета расположены рядом друг с другом на цветовом круге. Они обычно смотрят очень хорошо вместе. Использование таких цветовых сочетаний вызывает чувство комфорта у посетителей Вашего сайта.

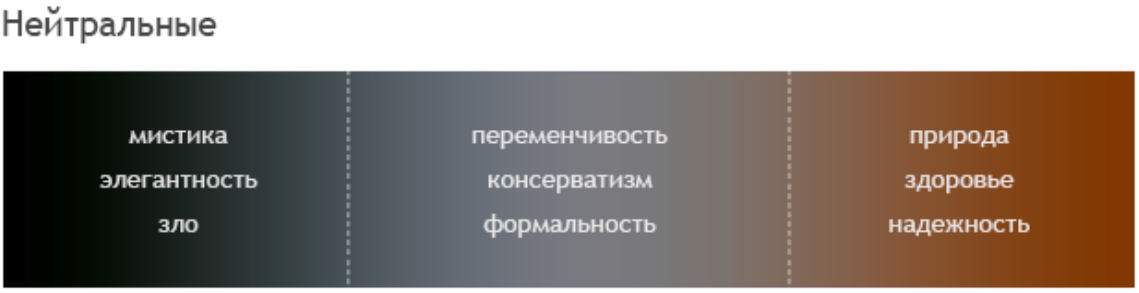
Несмотря на то, что цвета и их оттенки очень часто вызывают у разных людей разные эмоции и ассоциируются с разными вещами, все же на сайте они должны гармонично сочетаться, и нести смысловую нагрузку. Общепринятые значения цветов условно можно разделить на три группы по типу их восприятия: тёплые, холодные и нейтральные

цвета.

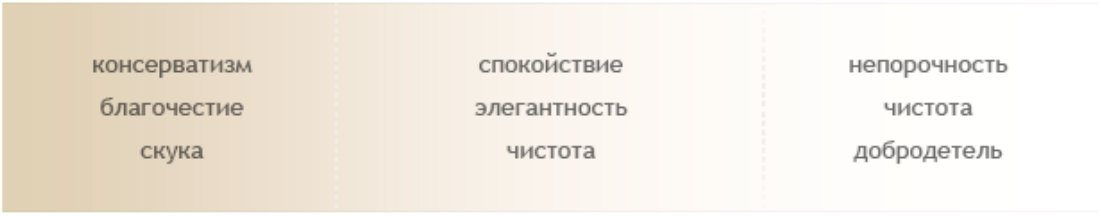
Красный Оранжевый Желтый



Зеленый Синий Пурпурный



Черный Серый Коричневый



Бежевый Кремовый Белый

Выбор цвета для сайта

1. Выбирать цвета для сайта нужно не наобум, а руководствуясь будущими целями сайта. Первое, с чего нужно начать выбор - узнать, не сделали ли его до вас). У многих коммерческих (и чуть реже неком-

мерческих) сайтов очень часто есть жесткая привязка к существующему логотипу, узнаваемым цветам и элементам предыдущей версии сайта, или попросту выработан фирменный стиль.

2. Если сайт в дальнейшем планируется продвигать в поиске или контекстной рекламе, то лучше просмотреть Топ 20, или Топ 30 (включая рекламные площадки) сайтов конкурентов. Нужно это для того чтобы найти цвета, оттенки и их сочетания которые реже используются в вашей тематике. Сайт должен запоминаться как самим дизайном, так и его уникальными цветовыми решениями, а не быть похожим на сайты конкурентов.

3. Далее следует выбрать основные цвета:

- подложки сайта (чаще всего берутся нейтральные цвета);
- основных элементов сайта (выбор зависит от задач);
- для выделения продающих и важных моментов (обычно используют яркие оттенки тёплых).

Подбирать цвета нужно руководствуясь правилами сочетания цветов. Начинающим дизайнерам советую использовать аналогичные (близкие по палитре) цвета или методы триад и квадратов по палитре цветового круга.

Аналогичные.

Этот метод основан на использовании двух – трех близких по палитре цветов и всего многообразия их оттенков.

Триады и квадраты.

Равносторонний треугольник или квадрат на цветовом круге подскажут правильное сочетание цветов, упростив выбор дизайнера.

И помните! Даже самое небольшое изменение одного цветового оттенка может вызвать абсолютное другое восприятие у пользователей. В практике нашей студии был случай, когда изменение основного цвета сайта дало увеличение конверсии и KPI почти в 2 раза.

Вопросы для самоконтроля:

1. Опишите типовую структуру сайта.
2. Какие виды меню существуют?
3. Перечислите требования, предъявляемые к сайтам.
4. Каковы критерии оценки сайтов?
5. Опишите цветовой круг.
6. Перечислите 16 критериев оценки сайта.

Тема 2. Инструменты разработки web-сайтов

Web-сервер. Виды и функции web-серверов. Инструменты разработки web-сайтов. Web-редакторы: SharePoint. Он-лайн констртор

2.2.1 Web-сервер. Виды и функции web-серверов.

Веб-сервер (**web-server**) – это сервер, отвечающий за прием и обработку запросов (HTTP-запросов) от клиентов к веб-сайту. В качестве клиентов обычно выступают различные веб-браузеры. В ответ веб-сервер выдает клиентам HTTP-ответы, в большинстве случаев – вместе с HTML-страницей, которая может содержать: всевозможные файлы, изображения, медиа-поток или любые другие данные.

Также веб-сервер выполняет функцию исполнения скриптов, например, таких как CGI, JSP, ASP и PHP, которые отвечают за организацию запросов к сетевым службам, базам данных, доступу к файлам, пересылке электронной почты и другим приложениям электронной коммерции.

Термин “веб-сервер” также применяется к техническим устройствам и программному обеспечению, которые выполняют функции веб-сервера. Это может быть какой-нибудь компьютер, который специально выделен из группы персональных компьютеров или рабочая станция, на которых установлено и работает сервисное программное обеспечение.

Клиент пользователя, которым преимущественно является веб-браузер, передает веб-серверу запросы на получение ресурсов, обозначенных URL-адресами. Ресурсы – это HTML-страницы, цифровой медиа контент, медиа-потоки, различные изображения, файлы данных, или любые другие данные, необходимые клиенту. В ответ веб-сервер передает клиенту запрошенные им данные. Этот обмен происходит с помощью протокола HTTP.

HTTP (англ. HyperText Transfer Protocol – протокол передачи гипертекста) – это сетевой протокол прикладного уровня передачи данных. Основным принципом протокола HTTP является технология «клиент-сервер», обеспечивающая взаимодействие сети и пользователя.

В случае малой организации веб-сервер может быть целостной системой, которая будет состоять из: HTTP-сервера – служит для запросов к веб-страницам; FTP-сервера – применяется для загрузки файлов через Интернет; NNTP-сервера – выполняет доступ к группам новостей; SMTP-сервера – для электронной почты.

Изобретателем первого веб-сервера считается британский ученый Тим Бернерс-Ли. Работая с 1980 года в Европейской лаборатории ядерных исследований (фр. Conseil Européen pour la Recherche Nucléaire, CERN) консультантом по программному обеспечению, он приступил к своим разработкам. В Женеве он для своих собственных потребностей разработал программу «Энквайр» (англ. enquire – спрашивать), которая использовала случайные ассоциации для хранения данных и заложила концепцию для основы Всемирной паутины.

В 1989 году Тим Бернерс-Ли, работал над внутренней сетью организации CERN и предложил основать глобальный гипертекстовый проект, который заключался в публикации гипертекстовых документов, связанных между собой гиперссылками. Внедрение этого проекта, по его мнению, облегчило бы объединение, поиск и обмен информацией для ученых CERN. Для осуществления проекта Тим Бернерс-Ли вместе со своими помощниками изобрел идентификаторы URI и URL, протокол HTTP, а также язык HTML. Все эти технологии теперь широко применяются в современном Интернете и без них уже не обойтись

В результате выполнения этого проекта Бернерс-Ли разработал первый в мире веб-сервер, называвшийся «httpd», а также первый в мире гипертекстовый веб-браузер для компьютера NeXT, получивший название WorldWideWeb (Всемирная паутина).

Первый веб-браузер работал на платформе NeXTSTEP – объектно-ориентированной, многозадачной операционной системе, и был разработан с помощью Interface Builder. Интерфейс веб-браузера был очень простым, и почти вся информация отображалась в текстовом формате только лишь с несколькими изображениями. Помимо стандартного протокола FTP, Тим Бернерс-Ли использовал новый, изобретенный им, протокол HTTP. В период с 1991 по 1993 год Бернерс-Ли усовершенствовал технические свойства своих новых разработок: идентификаторов URI и URL, протокола HTTP и языка HTML и опубликовал их. Позже веб-браузер был переименован в "Nexus", чтобы не

возникло путаницы с названием операционной системы, на которой был разработан браузер и его названием.

Первый в мире веб-сервер и первый веб-браузер работали на персональном компьютере NeXTSTEP; сейчас этот компьютер выставлен в музее CERN (Микрокосм).

Первый в мире веб-сайт Тим Бернерс-Ли разместил по адресу http://info.cern.ch; сейчас этот сайт хранится в архиве. Первый сайт появился в Интернете 6 августа 1991 года. На этом веб-сайте было дано:

- описание Всемирной паутины;
- инструкция правильной установки веб-сервера;
- информация о том, как приобрести веб-браузер;
- прочая техническая информация.

Этот сайт также представлял собой первый в мире интернет-каталог. Бернерс-Ли разместил на нем список ссылок на другие сайты и регулярно обновлял его.

12 декабря 1991 года в Стэнфордском центре линейного ускорителя (SLAC) в США был установлен первый в мире веб-сервер.

Основные и дополнительные функции

Все основные и дополнительные функции веб-сервера:

- Прием запросов от веб-браузеров по протоколу стандарта HTTP с использованием сетевых протоколов TCP/IP;
- Выполнение поиска и отсылки файлов с гипертекстом или каких-либо документов в браузер по протоколу HTTP;
- Обслуживание и обработка запросов, типа: mailto, FTP, Telnet и т. п.;
- Запуск прикладных программ на веб-сервере с последующей передачей и возвратом параметров обработки через стандарт интерфейса CGI;
- Работа и обслуживание навигационных карт изображений (Image map);
- Загрузка Java-приложений;
- Администрация и оперативное управление сервером;
- Авторизация пользователей и их аутентификация;
- Ведение регистрационного журнала обращений пользователей к различным ресурсам;
- Автоматизированная работа веб-страниц;

- Поддержка страниц, которые генерируются динамически;
- Поддержка работы протокола HTTPS для защищенных соединений с клиентами.

Описание работы веб-сервера

Веб-браузеры поддерживают связь с веб-серверами с помощью протокола передачи гипертекстовых сообщений (HypertextTransferProtocol, HTTP). Это простой протокол запросов и ответов для пересылки информации с использованием протокола TCP/IP. Веб-сервер получает запрос, обнаруживает файл, посылает его браузеру, а затем разрывает соединение. Графическая информация, которая имеется на странице, обрабатывается таким же образом. Далее настает очередь веб-браузера – вывести на монитор пользователя загруженный из сети HTML-документ.

Кроме HTML-страниц и графики, веб-серверы могут хранить любые файлы, в том числе текстовые документы, документы текстовых процессоров, видеофайлы и аудиоинформацию. На сегодняшний день, если не учитывать анкет, которые заполняют пользователи, основная часть веб-трафика передается в одном направлении – браузеры считывают файлы с веб-сервера. Но это положение изменится после общего принятия описанного в проекте HTTP 1.1 метода PUT, который позволяет записывать файлы на веб-сервер. Сегодня метод PUT используется в основном пользователями, создающими веб-страницы, но в перспективе он может пригодиться и остальным пользователям для обратной связи с информационными центрами. Запросы методом PUT намного проще, чем обыкновенная POST загрузка файлов на веб-сервер.

На веб-сервере также выполняют свою работу различные приложения, наибольшую популярность среди которых получили поисковики и средства связи с базами данных. Для разработки этих приложений применяются такие стандарты, как общий шлюзовой интерфейс (CommonGatewayInterface, CGI), языки сценариев JavaScript, а также языки программирования Java и VisualBasic. Кроме интерфейса стандарта CGI, некоторые фирмы-разработчики веб-серверов создали интерфейсы прикладного программирования (API) такие как, например, Netscape Server API и Internet Server API, которые созданы компаниями Microsoft и Process Software AG. Эти интерфейсы позволяют разработ-

чикам непосредственно обращаться к конкретным функциям веб-сервера. Некоторые веб-серверы обладают связующим программным обеспечением (middleware) для подключения к базам данных, работа с которыми может потребовать профессиональных знаний в программировании.

Базовые функции поиска помогают пользователям отсортировывать нужную им информацию, а утилиты для связи с базами данных предоставляют пользователям веб-браузеров доступ к этой информации.

Обзор веб-серверов

Критериями для выбора веб-сервера могут быть разные характеристики: установка, настройка конфигурации, управление сервером, администрирование, управление размещаемой на сервере информации, защита этой информации, контроль доступа, функции разработки приложений, а также производительность.

Большинство веб-серверов инсталлируется легко и быстро.

Самая сложная часть процесса инсталляции – это проведение конфигурации нескольких имен доменов на одном физическом устройстве или другими словами организация виртуальных серверов.

Веб-серверы имеют *средства для управления информационным модулем*, характеризующие общую организацию веб-узла, а также обладают инструментами для проверки правильности внутренних и внешних гипертекстовых связей. Пакет LiveWire фирмы Netscape Communications, который поставляется вместе с Novell Open Enterprise Server (OES) и дополнительно предлагаемый с сервером FastTrack, обладает утилитой управления узлом, которая формирует список всех связей выбранной страницы. Эта утилита также предоставляет общий перечень всех некорректных связей, которые обнаруживает. Программа WebView компании «O'Reilly & Associates» обладает такой же функцией и может выводить на экран подробное дерево файлов, в котором все некорректные связи выделяются красным цветом.

Также имеются и *элементарные средства для управления содержательным материалом.* Веб-администраторы должны выбирать, где хранить файлы и как именно будет осуществляться доступ к этим файлам со стороны пользователей, которые будут обращаться на веб-сер-

вер. Для этого требуется устанавливать соответствие между логическими URL и физическими каталогами файлов. Каждое программное обеспечение выполняет эту операцию своим уникальным способом.

С увеличением популярности веб-серверов и все более широкого их применения в интрасетях, усиливается коммерческая активность в Интернете, поэтому возрастает важность защиты информации. Чаще всего системы обеспечения безопасности веб-сервера оказываются или избыточными, или недостаточными для современных интрасетей. Если необходимо ограничить доступ к определенной информации внутри компании, то есть выбор между использованием незашифрованных паролей, которые передаются по каналам связи, и применением протокола SSL (англ. Secure Sockets Layer – уровень защищенных сокетов) – сложного и медленного метода, который используется для шифровки паролей и данных.

Для того чтобы организовать работу отдельных пользователей и их групп могут быть использованы внутренние приложения сервера или определенные функции операционной системы. Для того чтобы организовать работу отдельных пользователей и их групп могут быть использованы внутренние приложения сервера или определенные функции операционной системы. В пакетной службе Microsoft IIS предусмотрено применение средств базовой сетевой ОС Windows NT.

Пакет NetWare Web Server фирмы Novell, Inc. целиком интегрирован со службами адресных каталогов (NetWare Directory Services, NDS). Налаживать работу пользователей из общего центра удобно, но это может нести угрозу безопасности. Пароли распространяются по каналам связи в незашифрованном виде, и если их перехватят, то подвергнется риску не только веб-сервер, но и безопасность всей сетевой операционной системы.

Разработка приложений – это одна из основных функций веб-сервера. Среда разработки приложений и инструменты подключения к базам данных очень важны для расширения возможности веб-сервера, поскольку разработка приложений зависит от различных своеобразных деталей интерфейса прикладного программирования (англ. application programming interface, API), а также от особенностей языков программирования или индивидуальных предпочтений программистов.

Веб-серверы могут обслуживать различные системы от малой интрасети предприятия до крупных информационных веб-центров, которыми пользуются миллионы людей.

Для малых корпоративных интрасетей лучше всего подойдет пакет Internet Information Server (IIS), созданный и распространяемый компанией Microsoft. IIS отличается достаточно простой инсталляцией и простыми настройками конфигурации. Этот пакет веб-сервера отлично интегрирован со средствами управления доступом, инструментом контроля параметров системы Performance Monitor (Системный монитор), а также с программой просмотра журнала событий Event Viewer. Еще веб-сервером IIS представляется несколько инструментов для динамической передачи информации из баз данных. IIS отличается очень высоким быстродействием. Компоненты IIS поддерживают такие протоколы, как: HTTP, HTTPS, FTP, NNTP, SMTP, POP3.

С целью облегчить создание информационных веб-центров, с большинством веб-серверов поставляются утилиты и инструменты для управления содержательным материалом. Кроме HTML-редакторов и конвертеров форматов документов, самыми полезными являются средства контроля URL, которые гарантируют работоспособность всех гипертекстовых связей вашего веб-узла.

Любой персональный компьютер, который подключен к сети Интернет, можно сделать веб-сервером, если установить на него специальное серверное программное обеспечение.

Самые распространенные веб-серверы: Apache (компания Apache Software Foundation), IIS (компания Microsoft) и iPlanet server (от компаний Sun Microsystems и Netscape Communications Corporation). Сейчас на рынке программного обеспечения для веб-серверов, существует огромный выбор продуктов, как коммерческих, так и бесплатных.

Одним из самых распространенных веб-серверов, является *Apache* от компании Apache Software Foundation. По ориентировочным подсчетам, он используется на 65% всех веб-серверов в мире. Одно из основных достоинств программного обеспечения Apache – бесплатное распространение. Разработчики регулярно устраняют найденные ошибки и предоставляют хорошую поддержку пользователей. Данный веб-сервер поддерживает большое количество модулей, утилит и до-

полнений. Поскольку с самого начала Apache разрабатывался как программное обеспечение для администраторов и опытных пользователей, то есть недостаток – сложность настройки и обслуживания для неопытных вебмастеров.

Далее по популярности идет *веб-сервер IIS* от компании Microsoft. По данным компании Netcraft веб-сервер IIS составляет 12,46% от общего числа веб-серверов. Этот продукт входит в состав серверного программного обеспечения семейства Windows NT. Его основные преимущества – стабильность, высокая скорость работы, возможность подключения дополнительных модулей. Компания Microsoft стремится к тому, чтобы любой пользователь смог пользоваться ее продуктами без помощи специалистов, если ему нужно решить стандартные задачи. Поэтому система IIS очень проста в установке, настройке и обслуживании. Веб-сервер поддерживает технологию .NET, набирающую, в последнее время, популярность в среде разработчиков и профессиональных пользователей. Эти достоинства выводят веб-сервер IIS на новый уровень и можно ожидать, что его использование возрастет.

Другие известные веб-серверы:

- **nginx** — свободный веб-сервер и почтовый прокси-сервер, разрабатываемый Игорем Сысоевым. Простой, быстрый и надежный сервер. Работает в Linux и других Unix-подобных операционных системах, а также в Windows. Пользуется популярностью на крупных веб-сайтах;
- **lighttpd** — свободный веб-сервер. Разработчик Ян Кнешке. Быстрый и безопасный веб-сервер. Работает в Linux и других Unix-подобных операционных системах, а также в Windows;
- **Google Web Server** — веб-сервер, который основан на Apache и используется компанией Google для организации своей веб-инфраструктуры;
- **Resin** — свободный веб-сервер и сервер приложений для Java. Разработчик – компания Caucho Technology Inc.;
- **Cherokee** — свободный веб-сервер, который управляется только через веб-интерфейс. Написан на языке программирования Си;
- **Rootage** — веб-сервер, который написан на языке программирования Java. Работает в Linux и Windows;

- **THTTPD** — простой, маленький, быстрый и безопасный веб-сервер. Разработчик компания ACME Labs Software.

Клиенты веб-сервера

Обычно, клиентом является веб-браузер. Но также обращаться к веб-серверу могут и другие разнообразные устройства и программы:

- Веб-браузер, который установлен на стационарном персональном компьютере;
- Веб-браузер, который установлен на КПК или другом переносном устройстве;
- Мобильные телефоны и смартфоны, с помощью которых пользователь получает доступ к ресурсам веб-сервера по WAP-протоколу;
- Различные программы, которые могут обращаться к веб-серверу самостоятельно для обновления либо получения другой информации. Пример – различные антивирусы, которые периодически обращаются к веб-серверу, чтобы обновить базу данных;
- Разные цифровые устройства, а также некоторая бытовая техника.

2.2.2. Инструменты разработки web-сайтов. Web-редакторы: SharePoint. Он-лайн конструктор.

Подавляющее большинство web-страниц связаны в web-сайты.

Web-сайт - это система связанных между собой страниц, принадлежащих одной организации или лицу.

Рассмотрим инструменты и технологии для создания web-сайтов :

1. **Язык разметки гипертекста HTML**
2. **Визуальные HTML-редакторы**
3. **Прикладные технологии, которые могут применяться при создании web-сайтов**

Независимо от того при помощи какой технологии создаются web-сайты, все его web-страницы будут содержать разметку, написанную на языке разметки гипертекста (Hyper-text markup language - HTML).

Основу языка HTML составляют теги - специальные записи в угловых скобках. Их можно разделить на **парные** и **одиночные**.

Например тег абзаца парный:

Текст абзаца, текст абзаца

А тег переноса строки - одиночный:

Писать сайт на языке html можно непосредственно в текстовом редакторе "блокнот", а получившимуся файлу при сохранении указать расширение html.
Недостаток такого способа создания web-страниц: требуется знания большинства html-тегов. Поэтому, для упрощения создания сайтов появились дополнительные инструменты - *html редакторы.*

Виды html-редакторов (рис.8):

1. Редакторы исходного кода
2. Визуальные html-редакторы

Рис. 8

Редакторы исходного кода облегчают его написание при помощи дополнительной функциональности по сравнению с блокнотом. Например, они могут проверять его правильность. Так же они предлагают для использования некоторые шаблоны кода. Они могут использоваться не только для написания HTML-кода.
Их преимущества:

1. *имеют дополнительную функциональность,*
2. *имеют встроенные шаблоны кода*
3. *могут использоваться для написания кода на разных языках программирования*

Некоторые распространенные редакторы исходного кода:

1. *Notepad++*
2. *PSPad*

Редакторы исходного кода

Преимущества:

- ✓ имеют дополнительную функциональность;
- ✓ предлагают для использования некоторые шаблоны кода;
- ✓ многозадачность.

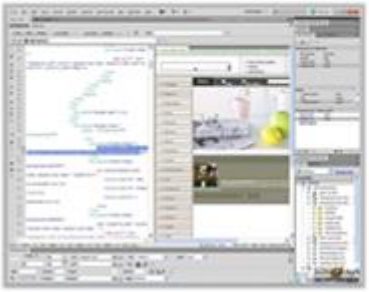
Распространенные редакторы исходного кода:

- ✓ NotePad++;
- ✓ PSPad.

Существует множество **визуальных HTML редакторов** различной сложности. Рассмотрим наиболее распространённые из них.

1.Начнём с визуального редактора **«Adobe DreamWeaver»**. Это одна из самых распространённых коммерческих программ для веб-дизайна. «Adobe DreamWeaver» предоставляет большой выбор инструментов, а также достаточно тонкие настройки для опытных веб-дизайнеров. В то же время начинающий пользователь может создавать веб-страницы с помощью этого редактора практически без знаний языка разметки гипертекста. В этом ему может помочь встроенный мастер настройки элементов веб-страницы. В нем достаточно выбрать нужный элемент из доступных, а затем настроить его основные параметры.

Визуальный html-редактор Adobe DreamWeaver



Dw Описание:

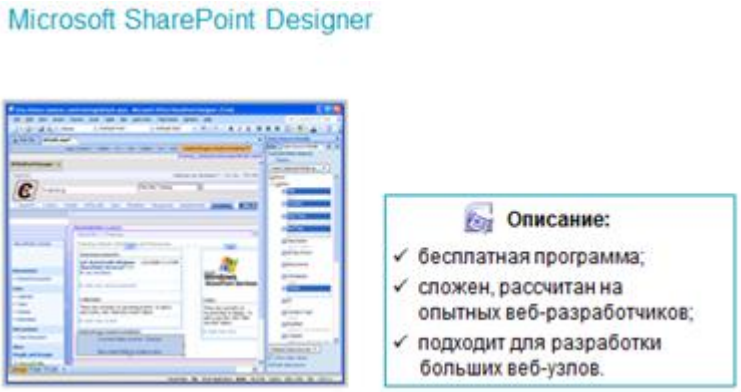
- ✓ коммерческий продукт;
- ✓ имеет достаточный функционал для опытных веб-разработчиков;
- ✓ доступен для начинающих пользователей благодаря мастеру настройки элементов веб-страницы.

Рассмотрим самые распространенные визуальные редакторы.
Adobe Dreamweaver:

1. **платная программа**
2. **имеет огромное количество настроек, под любые потребности опытного разработчика**

малоопытные web-разработчики быстро осваивают программу благодаря встроенному мастеру настроек элементов web-страницы

2.Визуальный редактор **«Microsoft SharePoint Designer»**, изначально был коммерческим проектом, но теперь доступен для свободного скачивания на официальном сайте корпорации Майкрософт. Данная программа считается достаточно сложным визуальным редактором, рассчитанным на опытных пользователей. С его помощью можно создавать не только простые веб-страницы, а также большие веб-узлы, которые предназначены для совместной работы большого числа пользователей.



Microsoft SharePoint Designer:

1. **Бесплатная программа**
2. **Достаточно сложный интерфейс, расчитанный на опытного web-разработчика.**
3. **Подходит для разработки больших web-узлов, предназначенных для совместной работы большого числа пользователей.**

3.Простым и быстрым коммерческим визуальным HTML-редактором является программа **«Web Page Maker»**. Чтобы разместить, элемент на веб-странице пользователю достаточно выбрать его в списке готовых и при помощи мыши перетащить на нужное место. В составе программы есть большое количество готовых шаблонов, которые можно использовать как основу для новых сайтов. Так же в данной программе имеется встроенный *FTP-менеджер*, который позволяет быстро загружать готовые сайты на веб-сервер. Сайты, которые созданы с помощью это программы, одинаково правильно отображаются на всех популярных браузерах. К преимуществам этой программы

можно отнести её малый размер. Установочные файлы занимают всего три с половиной мегабайта дискового пространства.

Web Page Maker

1. **Коммерческая (платная) программа,**
2. **Проста в использовании,**
3. **Имеет много заготовок готовых шаблонов web-страниц**
4. **Имеется встроенный FTP - менеджер, для удобства загрузки готового сайта на web-сервер.**
5. **Сайты, созданные в этой программе одинаково правильно отображаются на всех популярных браузерах.**
6. **Малый размер программы (около 3,5 МБ дискового пространства)**

4.Визуальный HTML-редактор **«Nvu»** с открытым исходным кодом изначально задумывался как бесплатная альтернатива редакторам «Adobe DreamWeaver» и «Microsoft SharePoint Designer». Однако намеченного уровня функциональности он так и не достиг. В две тысячи шестом году его поддержка была прекращена. Его главным преимуществом является кроссплатформенность. Данный редактор прост в применении. Он имеет большое количество встроенных шаблонов, а также встроенный FTP-менеджер.

Nvu

Описание:

- ✓ бесплатная программа с открытым кодом;
- ✓ поддержка прекращена;
- ✓ имеются версии для различных операционных систем;
- ✓ прост в использовании;
- ✓ имеется большое количество шаблонов и встроенный FTP-менеджер.

NVU

1. **Бесплатная программа**
2. **Поддержка прекращена в 2006 году.**
3. **Преимущество - кроссплатформенность (есть версии для Windows, Linux, MacOS)**
4. **Прост в использовании**
5. **Имеет большое количество встроенных шаблонов**

Имеет встроенный FTP-менеджер

6. В две тысячи шестом году на основе открытого кода редактора «Nvu» был выпущен бесплатный HTML-редактор **«Kompozer»**. В две тысячи седьмом году он был объявлен лучшей бесплатной альтернативой «Adobe DreamWeaver» по версии сайта «Download.com». По сравнению с «Nvu», «Kompozer» генерирует код на языке разметки гипертекста более рационально. Он очень прост в использовании, но при этом имеет достаточную функциональность для разработки небольших веб-проектов.

Kompozer

Описание:

- ✓ бесплатная программа с открытым кодом;
- ✓ рациональная генерация html-кода;
- ✓ прост в использовании;
- ✓ имеется достаточная функциональность для разработки небольших веб-проектов.

Несмотря на развитие языка разметки гипертекста, в две тысячи четырнадцатом году была выпущена его пятая версия, не только он используется при создании современных веб-сайтов.

Kompozer (появился в 2006г. на основе NVU)

1. **Считается лучшей бесплатной альтернативой Adobe Dreamweaver**
2. **Бесплатная программа**
3. **Более рациональная генерация кода**
4. **Прост в использовании**
5. **Имеет достаточную функциональность для разработки небольших web-проектов.**

Кратко рассмотрим другие технологии, применение которых, вместе с языком разметки гипертекста, так же достаточно распространено.

Каскадные таблицы стилей или «CSS». Это мощный инструмент для оформления блоков гипертекста. Они позволяют вынести все описание оформления в отдельные файлы, что позволяет не загромождать код страницы на языке разметки гипертекста.

Сценарные языки программирования (JavaScript) - позволяют "оживить" сайт, включив в него динамичные блоки, а также придав страницам интерактивность.

Для хранения различных данных и быстрого доступа к ним используются различные базы данных. Основной системой управления базами данных для веб-сайтов является «MySQL»

Для того чтобы реализовать логику работы различных инструментов веб-страницы, а так же совместить все эти технологии используются различные языки программирования. Самым популярным языком программирования для разработки веб-страниц является *PHP*.

Вопросы для самоконтроля:

1. Какими бывают теги для разметки html-страниц
2. Укажите разновидности html-редакторов.
3. При помощи какого языка пишется разметка веб-страниц?
4. Укажите преимущества визуальных HTML-редакторов, по сравнению с редакторами исходного кода при оформлении веб-страниц?
5. Как иначе называются визуальные html-редакторы?
6. Как называется параметр тега в языке HTML?
7. При помощи каких программ можно создавать веб-страницы на языке разметки гипертекста?
8. Как называется указание к оформлению разметки веб-страницы на языке HTML?
9. Укажите преимущества редакторов исходного кода перед текстовыми редакторами при создании веб-страниц?
10. Укажите недостатки визуальных HTML-редакторов.

Тема 3. Публикация сайтов

Хостинг. Бесплатный хостинг. FTP. Размещение Интернет-ресурса на сервере провайдера. Регистрация Интернет-ресурса в каталогах и поисковых системах

2.3.1 Хостинг. Бесплатный хостинг.

Само слово «хостинг» произошло от английского слова «host», в дословном переводе имеющего ряд значений: «хозяин», «содержатель постоялого двора», «главный компьютер» и «основное устройство». Если объяснять это понятие более простым языком, то «содержатель постоялого двора» подходит больше по смыслу, так как на сервере фирмы, предоставляющей услугу хостинга, размещены тысячи самых разных сайтов, своего рода жителей.

Следовательно, **хостинг** – это услуга, которую предоставляет так называемая хостинговая компания, позволяющая вам размешать свои веб-сайты на её серверах.

Речь идет об услугах размещения определенного ресурса на веб-сервере. *Сервер* - это программное и аппаратное решение (сочетание компьютера и особой системы или программы), главной задачей которого можно считать обработку, а также хранение информации. Кроме того, благодаря данному решению множество других компьютеров получают доступ к данным, которыми наполнен сайт.

Исходя из вышеизложенного, попробуем дать еще одно важное определение: *хостинг серверов* - это услуга по размещению серверов, а также оборудования на специальной площадке. Речь идет о комплексе мер по обеспечению стабильной работы сервера.

Хостинг серверов можно разделить *на 2 категории*. К первой относится предоставление клиенту определенного физического сервера с заданной конфигурацией, а также его размещение. Вторая категория - аппаратный хостинг. В таком случае на площадках провайдера размещается оборудование, которое принадлежит непосредственно клиенту. Данный вид хостинга называют *колокацией*. В любом случае клиент оставляет за собой права администратора сервера, осуществляя полный контроль над оборудованием.

Теперь мы попробуем разобраться с еще более сложным вопросом. Его можно сформулировать так: «VDS-хостинг - что это такое?».

Чтобы постараться ответить на данный вопрос, представим себе одну из комнат крупного хостинг-провайдера. Рядами в большом зале размещены серверы, их очень много. Так вот, если бы технологии VDS не существовало, каждый из компьютеров отвечал бы за работу всего одного сайта. Катастрофические последствия. Итак, представим, что VDS не существует и пространство серверной, которая обеспечивает хостинг, расходуется чрезвычайно неэкономно и все время требует новых территорий. Расход электроэнергии бьет все возможные рекорды, и это только для обслуживания одного ресурса. Бесплатный хостинг совсем исчезает, а цены на традиционный взлетают ввысь, и теперь содержать сайты могут себе позволить только миллионеры, поскольку интернет-хостинг - это основа любого виртуального ресурса. К счастью, серверы снабжены операционными системами, которые являются многозадачными и многопользовательскими: Windows Server, VXworks, BSD, Linux. Такой подход позволяет максимально использовать все ресурсы мощного компьютера, распределяя производительность между значительным количеством сайтов, тем самым осуществляется хостинг всего описанного многообразия на единой машине. Следует отметить, что владельцы сайтов не ощущают разницы, как будто их хостинг обеспечивается отдельными физическими серверами.

Существует также такое понятие *как хостинг сайтов.*

Без подобной услуги Интернет просто не смог бы существовать и так интенсивно развиваться, ввиду того, что веб-мастерам негде было бы размещать свои сайты. Даже разместив свой сайт на своем локальном (домашнем) компьютере, вы не сможете постоянно держать его в рабочем состоянии и подключенным к сети Интернет. А если у Вас несколько сайтов, и довольно посещаемых, то размещать сайты на домашнем компьютере вообще невозможно. А вот сервер – это компьютер, обладающий большой мощностью и рассчитанный на непрерывную и долгосрочную работу, он имеет высокоскоростное соединение с Интернетом, на нём установлено только специальное программное обеспечение и за ним круглосуточно следит администратор. Именно на нем размещаются ресурсы клиентов хостинговой компании.

Популярность и спрос на хостинг сайтов намного выше. Причина в том, что последнее решение оптимально подходит для любых ресурсов, размещенных во всемирной паутине, а хостирование серверов разумно только в случае крупных сайтов.

На данный момент существуют как платные хостинги, так и бесплатные.

Особенности VPS. Если постараться коротко ответить на вопрос «VPS-хостинг - что это такое», то можно сказать, что речь идет об очередном рывке в сфере информатизации общества. VPS с английского переводится как собственный виртуальный сервер. Суть данного решения заключается в делении одного сервера на определенное количество независимых виртуальных серверов. Данные виртуальные части обладают всеми возможностями обычного хостинга. Интересно, что виртуальные хостинги, расположенные рядом, работать могут под управлением разных операционных систем: Windows, Unix. Каждый собственный сервер обладает одним или несколькими выделенными IP-адресами, ему гарантируются минимальные ресурсы, а также быстрая перезагрузка. Как видим, VPS от традиционных решений значительно отличается. Здесь гарантируется стабильная работа и невысокая стоимость услуг. Кому это нужно? Давайте разберемся, кому и для каких целей может понадобиться подобная услуга. Необходимость в VPS возникает, если клиент желает управлять работой серверов собственноручно. Чаще всего, такая потребность появляется у людей, имеющих личный интернет-магазин. VPS нужен и тем, кто хочет применить нетрадиционное программное обеспечение либо нестандартные конфигурации.

Вообще, хостинг имеет название «виртуальный», что означает, что Вам предоставляется не весь сервер, а лишь определённая его часть, что представляет из себя обычно дисковое пространство. Так же по отношения к серверу иногда указывается лимит использования оперативной памяти и центрального процессора. При формировании цены на хостинг помимо дискового пространства учитываются так же количество доменов и поддоменов на аккаунт, количество паркованных доменов, e-mail и FTP-пользователей, трафик, программное обеспечение и т.д.

Виртуальный хостинг предназначен для сайтов средней посещаемости (хотя, многое зависит от движка). Если у Вас высоко посещаемый сайт, либо он просто сильно загружает сервер, то хостинговая компания может представить Вам выделенный сервер. Это означает, что будет специально собран из необходимых Вам элементов сервер, на котором будут размещены только Ваши сайты.

Сегодня существуют сотни хостинг-провайдеров. Выбор достаточно сложен, необходимо пользоваться отзывами.

2.3.2 FTP.

FTP (англ. *File Transfer Protocol* — протокол передачи файлов) — стандартный протокол, предназначенный для передачи файлов по TCP-сетям (например, Интернет). Использует 21-й порт. FTP часто используется для загрузки сетевых страниц и других документов с частного устройства разработки на открытые сервера хостинга.

Протокол построен на архитектуре «клиент-сервер» и использует разные сетевые соединения для передачи команд и данных между клиентом и сервером. Пользователи FTP могут пройти аутентификацию, передавая логин и пароль открытым текстом, или же, если это разрешено на сервере, они могут подключиться анонимно. Можно использовать протокол SSH для безопасной передачи, скрывающей (шифрующей) логин и пароль, а также шифрующей содержимое.

Первые клиентские FTP-приложения были интерактивными инструментами командной строки, реализующими стандартные команды и синтаксис. Графические пользовательские интерфейсы с тех пор были разработаны для многих используемых по сей день операционных систем. Среди этих интерфейсов как программы общего веб-дизайна вроде Microsoft Expression Web, так и специализированные FTP-клиенты (например, FileZilla).

FTP является одним из старейших прикладных протоколов, появившимся задолго до HTTP, и даже до TCP/IP, в 1971 году. В первое время он работал поверх протокола NCP. Он и сегодня широко используется для распространения ПО и доступа к удалённым хостам.

Первая реализация протокола предусматривала обмен между клиентом и сервером сообщениями, состоящими из заголовка (72 бит) и данных переменной длины. Заголовок сообщения включал в себя запрос к FTP-серверу или ответ от него, тип и длину передаваемых данных. В качестве данных передавались параметры запроса (например, путь и имя файла), информация от сервера (например, список файлов в каталоге) и сами файлы. Таким образом, команды и данные передавались по одному и тому же каналу.

В 1972 г. протокол был полностью изменён, и принял вид, близкий к современному. Команды с параметрами от клиента и ответы сервера передаются по TELNET-соединению (канал управления), для передачи данных создаётся отдельное соединение (канал данных).

В последующих редакциях была добавлена возможность работы в пассивном режиме, передачи файлов между FTP-серверами, введены команды получения информации, смены текущего каталога, создания и удаления каталогов, сохранения файлов под уникальным именем. Некоторое время существовали команды для передачи электронной почты через FTP, однако впоследствии они были исключены из протокола.

В 1980 г. FTP-протокол стал использовать TCP. Последняя редакция протокола была выпущена в 1985 г. В 1997 г. появилось дополнение к протоколу, позволяющее шифровать и подписывать информацию в канале управления и канале данных. В 1999 г. выпущено дополнение, посвящённое интернационализации протокола, которое рекомендует использовать кодировку UTF-8 для команд и ответов сервера и определяет новую команду LANG, устанавливающую язык ответов.

Чтобы передать файл по TCP-порту через протокол FTP, FTP-клиенту нужно связаться с настроенным и запущенным FTP-сервером. Подобная организация передачи файлов наиболее часто используется веб-разработчиками — когда с ПК нужно получить доступ к «внутренностям» сайта и что-нибудь там изменить, улучшить.

Как и HTTP, протокол FTP построен на «клиент-серверной» архитектуре и использует несколько сетевых соединений, чтобы передавать команды и файлы между «клиентом» и «сервером».

Доступ к сайту по FTP аккаунту. Это функция, позволяющая просматривать, изменять, скачивать и загружать файлы. Для получения клиентом доступа к FTP-серверу сайта нужно пройти аутентификацию — ввести логин и пароль. Для отправки имени используется команда «USER», для отправки пароля — команда «PASS».

Если сервер принимает пару логин:пароль, то он отправляет клиенту приглашение. Клиент принимает приглашение и начинается сессия (рис.9).

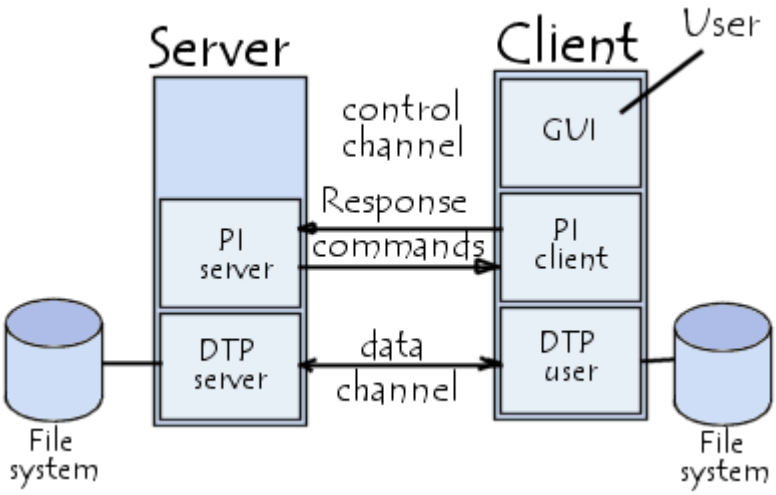


Рис.9

У FTP предельно простой синтаксис, описанный в спецификации RFC1738 (данные в квадратных скобках необязательны для заполнения) — ftp://[<POLZOVATEL>[:<PAROL>]@]<HOST>[:<PORT>]/<FTP_URL>.

Пример 1 — ftp://ftp.7bloggers.ru/FTP/humans.txt

Пример 2 — ftp://UNIQUE_USER:STRONG_PASS@ftp.7bloggers.ru/FTP/humans.txt

Также есть т.н. «анонимный FTP» — когда любой пользователь может анонимно подключиться к ФТП серверу без предоставления данных USER/PASS. Для сессий такого типа предоставляется ограниченный доступ.

Пользователи осуществляют анонимный вход как «anonymous» (имя пользователя), но бывает и так что просят ввести email-адрес вместо пароля. Конечно же, никто не проверяет эти адреса на предмет достоверности.

Анонимные FTP-хосты достаточно популярны, ведь они часто используются для скачивания и обновления ПО на ПК. Более того, доступ может быть организован и через обычные браузеры — ведь они могут напрямую извлекать файлы с FTP-серверов. Происходит это очень просто — при указании FTP-адреса FTP-контент предоставляется точно также как веб-контент при HTTP. А в браузере Фаерфокс можно даже установить полноценный FTP-клиент — FireFTP.

Несмотря на то что работать с FTP можно в браузере, FTP довольно сильно отличается от HTTP:

Особенности протокола FTP:

Свойство	FTP	HTTP
Основан на сессиях работы	Да	Нет
Встроена аутентификация пользователей	Да	Нет
В основном предусмотрен для передачи	Больших двоичных файлов	Небольших текстовых файлов
Модель соединения	Двойное подключение	Одиночное подключение
В основном приспособлен для приёма/передачи	Приёма и передачи	Приёма
Поддерживает текстовый и двоичный режимы передачи	Да	Нет
Поддерживает указание типов передаваемых данных (MIME заголовки)	Нет	Да
Поддерживает операции над файловой системой (mkdir, rm, rename, и т. д.)	Да	Нет

1. Использование множественного подключения, минимум — двойного. Один канал — управляющий, через него поступают команды для фтп-сервера и возвращаются ответы; другие каналы используются для передачи данных: одна передача — один канал (для каждого открывается TCP-порт). Благодаря этому свойству, в обоих направлениях одновременно можно передавать несколько файлов, а управляющий поток остается открытым на все время ftp-сессии.
2. В FTP-протоколе есть двоичный режим передачи данных — при котором уменьшается время передачи файлов и расход трафика. В HTTP такого нет.
3. По FTP-протоколу проводятся операции в рамках одной сессии с помощью TCP/IP — пока сервер «не забыл» текущее состояние и авторизованного пользователя. В HTTP сессий нет: он просто отдает данные.

FTP имеет три режима передачи данных:

- Поточный — непрерывная передача данных в виде потока (без обработки, обработка выполняется TCP);
- Блочный — FTP делит данные на блоки (заголовок, поле данных, размер файла в байтах) и передает их TCP;
- Режим сжатия данных единым алгоритмом.

Обычный FTP не является безопасным, потому что данные не шифруются при передаче. Это можно исправить, например используя протокол SSH, который зашифрует пару логин:пароль и передаваемое

содержимое. Рассмотрим подробнее обо всех безопасных FTP — FTPS, SFTP, SSH-FTP.

FTP был разработан до TLS/SSL и просто физически не может шифровать свой трафик, поэтому любой человек, который способен перехватить пакет по сети, получит данные к именам пользователей, паролям, командам, а значит обретет доступ к приватному FTP-серверу.

Решение этой проблемы — использование защищенных версий протокола ФТП. Например, неявный FTPS это TLS-защищенная версия FTP, а SFTP/SCP защищены Secure Shell. Расскажу подробнее о защищенных FTP.

Явный FTPS (FTPES, FTP over Explicit SSL/TLS) — расширенный FTP, создающий возможность «клиентам» требовать шифрование FTP-сессии при использовании команды «AUTH TLS». В ответ на нее, сервер может позволить создать такое соединение или отклонить запрос. Порт — 21. Есть еще неявный FTPS (требует SSL- или TLS-соединение), но он устарел.

SFTP (англ. «SSH File Transfer Protocol») — расширение протокола SSH. Он никак не связан с FTP, но точно также передает файлы и использует те же команды. SFTP использует Secure Shell (SSH) для передачи файлов, т.е. шифрует и файлы, и команды (не передает данные по сети в открытом виде). Порт — 22 или 2222. Функционально SFTP близок к FTP и очень похож на него, но клиенты стандартного ФТП не могут подсоединиться к SFTP-серверу, как и наоборот.

SFTP, используемый как подсистема второй версии реализации протокола SSH, имеет ряд преимуществ перед FTP:

- Поддержка аутентификации «без пароля» с помощью SSH-ключей более безопасная, чем если хранить пароль на диске или вводить вручную;
- Поддержка символических ссылок;
- SFTP-соединение более быстрое и надежное, когда в FTP бывают тормоза и перебои;
- SFTP-клиенты обладают возможностями прерывания и возобновления передачи файла, его удаления; загруженные файлы могут иметь метки времени, связанные с атрибутами файлов — а в FTP нет условйи для загрузок.

FTP через SSH (обратите внимание, SSH-FTP это не SFTP) — осуществляет обычную FTP-сессию, соединяясь через безопасный SSH-туннель. Такое тунеллирование затруднительно, т.к. FTP открывает несколько TCP-соединений. Это значит, что при установке несколькими SSH-клиентами туннеля для канала управления (по порту 21) защищенным будет только этот один канал, а передача данных пойдет по вновь созданным каналам данных (TCP-соединения), обходящим SSH и от этого являющимися небезопасными.

Проблема обмена данными, когда нужно оперативно передать коллегам, друзьям или знакомым какие-то объемные материалы (которые по почте не отправишь), знакома многим пользователям. Скажем, вам регулярно необходимо предоставлять рабочие материалы (презентации, изображения и т.п.) другим сотрудникам для их обсуждения. Или срочно захотелось поделиться с друзьями своими фотографиями, MP3-файлами или даже видеороликом с недавней вечеринки. Или потребовалось передать кому-то из знакомых дистрибутив некого приложения и т.п. Как поступить в таком случае? Вариантов множество -

можно воспользоваться услугами фото- или видеохостинговых сервисов, разместить файлы в онлайновом хранилище либо обратиться к файлообменным сервисам.

Но есть и другой вариант - *создать свой FTP-сервер,* который позволит сделать обмен данными более быстрым, безопасным и удобным.

FTP-сервер — «библиотека» файлов на хостинге, используется для хранения файлов разных форматов. Соединение между FTP-сервером и FTP-клиентом происходит по протоколу передачи данных FTP.

Самые популярные ftp-сервера это vsftpd и proftpd. Их настройка осуществляется в файлах ftpaccess.

ФТП-сервера нужны для того, чтобы размещать на них для публичного и приватного скачивания больших объемов данных: как по количеству файлов, так и по их размеру. Часто такие сервера используются для анонимного (гостевого) доступа к размещенным в открытом виде дистрибутивов ПО, музыки и фото. Доступ для анонимов как правило позволяет только просматривать каталоги и скачивать требуемую информацию, но на некоторых серверах наоборот — есть спецкаталоги, куда любой аноним может загрузить файл для обмена.

При неанонимном доступе возможности пользователя шире (можно загружать файлы), но строго ограничиваются тем каталогом, куда ему предоставлен доступ, даже если «выше» или «по соседству» есть другие каталоги с файлами других пользователей.

Несмотря на то, что работа с FTP-серверами может быть организована из браузера, лучше пользоваться программами-клиентами — в случае обрыва связи с сервером они позволят докачать файл, как только связь будет восстановлена.

Наиболее популярные FTP-клиенты — программы для доступа к FTP-серверам с шифрованием передаваемых данных:

- WS_FTP;
- LeechFTP;
- CuteFTP;
- FileZilla — самый популярный FTP-клиент для Windows/Mac/Linux. Есть поддержка FTPS, SFTP.
- FAR Manager — самый «древний» консольный файловый менеджер для Виндоуса. Очень простой, имеет много плагинов, поддерживает SFTP (нужен плагин WinSCP).

- Total Commander — теряет популярность (на любителя) и поддерживает FTPS. Плагины для SFTP, к сожалению, устаревшие.
- FireFTP — а это вообще плагин для Мозиллы Фаерфокс. Есть поддержка FTPS, SFTP.
- Cyberduck — фтп клиент для маков, поддерживает FTP/SFTP.
- WinSCP — минималистичный и красивый SCP.

С помощью FTP-сервера можно будет не только открывать доступ к определенным папкам на собственном компьютере, но и гибко управлять объемами трафика, а также списками доступных файлов и пользователей. Технология такой настройки довольно проста. Достаточно просто инсталлировать соответствующую утилиту, создать в ней нужный набор учетных записей (скажем, по числу пользователей, для которых вы планируете открыть доступ к серверу), определить для каждого из пользователей права доступа и указать домашний каталог (под ним понимается каталог, который вы выделите на жестком диске для обмена данными). Ну и, разумеется, потребуется сообщить утилите внешний (интернетовский) IP-адрес - то есть тот самый адрес, по которому ваши друзья и будут попадать на ваш FTP-сервер. Последнее, впрочем, может и не потребоваться, поскольку часть утилит данный адрес могут установить самостоятельно. Понятно, что спектр настроек любого FTP-сервера этим не ограничивается, и сразу оговоримся, что мы не ставим задачей при знакомстве с той или иной конкретной утилитой все их перечислять. Но в целом, указанных действий уже вполне достаточно для того, чтобы открыть доступ к серверу вашим друзьям, которые, указав в FTP-клиенте IP-адрес вашего FTP-сервера, смогут получить доступ к домашнему каталогу на вашем жестком диске. Правда, тут стоит отметить один нюанс - если у вас IP-адрес статический, то никаких проблем не возникнет. А вот если динамический - дело иное, поскольку динамический IP при каждом вашем подключении к интернету будет уже другим. Конечно, и с динамическим адресом FTP-сервер заработает как миленький, но в таком случае каждый раз после входа в сеть вам придется сообщать друзьям (например, по почте) новый адрес вашего FTP-сервера.

2.3.3. Размещение Интернет-ресурса на сервере провайдера.

Способы размещения сайтов в интернет:

Размещение своего сайта на сервере местного интернет-провайдера.

Такой способ является *наиболее предпочтительным*.

- Получаете зарегистрированное лично на Вас доменное имя второго уровня.

- Вы можете лично поговорить с людьми занимающимися обслуживанием сервера, на котором размещен Ваш сайт, и на месте обсудить с ними имеющиеся проблемы.

- Не тратится время на переписку по e-mail.

К недостаткам можно отнести:

- Необходимы денежные вложения для размещения и поддержки сайта и доменного имени.

- Вы сможете пользоваться только теми услугами, которые Вам может предоставить местный провайдер.

- Вы сможете пользоваться только тем оборудованием, которое Вам может предоставить местный провайдер. Например, скоростные характеристики сервера.

- При ограниченном количестве местных интернет-провайдеров бывает невозможно перейти к другому, если предыдущий Вас не устраивает.

- Вы тратите свое время на посещение офиса провайдера, заключения договоров и ожидания прохождения оплаты.

Размещение сайта на иногороднем или на зарубежном платном сервере.

Достоинства заключаются в большом количестве компаний предоставляющих платный хостинга следовательно:

- Можно выбрать ту фирму, которая бы полностью удовлетворяла Ваши запросы.

- Если Вы не довольны чем-либо, можно с легкостью покинуть одну фирму и перейти на другую.

- Вы не тратите времени на поездки. Общение происходит либо по e-mail, либо по телефону.

Недостатки:

- Самый большой это то, что Вы собственно покупаете кота в мешке. Вы сможете оценить качество предоставляемых услуг только после их оплаты и начав работать с этим провайдером.

- Необходимы денежные вложения для размещения и поддержки сайта и доменного имени.

- Чтобы не попасть впросак следует некоторое время понаблюдать за сайтом, его скоростью и стабильностью работы.

- Общаться с фирмой придется только по межгороду, icq или e-mail, а иногда только по e-mail и в этом случае можно потерять довольно много времени на ожидание ответа.

- Очень сложно или совсем невозможно приехать лично и на месте обговорить некоторые вопросы.

- Можно оказаться просто обманутым и просто потерять свои деньги.

Размещение сайта у себя на компьютере

Такая возможность имеет больше теоретическое значение, чем практическую реализацию.

Размещение сайта на бесплатном хосте

Если Вы хотите самостоятельно разместить уже имеющийся у Вас сайт, данный способ обладает следующими *достоинствами*:

- Абсолютно никаких денежных затрат.

- Очень быстрые сроки получения места для размещения сайта. В 90% случаев нет необходимости вести переписку с хост-провайдером для регистрации. Регистрация сайтов производится в автоматическом режиме.

- Более широкий выбор доменных имен.

Недостатки:

- Вы получаете доменное имя третьего уровня (типа http://vashsayt.domen.ru), которое невозможно будет сохранить при переходе на другой хост.

- Не все провайдеры, предоставляющие бесплатный хостинг имеют поддержку PHP, хотя в последнее время можно найти компании, которые на бесплатном хостинге включают поддержку языка PHP.

- Практически на всех бесплатных хостах имеются ограничения на получение контента с других сайтов и отсутствие поддержки баз данных MySQL.

Аренда места на сервере компании

Размещения web-страниц в Интернете — аренда места на сервере компании, специализирующейся на предоставлении подобных услуг.

Размещение на сервере сайтов других компаний и частных лиц называют хостингом. Такие компании предоставляют своим клиентам некоторый объем на жестком диске для размещения сайта, а также набор услуг, в который обычно входит доступ к файлам клиента по протоколу FTP (это позволяет легко копировать файлы на сервер, изменять их или удалять), обслуживание электронной почты, обработка команд программ, входящих в состав сайта и т. д.

Вы можете подключаться к серверу компании через Интернет и осуществлять управление своим сайтом.

Одним из важных критериев выбора хостинга является используемая ОС, поскольку от этого зависит ПО, которое будет поддерживать функциональность тех или иных сервисов. Важным аспектом описания хостинга является наличие тех или иных служб и возможностей: (поддержка CGI: Perl, PHP, Python, ASP, Ruby, JSP;поддержка .htaccess/.htpasswd (для Apache);поддержка баз данных).

Хостинг как услугу сравнивают, описывают и оценивают по количественным ограничениям:

• размер дискового пространства под файлы пользователя

• количество месячного трафика

• количество сайтов, которые можно разместить в рамках одной учетной записи

• количество FTP пользователей

• количество E-Mail ящиков и объём дискового пространства, предназначенного для почты

• количество баз данных и размер дискового пространства под базы данных

• количество одновременных процессов на пользователя

• количество ОЗУ, и максимальное время исполнения, выделяемое каждому процессу пользователя

• качественным ограничениям:

• свободные ресурсы CPU, оперативной памяти, которые влияют на быстродействие сервера

• пропускная способность каналов, которая влияет на загрузку информации.

• удаленность оборудования хостера от целевой аудитории сайта, которая влияет на загрузку информации.

Некоторые платные хостинговые компании предоставляют бесплатный тест на определённый период, по истечении которого пользователь должен определиться подходит ли для него выбранная хостинговая компания, и имеет ли смысл оплачивать большие периоды.

Помимо платных хостеров существуют также и бесплатные хостинг компании, поддерживающие большинство описанных веб-технологий.

Платный хостинг

Как правило, такие услуги предоставляются за определенную плату. Компании, предоставляющие вам хостинг (хостинг-провайдеры), обычно заводят у себя специальный счет, на который вы вносите оплату за поддержку сайта.

С этого счета снимается абонентская плата. Кроме того, некоторые провайдеры могут взимать плату за дополнительные услуги, например, за превышение установленного обмена данными между вашим сайтом и компьютерами пользователей.

Преимущества платного хостинга:

1. Высокая скорость

2.Низкая стоимость

Каждая фирма, предоставляющая хостинг, пытается создать максимально выгодные и удобный условия для клиентов(фирм много). Именно поэтому сравнительная стоимость хостинга не велика

3. Постоянная техническая поддержка

Приобретая платный хостинг, Вы тем самым получаете гарантию на получение постоянных технических услуг, связанный с Вашим хостингом.

4.Доменное имя второго уровня

Когда Вы пользуетесь платным хостингом, Вы получаете доменное имя второго уровня. Тем самым Ваш компьютер получает свой определенный идентификатор, по которому к нему можно обратиться. Преимущества доменного имя второго уровня заключается в том, что другие пользователи могут идентифицировать Вас по этому имени, это очень удобно при общении и обеспечении информационных связей между компьютерами.

5.Технические предупреждения

Эта услуга также предоставляется провайдером, который обеспечивает хостинг сайта.В нужный момент Вам могут сообщить о возможных неполадках в сети, о сбоях, о проведении технических работ и т.д. Это позволит заранее знать о том, что делать..

6.Огромные размеры дискового пространства

Наверное, одно из главных преимуществ и отличий платного хостинга от бесплатного заключается в том, что первый предоставляет Вам огромные ресурсы дискового пространства, что позволяет Вам создавать и размещать сайты любой сложности, с любой структурой, который может требовать большой объем дискового места.

Недостатки платного хостинга:

Наверное, единственным недостатком платного хостинга является то, что пропускная способность зависит от количества сайтов, размещенных на сервере хост-провайдера. Нередко бывает так, что сервер располагает большим количеством сайтов, поэтому не хватает времени для работы с каждым, что значительно снижает общую пропускную способность.

Если Вы работаете с большими объемами информации, если Ваш сайт также будет требовать довольно больших информационных ресурсов и дискового пространства, если для Вас и Вашего Интернет-проекта в первую очередь важна скорость доступа, качество и надежность работы сайта, то услуги хост-провайдером с предоставлением Вам платного хостинга это лучшее решение для Вас.

Бесплатный хостинг

В некоторых случаях размещение страниц на сервере производится бесплатно. Существуют компании, которые предоставляют всем желающим место на своих серверах, не требуя при этом оплаты. Чаще всего такие серверы используются для размещения домашних страничек. Предоставляемого такими компаниями объема вполне хватает для размещения небольшого фотоархива и биографии, но не более того. Часто встречаются и так называемые «гостевые книги» — страницы, на которых пользователь может не только ознакомиться с текстом, но и оставить запись от своего имени.

На бесплатном сервере можно разместить и сайт фирмы, но при этом появляются некоторые проблемы, *недостатки*:

- во-первых, набор предоставляемых услуг и места на диске, как правило, сильно ограничен.

- во-вторых, ваши клиенты могут подумать, что вы не желаете более серьезно заняться организацией своего представительства во Всемирной паутине.

- Еще одна проблема, связанная с использованием бесплатного хостинга, состоит в том, что компания, предоставляющая место под сайт, надеется на получение от этого некоторого дохода.

- Скорее всего, вас обяжут в обмен на гостеприимство разместить на вашей страничке рекламный баннер.

- Иногда баннеры внедряются в страницу «насильно» — код, выводящий баннер на экран, автоматически дописывается к загружаемой с сервера странице.

- Такая реклама может испортить внешний вид вашего Интернет-представительства.

Выбор провайдера

Выбирать хостинг-провайдера следует, ориентируясь на спектр предоставляемых услуг и цену размещения сайта. Чаще всего цена выражается в виде суммы, которую вам придется уплачивать за месяц (несколько месяцев, год и т. д.) использования ресурсов сервера.

В большинстве случаев провайдеры предоставляют возможность выбора условий работы (тарифа).

Если вы остановили свой выбор на бесплатном провайдере, то обратите внимание на объем места на диске сервера, который вы получите в свое распоряжение.

Выбирая себе провайдера, вы выбираете еще и имя будущего сайта.

От того, насколько оно лаконично и удобно для запоминания, во многом зависит коммерческий успех будущего сайта.

- Чем проще потенциальному клиенту запомнить и правильно набрать на клавиатуре имя, тем больше вероятность того, что он посетит ваш сайт.

- Наилучший вариант для имени сайта — название самой фирмы. На сайт с таким именем легко попасть, даже не зная о нем заранее.

- При этом следует избегать имен, которые могут записываться неоднозначно.

- Русские буквы Ё, И, Ц, Щ, Ь, Ы, Ъ, Э, Ю, Я могут быть записаны допустимыми в адресах доменов латинскими буквами (транслитерированы) неоднозначно.

- Также следует избегать использования в адресах символов «-» и «_».

Платные хостинг-провайдеры обычно дают вам возможность зарегистрировать домен второго уровня (например, www.alpha.com). За регистрацию и сохранение за вами этого имени придется заплатить организации, уполномоченной выдавать имена сайтов в соответствующей зоне (с данным расширением).

Иногда домен регистрируется как домен третьего уровня, но домен второго уровня считают отдельной зоной (например, www.alpna.spb.ru). Регистрация такого имени часто производится бесплатно.

При использовании бесплатного хостинга вам, скорее всего, придется довольствоваться доменом третьего уровня или сайтом, размещенном в подкаталоге домена провайдера.

Такие имена часто бывают слишком длинными и неудобными не только для пользователей, но и для хозяев.

Выбрав провайдера и имя сайта, вам остается только заключить с ним договор на обслуживание и приступить к строительству вашего представительства во Всемирной Паутине.

Процесс регистрации сайта на бесплатном хостинге

В настоящее время существует множество серверов, предоставляющих услуги бесплатного хостинга. Вот адреса некоторых из них: http://www.narod.ru; http://www.chat.ru; http://www.holm.ru.

Процесс регистрации

Прежде всего, откроем в браузере главную страницу выбранного нами сервера (пусть это будет narod.ru).

Личные данные

В процессе регистрации вам придется предоставить системе некоторые данные о себе.

Некоторые из них, такие как имя пользователя и пароль, являются строго обязательными и необходимыми для дальнейшего использования услуг сервера.

Другие данные, например, имя и фамилия, могут использоваться для подписи отправляемых вами писем или для обращения к вам на служебных страницах сайта.

Все эти сведения вводятся в систему при помощи расположенных на страницах регистрации форм (наборов элементов управления).

Доступ к органам управления вашего сайта

После того, как вы закончите регистрацию, вам, разумеется, захочется получить доступ к органам управления вашего сайта.

В большинстве систем это можно сделать при помощи ссылки (например, Создайте свой сайт), для зарегистрированных пользователей или введя имя пользователя и пароль в поля, расположенные прямо на заглавной странице сайта провайдера.

Социальные сети

Социальная Сеть - это социальная структура, состоящая из узлов (примерами узлов могут быть отдельные люди, группы людей или сообщества), связанных между собой одним или несколькими способами посредством социальных взаимоотношений.

Моделирование

- Социальную сеть, как и любую сеть, можно математически моделировать графом, в котором вершины представляют объекты сети, а рёбра — взаимоотношения.

- В зависимости от рода связей и их значимых аспектов для данного исследования, они могут быть ненаправленными или направленными.

- Ненаправленные отношения вроде регулярного личного общения, товарищества, соседства.

- Односторонне направленные отношения — это подчинение, обязанность.

- Двусторонними, то есть взаимными, могут быть отношения вроде известности, готовности помочь и т.д.

- Большинство видов социальных связей можно оценить количественно; социальным объектам тоже можно присвоить количественные характеристики.

Отображение

При отображении модели социальной сети целесообразным может быть:

¨ размещение узлов сети в двух измерениях соответственно их свойствам вроде места жительства,

¨ пространственное упорядочение объектов в одном измерении соответственно некоторому их количественному свойству, такому как возраст, положение в организационной иерархии или иная мера социального статуса,

¨ использование общих для всех сетевых диаграмм методов для отображения количественных и качественных свойств социальных объектов и отношений.

Азбука социальных сетей

Социальная стратификация - это существование структурированного неравенства между социальными группами в обществе, касающегося доступа к материальным или иным благам.

Социальные связи-это совокупность осознанных или неосознанных, необходимых и случайных, устойчивых и спонтанных зависимостей одних социальных субъектов и объектов от других, их воздействия друг на друга, это комплекс факторов, обеспечивающих совместную деятельность индивидов в социальных общностях, объединяя их в функциональное целое, способное к устойчивости и развитию.

Социальная структура

- внутреннее устройство общества или социальной группы;

- упорядоченная совокупность взаимосвязанных и взаимодействующих социальных групп, социальных институтов и отношений между ними.

Анализ социальных сетей

Это направление структурного подхода, основными целями которого являются исследование взаимодействий между социальными объектами и выявление условий возникновения этих взаимодействий.

Сеть социальных взаимодействий состоит из конечной совокупности социальных акторов[1] и набора связей между ними.

В качестве социальных акторов могут выступать индивиды, социальные группы, организации, события, города, страны. Под свя-

[1] действующий субъект (индивидуальный или коллективный); индивид, социальная группа, организация, институт, общность людей, совершающих действия, направленные на других, например, государство является главным политическим актором на поле политики и ведущим социальным актором в обществе.

зями понимаются не только коммуникационные связи между акторами, но и связи по обмену различными ресурсами и деятельностью, включая конфликтные отношения.

Подходы в анализе социальных сетей

1. Структурный подход акцентирует внимание на геометрической форме сети и интенсивности взаимодействий (весе ребер).

2. Ресурсный подход рассматривает возможности акторов по привлечению индивидуальных и сетевых ресурсов для достижения определенных целей и дифференцирует акторов, находящихся в идентичных структурных позициях социальной сети, по их ресурсам. В качестве индивидуальных ресурсов могут выступать знания, престиж, богатство, раса, пол. Под сетевыми ресурсами понимаются влияние, статус, информация, капитал.

3. Нормативное направление изучает уровень доверия между акторами, а также нормы, правила и санкции, которые влияют на поведение акторов в социальной сети и процессы их взаимодействий.

4. Динамический подход акцентирует внимание на изменениях в сетевой структуре с течением времени.

Методы анализа социальных сетей

§ методы теории графов, в частности, направленные графы и представляющие их матрицы, применяемые для изучения структурных взаимосвязей актора;

§ методы нахождения локальных свойств субъектов, например, центральности, престижа, положения, принадлежности к некоторым подгруппам;

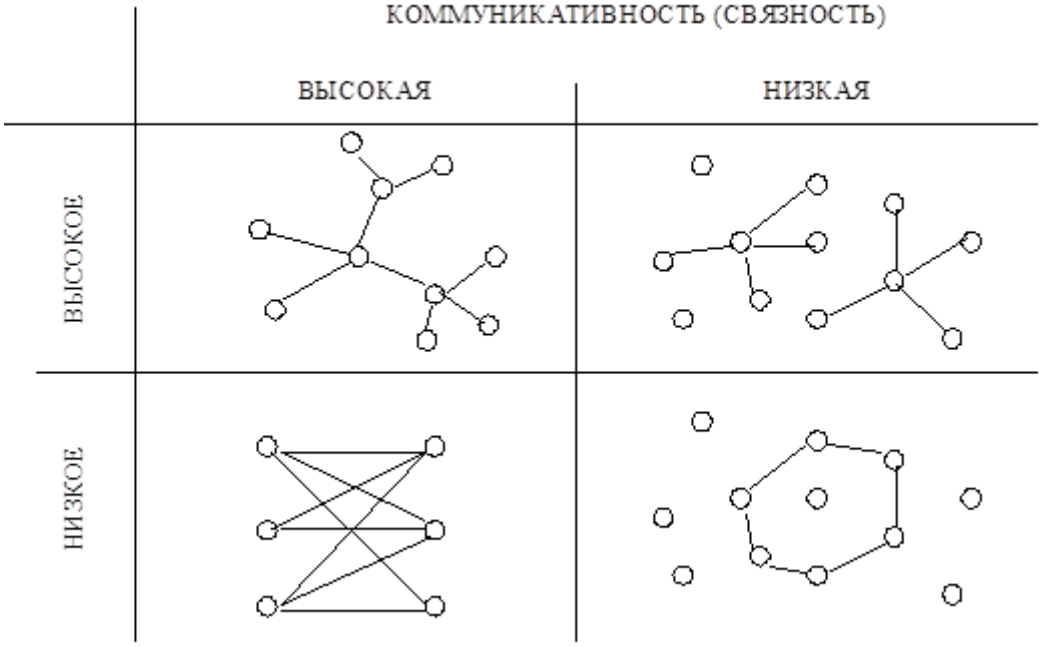
§ методы определения эквивалентности акторов, включая их структурную эквивалентность;

§ блоковые модели и ролевые алгебры;

§ вероятностные модели;

§ корреспондентский анализ и топологические метод.

Структуры и типы сетей



Доминирование — это отклонение от равномерного распределения связей между «центрами» и «кликами».

В системе, характеризующейся высокой степенью доминирования, большинство связей будут соединять «центры» и «клики».

Высокая доминантность и высокая коммуникативность присущи «спутниковой» структуре, где ресурсы перемещаются от центра к периферии.

Системы управления контентом (CMS)

CMS - это система управления контентом сайта, либо иначе ее еще называют «движок сайта». Позволяет просто, быстро и комфортно не только эксплуатировать сайт и обновлять информацию, но и совершенствовать его функционал**.**

Бесплатные CMS

Бесплатные CMS - системы управления контентом, которые распространяются бесплтно. Вот небольшой список популярных бесплатных CMS.

Drupal - написана на языке PHP, поддерживает MySQL, является модульной системой. **Недостаток данного движка**– не совсем понятный интерфейс и повышенная нагрузка на сервер.

Joomla - cms, написанная на PHP и JavaScript. Также, как и Друпал распространяется свободно под лицензией GNU GPL.

Недостаток -Слабая безопасность от взлома.

Достоинства :

§ Огромный выбор различных плагинов, модулей, компонентов, расширений и платных или бесплатных шаблонов;

§ Данная CMS постоянно обновляется, а также к ней чуть ли не каждый день появляются новые дополнительные возможности в лице свежих плагинов, расширений и т. д;

§ Большая часть дополнительных сприптов и софта для Joomla идут на русском языке

WordPress- имеет открытый исходный код, написана на PHP, поддердивает MySQL. Используется преимущественно для блогов, однако нередко ставят на новостные сайты и интернет-магазины. С помощью различных плагинов можно создавать проекты любой сложности.

Достоинства:

1. Удобный неперегруженный излишествами интерфейс административной части.

2. Просто колоссальное количество тем оформления.

3.Множество плагинов доступных в официальном репозитории, позволяющих расширять-наращивать имеющийся функционал.

4. Относительная неприхотливость к серверу. Кушает крайне мало ресурсов по сравнению скажем с Drupal.

Недостатки:

Для получения набора возможностей, вам придется поустанавливать вагоны сторонних плагинов. Это однозначно приведет к замедлению работы.

Уйма незакрытых узвимостей, недоработок.

Коммерческие CMS, примеры:

NetCat

UMI.CMS

HostCMS

AMIRO.CMS

Коммерческие системы управления контентом, как правило, имеют закрытый исходный код, но недостаток с лихвой компенсируется более высокой степенью защищенности. Коммерческие CMS регулярно обновляются и улучшаются. При этом принимаются во внимание пожелания пользователей. Это, естественно, положительно

сказывается как на уровне безопасности, так и на качестве функционала. Помимо этого, создатели традиционно ориентируют свои продукты на юзеров, не особо искушенных в программировании. Поэтому документация к CMS содержит описания мельчайших деталей по установке и настройке ПО.

2.3.4 Регистрация Интернет-ресурса в каталогах и поисковых системах

Каталог – это разбитые по тематикам ссылки на различные ресурсы Интернета. Каталоги, как правило, создаются вручную. Реже они собираются роботами. Изначально каталоги создавались как частная подборка наиболее интересных страниц и ресурсов. К сожалению, в настоящее время таких каталогов практически нет. Принцип организации каталогов универсален, но условно их можно разделить на:

1. каталоги поисковых систем (Яндекс.Каталог и .др);
2. каталоги обычных пользователей.

В настоящее время каталоги используются оптимизаторами как ресурс, который может предоставлять большое количество ссылок, при этом, не расцениваясь как запрещенный поисковый спам.

По негласному правилу, регистрация в каталогах добавляет сайту «веса». Модераторы каталога вручную проверяют данные, представленные владельцами. Если сайт не соответствует заявленной тематике или тематика обширна, то сайт просто не регистрируют. Если же сайт прошел проверку и внесен в каталог – это неплохо повлияет на общие позиции сайта.

Рассмотрев эти вопросы перейдем к конкретным каталогам известных **информационно-поисковых систем**.

Каталог Google

На главной странице **www.google.ru** (или **www.google.com**) есть ссылка на каталог. Так вот, этот каталог на самом деле принадлежит домену DMOZ.com. Насколько известно, компании Google и DMOZ заключили соглашение для создания самого большого в Интернете каталога **Open Directory Project**. Этот каталог пополняется изо дня в день вручную многими людьми. Каталог является модерируемым, то есть он действительно создается для облегчения поиска по определенной тематике. Как и все продукты, которые выпускает компания Google, каталог переведен на многие языки.

Регистрация в каталоге бесплатная, но вам придется соблюсти некоторые условия.

Рекомендации по регистрации в каталоге ODP:

1. Для начала вам необходимо выбрать точно подходящую рубрику каталога, и, только после этого (прямо находясь в данной рубрике), добавлять свой сайт в каталог (ссылка на добавление в самом низу страницы).

2. Обязательное условие при регистрации в этом каталоге: Вам необходимо уложиться в 25-30 слов при описании сайта. Также от вас потребуется имя домена и заголовок сайта.

Кстати, если кто заметил, Google создает свой каталог. В настоящее время он имеет статус "beta" и тоже является бесплатным. Основным преимуществом каталога Google является то, что он разбивает сайты в группах, на тематики.

Но в этот каталог Google невозможно самостоятельно добавить сайт. В него включаются сайты, которые уже входят в какие-либо некоммерческие каталоги (если Ваш сайт в каталоге Яндекс, в каталог Гугла его, скорее всего, не включат). Такая система, видимо, призвана сделать этот каталог максимально релевантным и удобным.

Так что данный каталог теоретически может помочь только для скорейшей индексации вновь появившегося сайта, то есть, вероятно, Google в какой-то мере синхронизирует свою базу доменов с базой каталога.

Каталог Яндекс

Разобравшись с каталогом Google, перейдем к каталогу Яндекс.

Здесь мы видим ту же ситуацию, что и с каталогом Google, лишь тем отличием, что это каталог ресурсов Рунета (в большей степени). Это также модерируемый каталог, он также разбит по тематикам. У каталога Яндекса есть небольшие преимущества для пользователя поисковой системы. Во-первых, возможен поиск сайтов, которые физически расположены в Вашем городе. Во-вторых, сама компания Яндекс заинтересована в поддержке актуальности и достоверности своего каталога – ведь это один из наиболее авторитетных сервисов компании, заслуженно пользующийся доверием пользователей.

Для продвижения каталог Яндекс, так же как и Google, будет полезен при появлении нового сайта в Сети. Между тем, существует мнение, что добавление сайта в каталог Яндекс придает сайту маленький вес при ранжировании документов.

Наверное, главным отличием каталога Яндекс от Google является платная регистрация.

Подытоживая все вышесказанное, стоит отметить, что каталоги поисковых машин используются ими в основном для повышения релевантности информации о сайтах, разделяя их по тематикам. Это действительно удобный инструмент. В тоже время, роль регистрации сайта в каталогах при поисковом продвижении хоть и положительна, но не велика.

Создав и разместив в Сети сайт, Вам придется задуматься о рекламе и его продвижении. Типичная ошибка многих новичков заключается в том, что после регистрации адреса сайта в поисковых и рейтинговых системах вся деятельность в этом направлении прекращается.

Продвижение сайта - наиважнейший процесс для любого сайта в Интернете. Это бесконечный процесс, который нельзя останавливать ни на день. Интернет - это огромное количество информации. Каждый день появляются новые сайты, которые в плане раскрученности становятся Вашими конкурентами. Если Вы просто разместили свой сайт в Интернете и решили, что этого достаточно, то Ваш сайт очень быстро окажется на самом дне Интернета, в самой глуши, где его никто никогда не найдет. Чтобы о Вашем сайте узнали, Вам нужно постоянно тянуть свой сайт на свет.

Платные сервисы по продвижению сайтов. В Интернете сейчас можно встретить очень много сайтов, предлагающих платные услуги - "раскрутку" и продвижение сайтов. Сайты, предлагающие услуги продвижения, поисковой оптимизации сайта и т.д., растут, как грибы. Настоящее продвижение сайта и вывод его на первые позиции стоит довольно дорого, в большинстве случаев в несколько раз дороже, чем создание самого сайта. Если Вы решили воспользоваться данными услугами по продвижению Вашего сайта, лучше это делать на серьёзных, проверенных сервисах (сайтах), существующих не один год.

Прежде, чем воспользоваться услугой "раскрутки" и продвижения своего сайта, проверьте сайт, который предлагает данные

услуги. Проверка сайта покажет Вам серьёзность проекта (сайта) и даст Вам хоть какие-то гарантии, что Вас не обманут. Бывает так, что сайт, предлагающий данные услуги, при поиске в поисковике будет сам находиться где-нибудь на 51-й странице или ещё дальше. Сайт, предлагающий данные услуги, в первую очередь сам должен иметь высокие показатели. Может получиться, что после такого продвижения Вашего сайта ("раскрутки"), через какое-то время поисковики Ваш сайт вообще запретят к индексации, ИЦ обнулят, а восстановить это будет очень и очень сложно. Да и деньги Вам никто не вернёт.

Продвижение сайта и вывод его на первые позиции в поисковых системах не может быть дешевым, эта услуга всегда стоит дороже, чем изготовление самого сайта в Веб-студии. Продвижение сайта - это целый комплекс услуг, который требует очень много времени и затрат. Результаты можно будет увидеть не ранее, чем через 2-3 месяца. Количество сайтов по некоторым популярным темам доходит до нескольких миллионов. Все владельцы хотят, чтобы их сайт был на первых позициях, ежемесячно тратя немалые деньги на продвижение, рекламу и т.д. своего сайта, но все сайты всё равно не могут быть первыми, это просто невозможно.

Продвижение в поисковых системах.

Поисковые системы - это самый удобный и самый распространенный способ поиска информации в Интернете. Именно регистрация сайта в поисковых системах даёт до 80% трафика на Ваш сайт. Поэтому после создания и проверки сайта первым делом зарегистрируйте его в поисковых системах. Самые популярные в России - **RAMBLER.RU, YANDEX.RU, GOOGLE.COM, MAIL.RU** и др. Англоязычное население Интернета использует Yahoo.com, Google.com и Altavista.com. В перечисленных выше поисковиках, и вообще во всех наиболее популярных системах, рекомендую регистрироваться вручную, несмотря на то, что существует множество сайтов и программ, позволяющих автоматически зарегистрироваться в десятках и сотнях поисковых систем. Зарегистрировавшись вручную в наиболее популярных поисковиках и каталогах, не помешает еще и зарегистрировать свой сайт при помощи специальных сайтов и служб автоматической регистрации, через данные службы раскручивать сайт намного легче, чем вручную. Проблема поисковых систем состоит в том, что в их базах данных содержатся

сотни тысяч документов. Поэтому по любому запросу поисковая машина выдаст пользователю огромное количество ресурсов (сайтов) по данной теме. Если учесть, что на каждой странице размещается в среднем 10 ссылок, шансов, что кто-нибудь просмотрит, например, тридцатую страницу и увидит ссылку номер 301, практически нет. По статистике около 10% пользователей просматривают результаты своего запроса дальше третьей страницы. Таким образом, Ваша задача - добиться того, чтобы Ваш ресурс показывался одним из первых. Повышенная значимость документа определяется ссылками извне на сайт, содержащий этот документ. Размещая ссылку на другой сайт, его как-бы рекомендуют посетителям своего сайта. Однако, ссылки тоже бывают разные. Количество внешних ссылок на сайт не годится для представления "цитируемости". Важность ссылок с сайтов на бесплатных хостингах и с низким тИЦ ничтожна по сравнению со ссылками с известных ресурсов. "Цитируемость" и есть такой параметр важности, который выражает авторитетность страницы. Все поисковики учитывают "цитируемость", но базы у них разные - поэтому цитируемый сайт для одного поисковика может не быть цитируемым для другого. Поэтому, очень важно то, чтобы ссылки на Ваш сайт были с наиболее цитируемых ресурсов. Ниже Вы можете зарегистрировать свой сайт в самых популярных поисковых системах.

Регистрация в поисковых системах:

Для регистрации Вашего сайта в популярных поисковых системах, нажмите ниже на ссылки поисковых систем (откроется в новом окне).

Яndex Rambler Google @mail.ru

Регистрация сайта в Рейтингах, Каталогах, ТОП-ресурсах.

Регистрация сайта - это создание ссылок на Ваш сайт с других сайтов. Каталог сайтов - это список сайтов по тематикам, Рейтинг - это Каталог, в котором положение Вашего сайта будет выше, чем будет больше посещаемость сайта. Считается, что регистрация в Рейтингах и Каталогах приносит наибольшее количество посетителей после поисковых систем. Регистрация в поисковых системах обеспечивает стабильный ежедневный приток посетителей на сайт, а регистрация в рейтингах приводит к резкому повышению трафика (при условии, если Ваш сайт в числе первых). Вы, наверное, видели, размещенные на многих сайтах небольшие счетчики. Эти счетчики устанавливаются на

сайт при его регистрации в том или ином рейтинге. Рейтинги или ТОП-листы представляют собой серверы, на которых расположены адреса страничек по какой-либо теме в соответствии с их популярностью. Это - своего рода хит-парад: каждый раз, когда посетитель заходит на Ваш сайт, позиция сайта в рейтинге изменяется, он поднимается вверх, обгоняя сайты, на которые было меньше заходов. Чем выше расположен сайт в рейтинге, тем больше вероятность, что очередной посетитель этого рейтинга зайдет на него. Желательно с первых дней Вашего проекта публиковать на сайте статистику по количеству посетителей, регистрациям и т.п., что даст дополнительный плюс и повысит доверие пользователей к Вашему проекту. Вот список наиболее популярных русскоязычных рейтингов: Rambler's Top100, Top.Mail.ru, Hotlog.ru, Spylog.ru, One.ru, Liveinternet.ru и т.д.

Продвижение сайта с помощью Досок объявлений.

Плюсы:они бесплатны, их много, объявление хранится достаточно долго, Ваше объявление индексируется поисковой системой, что повышает вероятность нахождения Вашего сайта через запрос в поисковой системе, немного повышается ТИЦ (индекс цитирования) Вашего сайта при хранении на досках Вашей ссылки.

Минусы: Ваше объявление будет видно на первых страницах лишь непродолжительное время, низкая откликаемость на Ваши объявления, на многих досках правила и формы размещения отличаются друг от друга, что приводит к трате драгоценного времени на приспособление к условиям конкретной доски, охота спаммеров на Ваш e-mail и т.д.

Не рекомендуется пользоваться данным видом продвижения Вашего сайта (с помощью рассылок на доски объявлений), т.к. в итоге может быть больше минусов, чем плюсов и это может только ухудшить существующие позиции Вашего сайта.

список всех поисковых систем, куда можно добавить ваш сайт:

yandex.ru – Пожалуй самая популярная поисковая система в России. Имеется расширенный поиск.

rambler.ru – Тоже одна из популярных русскоязычных поисковых систем.

google.ru – Также одна из популярных систем в России. И по всему миру (google.com)

www.yahoo.com– Еще один из крупных поисковых систем в мире.

www.apport.ru – Довольно распространенная поисковая система в Россия, имеется два языка.

msn.com — Также довольно популярная поисковая система по новостям и по сайтам.

www.altavista.com – Без сравнения самая крупная система в мире.

astalavista.box.sk — Поисковая система по поиску бесплатного программного обеспечения.

www.bigfoot.com – Социальная сеть, поиск людей.

www.excite.com – Удобный поиск.

www.filez.com — Поиск сайтов и бесплатного программного обеспечения.

www.hotbot.com – Одна из самых быстрых систем индексация сайтов.

infoseek.go.com – Проиндексировано более 60 мил. страниц.

www.Jassan.com – Поиск по фондовым биржам, корпорациям.

www.lycos.com – Удобный поиск музыки, видео, изображений и фотографий.

www.moneysearch.com — Поиск по сайтам финансового рынка, а также по сайтам компаний связанные с бизнесом.

www.travel-finder.com – Поиск по сайтом сосредоточенные на спорте, туризме.

www.atrus.ru – Поиск, каталоги.

www.list.ru – Каталог ресурсов.

www.ru – Поиск, также имеется каталог.

www.strars.ru — Поиск по сайтам, также имеется каталог.

sel.alfainter.net – Каталог, но индексации сайтов нет.

www.anet.donetsk.ua – Каталог сайтов сети интернет.

www.a-counter.kiev.ua – Каталог сайтов.

el.visti.net — Поиск по рефератам, дипломным работам.

www.internetri.net – Украинский каталог.

www.qp.dp.ua – Каталог по сайтам.

www.meta.kharkiv.net – Еще один поисковик

www.topping.com.ua – Каталог по сайтам. Поиск в интернете, имеется рейтинг.

poshuk.dnepr.net – Сравнительно молодой каталог.

sesna.kharkiv.org – Еще один поисковик

www.susanin.com Сусанин – Довольно известный каталог.

uahoo.gu.net UA – Каталог по сайтам в интернете

www.ukrainet.lviv.ua – Украинский каталог.

www.echo.com.ua — Регистрация в системах

www.allonesearch.com — Поиск по сайтам.

www.beaucoup.com – Поиск по сайтам в интернете. Имеется шесть языков.

www.deja.com – Одна из мощных поисковых систем по поиску новостей.

Поиск может проводится по авторам, дате добавления итд…

www.dogpile.com – Имеется логический поиск по сайтам.

Вопросы для самоконтроля:

1.Какими бывают теги для разметки html-страниц

2.Укажите разновидности html-редакторов.

3.При помощи какого языка пишется разметка веб-страниц?

4.Укажите преимущества визуальных HTML-редакторов, по сравнению с редакторами исходного кода при оформлении веб-страниц?

5.Как иначе называются визуальные html-редакторы?

6.Как называется параметр тега в языке HTML?

7.При помощи каких программ можно создавать веб-страницы на языке разметки гипертекста?

8.Как называется указание к оформлению разметки веб-страницы на языке HTML?

9.Укажите преимущества редакторов исходного кода перед текстовыми редакторами при создании веб-страниц?

10Укажите недостатки визуальных HTML-редакторов.

Модуль 3. ПРОГРАММИРОВАНИЕ НА JavaScript
Тема 1. Общий обзор языка

Основные определения. Понятие объектной модели применительно к JavaScript. Размещение операторов языка JavaScript на странице. Язык ядра JavaScript. Управляющие конструкции языка JavaScript. Стандартные объекты и функции ядра JavaScript. Объекты клиента. Обработка событий

3.1.1 Основные определения. Понятие объектной модели применительно к JavaScript. Размещение операторов языка JavaScript на странице

Взаимодействие клиента и сервера в Интернете осуществляется с помощью запросов, посылаемых клиентом серверу, и ответов сервера на запрос клиента.

Его основу составляют HTTP-сообщения, подразделяемые на:

• запрос (request) клиента к серверу;

• ответ (response) сервера клиенту.

Стандартный язык разметки HTML позволяет легко создавать статичные Web–страницы. Пользователь не может менять их содержимое, не может взаимодействовать с ними. Для того чтобы сделать страницу по-настоящему интерактивной, нужен еще один язык, выполняемый в контексте браузера, - скриптовый язык.

Исследования работы приложений интернета показали, что для выполнения определенных действий пользователя нет необходимости постоянно обращаться к серверу - эти действия можно реализовать на стороне клиента, если бы он позволял каким-то образом их запрограммировать. Так появился встроенный в программу просмотра Web-страниц (браузер) язык JavaScript, который расширил возможности языка разметки HTML, предоставляя разработчику возможность встраивать в документ HTML код программы, выполняющейся на клиенте. Скриптовый язык используется для создания интерактивных страниц. Обычно он не содержит всех возможностей настоящих языков программирования, таких, например, как работа с файлами или управление графикой. Созданные с помощью скриптовых языков программы не могут выполняться самостоятельно - они работают только в контексте браузера, поддерживающего выполнения скриптовых программ.

Создаваемые на скриптовых языках программы, называются сценариями или скриптами, включаются в состав Web-страниц и распознаются, и обрабатываются браузером отдельно от остального HTML - кода.

Язык программирования JavaScript - объектно-ориентированный язык разработки встраиваемых приложений, выполняющихся как на стороне клиента, так и на стороне сервера. Веб-обозреватель, работающий на компьютере-клиенте, обеспечивает среду, в которой JavaScript имеет доступ к объектам, которые представляют собой окна, меню, диалоги, текстовые области и т. д. Кроме того, обозреватель позволяет присоединить сценарии на языке JavaScript к таким событиям, как загрузка и выгрузка страниц и графических образов, нажатие клавиш и движение мыши, выбор текста и пересылка форм. При этом программный код сценариев только реагирует на события и поэтому не нуждается в главной программе. Набор объектов, предоставляемых обозревателем, известен под названием Document Object Model (DOM). Основная идея JavaScript состоит в возможности изменения значений атрибутов HTML-контейнеров и свойств среды отображения в процессе просмотра HTML-страницы пользователем. При этом перезагрузки страницы не происходит. Основные области использования JavaScript при создании интерактивных HTML- страниц:

• Динамического создания содержимого страницы во время ее загрузки или уже после того, как она полностью загружена;

• Отображения диалоговых панелей и сообщений в статусной строке браузера;

• Оперативная проверка достоверности заполняемых пользователем полей форм HTML до передачи их на сервер;

• Создание динамических HTML-страниц совместно с каскадными таблицами стилей и объектной моделью документа (DHTML);

Основные определения

Любая программа оперирует некими данными: именем стилевого класса, размерами элемента, цветом шрифта и прочие. JavaScript может манипулировать данными, относящимися к разным типам. Тип данных описывает их возможные значения и набор применимых к ним операций. Типы данных бывают простыми и сложными. Сущность, относящаяся к простому типу данных может хранить только одно значение (это строковые, числовые и логические типы данных). Сущность

сложного типа данных может хранить сразу несколько значений. Например – массивы. Другой пример сложного типа данных – объекты. Для создания механизма управления страницами на клиентской стороне было предложено использовать объектную модель документа. Суть модели в том, что каждый HTML-контейнер - это объект, который характеризуется тройкой:

- Свойства
- Методы
- События

Существование программных объектов самих по себе не имеет никакого смысла. Они дадут преимущества при программировании тогда, когда можно организовать их взаимодействие.

1. *Объекты* – это сложные сущности, позволяющие хранить сразу несколько значений разных типов данных, они представляют собой блоки, из которых строится JavaScript. Применяются для возвращения значений и изменения состояния форм, страниц, браузера и определенных программистом переменных. Объекты можно сопоставить с существительными. Кошка, автомобиль, дом, компьютер, форма – все это существительные, они могут быть представлены как объекты.

2. *Экземпляры объекта* – сущности, хранящие реальные данные и созданные на основе этого объекта. То есть конкретный, реально существующий дом, находящийся по заданному адресу можно рассматривать, как экземпляр объекта типа дом.

3. *Свойства* – набор внутренних параметров объекта. Используются для того, чтобы различать экземпляры одного объекта – например, все экземпляры типа дом. Свойства сравнимы с прилагательными и ссылаются на уникальные для каждого экземпляра объекта особенности. Один и тот же объект может обладать многими свойствами: дом может быть большим и маленьким, синим и красным. Разные объекты могут обладать одинаковыми свойствами: дерево, так же, как и дом, может быть большим и маленьким, синим и красным. Большинство свойств объекта мы можем изменять, воздействуя на них через методы.

4. *Методы* - это действие или способ, при помощи которого мы можем изменять определенные свойства объекта, то есть управлять этими объектами, а также в некоторых случаях менять их содержимое.

5. *События* – это очень важное в программировании на JavaScript понятие. События главным образом порождаются пользователем, являются следствиями его действий. Если пользователь нажимает кнопку мыши, то происходит событие, которое называется Click. Если экранный указатель мыши движется по ссылке HTML документа, происходит событие MouseOver. Существует несколько различных событий.

6. *Оператор* - это команда, инструкция для компьютера. Встретив в программе тот или иной оператор, машина четко его выполняет.

7. Функция - это определенная последовательность операторов, то есть набор команд, последовательное выполнение которых приводит к какому-то результату. Например, выполнение кем-то заданной Вами функции (процедуры) "возьми стакан, открой кран, набери в него воды и принеси мне" приведет к результату: Вы получите стакан воды из-под крана.

8. Переменная - в языках программирования переменные используются для хранения данных определенного типа, например, параметров свойств объекта. Каждая переменная имеет свое имя (идентификатор) и хранит только одно значение, которое может меняться в ходе выполнения программы. Данные могут быть разных типов: целое число, десятичная дробь, логическая константа, текстовая строка.

Понятие объектной модели применительно к JavaScript

При загрузке HTML-страницы в браузер интерпретатор языка создает объекты со свойствами, определенными значениями тэгов страницы. Для правильного использования объектных моделей следует четко понимать, как браузер компонует страницы и, тем самым, создает иерархию объектов. При загрузке страницы просматриваются сверху вниз, тем самым последовательно происходит компоновка страницы и ее отображение в окне браузера. А это означает, что и объектная модель страницы также формируется последовательно, по мере ее обработки. Поэтому невозможно обратится из сценария, расположенного ранее какой-либо формы на странице, к элементам этой формы. Всегда следует помнить о том, что браузер последовательно сверху вниз интерпретирует содержимое HTML-страницы. Еще один аспект работы с объектами языков сценариев заключается в том, что нельзя изменить свойства объектов. Браузер обрабатывает страницу только

один раз, компонуя и отображая ее. Поэтому попытка в сценарии изменить свойство отображенного элемента страницы, обречена на провал. Только повторная загрузка страницы приведет к желаемому результату.

Размещение операторов языка JavaScript на странице

Встроить сценарий JavaScript в HTML-страницу можно несколькими способами.

1. Задание операторов языка внутри тэга <script> языка HTML. Для внедрения в HTML-страницу сценария JavaScript в спецификацию языка HTML был введен тэг-контейнер <script>...</script>, внутри которого могут располагаться операторы языка JavaScript. Обычно браузеры, не поддерживающие какие-нибудь тэги HTML, просто их игнорируют, анализируя, однако, содержимое пропускаемых тэгов с точки зрения синтаксиса языка HTML, что может приводить к ошибкам при отображении страницы. Во избежание подобной ситуации следует помещать операторы языка JavaScript в контейнер комментария <!-- ... //->, как показано ниже

```
<script (language="javascript")>
 <!--
 операторы javascript
 //-->
 </script>
```

Параметр language задает используемый язык сценариев. В случае языка JavaScript его значение задавать не обязательно, так как этот язык используется браузерами по умолчанию.

Примечание:

символы // перед закрывающим тэгом комментария --> являются оператором комментария JavaScript. Он необходим для правильной работы интерпретатора.

Документ может содержать несколько тэгов <script>, расположенных в любом месте документа. Все они последовательно обрабатываются интерпретатором JavaScript по мере отображения частей документа в окне браузера. В связи с этим ссылка на переменную, определенную в сценарии, размещенном в конце документа, может привести к генерации ошибки интерпретатора при обращении к такой переменной из сценария в начале документа.

2. Задание файла с кодом JavaScript.

Тэг <script> имеет параметр src, позволяющий связать встраиваемый сценарий с внешним файлом, содержащим программный код на языке JavaScript. В качестве значения параметра задается полный или относительный URL-адрес ресурса. Задание закрывающего тэга </script> обязательно, независимо от того, заданы или нет операторы внутри тэга. Следующий фрагмент кода связывает документ HTML с файломисточником, содержащим некоторый набор функций: <script language="JavaScript" src="http://url/file.js"> операторы javascript </script> Примечание: связываемый внешний файл не должен содержать тэгов HTML и должен иметь расширение .js.

3. Использование выражений JavaScript в качестве значений параметров тэгов HTML. Переменные и выражения JavaScript можно использовать в качестве значений параметров тэгов HTML. Например:

```
<a href="javascript: window.open('name.htm', '_self')">
<img src="images/ch1.png" border="0">
</a>
```

4. Определение обработчика событий в тэге HTML.

3.1.2. Язык ядра JavaScript. Управляющие конструкции языка JavaScript.

Синтаксис языка

Язык JavaScript чувствителен к регистру.

Приложение JavaScript представляет собой набор операторов языка (команд), последовательно обрабатываемых встроенным в браузер интерпретатором. Каждый оператор можно располагать в отдельной строке. В этом случае разделитель ‘;’, отделяющий один оператор от другого, не обязателен. Его используют только в случае задания нескольких операторов на одной строке. Любой оператор можно расположить в нескольких строках без всякого символа продолжения. Например, следующие два вызова функции alert эквивалентны:

```
…
alert("Подсказка");
alert(
"Подсказка"
  );
…
```

Нельзя перемещать на другую строку единый строковый литерал - он должен располагаться полностью на одной строке текста программы или разбит на два строковых литерала, соединенных операцией конкатенации ‘+’:

```
…
alert("Подсказка"); // правильно
alert("Под
сказка"); // не правильно
alert("Под" +
"сказка");    // правильно (но браузер выведет текст одной строкой!)
…
```

Пробельные символы в тексте программы являются незначащими, если только они не используются в строковых литералах.

В JavaScript строковые литералы можно задавать двумя равноправными способами - последовательность символов, заключенная в двойные или одинарные кавычки:

```
"Анна"
'Анна'
```

В строковых литералах можно использовать ESC-последовательности, которые начинаются с символа обратной наклонной черты, за которой следует обычный символ. Некоторые подобные комбинации трактуются как один специальный символ. Таблица 13.

Esc-последовательности	Символ
\b	Возврат на один символ
\f	Переход на новую страницу
\n	Переход на новую строку
\r	Возврат каретки
\t	Горизонтальная табуляция Ctrl-I
\'	Апостроф
\"	Двойные кавычки
\\	Обратная наклонная черта

ESC-последовательности форматирования используются при отображении информации в диалоговых окнах, отображаемых функциями alert(), prompt() и confirm(), а также, если методом document.write() записывается содержимое элемента pre.

Комментарии в программе JavaScript двух видов - однострочные и многострочные:

// комментарий, расположенный на одной строке.

/*

комментарий, расположенный на нескольких строках.

*/

Ссылка на объект осуществляется по имени, заданному параметром name тэга HTML, с использованием точечной нотации. Например, пусть в документе задана форма с двумя полями ввода:

<form name="form1">

Фамилия: <input type = "text" name = "student" size = 20>

Курс: <input type = "text" name = "course" size = 2>

</form>

Для получения фамилии студента, введенного в первом поле ввода, в программе JavaScript следует использовать ссылку document.form.student.value, а чтобы определить курс, на котором обучается студент, необходимо использовать ссылку document.form. course.value.

Переменные и литералы в JavaScript

В JavaScript все переменные вводятся с помощью одного ключевого слова var. Синтаксическая конструкция для ввода в программе новой переменной с именем name1 выглядит следующим образом:

var name1;

Объявленная таким образом переменная name1 имеет значение 'undefined' до тех пор, пока ей не будет присвоено какое-либо другое значение, которое можно присвоить и при ее объявлении:

var name1 = 5;

var name1 = "новая строковая переменная";

JavaScript поддерживает четыре простых типа данных:

• Целый

• Вещественный

• Строковый

• Логический (булевый)

Для присваивания переменным значений основных типов применяются литералы – буквальные значения данных соответствующих типов. Выражения JavaScript

Выражение – комбинация переменных, литералов и операторов, в результате вычисления которой получается одно единственное значение. Переменные в выражениях должны быть инициализированы.

1. Присваивание Оператор присваивания (=) рассматривается как выражение присваивания, которое вычисляется равным выражению правой части, и в то же время он присваивает вычисленное значение выражения переменной заданной в левой части:

var name2=10;

2. Арифметическое выражение Вычисляемым значением арифметического выражения является число. Создаются с помощью арифметических операторов.

Таблица 14.

Оператор	Действие
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления целых чисел
++	Увеличение значения на единицу
--	Уменьшение значения на единицу

3. Логическое выражение Вычисляемым значением логического выражения может быть true или false. Для создания используются операторы сравнения или логические операторы, применяемые к переменным любого типа.

Таблица 15.

Операторы сравнения	Значение	Логические Операторы	Значение
= =	Равно	&&	логическое И
! =	Не равно	||	логическое ИЛИ
>=	Больше или равно	!	логическое НЕ

<=	Меньше или равно		
>	Строго больше		
<	Строго меньше		

4. Строковые выражения Вычисляемым значением строкового выражения является число. В JavaScript существует только один строковый оператор – оператор конкатенации (сложения) строк:

string1 = “Моя ” + “строка”

Операторы JavaScript

Операторы служат для управления потоком команд в JavaScript. Блоки операторов должны быть заключены в фигурные скобки.

1. Операторы выбора

• *условный оператор if*

Эта управляющая структура используется, когда необходимо выполнить некий программный код в зависимости от определенных условий. Также предусмотрена конструкция if-else (если-тогда-иначе).

```
if (условие_1)
{
оператор_1;    // эти операторы выполняются, если условие_1 верно
оператор_2;
}
else
{
оператор_3;   // эти операторы выполняются, если условие_1 ложно
оператор_4;
}
```

Условие для проверки (вопрос компьютеру) записывается сразу после слова if в круглых скобках. После этого в фигурных скобках пишется то, что будет предприниматься в случае выполнения условия. Далее else и снова в фигурных скобках то, что выполнится в случае, если условие не сработает. Количество различных действий между фигурными скобками неограниченно, фактически можно выполнить две различные программы. При сравнении можно использовать логические выражения. Например:

```
<script language="JavaScript">
```

```
var x = 5;
var y = 10;
if (x>y) {
alert('x - максимальное число')
}
else
{
alert('y - максимальное число')
}
  </script>
```

• *оператор выбора switch*

Это фактически несколько условных операторов, объединенных в одном. В данном операторе вычисляется одно выражение и сравнивается со значениями, заданными в блоках case. В случае совпадения выполняются операторы соответствующего блока case.

```
switch (выражение) {
case значение1:
       оператор_1;
      break;
case значение2
:      оператор_2;
      break;
…..
default:
     оператор;
}
```

Если значение выражения не равняется ни одному из значений, заданных в блоках case, то вычисляется группа операторов блока default, если этот блок задан, иначе происходит выход из оператора switch. Необязательный оператор break, задавемый в блоках case, выполняет безусловный выход их оператора switch.

2. Операторы цикла

Оператор цикла повторно выполняет последовательность операторов JavaScript, определенных в его теле, пока не выполниться некоторое заданное условие.

• *цикл for (цикл со счетчиком)*

```
for (i=1; i<10; i++){
```

<тело цикла>
}

Первый параметр (i=1) определяет счетчик и указывает его начальное значение. Этот параметр называется начальным выражением, поскольку в нем задается начальное значение счетчика (начальное значение в данном случае равно единице). Это выражение инициализации выполняется самым первым и всего один раз.

Второй параметр (i<10) - это условие, которое должно быть истинным, чтобы цикл выполнялся, как только условие цикла становится ложным, работа цикла завершается. Он называется условием цикла. Проверка условия цикла осуществляется на каждом шаге; если условие истинно, то выполняется тело цикла (операторы в теле цикла). Цикл в данном случае выполнится только девять раз так как задано условие i<10.

Третий параметр (i++) - это оператор, который выполняется при каждом последовательном прохождении цикла. Он называется выражением инкремента, поскольку в нем задается приращение счетчика (приращение счетчика в данном случае равно единице). Пример автоматической прорисовки нескольких линий с помощью цикла for.

```
<script language="JavaScript" type="text/JavaScript">
for (var i=1; i<10; i++){
document.write("<hr align='center' width='100'>");
}
</script>
```

• цикл while (цикл с предусловием)

```
while (условие)
{
<тело цикла>
}
```

Пока значение условия - true (истинно), выполняется тело цикла. Тело цикла может быть представлено простым или составным оператором. Оператор while содержит в скобках все необходимые параметры условия цикла (логическое выражение). После определения всех параметров цикла вводится открывающая фигурная скобка, символизирующая начало тела цикла. Закрывающая фигурная скобка вводится в конце тела цикла. Все операторы, введенные в скобках, выполняются при каждом прохождении цикла.

```
<script language="JavaScript">
var i=1;
 while(i<=10){
 document.write('число='+i+'<br>');
  i=i+2;
}
 </script>
```

• *прерывание и перезапуск цикла* Оператор прерывания break позволяет прервать выполнение цикла и перейти к следующему за ним выражению:

```
a = 10;
i = 1;
while (a<100){
a = a * i; if (i>4)
 break; ++i;
}
```

Если значение i превысит 4, то прерывается выполнение цикла. Оператор перезапуска continue позволяет перезапустить цикл, т.е. оставить невыполненными все последующие выражения, входящие в тело цикла, и запустить выполнение цикла с самого начала.

```
a = 10;
i = 1;
while (a<100){
 ++i;
 if (i>2 && i<11) continue;
a = a * i;
}
```

Создание и вызов функций в JavaScript

В JavaScript *функцией* называется именованная часть программного кода, которая выполняется только при обращении к ней посредством указания ее имени. Функции создаются с помощью ключевого слова function. Обычно функции располагают в секции <head>. Такое расположение функций в HTML-документе гарантирует их полную загрузку до того момента, когда их можно будет вызвать из секции

<body>. После названия функции (func_name) ставятся двойные круглые скобки, программный код при этом заключается в фигурные скобки:

```
<script language="JavaScript">
function func_name()
{
программный код функции (тело функции)
}
</script>
```

Для того, чтобы вызвать функцию в нужном месте, необходимо просто указать ее имя в тексте:

```
<script language="JavaScript">
func_name();
</script>
```

Второй вариант вызова функции непосредственно в HTML теге:

Текст ссылки. Ниже приведен код страницы HTML, после загрузки которой каждые три секунды будет появляться сообщение, генерируемое вызовом функции myMessage():

```
<script>
function myMessage()
{
alert("My Message")
}
</script>
<body onload='setTimeout ("myMessage()",3000)'>
<p>Каждые три секунды будет появляться сообщение</p>
</body>
```

Метод setTimeout() запускает выполнение кода JavaScript, задаваемого первым строковым параметром, через определенный промежуток времени после выполнения метода. Интервал задается в миллисекундах (1000 соответствует 1 секунде).

3.1.3 Стандартные объекты и функции ядра JavaScript. Объекты клиента. Обработка событий.

Объект Array

Массив - упорядоченный набор однородных данных, к элементам которого можно обращаться по имени и индексу. Язык JavaScript

не имеет встроенного типа данных для создания массивов, поэтому для решения используется объект Array и его методы.

Для создания объекта Array вызывается оператор new и конструктор массива - системная функция (ее имя совпадает с именем объекта), инициализирующая элементы массива:

```
m=new Array();
```

Заполнение массива происходит позже. Например:

```
<script language="JavaScript">
//создание нового массива
 m=new Array();
//заполнение массива
m[0]=1;
m[1]=2;
m[2]=4;
m[3]=56;
</script>
```

В приведенном выше примере с помощью команды new создается массив m, а затем происходит его заполнение - каждому элементу присваивается определенное значение. m=new Array(1,2,4,56);

Вызывается команда new и сразу задаются значения всех элементов массива.

```
<script language="JavaScript">
//создание нового массива и его заполнение
m=new Array(1,2,4,56)
 </script>
```

Объявление строковых массивов проводится тем же способом, что и объявление числовых массивов.

Таблица 16

Методы объекта Array	Действие
join()	Объединяет все элементы массива в одну строку с указанием разделителя.
reverse()	Изменяет порядок элементов в массиве - первый элемент становится последним, последний – первым

sort()	Выполняет сортировку элементов массива split() Разделяет строку на составные части
concat()	Объединяет два массива в один
slice()	Выделяет часть массивы
toString()	Возвращает строку -результат конкатенации всех элементов массива. Элементы массива в строке разделены запятой.
Свойство length	Возвращает длину массива (число элементов в нем).

Пусть определены два массива:

array1 = new Array("Первый","Второй","Третий");

array2 = new Array("Один","Два","Три");

Тогда метод join() первого массива array1.join() возвратит строку: "Первый, Второй, Третий"

Метод sort() первого массива array1.sort() упорядочит элементы массива array1 (переставив их местами непосредственно в самом массиве array1) в алфавитном порядке:

array1[0] = "Второй";

array1[1] = "Первый";

array1[2] = "Третий";

Поскольку некоторые методы массива возвращают массив, то к нему можно сразу же применить какой-либо метод, продолжив "точечную" нотацию. Например, array1.concat(array2).sort() объединит два массива в один новый и отсортирует его.

Объект Date

Используется для представления дат в программах JavaScript. Время храниться в виде числа миллисекунд, прошедших от 1 января 1970 года.

Данный объект создается также, как и любой объект в JavaScript – с помощью оператора new и конструктора, в данном случае Date():

date1 = new Date(); // значением переменной date1 будет текущая дата Параметром конструктора может быть строка, в которой записана нужная дата:

date1 = new Date(“january 14, 2000, 12:00:00”);

Можно задать список параметров:

date1 = new Date(2000, 1, 14, 12, 0, 0);

Объект Math

В свойствах данного объекта хранятся основные математические константы, а его методы вычисляют основные математические функции. При обращении к данному объекту, создавать его не надо, но необходимо явно указывать его имя Math. Например:

p = Math.PI; // хранится значение числа пи.

Объект String

Можно явно создавать строковый объект, используя оператор new и конструктор:

myString = new String(“Hello!”);

Данный объект имеет единственное свойство length, хранящее длину строки, содержащейся в строковом объекте, и два типа методов: одни непосредственно влияют на содержание самой строки, вторые возвращают отформатированный HTML-вариант строки.

Стандартные функции верхнего уровня

В JavaScript существуют несколько функций, для вызова которых не надо создавать никакого объекта, она находятся вне иерархии объектов.

Функция parseFloat(parameter) анализирует значение переданного ей строкового параметра на соответствие представлению вещественного числа.

Функция parseInt(parameter, base) пытается возвратить целое число по основанию, заданному вторым параметром.

Эти функции полезны при анализе введенных пользователем данных в полях формы до их передачи на сервер.

Функции Number(object) и String(object) преобразуют объект, заданный в качестве его параметра в число или строку.

Объекты клиента

При интерпретации страницы HTML браузером создаются объекты JavaScript, свойства которых представляют значения параметров тэгов языка HTML.

Иерархия объектов.

Созданные объекты существуют в виде иерархической структуры, отражающей структуру самой HTML-страницы.

На верхнем уровне расположен объект window, представляющий собой активное окно браузера.

Далее вниз по иерархической лестнице следуют объекты frame, document, location и history и т.д.

Значения свойств объектов отражают значения соответствующих параметров тэгов страницы или установленных системных параметров. На рисунке показана структура объектов клиента (браузера).

Особняком стоит объект navigator с двумя дочерними (подчиненными) объектами. Он относится к самому браузеру, и его свойства позволяют определить характеристики программы просмотра. Каждая страница в добавление к объекту navigator обязательно имеет еще четыре объекта:

 window — объект верхнего уровня, свойства которого применяются ко всему окну, в котором отображается документ;

 document — свойства которого определяются содержимым самого документа: связи, цвет фона, формы и т. д.;

 location — свойства которого связаны с url-адресом отображаемого документа;

 history — представляет адреса ранее загружавшихся HTML-страниц.

Кроме указанных объектов страница может иметь дополнительные объекты, зависящие от ее содержимого, которые являются дочерними объектами объекта document. Если на странице расположена форма, то все ее элементы являются дочерними объектами этой формы. Для задания точного имени объекта используется точечная нотация с полным указанием всей цепочки наследования объекта. Наиболее общий объект высшего уровня находится слева в выражении, и слева направо происходит переход к более частным объектам, являющимся при этом наследниками высших в иерархии объектов. Кроме этих классов объектов пользователь может создавать и свои собственные. Но обычно большинство программ используют эту систему классов и не создают новых.

Объект navigator

Этот объект применяется для получения информации о версиях.

Синтаксис: navigator.name_properties

Методы и события, догадаться не определены для этого объекта. Да и свойства только для чтения, так как ресурс с информацией о версии недоступен для редактирования. Свойства

• appCodeName - кодовое имя браузера;

• appName - название браузера;
• appVersion - информация о версии браузера;
• userAgent - кодовое имя и версия браузера;

Ниже приведен пример использования объекта navigator.

```
<html>
 <head>
 <title> navigator </title>
 <script language="javascript">
 var firstn = window.prompt("Введите ваше имя: ","ваше имя")
 function welcome(){
    var appname=navigator.appName
var appver=navigator.appVersion
 window.alert("Привет, " + firstn + ". Вы используете " +
  appname + ". Версия " + appver + ". Спасибо за визит.");
 }
function writename(){
  document.write(firstn + ".");
}
 </script>
  </head>
 <body onLoad="welcome()">
 Добро пожаловать,
 <script language="JavaScript">
 writename();
 </script>
   </body>
 </html>
```

Объект window

Объект window создается автоматически при запуске браузера, так как для отображения документа необходимо окно. Одно из назначений объекта окна - это создание нового окна. Новое окно браузера создается с помощью метода window.open(). Метод window.open() имеет ряд дополнительных аргументов, которые позволяют задать местоположение окна, его размер и тип, а также указывают, должно ли окно иметь полосы прокрутки, полосу команд и т. п. Помимо этого

можно задавать и имя окна. В общем виде данный метод можно представить следующим образом: window.open('url', 'name', 'parameters') Рассмотрим синтаксис более подробно:

• Первый параметр метода window.open() - это url документа, загружаемого в окне. Если его не заполнить, то окно останется пустым.

• Второй параметр определяет название окна (name). Это имя может использоваться для обращения к созданному окну.

• Третий параметр представляет список необязательных опций, разделенных запятой.

С их помощью Вы определяете вид нового окна: наличие в нем панелей инструментов, строки состояния и других элементов. Приведем таблицу с описанием параметров нового окна, задаваемого третьим параметром (parameters) метода open().

Таблица 16

параметр	значение		описание
fullscreen	yes	no	указывает, показывается ли новое окно на полный экран или как обычное окно. По умолчанию показывается обычное окно
	1	0	
channelmode	yes	no	позволяет указать, отображается ли полоса каналов
	1	0	
toolbar	yes	no	позволяет указать, отображается ли полоса каналов
	1	0	
location	yes	no	позволяет указать, отображается ли полоса для ввода адреса
	1	0	
directories	yes	no	позволяет указать, отображается ли полоса кнопок для выбора каталогов
	1	0	
status	yes	no	позволяет указать, отображается ли полоса статуса
	1	0	
menubar	yes	no	позволяет указать, отображается ли полоса меню
	1	0	
scrollbars	yes	no	задает отображение горизонтальной и вертикальной полос прокрутки
	1	0	

resizable	yes 1	no 0		позволяет указать, может ли окно изменять свой размер
width	yes 1	no 0		задает ширину окна в пикселах. Минимальное значение – 100
height	yes 1	no 0		задает высоту окна в пикселах. Минимальное значение – 100
top	yes 1	no 0		задает вертикальную координату левого верхнего угла окна
left	yes 1	no 0		задает горизонтальную координату левого верхнего угла окна

Объект window использует три метода отображения сообщений:

 метод prompt() – выводит диалоговое окно с полем ввода, куда пользователь может ввести информацию

 метод alert() – выводит на экран окно - сообщение с кнопкой ОК и определенным программистом текстом

 метод confirm() – выводит диалоговое окно с кнопками ОК и Cancel.

Дает возможность пользователю продолжить или отменить предложенную операцию. Сообщение, которое вы хотите вывести на экран, набирается в кавычках внутри круглых скобок. Данный скрипт запрашивает имя посетителя и выдает приветствие с введенным именем.

```
<script language="JavaScript">
 name=window.prompt ("Введите, пожалуйста, свое имя", "Ваше имя");
 window.alert ("Вас зовут, " + name);
 </script>
```

В этом фрагменте кода метод prompt имеет следующие параметры: текст запроса и значение, заполняющее поле ввода по умолчанию; переменная name - имя переменной, куда сохраняется введенная информация (имя может быть любым).

Объект document

Объект document имеет дело прежде всего с телом HTML-страницы. Он имеет несколько дочерних объектов (коллекций): all, images,

link, anchor и form. Пользуясь объектной моделью построения документа можно, например, обратиться к любой картинке на странице через следующий синтаксис: document.images.name.src Для document не существует никаких событий. Некоторые свойства и методы перечислены в таблице, из методов наиболее употребимы write и writeln .

Таблица 17

Свойства	Назначение
bgColor	Устанавливает цвет фона текущего документа. Этот цвет может иметь шестнадцатеричное представление #rrggbb или соответствующее название. Синтаксис: document.bgColor="#e7e6d8"
fgColor	Устанавливает цвет текста документа. Аналогичен по функциям свойству bgColor referrer
referrer	Указывает url документа, на который ссылается пользователь в настоящее время. Например, если кто-то обратился по адресу: http://www.nm.org/welcome.htm с сервера http://www.someplace.com, то свойством referrer будет: http://www.someplace.com, если это свойство было в странице вышеупомянутого расположения; в противном случае оно обращается в null
location	Соответствует адресу url текущего документа

Таблица 18.

Методы	Назначение
write() writeln()	Записывает HTML-текст в текущий документ и должен вызываться, когда документ открывается для записи. Синтаксис:

	document.write('somestring'), где somestring может быть одной строкой, переменной или же несколькими связанными строками в формате HTML, которые выводятся на экран
lastModified()	Показывает дату последней модификации документа: date1 = document. lastModified
open()	Открывает документ для записи дополнительных строк в формате HTML: document.
close()	Закрывает документ, который вызывался методом document.open(): document.close().

Методы write() / writeln(). Вызов метода document.write() с указанием определенных параметров приводит к отображению текста в окне браузера. В качестве параметра при вызове метода document.write() мы указываем строку, которую хотели бы увидеть на экране.

```
<script language="JavaScript">
 document.write('Пример вывода сообщения в окно браузера')
  </script>
```

Выводимая строка может содержать и тэги языка HTML. В этом случае браузер выведет данную строку точно так же, как если бы она была размещена непосредственно в HTML документе.

```
<script language="JavaScript">
document.write('<h1><b>< i>Пример вывода сообщения в окно браузера</i></b></h1>')
</script>
```

При написании скрипта, содержащего несколько команд document.write() подряд, при выводе в браузер текст окажется на одной строке

```
<script language="JavaScript">
document.write('Строка 1') ;
 document.write('Строка 2') ;
 </script>
```

Для размещения каждого куска текста в новом абзаце можно использовать 2 способа: 1. либо тег <p>, либо тег <br>, который включается в состав выводимой строки; 2. используя метод document.writeln().

Следующие примеры приведут к одинаковому результату:

```
<script language="JavaScript">
document.write('Строка 1<br>') ;
document.write('Строка 2<br>') ;
</script>
```

и

```
<script language="JavaScript">
 document.writeln('Строка 1') ;
 document.writeln('Строка 2') ;
</script>
```

Объект location

Объект location содержит информацию о местонахождении текущего документа, т.е. его интернет-адрес. Его также можно использовать для перехода на другой документ и перезагрузки текущего документа.

Таблица 19

Свойство	Описание
hash	Имя "якоря" в интернет-адресе документа, если оно есть.
host	Имя компьютера в сети интернет вместе с номером порта, если он указан.
hostname	Имя компьютера в сети Интернет. href Полный интернет-адрес документа.
href	Полный интернет-адрес документа
pathname	Путь и имя файла, если они есть.
port	Номер порта. Если не указан, возвращает номер 80 стандартный порт, через который работает протокол http.
protocol	Идентификатор протокола. Если не указан, возвращается "http:".

. search	Строка параметров, если она есть.

Таблица 10

Метод	Описание
assign(url)	Загружает документ, адрес которого передан в качестве параметра. Поддерживаетс только IE начиная с 4.0
reload()	Перезагружает документ с Web-сервера.
replace(url)	Загружает документ, адерс которого передан в качестве параметра, и заменяет в списке истории Web-обозревателя адрес предыдущего документа адресом нового.

Пользуясь объектом location, можно загрузить другой документ на место текущего. Для этого просто необходимо присвоить значение нового интернет-адреса свойству href.

document.location.href = "http://www.---.ru";

Если вы хотите полностью заменить текущий документ, чтобы даже адрес его не появлялся в списке истории, воспользуйтесь методом replace:

document.location.replace("http://www.--.ru");

Объект form

Каждая форма в документе, определенная тегом <form>, создает объект form, порождаемый объектом document. Ссылка на этот объект осуществляется с помощью переменной, определенной в атрибуте name тега <form>. В документе может быть несколько форм, поэтому для удобства ссылок и обработки в объект document введено свойство-массив forms, в котором содержатся ссылки на все формы документа. Ссылка на первую форму задается как document.forms[0], на вторую - document.forms[1] и т.д. Вместо индекса в массиве forms можно указывать имя формы. Например, если в документе присутствует единственная форма со значением атрибута name=form1, то любой из следующих операторов JavaScript содержит ссылку на эту форму:

document.forms[0];

document.forms["form1"];

document.form1;

Последний оператор возможен в силу того, что объект document порождает объект form (как и все остальные объекты, соответствующие элементам HTML страницы) и ссылку на него можно осуществлять по обычным правилам наследования языка JavaScript.

Все элементы формы порождают соответствующие объекты, подчиненные объекту родительской формы. Таким образом, для ссылки на объект text (с параметром name = text1) формы form1 можно пользоваться любым из нижеприведенных операторов:

document.forms[0].text1;

document.forms["form1"].text1;

document.form1.text1;

Кроме имени элементы формы, имеют свойство value, значение которого определяется смыслом атрибута value элемента формы.

Например, для элементов text и textarea значением этого свойства будет строка содержимого полей ввода этих элементов; для кнопки подтверждения - надпись на кнопке и т.д. Обратиться к свойству value можно по тому же принципу, например

: document.form1.text1.value

Обработка событий

Использование языка JavaScript при обработке событий значительно расширило возможности языка HTML. Чаще всего программы создаются для обработки информации, вводимой пользователем в поля форм. Возможности управления элементами форм обеспечиваются главным образом за счет функций обработки событий, которые могут быть заданы для всех элементов формы. События делятся на несколько категорий:

• события, связанные с документами (события документа) - загрузка и выгрузка документов;

• события, связанные с гиперсвязью (события гиперсвязи) - помещение указателя мыши на гиперсвязь и активизация гиперсвязи;

• события, связанные с формой (события формы) – о щелчки мыши на кнопках; о получение и потеря фокуса ввода и изменение содержимого полей ввода, областей текста и списков; о выделение текста в полях ввода и областях текста;

События, связанные с документами, возникают при загрузке и выгрузке документа, в то время как события гиперсвязей возникают при их активизации или при помещении на них указателя мыши. Чтобы обеспечить перехват события, необходимо написать функцию-обработчик события. В качестве обработчиков событий могут быть заданы целые функции языка JavaScript или только группы из одного или нескольких операторов. В таблице перечислены имена событий и условия их возникновения: Таблица 21

Имя события	Атрибут HTML	Условие возникновения события
Blur	onBlur	Потеря фокуса ввода элементом формы
Change	onChange	Изменение содержимого поля ввода или области текста, либо выбор нового элемента списка
Click	onClick	Щелчок мыши на элементе формы или гиперсвязи Focus onFocus Получение фокуса ввода элементом формы
Load	onLoad	Завершение загрузки документа
Unloaqd	onUnload	Выгрузка текущего документа и начало загрузки нового
MouseOver	onMouseOver	Помещение указателя мыши на гиперсвязь
MouseOut	onMouseOut	Помещение указателя мыши не на гиперсвязь
Select	onSelect	Выделение текста в поле ввода или области текста

Атрибут onClick

Атрибут onClick может использоваться в следующих тегах HTML:

- . . .
- <input type="checkbox" onClick=" function()">
- <input type="radio" onClick="function()">
- <input type="reset" onClick=" function()">
- <input type="submit" onClick=" function()">
- <input type="button" onClick=" function()">

Операторы языка JavaScript, заданные в атрибуте onClick, выполняются при щелчке мыши на таких объектах как гиперсвязь, кнопка перезагрузки формы или контрольный переключатель. Для контрольных переключателей и селекторных кнопок событие Click возникает не только при выборе элемента, но и при разблокировании. Разберем пример использования атрибута onClick для кнопок, определенных тегами <input type="button"> в контейнере <form> . . . </form>: …

```
<script language="JavaScript">
function but1() {
alert("Вы нажали первую кнопку");
}
function but2() {
 alert("Вы нажали вторую кнопку");
}
 </script> …

<form>          <input type="button" value="Первая кнопка" onClick="but1()">
<input type="button" value="Вторая кнопка" onClick="but2()">
 </form>
```

… Работа с меню

Список в форме задается с помощью объекта select, обработка событий выполняется с помощью следующих параметров: onChange - вызывается при изменении выбора; onBlur - вызывается при снятии фокуса с объекта; onFocus - вызывается при перемещении фокуса на объект. Рассмотрим следующий пример:

```
 …
<script language="JavaScript">
function selectBlur()
{
 document.myForm7.myText.value="Вы нажали поле вне списка ";
}
function selectFocus()
{
 document.myForm7.myText.value="Вы нажали ту же кнопку ";
```

```
}
function selectChange()
{
document.myForm7.myText.value="Вы нажали другую кнопку ";
}
</script>
 …
<form name="myForm7">
<input type="text" name="myText" size=40 value="Город"><br>
<select name="script" multiple onBlur="selectBlur()"
onFocus="selectFocus()" onChange="selectChange()">
<option value="town1" selected>Париж
<option value="town2">Лондон
<option value="town3">Рим
 <option value="town4">Берлин
</select>
</form>
…
```

Управление логикой программного кода при помощи событий

В объектно-ориентированном программировании нет единой структуры управления работой программы. Есть независимые друг от друга объекты. Когда пользователь щелкает, например, по ссылке на экране, браузер передает событие Click объекту, тега <a>. Для события “щелчок мыши” в этом объекте предусмотрен стандартный обработчик — он загружает в окно новый документ.

Давайте попробуем “перехватить” это событие:

```
<a href=”page1.htm” onClick="alert('Хода нет?')">документ page1</a>
```

Если щелкнуть по ссылке, на экране возникнет надпись “Хода нет?”. Событие перехвачено, но, при закрытии окна alert, видим, что браузер по-прежнему грузит документ page1.htm. При помощи атрибута onClick мы установили в объекте, “отвод” на собственный обработчик. Но когда скрипт нашего обработчика выполнен, управление возвращается к стандартному обработчику, и это вызывает загрузку документа page1.htm. Отключение стандартной обработки кодируется так:

документ page1

Оператор return указывает возвращаемое функцией значение. Если ее операнд true, то документ загружается, если false, нет.

Подтверждение активизации гиперсвязи.

Аналогичный пример управления логикой программного кода при помощи событий рассмотрен и в следующем примере. Гиперссылка обычно всегда срабатывает по клику мыши, но иногда нужно, чтобы пользователь был уверен, что хочет перейти по ссылке в следующий документ. Для этого существует метод confirm(), который отображает на экране окно сообщения с кнопками "Ok" и "Cancel". Для перехвата события в теге <a href= ... > мы применим событие onClick. Рассмотрите пример подтверждения активизации гиперсвязи:

 Подтверждение активизации гиперсвязи

Определение событий формы

Объект form имеет два обработчика событий: onSubmit и onReset. В эти обработчики событий, задаваемые в пределах дескриптора <form>, добавляется группа операторов JavaScript или функция, управляющая формой.

Если вы добавите оператор (или функцию) в обработчик onSubmit, то он (или она) вызывается до отправки данных в сценарий CGI. Для того чтобы отменить отправку данных на обработку сценарием CGI, обработчик событий onSubmit должен возвратить значение false. Если же он возвращает значение true, то данные отправляются на сервер. В некоторых случаях необходимо добавить в форму кнопку reset, запускающую обработчик событий onReset.

Для формы одним из важных действий на странице является проверка правильности заполнения полей пользователем на машине клиента до пересылки их на сервер. В следующем примере разъясняется, как выполнять эту процедуру.

Рассмотрим скрипт, который будет проверять правильность заполнения формы. Необходимо проверить нет ли пустых строк и правильно ли введен e-mail:

```
<html>
    <head>
```

```
        <title>пример формы</title>
         <script language="JavaScript">
function doSend(){
var v=document.user.e.value.indexOf("@",1)
         if(document.user.f.value==""){
         alert('Вы должны заполнить поле ФИО')
        document.user.f.focus()
          }
 if(document.user.a.value==""){
alert('Вы должны заполнить поле адреса')
 document.user.a.focus()
         }
if(document.user.e.value==""){
alert('Вы должны заполнить поле e-mail')
 document.user.e.focus()
         }
if(v==-1){
alert('Адрес e-mail указан неверно')
 document.user.e.select()       document.user.e.focus()
 }
else
  document.user.submit()
        }
</script>
 </head>
 <body>
 <p align="center">
<font size=6>Данные о пользователе</font>
<form name="user">
<b>Пожалуйста, укажите данные о себе:
</b>
<br>
ФИО<input type="text" name="f" size="30"><br>
Адрес<input type="text" name="a" size="35"><br>
e-mai<input type="text" name="e" size="30"><br>
<input type="button" value="Послать" onClick="doSend()">
<input type="reset" value="
```

```
</form>
</p>
</body>
</html>
```

Вставка звука

Если вам необходимо озвучить страницу, вот простейшая инструкция: <bgsound src="music/gimn.mid" loop=infinite>

С помощью JavaScript можно разнообразить страницы сайта.

Пример скрипта проигрывания музыки при наведении на заголовок текста:

```
<script>
function playHome() {
 document.all.sound.src = "music/file.mid"}
 </script>
 …
 <bgsound id=sound>
 <h1 onMouseover=playHome()>Заголовок с музыкой</h1>
```

Вопросы для самоконтроля:

1. В чем состоят преимущества и ограничения программ, работающих на стороне клиента?
2. Опишите язык JavaScript?
3. Расскажите об основах синтаксиса JavaScript

Тема 2. Модели DHTML

Объектная модель HTML страницы; Событийная модель DHTML: связывание событий с кодом, всплытие событий, объект Event

3.2.1 Объектная модель HTML страницы.

DHTML (динамический HTML) – это набор средств, которые позволяют создавать более интерактивные Web-страницы без увеличения загрузки сервера. Другими словами, определенные действия посетителя ведут к изменениям внешнего вида и содержания страницы без обращения к серверу. DHTML построен на *объектной модели документа (Document Object Model, DOM),* которая расширяет традиционный статический HTML документ. DOM обеспечивает динамический доступ к содержимому документа, его структуре и стилям. В DOM каждый элемент Web-страницы является объектом, который можно изменять. DOM не определяет новых тэгов и атрибутов, а просто обеспечивает возможность программного управления всеми тэгами, атрибутами и каскадными таблицами стилей (CSS).

Динамический HTML (Dynamic HTML или DHTML) не является каким-то особым языком разметки страниц. Это всего лишь термин, применяемый для обозначения HTML-страниц с динамически изменяемым содержимым.

Реализация DHTML покоится на трех "китах": непосредственно HTML, каскадных таблицах стилей (Cascade Style Sheets — CSS) и языке сценариев (JavaScript или VBScript). Эти три компонента DHTML связаны между собой объектной моделью документа (Document Object Model — DOM), являющейся, по сути, интерфейсом прикладного программирования (API). DOM связывает воедино три перечисленных компонента, придавая простому документу HTML новое качество, — возможность динамического изменения своего содержимого без перезагрузки страницы. Символически подобное единство показано на рис. 10.

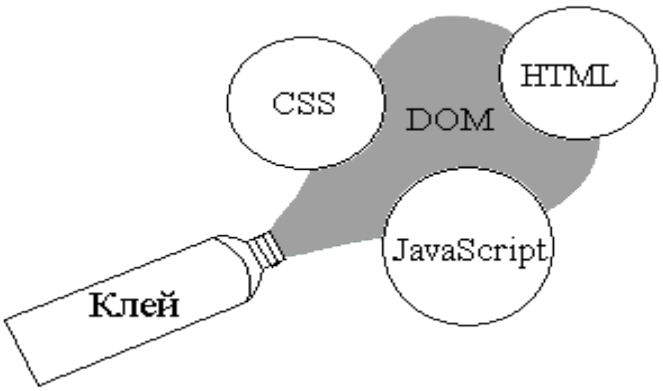


Рис. 10. Компоненты динамического HTML

Каскадные таблицы стилей можно сравнить со стилевыми файлами любого текстового редактора. С их помощью определяется внешний вид отображаемого HTML-документа: цвет шрифта и фона документа, сам шрифт, разбивка текста и многое другое. Для каждого элемента, задаваемого определенным тэгом HTML, можно определить свой стиль отображения в окне браузера. Например, заголовки первого уровня будут отображаться шрифтом Arial I6pt синего цвета, заголовки второго уровня — Arial 14pt красного цвета, основной текст — Times New Roman 10pt черного цвета с одинарным интервалом между строками. Можно создать таблицу стилей и использовать ее для всех документов, расположенных на сервере, что придаст стройность и строгость всему Web-сайту.

Объектная модель документа делает все элементы страницы программируемыми объектами. С ее помощью через языки сценариев можно получить доступ и управлять всем, что есть в документе. Каждый элемент HTML доступен как индивидуальный объект, а это означает, что можно изменять значение любого параметра любого тега HTML-страницы, и, как следствие, документ действительно становится динамическим. Любое действие пользователя (щелчок кнопкой мыши, перемещение мыши в окне браузера или нажатие клавиши клавиатуры) объектной моделью документа трактуется как событие, которое может быть перехвачено и обработано процедурой сценария.

DHTML достаточно новая технология, и не все браузеры поддерживают объектную модель документа и каскадные таблицы стилей. Однако DHTML использует стандартные теги HTML, и поэтому пользователи браузеров, не поддерживающих DOM, практически увидят все, что задумано разработчиком динамической страницы, но только в статическом виде.

Есть еще одна "неприятность", связанная с тем, что разные фирмы-разработчики браузеров могут реализовывать собственную объектную модель документов, как это произошло с двумя популярными браузерами Internet Explorer и Netscape Navigator. Поэтому разработчикам динамических страниц приходится, в конечном счете, писать два варианта своих приложений, чтобы пользователи указанных браузеров могли правильно просматривать их страницы.

Структура документа

В объектной модели документа любой документ представляется в виде логической древовидной структуры. Например, следующий фрагмент документа HTML:

```
<BODY>
.   .   .
<P ID='p1'> В блоковый элемент, каким является абзац,
можно добавлять <B ID='b1 'Устраиваемые элементы</B> и
даже другие блоковые элементы <IMG ID='img1' SRC="my.gif">
</P>
<IMG ID='img2' SRC="my-1.gif">
.   .   .
</BODY>
```

будет представлен в объектной модели документа логической структурой, показанной на рис.11.

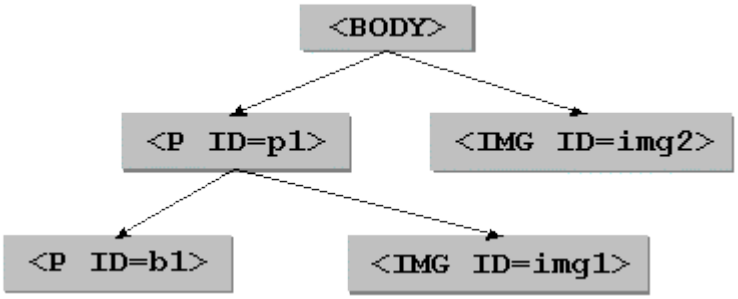


Рис. 11. Логическая структура фрагмента документа

Для сложного документа логическая структура будет, естественно, сложнее. В ней может оказаться много "деревьев", которые в сумме будут представлять уже некоторый "лес".

Для понимания объектной модели документов важно осознавать, что логическая древовидная структура представления документа никак не связана с реализацией этой модели именно в виде древовидной структуры. Рекомендации не регламентируют способ реализации модели, она может быть произвольной. Основное — это принцип

структурного изоморфизма: две реализации объектной модели документа, используемые для представления одного и того же документа, создадут одну и ту же структурную модель с одинаковыми объектами и их связями.

Другой важный аспект модели документов — она оперирует с объектами в полном соответствии с традиционными объектно-ориентированными технологиями: все элементы документа представляются в виде объектов. В узлах структурной логической схемы находятся объекты, а не данные, со всеми присущими объектам свойствами и поведением.

Объектная модель документов, таким образом, определяет:

- интерфейсы и объекты, используемые для представления документа и манипулирования с ним;
- семантику (смысл) этих интерфейсов и объектов, включая и поведение, и параметры;
- "родственные" связи и взаимодействие между этими интерфейсами и объектами.

Основное назначение реализации объектной модели документов — предоставить возможность доступа и манипулирования элементами документа из программы с помощью объектов, выстроенных в некоторую иерархическую структуру, а также обеспечить взаимодействие между объектами. Поэтому любая реализация модели включает в свою очередь и управление событиями, представленными также в виде объектов.

3.2.2. Событийная модель DHTML: связывание событий с кодом, всплытие событий, объект Event.

В объектной модели DHTML с каждым элементом страницы можно связать определенное действие пользователя: щелчок кнопкой мыши, нажатие клавиши клавиатуры, перемещение в области элемента курсора мыши и т. д. Эта технология основана на фундаментальном понятии *события* в операционных системах с графическим интерфейсом пользователя.

Одна из сильных сторон Dynamic HTML - это события. Как вы, наверное, знаете, Windows - операционная система, управляемая событиями. Когда вы выполняете некоторое действие в среде Windows, например, щелкаете на окне мышью, операционная система формирует

сообщение о событии. Получая сообщения Windows, WebBrowser формирует на их основе события своей объектной модели. Например, когда вы щелкаете мышью внутри страницы, браузер получает это событие и пропускает его через всю иерархию объектов страницы, инициируя, таким образом, множество событий onmousedown, onmouseup и onclick для всех элементов страницы, которых это касается. Если нужно обработать событие, возвратив некоторый результат, то инструкции идут от кода обратно к браузеру с помощью все той же объектной модели.

Свойства объектов-событий можно использовать во встраиваемых сценариях для получения информации о событии. При возникновении любого события динамически создается свойство event объекта window, входящего в объектную модель и представляющего окно браузера. Это свойство и является объектом, соответствующим сгенерированному событию.

Цикл жизни события

Любое событие имеет свой "жизненный" цикл: от момента возникновения действия или условия, являющегося причиной генерирования события, до выполнения последнего оператора обработчика события или финальных действий браузера. *Цикл жизни любого типичного события включает следующие этапы:*

1. Происходит действие пользователя или возникает условие, которое возбуждает событие.
2. Тотчас же корректируется объект event, чтобы отразить параметры возникшего события.
3. Событие генерируется — это и есть истинное сообщение о возникшем событии.
4. Вызывается обработчик событий элемента-источника события, который выполняет определенные программистом действия и завершает свою работу.
5. Событие передается вверх по иерархии объектов (bubble up) и вызывается обработчик события объекта, являющегося родителем объекта-источника события. Это "всплытие" вверх по иерархии объектов продолжается, пока не будет достигнут самый верхний объект иерархии — объект window, или обработчик события какого-либо объекта не аннулирует событие.

6. Выполняются заключительные действия по умолчанию, если таковые определены, но при условии, что событие не было аннулировано.

Если для элемента-источника события не определен обработчик событий, то в иерархии объектов определяется его родитель, и обработчик событий родителя выполняет соответствующие действия по обработке события и так происходит до корневого объекта иерархии.

Какие удобства предоставляет подобная технология обработки событий? Прежде всего, нет необходимости для каждого элемента писать процедуру обработки события и присоединять ее к нему. Достаточно написать одну процедуру для элемента-родителя, и она будет обрабатывать события, возбуждаемые всеми порожденными родителем элементами. Это позволяет централизованно обрабатывать наиболее часто возникающие события, и, как результат, требует меньше усилий и времени для написания и поддержки кода процедур обработки событий.

Пример 1 демонстрирует технику передачи события вверх по иерархии объектов. В нем щелчки кнопкой мыши на всех элементах страницы обрабатываются централизованно обработчиком события элемента <BODY>, который является родителем всех элементов страницы.

Пример 1 *Передача обработки события родителю*

```
<HTML>
<HEAD><TITLE>Всплывание события</TITLE>
</HEAD>
<BODY ID='body' onclick="alert('Не надо щелкать!');">
<H1 ID='head1'>Привет!</H1>
<P ID='parag1'>Это простой пример, <B ID='bold1'>ну очень простой</B>
пример.
</BODY>
</HTML>
```

Щелчок на любом элементе документа приводит к отображению диалогового окна предупреждений из процедуры обработки события click объекта body.

Если к какому-нибудь элементу добавить собственный обработчик событий, то будут выполнены две процедуры: самого элемента и

элемента родителя. Если элемент расположен достаточно глубоко в иерархии объектов, и каждый элемент, расположенный выше него, имеет также собственный обработчик событий, то неужели событие будет обрабатываться всеми обработчиками? Да, именно это и произойдет, если только какой-то обработчик не аннулирует "всплывающее" вверх по иерархии событие. Объект event имеет свойство cancelBubble, которое позволяет аннулировать событие, если установить его значение равным true. После этого соответствующее событие не существует, и обработчики этого события для всех, расположенных выше элементов, не вызываются. В примере 2 обработчик щелчка мыши выделенного элемента <в> аннулирует данное событие. Это приводит к тому, что при щелчке на нем никакого диалогового окна с сообщением не отображается.

Пример 2 *Аннулирование события*

```
<HTML>
<HEAD><TITLE>Аннулирование события</TITLE>
</HEAD>
<BODY ID='body' onclick="alert('Не надо щелкать!');">
<H1 ID='headl'>Привет!</Hl>
<P ID='parag1'>Это простой пример,
<B ID='bold1' onclick="window.event.cancelBubble=true" >ну очень простой</B> пример.
</BODY>
</HTML>
```

Аннулирование события достаточно частое действие в сценариях обработки событий, и будет интенсивно использоваться далее в примерах раздела "Динамический HTML в Internet Explorer".

Присоединение обработчика события к элементу страницы осуществляется установкой значения параметра обработки события тега элемента или соответствующего свойства объекта.

Каждый элемент HTML может быть источником многих событий. В табл. 22 представлены события, которые могут генерировать и обрабатывать все элементы страницы, и соответствующие им параметры обработки событий.

Таблица 22 Общие события всех HTML-элементов

Параметр	**Условие возникновения**
onmouseover	Курсор мыши перемещается в область элемента (т. е. находится внутри элемента)
onmouseout	Курсор мыши выходит за пределы элемента
onmousedown	Нажата любая кнопка мыши, когда курсор находится в пределах элемента
onmouseup	Отпущена ранее нажатая любая кнопка мыши, когда курсор находится в пределах элемента
onmousemove	Курсор мыши перемещается в пределах области элемента
onclick	Щелчок левой кнопкой мыши на элементе
ondblclick	Двойной щелчок левой кнопкой мыши на элементе
onkeypress	Нажата и отпущена клавиша клавиатуры. Если клавиша удерживается, то генерируется серия события Keypress
on key down	Нажата клавиша клавиатуры. Генерируется только одно событие, даже если клавиша удерживается
onkeyup	Отпущена ранее нажатая клавиша клавиатуры

Объект event

Особенностью программ, создаваемых для среды веб, является то, что они управляются событиями. Чтобы узнать, какое событие произошло, в DOM имеется объект события event. Объект event является локальным и его следует явным образом передавать в обработчик события.

Объект event создается автоматически всякий раз, когда возникает какое-либо событие. Этот объект не зависит от используемого языка создания сценария, и его использование в процедурах обработки событий для получения информации о сгенерированном событии является предпочтительным способом получения достоверной информации о событии.

Каждое событие характеризуется параметрами, которые передаются в сценарий через свойства объекта event. Существуют параметры, общие для всех типов событий (например, координаты курсора мыши в окне браузера) и специфические для определенного события (например, код нажатой клавиши для событий клавиатуры). Свойства объекта event, как и сам он, являются динамическими и создаются в зависимости от типа произошедшего события. При описании свойства, если не оговорено противное, подразумевается, что оно является общим для всех типов событий.

Свойство srcEiement определяет элемент документа, явившийся источником события. Оно может быть полезным при централизованной обработке событий элементом, расположенным выше в иерархии объектов документа истинного "виновника" события, и, в зависимости от типа элемента, программа-обработчик может предпринять соответствующие действия.

Важное свойство cancelBubble, аннулирующее событие и прекращающее передачу его на обработку вверх по иерархии объектов, рассмотрено немного ранее в этом же разделе.

Свойство returnvalue является булевым и возвращает значение true или false после завершения выполнения процедуры обработки события. При передаче события вверх по иерархии значение этого свойства можно использовать для альтернативной обработки события. Кроме того, если в обработчике события элемента, для которого определены действия по умолчанию, это свойство устанавливается равным false, то это отменяет выполнение действий по умолчанию. Одним из

таких элементов является тег <A>, действием по умолчанию которого является переход по ссылке, задаваемой параметром HREF.

По значениям свойств aitKey, ctriKey и shiftKey элемента-источника события определяется, была ли нажата, соответственно, клавиша <Alt>, <Ctrl> или <Shift> в момент возникновения события. Значение свойства равно true, если клавиша была нажата, и false — в противном случае.

Следующий фрагмент сценария отменяет переход по любой связи в документе, если при щелчке на ней была нажата клавиша <Shift>:

```
document.onclick=click;
function click() {
    if((window.event.srcElement.tagName=='A') && win-
dow.event.shiftKey) {window.event.returnValue=false;
    }
}
```

Для событий мыши определены свойства, значениями которых являются координаты положения курсора в момент возникновения события.

Свойства clientX и clientY представляют координаты относительно области отображения браузера, offsetX и offsetY являются координатами относительно элемента-контейнера, в котором расположен элемент-источник события, screenX и screenY — абсолютные координаты курсора мыши, т. е. координаты экрана монитора. Все значения этих свойств определены в пикселах.

Свойства x и y определяют положение курсора мыши по горизонтали и вертикали относительно позиционированного контейнера, содержащего элемент-источник события. Если ни один контейнер не позиционирован, то положение определяется относительно тела документа <BODY>.

Полезное свойство событий мыши button определяет нажатую кнопку мыши. Его возможные значения представлены в табл. 23

Таблица 23 Значения свойства button

Значение		**Нажаты кнопки мыши**
0		Ни одна

1		Левая
2		Правая
3		Одновременно левая и правая
4		Средняя
5		Одновременно левая и средняя
6		Одновременно правая и средняя
7		Все три одновременно

Свойства toElement и fromElement Применимы ТОЛЬКО К событиям onmouseover и onmouseout. Они определяют, от какого элемента и к какому перемещался курсор мыши, и полезны при анализе этих действий для вложенных элементов.

Некоторые элементы на HTML-странице могут получить фокус. В каждый момент времени только один элемент может обладать фокусом, и ввод данных с клавиатуры направляется именно этому элементу. Наиболее часто используемыми элементами с фокусом являются <BUTTON> и некоторые типы элемента <INPUT>. Для таких элементов при получении ими фокуса генерируется событие onfocus, а при потере фокуса — событие onbiur. Элемент получает фокус при щелчке на нем кнопкой мыши или перемещением на этот элемент с помощью клавиш клавиатуры.

При выделении на странице части документа возникают события, регистрирующие эти действия. Событие onselectstart генерируется, когда пользователь нажимает кнопку мыши при расположении курсора в области документа. Если после этого нажатия он перемещает курсор мыши по документу (не отпуская нажатую кнопку), то инициируется событие onselect, регистрирующее выделение части документа. У этого события есть действия по умолчанию: визуально отметить выделенную часть документа изменением ее фона. Это действие можно отменить, установив значение свойства returnValue события равным false в процедуре обработки этого события.

Для самого документа существуют два события, отмечающие некоторые стадии его обработки браузером. На самом деле эти события относятся к объекту window, находящемуся на вершине иерархии объектов, но обработчики этих событий задаются в тэге <BODY> документа. Событие onload происходит сразу же после того, как были загружены в окно браузера все элементы страницы. Его можно использовать для выполнения действий при первоначальной или повторной загрузке страницы. Событие unload генерируется до начала выгрузки документа, когда пользователь желает загрузить другой документ, и в процедуре обработки этого события можно, например, напомнить пользователю о выполнении некоторых действий перед окончательной выгрузкой страницы.

Всплывание событий заключается в том, что на событие может быть получена реакция не только от элемента-источника события, но также и от всех его родительских элементов вплоть до тела документа и самого документа. Событие может быть обработано на любом уровне. В приведенном ниже примере обработчик щелчков мышью на ссылке будет также обрабатывать щелчки мышью на изображении.

Пример 2

```
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=win-
dows-1251">
</head>
<body>
<H2>Всплывание события</H2>
<b>Для получения информации можно щелкнуть мышкой как на
изображении,
   так и на тексте</b><br>
<a href="be.htm"><img src="be.gif" align="left">Кто это?</a>
</body>
        </html>
```

Действие по умолчанию обеспечивается встроенной в браузер обработкой событий. Например, действием по умолчанию на нажатие ссылки <A href="…"> является переход по указанному адресу и загрузка страницы. Многие события позволяют заменить встроенные действия по умолчанию на индивидуальную обработку.

Вопросы для самоконтроля:

1. В чем заключается разница между объектной моделью документа DOM и объектной моделью браузера BOM?

2) Каким образом строится дерево DOM?

3) Как будет выглядеть дерево DOM, если html-разметка документа не будет соответствовать стандарту языка?

4) Сколько типов узлов может присутствовать в дереве DOM?

5) Каким образом можно редактировать дерево DOM?

Тема 3. Применение DHTML

Программное изменение содержания документа; программное изменение формата документа; программное изменение положения элементов

Особенностью программ, создаваемых для среды веб, является то, что они управляются событиями. Чтобы узнать, какое событие произошло, в DOM имеется объект события event (табл.24). Объект event является локальным и его следует явным образом передавать в обработчик события.

Таблица 24 Свойства объекта event

Свойство	**Описание**
bubbles	Указывает возможность «всплывания» события (передачи управления вверх по иерархической структуре
cancelable	Указывает возможность отмены действия события, заданного по умолча-нию
currentTarget	Указывает событие, обрабатываемое в данный момент
eventPhase	Указывает фазу возбуждения события
target (только NN 6)	Указывает элемент, вызвавший собы- тие
timestamp (только NN 6)	Указывает время возникновения события
type	Указывает имя события

Установление связи между определенным событием и сценарием называется связыванием событий. События можно связать с помощью специальных атрибутов любого элемента или с помощью тэга **SCRIPT**.

Связывание событий с атрибутами удобно, но требует расширения языка HTML каждый раз при изобретении нового события. А так как HTML развивается медленно, данный подход используется только

для небольшого набора встроенных событий. Как атрибуты любого элемента в DHTML представлены события мыши и клавиатуры.

Пример 3

```
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=win-
dows-1251">
</head>
<body>
<H2>Смена графических объектов, обтекаемых текстом</H2>
<b>Для смены графического объекта переместите на него мышку</b>
<p>Иоганн Себастьян Бах сменит Людвига ван Бетховена.
   <a href="#"
     OnMouseOver="document.getElementById('B_B').src='ba.gif'"
     OnMouseOut="document.getElementById('B_B').src='be.gif'">
      <img id="B_B" src="be.gif" border=0 align="left"></a>
   Людвиг ван Бетховен сменит Иоганна Себастьяна Баха.
   Иоганн Себастьян Бах сменит Людвига ван Бетховена.
   Людвиг ван Бетховен сменит Иоганна Себастьяна Баха.
   Иоганн Себастьян Бах сменит Людвига ван Бетховена.
   Людвиг ван Бетховен сменит Иоганна Себастьяна Баха.
   Иоганн Себастьян Бах сменит Людвига ван Бетховена.
   Людвиг ван Бетховен сменит Иоганна Себастьяна Баха.
   Иоганн Себастьян Бах сменит Людвига ван Бетховена.
   Людвиг ван Бетховен сменит Иоганна Себастьяна Баха.
   Иоганн Себастьян Бах сменит Людвига ван Бетховена.
   Людвиг ван Бетховен сменит Иоганна Себастьяна Баха.
   Иоганн Себастьян Бах сменит Людвига ван Бетховена.
   Людвиг ван Бетховен сменит Иоганна Себастьяна Баха.
</body>
         </html>
```

Пример 4

```
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=win-
dows-1251">
</head>
```

```
<body>
<H2>Изменение вида данного элемента</H2>
<p onmousedown="this.style.fontStyle='italic';
   this.style.color='red'" onmouseup="this.style.fontStyle='';
   this.style.color='blue'">
      Для изменения цвета и стиля шрифта данного текста нажимайте
и отпускайте
      кнопку мыши</p>
</body>
        </html>
```

Обратите внимание на то, что в примере 2 область действия обработчика события определена с помощью идентификатора **B_B**, а в примере 3 – с помощью указателя **this**.

Использование тэга **SCRIPT** является более общим механизмом связывания события со сценарием. При этом можно использовать новые атрибуты такие, как **FOR** и **EVENT**:

- **FOR** указывает имя или идентификатор (ID) элемента, для которого описывается событие;
- **EVENT** указывает событие и все параметры, которые ему могут быть переданы.

Пример 3 теперь примет следующий вид:

```
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=win-
dows-1251">
   <script for="B" event="onmousedown()">
      <!--
      this.style.fontStyle='italic'; this.style.color='red';
      //-->
   </script>
   <script for="B" event="onmouseup()">
      <!--
      this.style.fontStyle=''; this.style.color='blue';
      //-->
   </script>
</head>
<body lang=RU>
```

```
<H2>Изменение вида данного элемента</H2>
<p id="B">Для изменения цвета и
      шрифта данного текста нажимайте и
      отпускайте кнопку мыши</p>
</body>
        </html>
```

Все события также представлены как свойства в DOM. Используя это, перепишем наш пример в таком виде:

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=win-
dows-1251">
</head>
<body>
<H2> Изменение вида данного элемента</H2>
<p id="B">Для изменения цвета и шрифта данного текста
      нажимайте и отпускайте кнопку мыши<br></p>
<script>
  <!--
  document.getElementById('B').onmousedown=new
     Function("this.style.fontStyle='italic'; this.style.color='red';");
  document.getElementById('B').onmouseup=new
     Function("this.style.fontStyle=''; this.style.color='blue';");
  //-->
</script>
</body>
        </html>
```

Не следует забывать, что все имена свойств надо вводить в нижнем регистре.

События мыши

- **OnMouseOver**. Перемещение указателя мыши на элемент.
- **OnMouseOut**. Перемещение указателя мыши за пределы элемента.
- **OnMouseDown**. Нажатие любой кнопки мыши. Внутри обработчика event.button указывает, какая кнопка нажата: 1 = левая, 2 = правая, 4 = средняя.

- **OnMouseUp**. Отпускание любой кнопки мыши. Внутри обработчика event.button указывает, какая кнопка отпущена: 1 = левая, 2 = правая, 4 = средняя.
- **OnMouseMove**. Перемещение указателя мыши. Внутри обработчика event.x и event.y - текущие координаты "горячей" точки курсора на экране.
- **OnClick**. Щелчок левой кнопкой мыши на элементе или нажатие <Enter> при каком-то элементе в фокусе.
- **OnDblClick**. Двойной щелчок левой кнопкой мыши на элементе.
- **OnDragStart**. Запуск операции перетаскивания. Цель – запретить операцию перетаскивания путем возвращения значения false.
- **OnSelectStart**. Запуск операции выделения элемента. Цель – запретить выделение области документа. Важно учесть, что отменяется лишь инициализация выделения, т.е. если выделение начато за пределами данной области, а на ней лишь продолжается, то помешать выделению нельзя. Событие **OnSelectStart** всплывающее, поэтому можно перехватить его и вернуть значение false. Это лишит посетителя возможности выделять текст на странице.
- **OnSelect**. Выделение элемента. Следует за **OnSelectStart** и возникает много раз по мере того, как посетитель расширяет или сужает выделение. Событие **OnSelect** не всплывает. Оно возникает лишь в том разделе документа, где происходит выделение.

События клавиатуры

- **OnKeyPress**. Нажатие и отпускание клавиши. Событие возникает много раз, если клавиша удерживается.
- **OnKeyDown**. Нажатие клавиши. Событие возникает один раз, даже если клавиша продолжает удерживаться.
- **OnKeyUp**. Отпускание клавиши.

Событие прокручивания

- **OnScroll**. Использование полосы прокрутки или прокручивание элемента неявно посредством другого действия не может отменить прокручивания, так как возникает после завершения прокручивания. Не всплывает.

События фокуса

- **OnFocus**. Возникает, когда элемент активизируется после щелчка по нему мышью или с помощью клавиатуры. Фокус могут получить только элементы пользовательского ввода и тело документа, а не элементы содержания документа.
- **OnBlur**. Возникает, когда элемент теряет фокус. Используется для контроля корректности ввода.

Пример 5

```
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
   <script>
      <!--
      function verify()
      {
         //Ввод строки
         var str=document.inp.m.value;
         //Преобразование строки в число
         var int=parseInt(str);
         //Проверка
         if ( int == 1 || int == 2 || int == 12 )
            {alert("Зима");}
         else if ( 3 <= int && int <= 5 )
            {alert("Весна");}
         else if ( 6 <= int && int <= 8 )
            {alert("Лето");}
         else if ( 9 <= int && int <= 11 )
            {alert("Осень");}
         else
            {alert("Неверно!");}
      }
      //-->
   </script>
</head>
<body>
<H2>Контроль ввода</H2>
```

```
<form name="inp">
  <label>Введите номер текущего месяца:
    <input name="m" size="2" OnBlur="verify()">
  </label>
</form>
</body>
</html>
```

Каждой гиперссылке, заголовку или текстовому параграфу можно присвоить имя, атрибуты стиля или цвета текста и указать это имя в сценарии, имеющемся на данной странице. Сценарий может быть написан на любом существующем скриптовом языке, но здесь подразумевается использование языка JavaScript. Элементы страницы впоследствии могут изменяться в результате определенного события, например, при наведении курсора мыши, при щелчке, нажатии клавиши на клавиатуре либо после истечения заданного промежутка времени. Изменяться может не только стиль и цвет текста, но и весь объект, текст или рисунок.

Например, при расположении курсора мыши над каким-либо текстом последний меняет цвет или размер и появляется всплывающая подсказка, что при щелчке кнопкой мыши можно увидеть элементы раскрывающегося списка.

Раскрывающийся список

Создать простое меню можно на основе HTML-элемента <UL> и свойства display каскадных таблиц стилей, которое позволяет скрывать элементы страницы. Поместим на страницу вложенный список:

```
UL ID="idList" NAME="idList">
      <LI TITLE="Щелкни и раскрой"
          STYLE="cursor: hand;"> Один
<UL ID="idListOneA" NAME="idListOneA"
          STYLE="display: none; cursor: default;">
          <LI TITLE="Файл А">А
          <LI TITLE="Файл Б">Б
       </UL>
       <LI TITLE="Нераскрывающийся список"> Два
       <LI TITLE="Нераскрывающийся список"> Три
</UL>
```

Список idList составлен из трех элементов <LI> и вложенного списка idListOneA, который не отображается (его свойство isplay равно none) и будет использован для создания раскрывающегося списка при щелчке на первом элементе списка. Строка, заданная в параметре TITLE, отображается в виде всплывающей подсказки при расположении курсора мыши над элементом. Этот фрагмент отображается как простой статический список из трех элементов (вложенный список не отображается) с маркерами в виде закрашенных кружков.

Чтобы сделать список раскрывающимся, необходимо добавить к первому элементу внешнего списка и к элементам внутреннего списка обработчики событий и определить в них необходимые действия.

Прежде всего, запрограммируем изменение цвета этих элементов при расположении над ними курсора мыши, чтобы пользователь обратил внимание на их динамичность. Для этого добавим в теги элементов параметры обработчиков событий:

```
ONMOUSEOVER="flashMe(this,'red')"
ONMOUSEOUT="flashMe(this,'black')"
```

В обработчиках событий вызывается функция fiashMe(), изменяющая цвет элемента. имя элемента (параметр this) и цвет передаются в качестве параметров функции:

```
function flashMe(eSrc, sColor) {
eSrc.style.color=sColor
idList;OneA.style.color="black"
}
```

Обратим внимание на оператор установки черного цвета списка idListOneA. Если его не будет, то при раскрытии первого элемента внешнего списка цвет элементов вложенного списка idListOneA будет таким же, как и цвет элемента-родителя — красным (свойство color наследуется потомками).

Теперь остается добавить обработчик щелчка кнопки мыши в первый элемент списка idList, выполняющий функцию отображения вложенного списка idListOneA, если он скрыт, и скрывающий его, если он видим:

```
ONCLICK="toggleListOneA()"
```

исходный текст функции toggieListoneA () приведен ниже и не требует комментариев:

```
function toggleListOneA() {
        eTarget=idListOneA
        eTarget.style.display == "none" ? eTarget.style.display="block":
              eTarget.style.display="none"
eTarget.display == "none" ? eTarget.display="block":
              eTarget.display="none"
        }
```

Итак, у нас есть все функции, реализующие раскрывающийся список. Однако мы не учли одного обстоятельства. Если пользователь щелкнет на любом из элементов раскрывшегося списка, то список закроется. Вспомним "подъем" события по иерархии объектов. Хотя в элементе раскрывающегося списка нет обработчика события ONCLICK, в любом случае оно передается вверх по иерархии и обрабатывается всеми встречающимися обработчиками этого события. Поэтому процедура обработки события ONCLICK внешнего списка закроет внутренний (установит значение его свойства display равным none). Чтобы этого не происходило, следует в обработчиках событий элементов внутреннего списка отменить передачу события на обработку вверх по иерархии объектов:

```
ONCLICK="window.event.cancelBubble=true"
```

Окончательно исходный текст нашего раскрывающегося списка выглядит так:

Пример 5. *Раскрывающийся список*

```
<HEAD>
<SCRIPT LANGUAGE=' JavaScript' >
<!--
function toggleListOneA(){
        eTarget=idListOneA
        eTarget.style.display == "none" ? eTarget.style.display="block":
              eTarget.style.display="none"
        eTarget.display == "none" ? eTarget.display="block":
              eTarget.display="none"
}
function flashMe(eSrc, sColor) {
eSrc.style.color=sColor
idListOneA.style.color="black"
}
```

```
//-->
</SCRIPT>
<STYLE TYPE="text/css"><!--
        H1 {background-color:lightgrey;
            font-family: Arial;
            font-size: 11pt;
            color: indianred;
        }
            #idListOne {
        list-style-image:url(item, jpg);
        color: black;
        list-style-position:inside;
        }
--></STYLE>
</HEAD>
<H1> Пример 2.4: Раскрывающийся список </H1>
<UL ID="idList" NAME="idList">
        <LI ONCLICK="toggleListOneA()"
            ONMOUSEOVER="flashMe(this,'red')"
            ONMOUSEOUT="flashMe(this,'black')"
            TITLE="Щелкни и раскрой"
            STYLE="cursor: hand;"> Один
        <UL ID="idListOneA"
            NAME="idListOneA"
        STYLE="display:none; cursor:default;">
        <LI TITLE="Файл А"
            ONCLICK="window.event.cancelBubble=true"
            ONMOUSEOVER="flashMe(this,'red')"
            ONMOUSEOUT="flashMe(this,'black')">А
        <LI TITLE="Файл Б"
            ONCLICK="window.event.cancelBubble=true"
            ONMOUSEOVER="flashMe(this,'red')"
            ONMOUSEOUT="flashMe(this,'black')">Б
          </UL>
          <LI TITLE="Нераскрывающийся список"> Два
          <LI TITLE="Нераскрывающийся список"> Три
</UL>
```

Графический файл item.jpg в свойстве каскадных таблиц стилей tem-style-image задает изображение маркера в списках. Свойство cursor определяет тип курсора мыши, когда он располагается над элементом. В нашем примере курсор будет меняться на изображение руки (значение свойства hand).

Раскрывающийся список в Web-документах можно использовать в качестве простого меню, если в процедурах событий ONCLICK его элементов задать отображение каких-либо документов в отдельных окнах или во фрейме страницы.

Аналогичную технику можно применить для динамического отображения частей документа, связав отображение, например, какого-либо абзаца, с расположением курсора мыши на определенном слове документа, а его скрытие — с перемещением курсора от этого слова. Подобные приемы динамического раскрытия документа позволяют на одном экране представить краткую полную информацию, а детали показать в больших раскрывающихся частях документа.

Объединение JavaScript и CSS

1. Пример изменения цвета текста.

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
<h1 style="color: red">Добро пожаловать на нашу страницу!</h1>
<p>Здесь много интересной информации.
        Здесь много интересной информации.
        Здесь много интересной информации.
        Здесь много интересной информации.
        Здесь много интересной информации. </p>
        </body>
 </html>
```

Данный пример применения CSS позволяет сделать заголовок красного цвета. Допустим, вы хотите, чтобы текст заголовка только тогда становился красным, когда пользователь наводит на него курсор. Этого можно добиться с помощью CSS и JavaScript.

Шаг 1. Удаление существующей информации о стиле Это действие может показаться вам шагом назад, но оно действительно необходимо:

```
<html>
 <head>
        <title>Простая страница</title>
 </head>
 <body>
 <h1>Добро пожаловать на нашу страницу! </h1>
  <p>3десь много интересной информации.
        3десь много интересной информации.
        3десь много интересной информации.
        3десь много интересной информации.
       3десь много интересной информации. </p>
</body>
</html>
```

Шаг 2. Добавление идентификатора. Поскольку вам нужно както обращаться к элементу, с которым будут производиться манипуляции, необходимо в тэг <hl> добавить атрибут id - это краткое обозначение, позволяющее указать нужный элемент:

```
<html>
 <head>
<title>Простая страница</title>
 </head>
<body>
 <hl id="head1">Добро пожаловать на нашу страницу!</h1>
 <p>3десь много интересной информации.
      3десь много интересной информации.
      3десь много интересной информации.
      3десь много интересной информации.
      3десь много интересной информации. </p>
       </body>
 </html
```

Шаг 3. Добавление обработчика событий Следующий шаг — добавление обработчика событий. Этому действию соответствует событие onMouseover. Также следует указать имя функции, которая будет вызываться при выполнении события:

```
<html>
<head>
<title>Простая страница</title>
</head>
<body>
 <h1 id="head1" onMouseover ="colorchange ()">Добро пожаловать на
нашу
 страницу!</h1>
<p>3десь много интересной информации.
    3десь много интересной информации.
    3десь много интересной информации.
    3десь много интересной информации.
    3десь много интересной информации. </p>
</body>
 </html>
```

Шаг 4. Написание сценария JavaScript Вам потребуется единственная строка, состоящая из следующих частей:

 имя объекта на странице, с которым должен выполняться ваш сценарий - в данном случае head1;

 применяемый аспект JavaScript - в данном случае style;

 атрибут стиля, который будет изменяться - color;

 новое значение, принимаемое атрибутом стиля - red.

Соедините это, и получится следующая строка:

```
 Head1.style.color = "red"
```

Добавьте ее в функцию и сохраните файл. В окончательном варианте страница должна выглядеть так:

```
<html>
 <head>
 <title>Простая страница</title>
 <script language="JavaScript">
function colorchange()
{
head1.style.color = "red";
 }
 </script>
</head>
```

```
 <body>
 <h1 id="head1" onMouseover="colorchange()">Добро пожаловать на нашу страницу!</h1>
<p>3десь много интересной информации.
    3десь много интересной информации.
    3десь много интересной информации.
    3десь много интересной информации.
    3десь много интересной информации. </p>
</body>
</html>
```

Откройте эту страницу в браузере и посмотрите, что происходит, когда вы наводите курсор на заголовок. Если все было сделано правильно, то цвет заголовка изменится. Но обратите внимание, что цвет текста не становится прежним, когда вы убираете курсор с заголовка.

2. Пример использования атрибута text-decoration.

Используя в сценариях JavaScript атрибуты, название которых пишется через дефис, убирайте дефис и пишите оба слова слитно, причем второе слово должно начинаться с заглавной буквы. Таким образом, textdecoration в сценариях должно выглядеть как textDecoration.

```
<html>
 <head>
 <title>Простая страница</title>
 <script language="JavaScript">  function addunderline()
{
head.style.textDecoration = "underline";
 }
function removeunderline()
 {
head.style.textDecoration = "none";
 }
</script>
 </head>
 <body>
<h1 id="head" onMouseover="addunderline()"  onMouseout="removeunderline()">Добро пожаловать на нашу
страницу! </h1>
```

Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
Здесь много интересной информации.
</body>
</html>

Необходимо обратить внимание на прописную букву в слове textDecoration — если все слово набрать в нижнем регистре, сценарий выполняться не будет. Теперь при наведении курсора заголовок станет подчеркнутым, а затем, если убрать курсор, вернется в прежнее состояние.

3. Пример точного позиционирования текста. Сначала необходимо рассмотреть, каким образом осуществляется позиционирование текста. Наиболее часто применяются следующие атрибуты позиционирования:

 position - имеет два интересующих нас значения: absolute и relative (по умолчанию значение static). Для значения absolute в качестве точки отсчета используется верхний левый угол окна браузера, и все параметры местоположения отмеряются от него. В свою очередь, для relative точкой отсчета является то место, в котором разместился бы элемент страницы, если бы не было представлено никакой информации о местоположении;

 top - используется для указания вертикального смещения элемента от точки отсчета. Величина смещения может выражаться в различных единицах (пикселы, дюймы, сантиметры, миллиметры и т.п.). В наших примерах используются пикселы. Положительное значение top соответствует смещению элемента страницы вниз, в то время как отрицательное - по направлению к верхней границе окна браузера;

 left - подобен атрибуту top, но применяется для указания горизонтального направления. Положительное значение соответствует сдвигу элемента вправо, отрицательное – влево.

<html>
<head>
<title>Простая страница</title>
</head>
<body>

<h1 style=" text-decoration: underline">Добро пожаловать на нашу страницу!</h1>

Простой текст для позиционирования.

</body>

</html>

С помощью CSS текст расположен со значением absolute, то есть его положение отсчитывается от верхнего левого угла окна браузера. Значение атрибута top равно 125px, таким образом, текст будет расположен на 125 пикселов ниже верхнего края страницы. Значение атрибута left равно 200px, то есть текст будет сдвинут на 200 пикселов от левого края окна браузера.

Вопросы для самоконтроля:

1.Как происходит программное изменение содержания документа?

2. Как осуществляется изменение формата документа?

3. Как изменить положение элементов?

Модуль 4. ПРОГРАММИРОВАНИЕ НА PHP. MySQL & PHP
Тема 1. Язык PHP

Введение в программирование на стороне сервера на примере PHP. Принцип работы.
Синтаксис языка программирования PHP4

4.1.1 Введение в программирование на стороне сервера на примере PHP. Принцип работы.

PHP (Hypertext Preprocessor) – наиболее простой скриптовый язык программирования, широко применяющийся при создании динамически генерируемых веб-страниц. Основная масса Интернет ресурсов, на данный момент, написана с использованием именно этого языка программирования. При всей своей простоте, PHP позволяет разрабатывать профессиональные веб-проекты любой сложности, от небольших сайтов до крупных порталов.

PHP-код программы выполняется на стороне сервера. После того, как пользователь совершил на сайте некое действие, например, клик по ссылке в меню, с целью перейти на другую страницу сайта, браузер посылает запрос серверу на соответствующую страницу с PHP-кодом. Далее, PHP-код обрабатывается интерпретатором PHP и генерируется HTML-код, который возвращается серверу. Сервер в свою очередь, передаёт этот HTML-код обратно браузеру. В результате пользователь видит отображение в браузере новой страницы, имеющей свой HTML-код. При просмотре же исходного кода этой страницы виден будет только HTML-код, а PHP-код остается недоступен для просмотра.

Большой плюс языка PHP состоит в том, что PHP-код можно внедрять непосредственно в HTML-файлы. PHP-код встраивается в HTML-страницы при помощи угловых скобок и знака вопроса:

```
<?php
...здесь находится код программы php...
?>
```

Сами же файлы, в которых присутствует PHP-код, имеют расширение ***.php.

Для создания веб-проектов на языке php, необходимо программировать, используя либо установленный локальный сервер на своём компьютере, либо работая с помощью удалённого сервера. Удалённый

сервер не всегда удобен, да и, как правило, за это надо платить. Для создания сервера на своём компьютере понадобятся следующие программы: Apache или Denver (сервер), MySQL (базы данных), PHP. Все программы актуальных версий можно найти на официальных сайтах разработчиков. Создание PHP-файлов, написание кода и работа с ним ничем не отличается от того же процесса, что и при работе с HTML. Работать с php-кодом можно также в обычном текстовом редакторе, но делать это с помощью php-редактора намного удобнее.

Можно в принципе обойти процесс установки серверных программ на компьютер и долгой их настройки. Это возможно при использовании уже готовых сборок в виде одной целой программы. Например, Top Server, и подобные программы. Установив такую программу на компьютер, вы получаете уже готовый набор всех серверных программ, с отдельной панелью управления ими. Такой вариант идеален для создания и разработки сайтов на компьютере и проверки их в «живую». Для использования компьютера в качестве сервера такой вариант, конечно, использовать нельзя. Для тех, кто только начинает знакомиться с языком программирования PHP и не имеет особого навыка в программировании, кому, нужно лишь только написав некоторый код проверить его в действии, такой вариант будет самым удобным.

4.1.2 Синтаксис языка программирования PHP.

PHP – это широко используемый язык сценариев общего назначения с открытым исходным кодом.

Говоря проще, PHP это язык программирования, специально разработанный для написания web-приложений (сценариев), исполняющихся на Web-сервере.

Аббревиатура PHP означает “Hypertext Preprocessor (Препроцессор Гипертекста)". Синтаксис языка берет начало из C, Java и Perl. PHP достаточно прост для изучения. Преимуществом PHP является предоставление web-разработчикам возможности быстрого создания динамически генерируемых web-страниц.

Важным преимуществом языка PHP перед такими языками, как языков Perl и C заключается в возможности создания HTML документов с внедренными командами PHP.

Значительным отличием PHP от какого-либо кода, выполняющегося на стороне клиента, например, JavaScript, является то, что PHP-

скрипты выполняются на стороне сервера. Вы даже можете сконфигурировать свой сервер таким образом, чтобы HTML-файлы обрабатывались процессором PHP, так что клиенты даже не смогут узнать, получают ли они обычный HTML-файл или результат выполнения скрипта.

PHP позволяет создавать качественные Web-приложения за очень короткие сроки, получая продукты, легко модифицируемые и поддерживаемые в будущем.

PHP прост для освоения, и вместе с тем способен удовлетворить запросы профессиональных программистов.

Язык PHP постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков web -программирования, по крайней мере, в ближайшее время.

Возможности PHP

Главным образом, область применения PHP сфокусирована на написание скриптов, работающих на стороне сервера.

PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix (такие, как HP-UX, Solaris и OpenBSD), Microsoft Windows, Mac OS X, RISC OS, и многих других. Также в PHP включена поддержка большинства современных вебсерверов, таких, как Apache, Microsoft Internet Information Server, Personal Web Server, серверов Netscape и iPlanet, сервера Oreilly Website Pro, Caudium, Xitami, OmniHTTPd и многих других. Для большинства серверов PHP поставляется в качестве модуля, для других, поддерживающих стандарт CGI, PHP может функционировать в качестве процессора CGI.

Таким образом, выбирая PHP, вы получаете свободу выбора операционной системы и вебсервера. Кроме того, у вас появляется выбор между использованием процедурного или объектно-ориентированного программирования, или же их сочетания.

PHP способен не только выдавать HTML. Возможности PHP включают формирование изображений, файлов PDF и даже роликов Flash (с использованием libswf и Ming), создаваемых "на лету". PHP также способен выдавать любые текстовые данные, такие, как XHTML и другие XML-файлы. PHP способен осуществлять автоматическую генерацию таких файлов и сохранять их в файловой системе вашего сервера, вместо того, чтобы отдавать клиенту, организуя, таким образом, кеш динамического содержания, расположенный на стороне сервера.

Одним из значительных преимуществ PHP является поддержка широкого круга баз данных. Создание скрипта, использующего базы данных, - очень просто. В настоящее время PHP поддерживает следующие базы данных:

Преимущества PHP

Главным фактором языка PHP является практичность. PHP должен предоставить программисту средства для быстрого и эффективного решения поставленных задач. Практический характер PHP обусловлен пятью важными характеристиками:

- традиционностью;
- простотой;
- эффективностью;
- безопасностью;
- гибкостью.

Существует еще одна «характеристика», которая делает PHP особенно привлекательным: он распространяется бесплатно! Причем, с открытыми исходными кодами (Open Source).

Традиционность

Язык PHP будет казаться знакомым программистам, работающим в разных областях. Многие конструкции языка позаимствованы из Си, Perl.

Код PHP очень похож на тот, который встречается в типичных программах на C или Pascal. Это заметно снижает начальные усилия при изучении PHP. PHP — язык, сочетающий достоинства Perl и Си и специально нацеленный на работу в Интернете, язык с универсальным (правда, за некоторыми оговорками) и ясным синтаксисом.

И хотя PHP является довольно молодым языком, он обрел такую популярность среди web-программистов, что на данный момент является чуть ли не самым популярным языком для создания web-приложений (скриптов).

Простота

Сценарий PHP может состоять из 10 000 строк или из одной строки — все зависит от специфики вашей задачи. Вам не придется подгружать библиотеки, указывать специальные параметры компиляции или что-нибудь в этом роде. Механизм PHP просто начинает выполнять код после первой экранирующей последовательности (<?) и продолжает выполнение до того момента, когда он встретит парную

экранирующую последовательность (?>). Если код имеет правильный синтаксис, он исполняется в точности так, как указал программист.

PHP — язык, который может быть встроен непосредственно в html -код страниц, которые, в свою очередь будут корректно обрабатываться PHP -интерпретатором. Мы можем использовать PHP для написания CGI-сценариев и избавиться от множества неудобных операторов вывода текста. Мы можем привлекать PHP для формирования HTML-документов, избавившись от множества вызовов внешних сценариев.

Большое разнообразие функций PHP избавят вас от написания многострочных пользовательских функций на C или Pascal .

Эффективность

Эффективность является исключительно важным фактором при программировании для многопользовательских сред, к числу которых относится и web

Очень важное преимущество PHP заключается в его **«движке»**. «Движок» PHP не является ни компилятором, ни интерпретатором. Он является транслирующим интерпретатором. Такое устройство «движка» PHP позволяет обрабатывать сценарии с достаточно высокой скоростью.

По некоторым оценкам, большинство PHP-сценариев (особенно не очень больших размеров) обрабатываются быстрее аналогичных им программ, написанных на Perl. Однако, чтобы не делали разработчики PHP, откомпилированные исполняемые файлы будут работать значительно быстрее – в десятки, а иногда и в сотни раз. Но производительность PHP вполне достаточна для создания вполне серьезных web-приложений.

Безопасность

PHP предоставляет в распоряжение разработчиков и администраторов гибкие и эффективные средства безопасности, которые условно делятся на две категории: средства системного уровня и средства уровня приложения.

1. Средства безопасности системного уровня

В PHP реализованы механизмы безопасности, находящиеся под управлением администраторов; при правильной настройке PHP это обеспечивает максимальную свободу действий и безопасность. PHP

может работать в так называемом безопасном режиме (safe mode), который ограничивает возможности применения PHP пользователями по ряду важных показателей. Например, можно ограничить максимальное время выполнения и использование памяти (неконтролируемый расход памяти отрицательно влияет на быстродействие сервера). По аналогии с cgi-bin администратор также может устанавливать ограничения на каталоги, в которых пользователь может просматривать и исполнять сценарии PHP, а также использовать сценарии PHP для просмотра конфиденциальной информации на сервере (например, файла passwd).

2. Средства безопасности уровня приложения

В стандартный набор функций PHP входит ряд надежных механизмов шифрования. PHP также совместим с многими приложениями независимых фирм, что позволяет легко интегрировать его с защищенными технологиями электронной коммерции (e-commerce). Другое преимущество заключается в том, что исходный текст сценариев PHP нельзя просмотреть в браузере, поскольку сценарий компилируется до его отправки по запросу пользователя. Реализация PHP на стороне сервера предотвращает похищение нетривиальных сценариев пользователями, знаний которых хватает хотя бы для выполнения команды View Source.

Гибкость

Поскольку PHP является встраиваемым (embedded) языком, он отличается исключительной гибкостью по отношению к потребностям разработчика. Хотя PHP обычно рекомендуется использовать в сочетании с HTML, он с таким же успехом интегрируется и в JavaScript, WML, XML и другие языки. Кроме того, хорошо структурированные приложения PHP легко расширяются по мере необходимости (впрочем, это относится ко всем основным языкам программирования).

Нет проблем и с зависимостью от браузеров, поскольку перед отправкой клиенту сценарии PHP полностью компилируются на стороне сервера. В сущности, сценарии PHP могут передаваться любым устройствам с браузерами, включая сотовые телефоны, электронные записные книжки, пейджеры и портативные компьютеры, не говоря уже о традиционных ПК. Программисты, занимающиеся вспомогательными утилитами, могут запускать PHP в режиме командной строки.

Поскольку PHP не содержит кода, ориентированного на конкретный web-сервер, пользователи не ограничиваются определенными серверами (возможно, незнакомыми для них). Apache, Microsoft IIS, Netscape Enterprise Server, Stronghold и Zeus — PHP работает на всех перечисленных серверах. Поскольку эти серверы работают на разных платформах, PHP в целом является платформенно-независимым языком и существует на таких платформах, как UNIX, Solaris, FreeBSD и Windows 95/98/NT/2000/XP/2003.

Наконец, средства PHP позволяют программисту работать с внешними компонентами, такими как Enterprise Java Beans или COM-объекты Win32. Благодаря этим новым возможностям PHP занимает достойное место среди современных технологий и обеспечивает масштабирование проектов до необходимых пределов.

Бесплатное распространение

Стратегия Open Source, и распространение исходных текстов программ в массах, оказало несомненно благотворное влияние на многие проекты, в первую очередь — Linux, хотя и успех проекта Apache сильно подкрепил позиции сторонников Open Source. Сказанное относится и к истории создания PHP, поскольку поддержка пользователей со всего мира оказалась очень важным фактором в развитии проекта PHP.

Принятие стратегии Open Source и бесплатное распространение исходных текстов PHP оказало неоценимую услугу пользователям.

Adabas D	Ingres	Oracle (OCI7 и OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (только чтение)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

PHP также поддерживает "общение" с другими сервисами с использованием таких протоколов, как LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (на платформах Windows) и многих других. Кроме того, вы получаете возможность работать с сетевыми сокетами "напрямую". PHP поддерживает стандарт обмена сложными структурами данных WDDX. Обращая внимание на взаимодействие между различными языками, следует упомянуть о поддержке объектов Java и возможности их использования в качестве объектов PHP. Для доступа к удаленным объектам вы можете использовать расширение CORBA.

PHP включает средства обработки текстовой информации, начиная с регулярных выражений Perl или POSIX Extended и заканчивая парсером документов XML. Для парсинга XML используются стандарты SAX и DOM. Для преобразования документов XML вы можете использовать расширение XSLT.

Последним по порядку, но не по значению, является поддержка многих других расширений, таких, как функции поисковой машины mnoGoSearch, функции IRC Gateway, функции для работы со сжатыми файлами (gzip, bz2), функции календарных вычислений, функции перевода и многое другое.

Синтаксис PHP очень напоминает синтаксис языка C и во многом заимствован из таких языков как Java и Perl.

В принципе, в PHP есть практически все операторы и функции, имеющиеся в стандартном GNU C (или их аналоги), например есть циклы (while, for), операторы выбора (if, switch), функции работы с файловой системой и процессами (fopen, *dir, stat, unlink, popen, exec), функции ввода-вывода (fgets,fputs,printf) и множество других...

Синтаксис любого языка программирования гораздо легче "почувствовать" на примерах, нежели используя какие-то диаграммы и схемы. Поэтому приведем пример простейшего скрипта на PHP:

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

<?
echo "Привет, я - скрипт PHP!";
```

```
?>

</body>
</html>
```

Обратите внимание, что HTML-код корректно обрабатывается интерпретатором PHP.

Начало сценария вас может озадачить: разве это сценарий? Откуда HTML-тэги <html> и <body>? Вот тут-то и кроется главная особенность (кстати, чрезвычайно удобная) языка PHP: PHP-скрипт может вообще не отличаться от обычного HTML-документа.

Идем дальше. Вы, наверное, догадались, что сам код сценария начинается после открывающего тэга **<?** и заканчивается закрывающим **?>**. Итак, между этими двумя тэгами текст интерпретируется как программа, и в HTML-документ не попадает. Если же программе нужно что-то вывести, она должна воспользоваться оператором echo.

Итак, PHP устроен так, что любой текст, который расположен вне программных блоков, ограниченных **<?** и **?>**, выводится в браузер непосредственно. В этом и заключается главная особенность PHP, в отличие от Perl и C, где вывод осуществляется только с помощью стандартных операторов.

Разделение инструкций

Инструкции разделяются также, как и в C или Perl - каждое выражение заканчивается точкой с запятой.

Закрывающий тег (?>) также подразумевает конец инструкции, поэтому два следующих фрагмента кода эквиваленты:

```
<?php
    echo "Это тест";
?>

<?php echo "Это тест" ?>
```

Комментарии в PHP скриптах

Написание практически любого скрипта не обходится без комментариев.

PHP поддерживает комментарии в стиле 'C', 'C++' и оболочки Unix. Например:

```
<?php
    echo "Это тест"; // Это однострочный комментарий в стиле c++
    /* Это многострочный комментарий
       еще одна строка комментария */
    echo "Это еще один тест";
    echo "Последний тест"; # Это комментарий в стиле оболочки Unix
?>
```

Однострочные комментарии идут только до конца строки или текущего блока PHP-кода, в зависимости от того, что идет перед ними.

```
<h1>Это <?php # echo "простой";?> пример.</h1>
<p>Заголовок вверху выведет 'Это пример'.
```

Будьте внимательны, следите за отсутствием вложенных 'C'-комментариев, они могут появиться во время комментирования больших блоков:

```
<?php
  /*
    echo "Это тест"; /* Этот комментарий вызовет проблему */
  */
?>
```

Однострочные комментарии идут только до конца строки или текущего блока PHP-кода, в зависимости от того, что идет перед ними. Это означает, что HTML-код после // ?> БУДЕТ напечатан: ?> выводит из режима PHP и возвращает в режим HTML, но // не позволяет этого сделать.

Переменные в PHP

Имена переменных обозначаются знаком **$**. То же самое "Привет, я - скрипт PHP! " можно получить следующим образом:

```
<?php
$message = "Привет, я - скрипт PHP!";
echo $message;
?>
```

Типы данных в PHP

PHP поддерживает восемь простых типов данных:

Четыре скалярных типа:

- boolean (двоичные данные)
- integer (целые числа)
- float (числа с плавающей точкой или 'double')
- string (строки)

Два смешанных типа:
- array (массивы)
- object (объекты)

И два специальных типа:
resource (ресурсы)
NULL ("пустые")

Существуют также несколько псевдотипов:
- mixed (смешанные)
- number (числа)
- callback (обратного вызова)

Выражения в PHP

Основными формами выражений являются константы и переменные. Например, если вы записываете "$a = 100", вы присваиваете '100' переменной $a:

```
$a = 100;
```

В приведенном примере $a - это переменная, = - это оператор присваивания, а 100 - это и есть выражения. Его значение 100.

Выражением может быть и переменная, если ей сопоставлено определенное значение:

```
$x = 7;
$y = $x;
```

В первой строке рассмотренного примера выражением является константа 7, а во второй строке - переменная $x, т.к. ранее ей было присвоено значение 7. $y = $x также является выражением.

Операторы PHP

Оператором называется нечто, состоящее из одного или более значений (выражений, если говорить на жаргоне программирования), которое можно вычислить как новое значение (таким образом, вся конструкция может рассматриваться как выражение).

Таблица 13

Операторы PHP	
Синтаксис	Оператор
if (cond1) {...} **elseif** (cond2) {...} ... **else** {...}	Ветвление
while (cond) {...}	Цикл с предусловием (цикл ПОКА)
do {...} **while** (cond)	Цикл с постусловием (цикл ДО)
for (init; cond; expr) {...}	Цикл с предусловием
break	Безусловный переход на конец операторов while, do while, for и switch
continue	Безусловный переход на следующий цикл

Окончание табл. 11

Синтаксис	Оператор

switch (var) {
case val1: ...
case val2: ...
...
default: ...
}
	Выбор

function name(parameters) {...}	Определение функции
return val	Возврат значения

Примеры операторов PHP:
Операторы присвоения:

```
<?php

$a = ($b = 4) + 5; // результат: $a установлена значением 9, перемен-
ной $b присвоено 4.

?>
```

Комбинированные операторы:

```
<?php

$a = 3;
$a += 5; // устанавливает $a значением 8, аналогично за-
писи: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // устанавливает $b стро-
кой "Hello There!",  как и $b = $b . "There!";

?>
```

Строковые операторы:

```
<?php
$a = "Hello ";
$b = $a . "World!"; // $b содержит строку "Hello World!"

$a = "Hello ";
$a .= "World!";     // $a содержит строку "Hello World!"
?>
```

Существуют также логические операторы и операторы сравнения, однако их принято рассматривать в контексте управляющих конструкций языка.

Управляющие конструкции языка PHP

Основными конструкциями языка PHP являются:

1. Условные операторы (if, else);
2. Циклы (while, do-while, for, foreach, break, continue);
3. Конструкции выбора (switch);
4. Конструкции объявления (declare);
5. Конструкции возврата значений (return);
6. Конструкции включений (require, include).

Примеры конструкций языка PHP:

```
<?php
if ($a > $b) echo "значение a больше, чем b";
?>
```

Приведенный пример наглядно показывает использование конструкции **if** совместно с оператором сравнения ($a > $b).

В следующем примере если переменная *$a* не равна нулю, будет выведена строка "значение a истинно (true), то есть показано взаимодействие условного оператора (конструкции) if с логическим оператором:

```
<?php
if ($a) echo "значение a истинно (true) ";
?>
```

А вот пример цикла while:

```
<?php
$x=0;
while ($x++<10) echo $x;
// Выводит 12345678910
?>
```

Пользовательские функции в PHP

В любом языке программирования существуют подпрограммы. В языке C они называются функциями, в ассемблере - подпрограммами, а в Pascal существуют два вида подпрограмм: процедуры и функции.

В PHP такими подпрограммами являются пользовательские функции.

Подпрограмма - это специальным образом, оформленный фрагмент программы, к которому можно обратиться из любого места внутри программы. Подпрограммы существенно упрощают жизнь программистам, улучшая читабельность исходного кода, а также сокращая его, поскольку отдельные фрагменты кода не нужно писать несколько раз.

Приведем пример пользовательской функции на PHP:

```
<?php

function funct() {
$a = 100;
 echo "<h4>$a</h4>";
}
funct();

?>
```

Сценарий выводит 100:

100

Пользовательским функциям в PHP можно передавать аргументы и получать возвращаемые функциями значения.

Встроенные (стандартные) функции PHP

PHP содержит огромное количество встроенных функций, способных выполнять задачи различного уровня сложности.

Таблица 14

Встроенные функции PHP	
Функция	Описание
Математические функции	
mixed abs(mixed x)	Абсолютное значение аргумента x
int ceil(double x)	Наименьшее целое, большее x
int floor(double number)	Наибольшее целое, меньшее x

Продолжение табл. 2

Функция	Описание
double round(double x)	Округление

string decbin(int dec)	Преобразование десятичного числа в двоичное
string dechex(int dec)	Преобразование десятичного числа в шестнадцатеричное
string decoct(int dec)	Преобразование десятичного числа в восьмеричное
int bindec(string bin)	Преобразование двоичного числа в десятичное
int hexdec(string hex)	Преобразование шестнадцатеричного числа в десятичное
int octdec(string oct)	Преобразование восьмеричного числа в десятичное
int rand([int min, int max])	Получить случайное число
void srand(int seed)	Инициализировать генератор случайных чисел
int getrandmax()	Получить максимальное число, возвращаемое rand
int mt_rand([int min, int max])	Получить случайное число
void mt_srand(int seed)	Инициализировать генератор случайных чисел
int mt_getrandmax()	Максимальное число, возвращаемое mt_rand
double sqrt(double x)	Квадратный корень
double exp(double x)	Экспонента
double pow(double x, double y)	x^y
double log(double x)	Натуральный логарифм
double log10(double x)	Десятичный логарифм
double pi()	π

double cos(double x)	Косинус
double sin(double x)	Синус
double tan(double x)	Тангенс
double acos(double x)	Арккосинус
double asin(double x)	Арксинус
double atan(double x)	Арктангенс
double atan2(double y, double x)	Арктангенс y/x
Обработка строк	
void print(string str)	Вывести строку в стандартный поток вывода
int printf(string format, mixed arg, ...)	Форматированный вывод
string sprintf(string format, mixed arg, ...)	Форматировать строку

Продолжение табл. 2

Функция	Описание
string addslashes(string str)	Экранирование символов "$", "\", "'" и 0 в строке str
string stripslashes(string str)	Удалить экранирующие символы из строки
string chr(int ascii)	Получить символ с заданным кодом
int ord(string ch)	Код символа
string convert_cyr_string(	Перевод строки из одной русскоязычной кодировки в другую

string str, string from, string to)	
void parse_str(string str)	Разбить строку запроса и создать соответствующие переменные
Обработка массивов	
int count(mixed arr)	Число элементов массива
mixed current(array arr)	Текущий элемент массива
mixed pos(array arr)	Псевдоним current
array each(array arr)	Получить в виде массива пару ключ/значение текущего элемента arr
mixed key(array arr)	Ключ текущего элемента массива
mixed reset(array arr)	Установить внутренний указатель массива на начальный элемент
mixed prev(array arr)	Переместить указатель массива на предыдущий элемент
mixed next(array arr)	Переместить указатель массива на следующий элемент
mixed end(array arr)	Установить внутренний указатель массива на последний элемент
void sort(array arr)	Сортировка массива
void rsort(array arr)	Сортировка массива в обратном порядке
void asort(array arr)	Сортировка ассоциативного массива array
void arsort(array arr)	Сортировка ассоциативного массива arr в обратном порядке
int ksort(array arr)	Сортировка ассоциативного массива по ключам

Процессы	
string exec(string command [, array output [, int return_code]])	Выполнить команду command оболочки UNIX. Стандартный вывод будет записан в массив строк output, код возврата – в переменную result_code. Возвращается последняя строка стандартного потока вывода
string system(string cmd [, int ret])	Выполнить команду и возвратить результат
int chdir(string dir)	Изменить текущий рабочий каталог
void putenv(string str)	Установить значение переменной окружения

Продолжение табл. 2

Функция	Описание
string getenv(string var)	Получить значение переменной окружения
void sleep(int sec)	Задержка выполнения текущего процесса на sec секунд
void usleep(int us)	Задержка выполнения текущего процесса на us микросекунд
int sem_get(int key [, int max [, int perm]])	Получить идентификатор семафора
int sem_acquire(int sem)	Уменьшение счетчика семафора
int sem_release(int sem)	Увеличение счетчика семафора
Файлы и каталоги	
int copy(string src, string dest)	Копировать файл

int rename(string old, string new)	Переименовать файл
int unlink(string filename)	Удалить файл
int readfile(strinf filename)	Прочитать текстовый файл в стандартное устройство вывода
array file(string filename)	Прочитать текстовый файл в массив строк
int filesize(string filename)	Размер файла
int file_exists(string filename)	TRUE, если файл существует
int fopen(string file-name, string mode)	Открыть файл
string fgetc(int fd)	Прочитать символ из файла
string fgets(int fd, int maxlen)	Прочитать из файла строку максимальной длиной maxlen
int fputs(int fd, string str [, int length])	Записать строку в файл
string fread(int fp, int len)	Бинарное чтение файла
int fwrite(int fd, string str, int len)	Бинарная запись в файл
int ftell(int fd)	Текущая позиция указателя
int fseek(int fd, int off-set)	Переместить внутренний указатель файла

int rewind(int fd)	Переместить указатель файла на начало
int feof(int fd)	TRUE, если дескриптор fd указывает на конец файла
int fclose(int fd)	Закрыть файл
int popen(string command, string mode)	Запустить процесс command и возвратить дескриптор стандартного потока ввода или стандартного потока вывода созданного процесса

Продолжение табл. 2

Функция	Описание
int pclose(int fd)	Закрыть поток, открытый popen
int mkdir(string dir, int mode)	Создать директорию
int rmdir(string dir)	Удалить директорию
int opendir(string dir)	Открыть директорию
string readdir(int fd)	Получить имя очередного файла в каталоге
void rewinddir(int fd)	Переместить указатель каталога на первый файл
void closedir(int fd)	Закрыть директорию
void clearstatcache()	Очистить файловый кэш
flush()	Очистка буфера стандартного вывода
int fsockopen(string host, int port)	Открыть сокет
Дата и время	
int time()	Получить текущее время в формате UNIX
string date(string for-mat,	Строковое представление локальной даты и времени в указанном формате

int timestamp)	
string gmdate(string format, int timestamp)	Преобразовать дату и время GMT в формате UNIX в форматированную строку
array getdate(int timestamp)	Преобразовать дату и время в ассоциативный массив
int mktime(int hour, int minute, int second, int month, int day, int year)	Дата и время в формате UNIX
int gmmktime(int hour, int minute, int second, int month, int day, int year)	Дата и время GMT в формате UNIX
TCP/IP, HTTP, CGI, PHP	
string gethostbyname(string host)	Получить IP-адрес хоста
string gethostbyaddr(string addr)	Получить имя хоста по IP-адресу
int header(string hdr)	Отправить заголовок HTTP
string htmlspecialchars(string str)	Преобразовать специальные символы HTML в строке str в escape-последовательности
int isset()	TRUE, если переменная определена
void eval(string code)	Обработка code в качестве вложенного сценария
void exit()	Завершить текущий сценарий
Функция	Описание

int phpinfo()	Вывод информации о текущей реализации PHP
string phpversion()	Вывод информации о текущей версии PHP

ООП и PHP

PHP имеет достаточно хорошую поддержку объектно-ориентированного программирования (ООП).

В PHP можно создавать классы различных уровней, объекты и достаточно гибко ими оперировать.

Вот пример PHP класса и его использования:

```
<?php
// Создаем новый класс Coor:
class Coor {
// данные (свойства):
var $name;

// методы:
 function Getname() {
 echo "<h3>John</h3>";
 }

}

// Создаем объект класса Coor:
$object = new Coor;
// Получаем доступ к членам класса:
$object->name = "Alex";
echo $object->name;
// Выводит 'Alex'
// А теперь получим доступ к методу класса (фактически, к функции внутри класса):
$object->Getname();
// Выводит 'John' крупными буквами
?>
```

Вопросы для самоконтроля:

1. Опишите принцип работы PHP.
2. Каковы возможности языка PHP?
3. В чем заключаются преимущества языка PHP?
4. Опишите синтаксис языка программирования PHP.
5. Какие вы можете назвать управляющие конструкции языка PHP?

Тема 2. Взаимодействие с пользователем

Методы передачи параметров между страницами (GET, POST). Обработка действий пользователя при помощи форм. Использование вспомогательных переменных

4.2.1. Методы передачи параметров между страницами (GET, POST). Обработка действий пользователя при помощи форм.

Чтобы организовать передачу данных на сервер с помощью формы, потребуется реализовать HTML форму, в которую посетители сайта будут вводить свою информацию и PHP код, назначение которого в принятии и обработке полученных данных на сервере.

HTML форма отправки данных

Форма на странице формируется тегами <form>...</form>, внутри которых помещаются теги полей для ввода текстовой информации, теги специальных компонентов (например, поле со списком), теги для поля выбора и загрузки файла.

* Для HTML5 так же существует возможность размещать теги полей формы не внутри тегов формы, а в любом месте на странице. При этом для каждого такого поля нужно указывать атрибут "form", чтобы определить с какой формой отправки он должен взаимодействовать.

Итак, простейшая форма отправки может содержать следующий код:

```
<form action="myform.php" method="post">
    Значение А: <input type="text" name="data1">
    Значение Б: <input type="text" name="data2">
   <input type="submit" value="Отправить">
</form>
```

Элементы формы и их параметры:

action="myform.php" – атрибут "action" определяет, какой php-файл будет обрабатывать отправляемые данные. В этом примере, данные будут отправлены в файл "myform.php", находящийся в той же директории что и страница с формой. Если этот атрибут не указать явно, данные формы будут отправлены по адресу страницы самой формы.

method="post" – параметр method определяет метод передачи данных POST или GET. Более подробно об этом в статье "Отличия методов POST или GET". Если не указывать атрибут явно, по умолчанию будет использоваться метод GET.

Текст **"Значение А:"** и **"Значение Б:"** добавлен только с целью оформления и понятности формы для пользователя. Добавлять это для передачи данных не обязательно, но для того, чтобы пользователю стало понятно, что вводить, стоит указывать.

Теги **<input>** используются для формирования различных управляющих элементов формы.

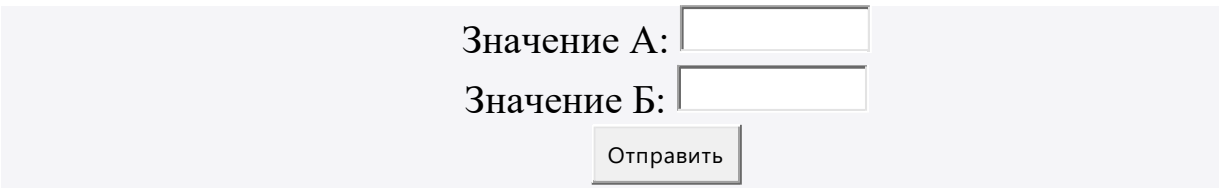
type="text" – атрибут "type" определяет вид поля. В зависимости от того, какой тип указан, меняется и внешний вид элемента, и его назначение. Значение атрибута "text" указывает, что в браузере элемент будет отображаться однострочным текстовым полем, куда пользователь сможет ввести свою строку.

name="data1" – атрибут "name", указывает имя, вернее индекс данных в массиве, полученных сервером. Это обязательный параметр, по которому в php-обработчике можно будет затем получить доступ переданному значению. Имя может быть выбрано произвольно, однако, удобнее когда это значение имеет какой-то понятный смысл.

type="submit" – тег <input> с таким значением параметра "type" будет отображаться на странице как кнопка. На самом деле на форме можно обойтись и без кнопки. Если, например, в форме есть текстовые поля, то отправку можно осуществить, просто нажав "Ввод" на клавиатуре. Но наличие кнопки делает форму более понятной.

value="Отправить" – в данном случае (для type="submit") определяет только надпись на кнопке. Для type="text", например, это будет текст, который будет выведен в текстовом поле.

В итоге, на странице этот код будет выглядеть приблизительно так:



При нажатии на кнопку, будет выполнена отправка данных на указанную страницу, и если она существует и корректно работает, данные будут обработаны.

Обработка отправленных HTML формой данных в PHP

Отправленные описанным способом данные, помещаются в суперглобальные массивы $_POST, $_GET и $_REQUEST. $_POST или $_GET будут содержать данные в зависимости от того, каким методом осуществлялась отправка. $_REQUEST содержит отправленные данные любым из указанных методов.

$_POST, $_GET и $_REQUEST – это ассоциативные массивы, поля-индексы которых совпадают с атрибутами "name" тегов <input>. Соответственно, для работы с данными в файле myform.php можно присвоить переменным значения элементов такого массива указав в качестве индекса имя поля:

```
// для метода GET
$a = $_GET['data1'];
$b = $_GET['data2'];

// для метода POST
$a = $_POST['data1'];
$b = $_POST['data2'];

// при любом методе
$a = $_REQUEST['data1'];
$b = $_REQUEST['data2'];
```

Проверка заполнения полей формы

Иногда при получении данных нужно проверить, не отправил ли пользователь пустую форму. Для этого можно использовать функцию empty.

```
if (empty($_REQUEST['data1'])) {
    echo 'Поле не заполнено';
} else {
    echo 'Поле было заполнено';
    $a = $_REQUEST['data1'];
}
```

Обычно этого решения достаточно. Если нужно вводить текст, то будет понятно, введен он или нет. Однако, если пользователь наме-

ренно для вычислений введет ноль, то функция empty покажет, что значения нет. Поэтому для таких ситуаций лучше использовать функцию isset. Она будет явно проверять, задано ли значение или нет.

```
if (isset($_REQUEST['data1'])) {
    echo 'Поле было заполнено';
    $a = $_REQUEST['data1'];
} else {
    echo 'Поле не заполнено';
}
```

Методы GET и POST используются для отправки данных HTML формы на сервер.

Для начала разберем метод GET. Это когда все переменные и их значения передаются прямо через адрес. Сейчас на примере вы все увидите, как работает большинство сайтов и форумов.

К примеру, есть у нас html страничка такого вида:

```
<html>
<head>
<title>Страница с примером передачи переменных с помощью Get</title>
</head>
<body>
<a href=http://myblaze.ru/index.php?name=Sergey&age=22>ссылка</a>
</body>
</html>
```

Видите ссылку? Она сложная и состоит из нескольких частей. Давайте разберем все по полочкам:

http://myblaze.ru — адрес домена или, как его еще называют, хост.

index.php — страница на php, которая будет обрабатывать запрос.

? — символ разделения между адресом и блоком с переменными.

Далее идут переменные и их значения, которые разделены символом *&*.

name=Sergey — переменная name и ее значение Sergey.

age=22 — то же самое, переменная age, значение 22.

Теперь посмотрим, как это обрабатывается в php, с помощью метода GET.

Страница index.php, как вы помните, мы передавали ей:

```
<?php

    if (!empty($_GET["name"])&&!empty($_GET["age"]))
    { echo " Получены новые вводные: имя - ".$_GET["name"].", возраст - ".$_GET["age"]." лет";}
    else { echo "Переменные не дошли. Проверьте все еще раз."; }

?>
```

Для демонстрации работы метода POST нам понадобится немного больше, чем простая строчка с адресом :) Нужно будет создать html страницу с формой для заполнения. Рассмотрим пример:

```
<html>
<head>
<title>Страница с примером передачи переменных с помощью Post</title>
</head>
<body>

<form method="post" action="index.php">Заполняем поля для передачи информации:<br><br>
 Укажите Ваше имя: <input name="user_name" type="text" maxlength="20" size="25" value="" />
<br><br> Укажите Ваш возраст: <input name="age" type="text" maxlength="2" size="3" value="" />
<br><br> <input type=submit value="Передать информацию"></form>
</body>
</html>
```

Итак, мы создали html страничку с простой формой. Запомните, метод POST может использоваться только в форме.

Первый параметр формы — «method», он определяет метод, который мы будем использовать для передачи. Как вы могли догадаться, это либо GET, либо POST. При этом, если установлен GET, то все имена полей (в виде названий переменных), а также их значения, передаются по ссылке, как в разделе про метод GET. Если же установлен POST, то все названия переменных и значения будут передаваться как запрос браузера к веб-серверу. То есть в адресной строке их видно не будет. Во многих случаях это очень полезно. Также POST безопаснее,

оно и понятно, ведь переменные с их значениями уже не так просто отредактировать, хотя тоже можно.

Второй параметр формы — «action». Это путь и имя файла скрипта, которому мы передаем данные. В нашем случае это index.php. Этот путь можно передавать и полностью, то есть так: action=«http://my_site.ru/index.php». Если не указать значение параметра «action», то вся информация будет передаваться главному скрипту, то есть индексной странице index.php вашего сайта, что вполне логично.

Теперь получим данные из нашей формы. Раз передавали мы в index.php, значит ниже будет код именно этой страницы:

```
<?php

 if (!empty($_POST["user_name"])&&!empty($_POST["age"]))
 {
 echo "Получены новые вводные:<br>";
 echo "имя - ";
 echo $_POST["user_name"];
 echo "<br>возраст - ";
 echo $_POST["age"];
 echo " лет";
 }
 else
 {
 echo "Переменные не дошли. Проверьте все еще раз.";
 }

?>
```

Не забываем проверять на пустоту и допустимые значения. Далее нужно уточнить, почему наши переменные называются именно user_name и age? А вы посмотрите на поля формы, которую мы создавали выше. Видите там *input name=«user_name» type=«text»* ? Вот здесь параметр name и задает имя переменной, которую мы получим с помощью этого поля. То же самое и с age. Надеюсь понятно. Ну а получение переменной и ее значения через POST почти не отличается от GET, который мы рассмотрели выше.

В целом оба метода выполняют аналогичную функцию – передают на сервер введенные в форме данные. Отличия определяются применением каждого из методов.
Например, для формы:

```
<form action="myform.php" method="post">
    <input type="text" name="data1">
    <input type="text" name="data2">
    <input type="submit" value="Отправить">
</form>
```

Если, в поля два текстовых поля формы ввести значения 15 и 20, то при выполнении **GET-запроса,** в адресной строке браузера будет явно виден url (адрес) страницы. Для такой формы это будет:

```
http://my_site.ru/myform.php?data1=15&data2=20
```

При выполнении передаче **методом POST** мы увидим лишь:

```
http://my_site.ru/myform.php
```

Никаких сведений о самих передаваемых данных здесь не увидеть.

Если, например, создается форма авторизации, то удобнее будет использовать POST запрос, т.к. в сохраненной браузером строке можно будет явно увидеть и логин и пароль.

GET же стоит использовать тогда, когда его результат можно полезно использовать для получения необходимой страницы повторно. Например, с параметрами необходимой сортировки или выборкой. Строку GET запроса можно увидеть в любом интернет поисковике.

Сохранив url, полученный методом GET, из адресной строки, можно всегда получить ту же страницу с уже подставленными данными, не заполняя форму отправки данных заново.

* Это не всегда срабатывает с поисковиками. Через некоторое время результат для сохраненной страницы будет меняться.

Если же форма будет отправляться методом POST, адрес полученной страницы будет всегда один, какие бы данные не вводились.

Объем передаваемой информации у этих методов тоже различен. С помощью GET лучше отправить небольшие тестовые данные. Максимальный объем здесь 4 Кб.

Для POST такого явного ограничения нет. Максимальный размер для него задается настройками сервера. Поэтому он подходит для загрузки файлов на сервер и передачи больших объемов текста.

4.2.2 Обработка действий пользователя при помощи форм.

Получение и обработка данных, введённых пользователем, стали неотъемлемой частью большинства успешных Web-сайтов. Бесспорно, возможность накопления статистики, проведения опросов, хранения персональных настроек и поиска выводят Web на принципиально новый уровень - без них эта среда обладала бы минимальной интерактивностью.

Ввод информации в основном реализуется с применением форм HTML. Как правило, пользователь заполняет в форме одно или несколько полей (например, имя и адрес электронной почты), нажимает кнопку отправки данных, после чего получает ответное сообщение.

При вводе данных в форму используются различные управляющие элементы. В одних элементах пользователь вводит информацию с клавиатуры, в других он выбирает нужный вариант, щёлкая кнопкой мыши. В формах могут присутствовать скрытые поля, которые поддерживаются самой формой; содержимое скрытых полей не должно изменяться пользователем.

Одна страница может содержать несколько форм, поэтому необходимы средства, которые позволили бы отличить одну форму от другой. Более того, вы должны как-то сообщить форме, куда следует перейти, когда пользователь выполняет действие с формой (как правило, нажимает кнопку отправки данных). Обе задачи решаются заключением форм в следующие теги HTML.

<form action ="действие" method="метод"></form>

Как видно из приведённого фрагмента, в тегах форм указывается два важных элемента: действие и метод. *Действие* указывает, какой сценарий должен обработать форму, а *метод* определяет способ передачи данных по этому сценарию. Существует два метода:

 Метод **get** передаёт все данные в конце URL. Из-за различных ограничений, связанных со спецификой языков и длиной данных этот метод применяется редко.
 Метод **post** передаёт все данные формы в теле запроса. Это метод используется чаще, чем **get**

Элементы форм, ориентированные на ввод с клавиатуры

Сейчас вам необходимо вспомнить, что построение форм начинается с элементов, ориентированных на ввод с клавиатуры. Таких элементов всего два - текстовое поле (**text box**) и текстовая область (**text area**).

В текстовых полях обычно водится короткая текстовая информация - скажем, адрес электронной почты, почтовый адрес или имя. Синтаксис определения текстового поля:

<input type="text" name="имя переменной" size="N"
maxlenght="N" value="">

Определение текстового поля включает пять атрибутов:

- type - тип элемента (для текстовых полей - **text**);
- name - тип переменной, в которой сохраняются введённой данные;
- size - общий размер текстового поля в браузере;
- maxlength - максимальное количество символов, вводимых в текстовом поле;
- value - значение, отображаемое в текстовом поле по умолчанию;

Особой разновидностью текстовых полей является поле для ввода паролей. Оно работает точно также, как обычное текстовое поле, однако вводимые символы заменяются звёздочками. Чтобы создать в форме поле для ввода паролей, достаточно указать

type="password" вместо type="text".

Введите следующий код HTML:

```
<html>
<head>
<title>Текстовое поле</title>
</head>
<body>
<form>
<label for = "Name">Ваше имя</label>
<input type = "text" name = "name" value = "Аноним"
size = "30" maxlenght = "20"><br>
<input type = "reset" value = "Очистить форму">
</form><br>
<form>
<label for = "Name">Введите пароль</label>
<input type = "password" size = "30" maxlenght = "10"><br>
<input type = "reset" value = "Сброс пароля">
</form>
</body>
</html>
```

Сохраните файл под именем **vvod_s_klaviatury.html**, откройте его в браузере. Если Вы всё сделали правильно, Вы увидите в окне браузера страничку, приведённую на рисунке 1. Протестируйте текстовые поля при вводе разных имён и паролей.

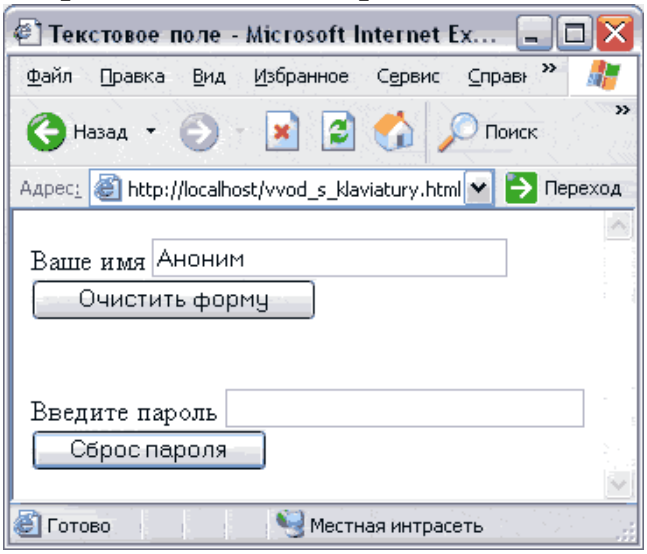


Рис. 1. Текстовые поля.

Текстовая область (text area) используется для ввода небольших объемов текста, не ограничивающихся простым именем или адресом электронной почты, а например, для ввода текстового сообщения. Синтаксис определения текстовой области следующий:

<textarea name="*имя_переменной*" rows="*N*" cols="*N*" value=""></textarea>

Определение текстового поля включает три основных атрибута:

- name - имя переменной, в которой сохраняются введённые данные;
- rows - количество строк в текстовой области;

- cols - количество столбцов (букв) в текстовой области.

Введите следующий текст HTML:

```
<html>
<head>
<title>Текстовое поле</title>
</head>
<body>
<form>
<label for = "Name">Ваше имя</label>
<input type = "text" name = "name" value = "Аноним"
size = "30" maxlenght = "20"><br>
<input type = "reset" value = "Очистить форму">
</form><br>
<form>
<label for = "Name">Введите пароль</label>
<input type = "password" size = "30" maxlenght = "10"><br>
<input type = "reset" value = "Сброс пароля">
</form><br>
<label for = "name">Для сообщений:</label><br>
<textarea name = "message" rows="5" cols = "50"></textarea>
</body>
</html>
```

Сохраните файл под именем **text_oblast.html**, откройте его в браузере. Если Вы всё сделали правильно, Вы увидите в окне браузера страничку, приведённую на рисунке 2, в которой к полям для ввода имени и пароля добавилось поле ввода сообщений.

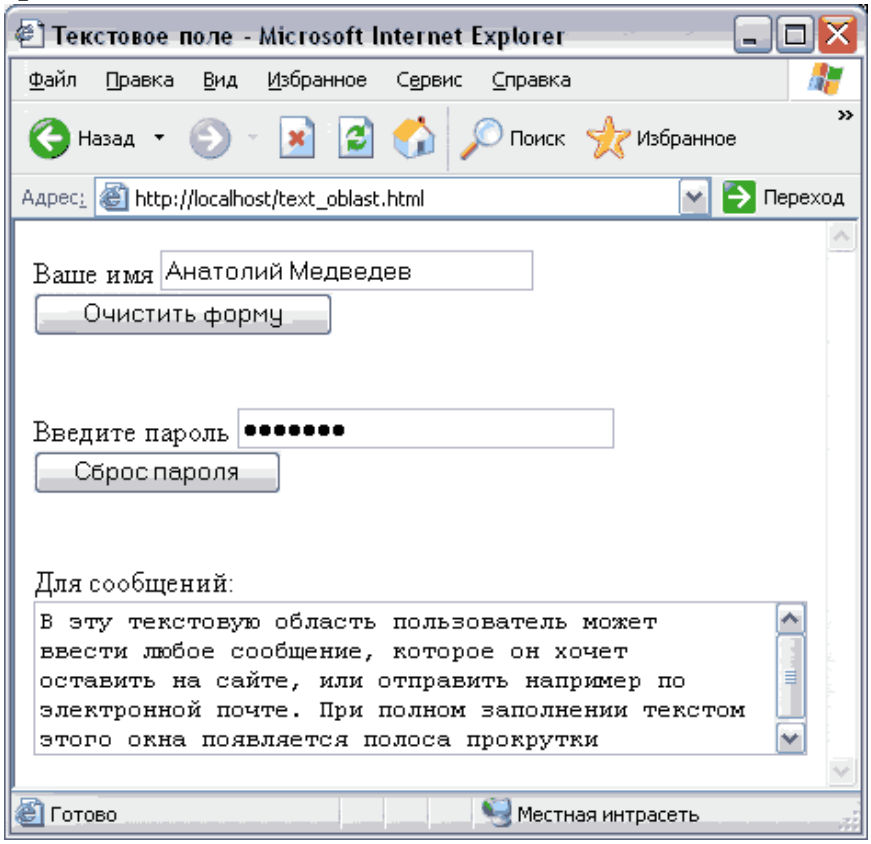


Рис. . Текстовая область.

В других элементах форм пользователь выбирает один из заранее определённых вариантов при помощи мыши. Ограничимся описанием флажков, переключателей и раскрывающихся списков.

*Флажки (**checkboxes**)* используются в ситуациях, когда пользователь выбирает один или несколько вариантов из готового набора - по аналогии с тем, как ставятся "галочки" в анкетах.

Синтаксис определения флажка:

<input type="cheskbox" name=*"имя_переменной"* value=*"начальное_значение"*>

Определение флажка включает три атрибута:

- type - тип элемента (для флажков - **checkbox**);
- name - имя переменной, в которой сохраняются введённые данные (в данном случае - состояние элемента);
- value - значение, присваиваемое переменной по умолча-нию. Если флажок установлен, именно это значение будет присвоено переменной с указанным именем. Если флажок не установлен, значение атрибута value не используется.

Введите следующий текст HTML:

```
<html>
<head>
<title>Флажки</title>
</head>
<body>
<label for = "interests1">Ваши интересы:</label><br>
<input type = "checkbox" name = "interests1" value = "computers">Компьютеры<br>
<input type = "checkbox" name = "interests2" value = "sport">Спорт<br>
<input type = "checkbox" name = "interests3" value = "art">Искусство<br>
<input type = "checkbox" name = "interests4" value = "science">Наука<br>
<input type = "checkbox" name = "interests5" value = "auto">Автомобили<br>
</body>
</html>
```

Сохраните файл под именем **flagi.html**, откройте его в браузере. Если Вы всё сделали правильно, Вы увидите в окне браузера страничку, приведённую на рисунке 3, в которой организован список из нескольких пунктов. Пользователь может отметить один, несколько, или даже все пункты флажками.

В данном случае для каждого пункта установлены разные значения атрибута **Name=**, но они могут быть и одинаковыми.

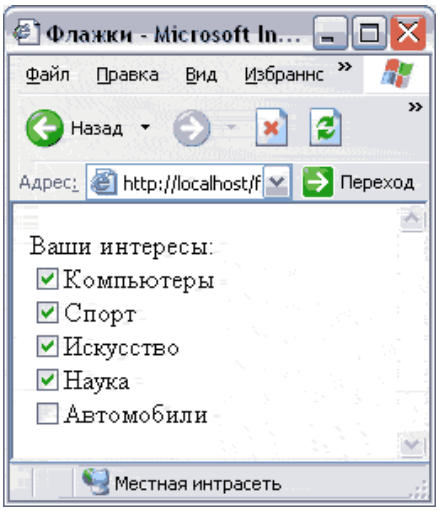


Рис. 2.3. Флажок.

*Переключатель (**radio buton**)* представляет собой разновидность флажка; он работает практически так же за одним исключением - в любой момент времени в группе может быть установлен лишь один переключатель.

Синтаксис определения переключателя:

<input type="radio" name=*"имя_переменной"* value=*"начальное_значение"*>

Как видите, синтаксис определения переключателя почти не отличается от синтаксиса определения флажка.

Определение переключателя поля включает три атрибута:

- type - тип элемента (для переключателей - radio);
- name - имя переменной, в которой сохраняются введённые данные (в данном случае - состояние элемента);
- value - значение, присваиваемое переменной по умолчанию. Если переключатель установлен, именно это значение будет присвоено переменной с указанным именем. Если переключатель не установлен, значение атрибута value не используется.

Введите следующий текст HTML:

```
<html>
<head>
<title>Переключатели</title>
</head>
<body>
<label for="Sex">Ваш пол:</label><br>
<input type="radio" Name="Sex" value="M">Мужской.<br>
<input type="radio" Name="Sex" value="F">Женский.<br><br>
<label for="Age">Ваш возраст:</label><br>
<input type="radio" Name="Age" value="10">менее 10 лет<br>
<input type="radio" Name="Age" value="10">20&ndash;30 лет<br>
<input type="radio" Name="Age" value="10">30&ndash;40 лет<br>
<input type="radio" Name="Age" value="10">40&ndash;50 лет<br>
<input type="radio" Name="Age" value="10">50&ndash;60 лет<br>
</body>
</html>
```

Сохраните файл под именем **switsh.html**, откройте его в браузере. Вы увидите страничку, приведённую на рисунке 4, в которой организованы два переключателя.

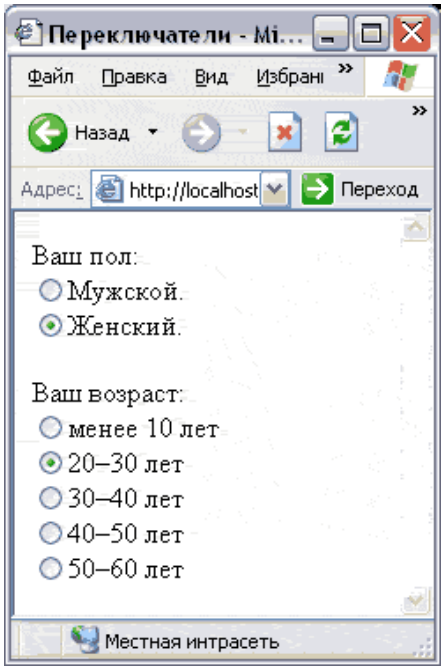


Рис. 4. Переключатели.

Каждый из переключателей создается с помощью тега **<input>** с атрибутом **type="radio"**, однако в одну группу объединяются те переключатели, теги которых имеют одинаковые значения атрибута **name=**.

Раскрывающиеся списки особенно удобны в ситуации, когда у Вас имеется длинный перечень допустимых вариантов, из которых

пользователь должен выбрать один вариант. Как правило, раскрывающиеся списки применяются при работе с относительно большими наборами данных - например, при перечислении областей или стран.

Синтаксис определения раскрывающегося списка:

<select name=*"имя_переменной"*>
<option value=*"имя_переменной1"*>
<option value=*"имя_переменной2"*>
"option value=*"имя_переменной3"*>
.....
<option value=*"имя_переменнойN"*>
</select>

Определение раскрывающегося списка включает два атрибута:

- name - имя переменной, в которой сохраняются введённые данные (в данном случае - строка, выбранная в списке);
- value - значение, отображаемое в списке по умолчанию.

Введите следующий текст HTML:

```
<html>

<head>
  <title>Раскрывающиеся списки</title>
</head>

<body>
<select name="SP">
<option value="SP1">Базы данных
<option value="SP2">Высшая математика
<option value="SP3">Компьютерная графика
<option value="SP4">Операционные системы и среды
<option value="SP5">Численные методы
<option value="SP6">Дискретная математика
<option value="SP7">Алгоритмизация и программирование
</select><br><br><br>

<select name="SPP">
<option value="SPP1">1-й урок: 09.00 - 09.45
<option value="SPP2">2-й урок: 09.50 - 10.35
<option value="SPP3">3-й урок: 10.45 - 11.30
<option value="SPP4">4-й урок: 11.35 - 12.20
<option value="SPP5">5-й урок: 12.30 - 13.35
</select>
</body>

</html>
```

Сохраните файл под именем **spiski.html**, откройте его в браузере. Вы увидите страничку, приведённую на рисунке 5, в которой организованы два раскрывающихся списка.

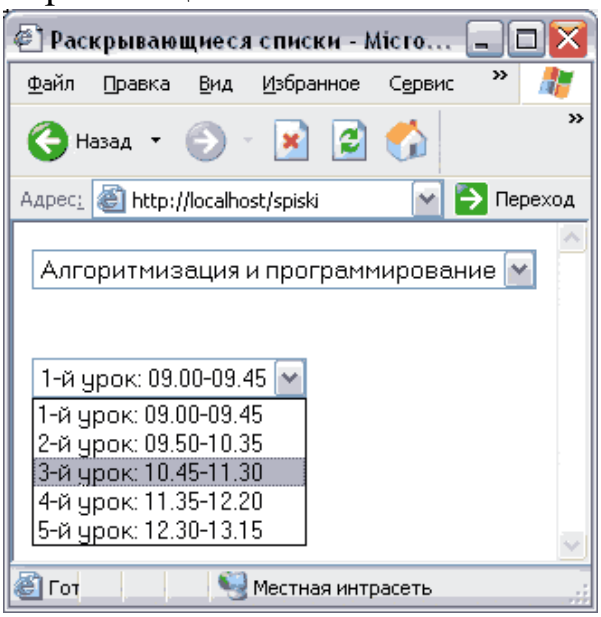


Рис. 5. Раскрывающиеся списки.

Скрытые поля не отображаются в браузере и обычно используются для передачи данных между сценариями. Хотя передача в скрытых полях работает вполне нормально, в PHP существует другое, более удобное средство - сеансовые переменные, но об этом средстве Вы узнаете несколько позже. Впрочем, скрытые поля также используются в некоторых ситуациях и потому заслуживают внимания.

Синтаксис определения скрытого поля практически идентичен синтаксису текстовых полей, отличается только атрибут поля. Поскольку текстовые поля не отображаются в браузере, привести пример на страницах методического пособия невозможно.

Синтаксис определения скрытого поля:

<input type="hidden" name="имя_переменной" value="начальное_значение">

Определение скрытого поля включает три атрибута:

- type - тип элемента (для скрытых полей - **hidden**);
- name - имя переменной, в которой сохраняются скрытые данные;
- value - значение, по умолчанию сохраняемое в скрытом поле.

Вообще говоря, название этого элемента - скрытое поле - несколько неточно. Хотя скрытые поля не отображаются в браузерах, пользователь может просто выполнить команду **View Source** и увидеть, какие скрытые значения хранятся в форме.

Кнопка отправки формы инициирует действие, заданное атрибутом **action** тега **<form>**.

Синтаксис определения кнопки отправки формы:

<input type="submit" value="текст на кнопке">

Определение кнопки включает два атрибута:

- type - тип элемента (для кнопки сброса - **submit**);
- value - текст, по умолчанию отображаемый на кнопке.

Кнопка сброса отменяет все изменения, внесённые в элемент формы.

Синтаксис определения кнопки сброса:

<input type="reset" value="текст на кнопке">

Определение кнопки сброса включает два атрибута:

- type - тип элемента (для кнопки сброса - **reset**);
- value - текст, по умолчанию отображаемый на кнопке.

Кнопка сброса выглядит точно также, как и кнопка отправки данных, если не считать того, что на ней обычно выводится слово Reset.

Вопросы для самоконтроля:

1.Опишите методы передачи параметров между страницами.

2.Как происходит обработка действий пользователя при помощи форм?

3.Каким образом происходит использование вспомогательных переменных?

Тема 3. База данных в MySQL

Варианты хранения информации в сети Internet. Принципы хранения информации в базах данных MySQL. Архитектура базы данных MySQL (таблицы, связи, триггеры).
Проектирование баз данных. Нормализация таблиц

4.3.1. Варианты хранения информации в сети Internet.

Интернет сегодня представляет собой совокупность соединенных между собой информационных серверов – компьютеров, на которых хранится различная информация, и самих пользователей информации. В основном эта информация доступна через технологию «мировая паутина» *World Wide Web* (сокращенно - *WWW* или *WEB).* Такое название эта технология получила потому, что каждый пользователь может свободно переключаться от одного сервера к другому независимо от географического местоположения сервера, как бы «опутывая» Землю паутиной своих переходов.

1. ***Сети хранения данных.*** Основной движущей силой, стимулирующей развитие сетей хранения данных, является непрекращающийся рост объемов информации, к которой требуется высокоскоростной доступ. Современные технологии хранения данных позволяют объединить отдельные массивы в высокопроизводительные сети.

Различают три основных типа обеспечения доступа к данным систем хранения:

- SAS (Server Attached Storage) – массив, подключенный к серверу;
- NAS (Network Attached Storage) – массив, присоединенный к сети;
- SAN (Storage Area Network) – группа массивов, объединенных в сеть.

Существуют разнообразные вариации каждого вида, которые имеют свои плюсы и минусы относительно использования в определенных условиях.

Технология SAS – традиционная система хранения данных, присоединенная к серверу. Основное преимущество – низкая цена и простота организации. Особенности использования:

- зависимость от операционной системы сервера и файловой системы;
- уменьшение скорости отклика при высокой загрузке сервера;
- ограниченные возможности при формировании сложных систем.

Технология NAS отличается улучшенной архитектурой файл-сервера, это идеальный вариант для организации работы серверов с минимальными функциями. К преимуществам относят независимость от операционной системы рабочих станций и сервера, простые администрирование и установка. Из минусов – конфликты с трафиком локальной и беспроводной сети.

Сеть хранения данных SAN — это система, которая позволяет организовать распределенный доступ к устройствам хранения данных между серверами и рабочими станциями, независимая от локальной и беспроводной сети.

Основой данной сети является протокол Fibre Channel. Передача данных межу серверами-коммутаторами осуществляется при помощи оптических каналов связи. Это позволяет обеспечить высокую готовность систем для станций с большой интенсивностью запросов.

Более прогрессивная и функциональная технология SAN все чаще вытесняет традиционные, и позволяет во много раз увеличить производительность и надежность системы хранения данных. К недостаткам, пожалуй, можно отнести только высокую стоимость системы.

Основные преимущества следующие:

- независимость от технологии от систем хранения данных и серверов;
- централизованное управление сетью;
- быстродействие;
- отсутствие конфликтов с локальными сетями;
- гибкость и вариативность;
- высокая отказоустойчивость.

Среди топологий сети хранения данных SAN различают: одно-коммутаторную структуру, каскадную или дерево, решётку, кольцо, центрально-распределённую. Топологии отличает количество ISL соединений.

В зависимости от требований, предъявляемых пользователями сети, выбор и разработка системы хранения данных может отличаться.

2. Облачные хранилища. Постараемся описать несколько вариантов, популярных сервисов, предоставляющих "Облачное хранилище".

Итак, что же такое это "Облачное хранилище"? По сути, это способ для резервного копирования ваших самых важных файлов (фотографий, видеороликов, документов, программ и пр.). К примеру, раньше вы собираясь пойти к друзьям в гости, брали с собой фотоальбомы или скажем видеозаписи на CD дисках, с появлением компьютеров и компактных накопителей (флешек, карт памяти и пр.) брали их. А теперь представьте, что сейчас достаточно лишь иметь доступ к виртуальному хранилищу вы сможете показать любые файлы кому угодно, даже с чужого компьютера, или скажем планшета.

Принцип, который используют эти сервисы сводится к *синхронизации данных* на различных устройствах. Файлы в этом случае не привязаны к одному компьютеру или телефону. Можно начать работать с документом дома, а продолжить работать с ним уже на работе с того же места, на котором остановились дома. Легкость использования данными сервисами заключается в том, что достаточно например, работая на локальном компьютере, перетянуть редактируемый файл в предназначенную для этого папку, и продолжить работу с ним уже на другом устройстве через веб-интерфейс программы, обратившись к той же папке. Будут сохранены все вносимые изменения автоматически. Это и есть синхронизация.

Также стоит отметить что польза подобных хранилищ очень велика еще и из-за того, что самые крупные из них предоставляют возможность использования на мобильных устройствах, таких как смартфоны, планшеты и мобильные телефоны. Установив программу клиент на своем компьютере и загрузив необходимые для хранения или демонстрации файлы, вы сможете также видеть и управлять ими на мобильных устройствах. Т.е. самое необходимое теперь может быть всегда с вами, даже не зависимо от того есть у вас с собой телефон или нет. Ведь даже оставив телефон дома (случайно забыв его, с кем не бывает), вы сможете, придя на работу получить доступ к необходимой информации и продолжить работу с ней.

Одним из первых таких сервисов **стал "DropBox"**. В нем предоставляется изначально 3Gb свободного пространства для хранения ваших файлов. В дальнейшем его можно расширить приглашая друзей, подписываясь на различные акции, используя его на мобильных устройствах или попросту купить премиум пакет.

Dropbox работает практически на любой операционной системе, что очень удобно если у вас есть несколько устройств от разных производителей.

Поддерживаемые операционные системы:

- Windows - XP, Vista, Win7, Win8, Win8.1
- Apple - MacOSX, iOS
- Linux - поддерживаются наиболее популярные дистрибутивы, такие как Ubuntu, Linux Mint и пр.
- Android - версии начиная с 2.2 (возможна работа и на более ранних, но качество работы не гарантируется)
- и др.

**Также есть возможность управления файлами через браузер.*

Стоит отметить что качество и скорость синхронизации на высоте, даже если вы оперируете большим количеством файлов маленького объема, например, бухгалтерские отчеты, файлы драйверов, или любые другие. Можно использовать одновременно на нескольких компьютерах, предоставлять общий доступ другим пользователям, с мобильных устройств можно воспроизводить медиафайлы не загружая их.

Кроме того, что DropBox на самых популярных операционных системах, его разработчики позаботились об удобстве пользователей и сделали доступным практически в любой стране мира. А также перевели на несколько языков, для того чтобы вы могли делиться своими файлами с людьми любой национальности.

"Облачное хранилище" SkyDrive это банк данных от компании Microsoft, который, пожалуй наиболее качественно интегрирован с операционными системами семейства Windows. Мало того, в последней на данный момент версии Windows 8.1 уже установлена программа для синхронизации с сервером SkyDrive. Для его использования пользователю достаточно лишь пройти простую регистрацию.

Изначально, после регистрации, пользователю предоставляется довольно крупный объем хранилища (7 Гб.), который можно расширить пригласив друзей. Всего бесплатно можно увеличить размер хранилища до 12 Гб., по 500 Мб. за каждого зарегистрировавшегося пользователя. Мало того, SkyDrive добавляет 500 Мб. не только вам, но и пользователю которого вы пригласили. Если вам недостаточно стартового объема и вы не хотите тратить свое время на приглашение друзей и знакомых, то у компании Microsoft можно приобрести различные пакеты которые существенно увеличат объем вашего виртуального хранилища.

Поддерживаемые операционные системы:

- Andriod - версии начиная с 2.2 (возможна работа и на более ранних, но качество работы не гарантируется)
- Windows - XP, Vista, Win7, Win8, Win8.1, а также Windows для мобильных устройств
- Apple - MacOSX, iOS
- для остальных операционных систем доступна веб версия, т.е. работа через интернет браузер

А теперь хотелось бы отметить что качество работы сервиса, интерфейса и собственно функциональность выполнена на высочайшем уровне, как впрочем, большинство продуктов от Microsoft. Особенно приятно и удобно то что Microsoft интегрировала данный "Виртуальный облачный диск" в свою последнюю версию операционной

системы. Впрочем, скорее всего все последующие версии, как мобильных, так и настольных ОС будут оснащены этим замечательным сервисом.

Следующим сервисом мы рассмотрим продукт от всемирно известной компании Google, который носит название **Google Drive** или **Google Диск**.

Достаточно неплохой сервис для хранения файлов в котором предоставляется 15Gb, правда этот объем распределяется по всему аккаунту, т.е на почту Gmail, социальную сеть Google +, Google фото и прочие приложения от Google.

Поддерживаемые операционные системы:

- Andriod - версии начиная с 2.2 (возможна работа и на более ранних, но качество работы не гарантируется)
- Windows - XP, Vista, Win7, Win8, Win8.1, а также Windows для мобильных устройств
- Apple - MacOSX, iOS
- Linux - для более точной информации смотрите на сайте вашей операционной системы

Из приятных и удобных компонентов стоит отметить приложение Google Docs, который позволяет просматривать и редактировать документы используя интернет браузер. Т.е. вы можете, к примеру, создать таблицу (наподобии MS Exel) непосредственно через ваш интернет браузер и предоставить доступ другим пользователям. Причем можно как ограничить права на управление и редактирование документа, так и дать только просмотр. Эта функция очень полезна для тех пользователей которые регулярно заполняют какие-либо отчеты (финансового, номинального и пр. характера) и необходимо чтобы руководство могло контролировать продажи или остатки на складах. Конечно можно использовать обычный Microsoft Office с его таблицами, документами и презентациями, но тогда придется регулярно пересылать такой отчет по почте, или печатать. С документами от Google вы

будете экономить как время, так и деньги на покупку офисных программ. Основной функционал таблиц практически не отличается.

Еще одним приятным дополнением является автосинхронизация с аккаунтом Google и всеми устройствами на операционной системе Android. В принципе данной функцией сейчас сложно удивить даже самого заурядного пользователя, но с другой стороны, стоит отметить что синхронизация приложения Google диск создана непосредственно первоисточником операционной системы, т.е. производителем. А производитель, как правило, наилучшим образом оптимизирует свое программное обеспечение под свою же операционную систему.

Качество синхронизации, т.е. загрузки и выгрузки файлов выполнено на достаточно высоком уровне. В сравнении с предыдущим сервисом Dropbox, Google диск способен выполнять все те же функции, но с некоторыми приятными дополнениями. Из некоторых недостатков (опять же в сравнении с Dropbox) можно отметить не такое высокое качество исполнения при работе с большим количеством мелких файлов, а также при работе с офисными документами в формате XLS и ему подобными. Иногда возникает ситуация когда документ попросту не способен осуществить сохранение после внесения в него корректировок. Такое случается если этот же документ открыт на другом компьютере и также в данный момент редактируется.

4.3.2 Принципы хранения информации в базах данных MySQL.

MS— это система управления контентом/содержимым сайта. Под контентом и содержимым сайта понимают: текст, картинки, видео – данные и файлы.

CMS— это система, которая позволяет в удобном виде создавать и управлять (редактировать, удалять) текстовыми материалами и мультимедиа документами (содержимое или контент) на сайте. Аббревиатура «CMS» появилась от англ. фразы **Content Management System**, что и переводится как система управления контентом.

Для того, чтобы добавить новую статью на ваш сайт нужно создавать новый файл html. Чтобы отредактировать статью, нужно открывать html файл и искать в нем тот фрагмент, который нуждается в изменении. На эту работу уходит много времени и необходимо знание html и css. Чтобы облегчить эту работу были придуманы системы, которые позволяли всего один раз создать дизайн сайта и, если нужно

написать/отредактировать статью, совсем не нужно создавать/лезть в html файлы.

Таким образом, в CMS внутренняя структура и дизайн отделены от контента, и, чтобы управлять сайтом, не нужно каких-то дополнительных знаний в технологиях интернет-разработки.

Все системы управления контентом разделяются на *платные и бесплатные.*

К примеру, этот блог создан на популярном движке WordPress, который распространяется бесплатно. Из бесплатных CMS еще можно назвать: Joomla, Drupal, 2z-project и др.

Из платных распространенные CMS – это DLE (Data Life Engine), которая больше подходит для развлекательных сайтов, UMI.CMS, NetCat и другие. Также сайт помогает подобрать CMS для своего сайта.

Разработчики платных CMS часто защищают свои скрипты специальным кодом, который называют **звонилками**. Звонилки сообщают разработчику на каком сайте установлена их система, лицензионная ли она или нет. Это им помогает пресечь бесплатное использование коммерческого движка. Если вы увидите рядом с названием CMS слово **null**(нулл) или **nulled**– это означает, что система была как бы «крякнута», т.е. все звонилки убраны из движка и почти все функции лицензионной версии будут доступны в бесплатном варианте.

Отличие нелицензионной версии от лицензионной в том, что для null не будет поддержки и обновлений от разработчика CMS. А также никто не даст вам гарантий на то, что система будет безопасна, и что сайт не увидят разработчики и попытаются его закрыть, предварительно написав жалобу с предложением вам перейти на лицензионную версию своего продукта.

Принцип работы любого движка прост. Пользователь системы добавляет контент на сайт. Вся информация, которую ввел пользователь, сохраняется в базе данных или файлах. Когда посетитель заходит на сайт, информация читается из базы данных и отображается на сайте. Вид отображения информации зависит от шаблона.

Шаблон сайта— это заготовка дизайна сайта, без наполнения её информацией. Почти во всех CMS шаблоны сайта легко меняются. И вы можете подобрать для себя понравившийся шаблон или сверстать его самостоятельно.

Во многих движках есть система модулей. То есть, функционал системы можно расширить, подключая дополнительные модули. Например, модуль «Чат» или модуль «Обратная связь» и т.д. Модули часто называют плагинами, расширениями или дополнениями.

Пишутся CMS чаще всего на одном из серверных языков программирования (PHP, Perl и др.).

Таким образом,

- ***CMS или движок сайта***– это система управления сайтом, которая позволяет пользователю эффективно управлять содержимым сайта без дополнительных навыков интернет-разработки.
- Бывают платные и бесплатные CMS. Выбирать движок нужно исходя из требований к сайту.
- Хранение информации происходит в базе данных (чаще всего в MySQL) или в файлах (txt либо других). Для работы многих CMS нужен особый хостинг. Если движок написан на языке PHP и требует базу MySQL для работы, то нужно, чтобы хостинг включал в себя эти функции.

База данных (**БД**) вещь крайне полезная для ведения своего сайта. Работать со статическим проектом, состоящим из нескольких html страничек легко и без применения баз данных. Однако сайты имеют тенденцию разрастаться. С динамичным проектом такой метод работы уже вряд ли себя оправдает. Хранить массивы различной информации в сотнях файлов, а затем требовать от них определенные строки при работе вэб-сервера – дело хлопотное и медленное. БД позволяют структурировать и систематизировать информацию. Код для использования БД намного легче, чем аналогичный для работы с файлами, да и времени на запрос уходит куда меньше.

База данных*(database) – это совокупность связанных между собой таблиц. Например, в одной таблице может храниться информация о пользователе, зарегистрированном на сайте, а в другой – информация о комментариях, которые оставил пользователь на сайте.*

Наилучшее решение — хранить информацию типа списков, комментариев и т.д. в БД. Однако база данных далеко не всегда статичное образование, чаще наоборот, она регулярно пополняться и корректируется. Для легкости управления этими БД, изменения и добавления данных существуют системы управления базами данных (**СУБД**).

Одна из самых популярных СУБД в современных интернет-технологиях, бесспорно, **MySQL**.

Web-мастера даже придумали альтернативное название MySQL – «мускул». Поэтому, если вы когда-нибудь услышите выражение «движок на мускуле», это означает, что CMS использует базу данных MySQL.

Для примера можно взять Facebook. Данные хранятся там именно в MySQL. MySQL оптимизирована для получения данных из БД. Практически на всех сайтах мы в основном читаем данные из БД, прежде чем отправить их на страницу.

К основным плюсам MySQL можно отнести высокую скорость работы, быстроту обработки данных и оптимальную надежность. Немаловажно и то, что данная СУБД распространяется бесплатно и представляет собой программное обеспечение с открытым кодом. За счет этого Вы можете вносить свои изменения и модифицировать код, что весьма полезно для веб-мастеров.

MySQL– это один из множества ПО для работы с SQL базами данных.

SQL – это структурированный язык запросов, созданный для управления реляционными БД. Он обладает широким перечнем возможностей, например, создать таблицу, редактировать и удалять данные, производить запросы из таблиц и многое другое.

Как же хранятся данные в реляционных базах данных?

Представьте себе самую простую таблицу имен, номеров телефонов, адресов и т.д. Именно так и хранятся данные реляционных БД – в таблице, организуются посредствам столбцов и строк. Каждому столбцу присвоено имя, которое отображается в названии, все значения в этом столбце принадлежат к переменным только одного типа. Столбцы расположены в определенном строгом порядке, в то время как строки неупорядочены. Зачастую данные некоторых ячеек в одной таблице связаны со значением ячеек другой таблицы и так далее. Запросы к БД возвращают результат в виде таблицы.

Данные в БД делятся на уникальные или неуникальные. Неуникальные – это имя, год рождения, время и т.д., в то время, как *уникальные* – номер кредитки, договора хостинг-услуг. Уникальные значения присутствуют в списках так называемого «уникального индекса».

Большим достоинством MySQL является возможность работы с интерфейсом программного приложения API (Application Program Interface). API может обеспечить простой доступ из программы пользователя к СУБД. Пусть даже эти программы будут написаны на на Perl, C и т.д.

Самой популярной «связкой» для управления сайтами считается MySQL с языком PHP. Многие CMS написаны на PHP в связке с БД MySQL. Одним из самых ярких примеров данного «союза» может служить движок для сайтов и блогов WordPress, завоевавший огромную популярность в мире. Взаимодействие с MySQL в данном случае ведется посредством совокупности функций. Примером такой функции может служить «mysql_connect», которая соединяется с сервером БД и возвращает дескриптор соединения с ней.

Существует множество СУБД поддерживающих SQL язык запросов: *MySQL, mSQL, PostgreSQL, MSSQL*и многие другие. Каждая из них имеет преимущества в определенной сфере. И все же именно MySQL завоевала широкое признание и популярность в Интернете благодаря своей гибкости и универсальности.

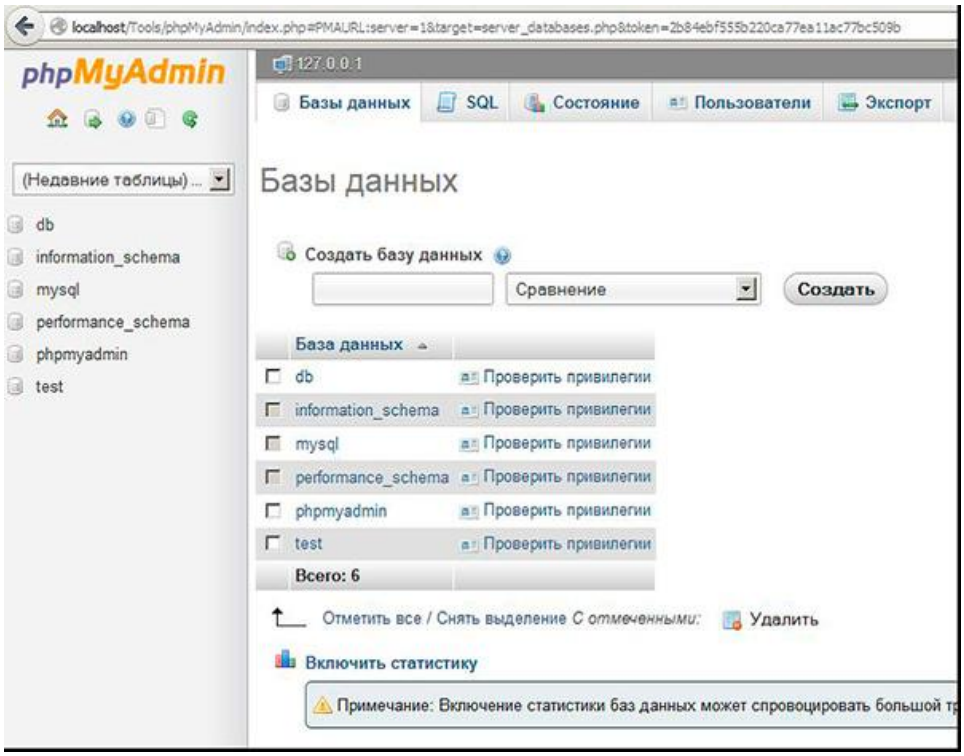
Электронные базы данных позволяют хранить на жестком диске сервера данные различных типов, а также устанавливать взаимосвязь между ними. *MySQL – наиболее распространенная система управления базами данных.*

Для управления базами данных существует специальный язык запросов –
SQL (structured query language), который имеет довольно обширный синтаксис и позволяет производить все необходимые манипуляции с данными в таблицах. Также существует прекрасное приложение phpMyAdmin, которое освобождает оператора и программиста от ввода огромного количества команд sql – в несколько кликов можно легко создавать базы данных, таблицы, вставлять и удалять записи. Тем не менее, php-программист обязан знать все команды и управляющие конструкции языка sql, а данная статья поможет вам быстро понять и освоить основные принципы работы с MySQL.

Для начала установим MySQL. Проще всего это сделать установив т.н. джентльменский набор веб-разработчика. Дистрибутив и порядок установки вы можете найти на сайте http://www.denwer.ru/.

Для работы с БД из всего набора Денвера нам нужна только утилита *phpMyAdmin*. Запускаем ее в браузере по адресу *http://localhost/Tools/phpMyAdmin/index.php*

Для начала создадим БД, с которой будем работать. Для этого на вкладке *Базы данных*(выбрана на рисунке) укажем реквизиты новой БД. Назовем ее *test*.



Типы данных

В MySQL, как и в других средах разработки, различают несколько типов данных. Изучая СУБД, в первую очередь необходимо изучить все типы данных. В общем, они идентичны во всех СУБД, но у каждой есть свои особенности.

1) INT - Целочисленный формат. В основном используется для идентификаторов и переключателей (0 - нет, 1 - есть).

2) CHAR – Текстовый формат. Ограничен 32 тыс. символов. Используется для хранения небольшого объема текстовой информации. Пример CHAR(128).

3) TEXT - Текстовый формат. Практически неограничен. Используется для хранения большого объема текстовой информации.

4) DATE – Дата.

5) DATETIME – Дата и время.

6) DECIMAL – Числовой формат с разделителем. Пример DECIMAL(14, 2) позволит нам записать число, длинной 14 символов с двумя знаками после запятой.

На самом деле, все типы данных я описывать не стал, этого набора вполне достаточно для реализации практически любого приложения.

Таблицы

Основная сущность БД это таблица. Остальные, как правило, вспомогательные.

Таблица - некоторая группа данных, объединенная по каким-либо общим свойствам. Например, список товаров на складе или список пользователей системы. Такие таблицы при проектировании приложения зовутся справочниками.

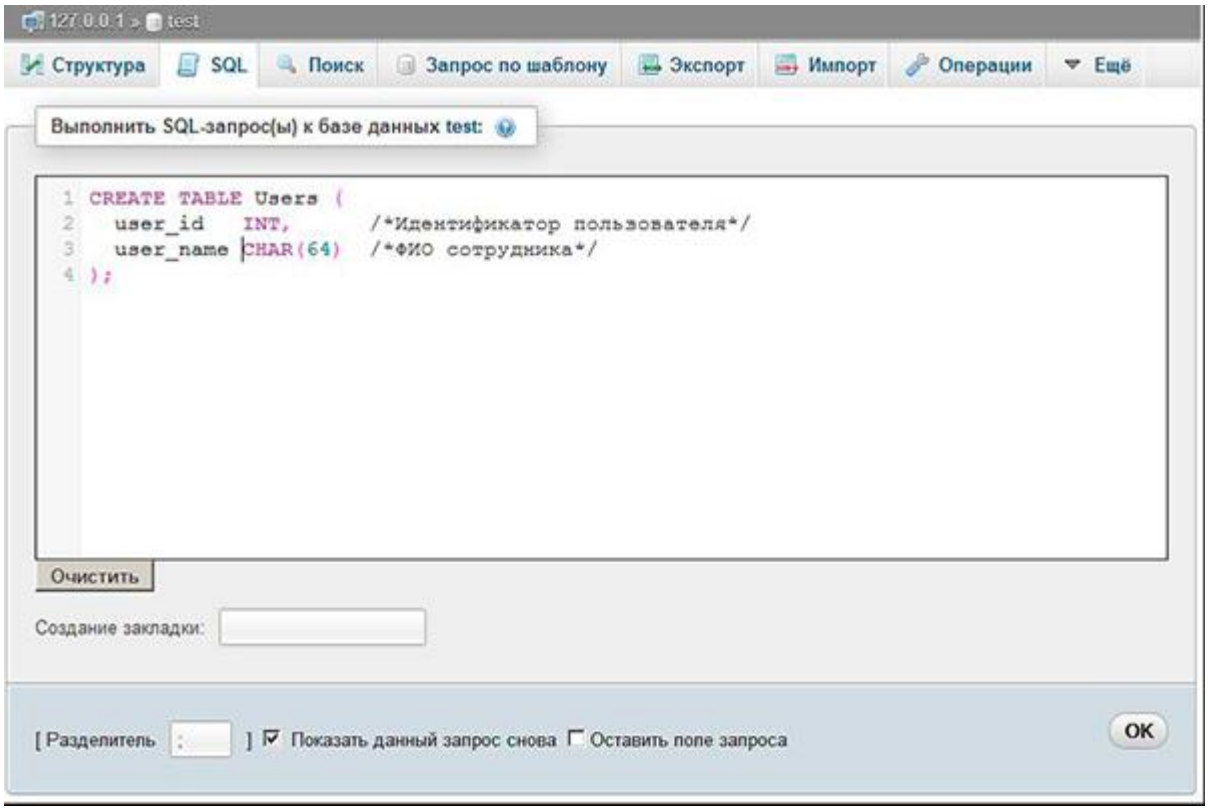
Роль таблицы в БД - *хранение структурированного набора данных* и только. В MySQL таблица представляет собой стандартную таблицу с колонками и строками.

Строки указывают на запись, колонки указывают на реквизиты записи.

Рассмотрим пример. Создадим таблицу "*Users*" (справочник пользователей) с помощью следующего кода:

```
CREATE TABLE Users (
   user_id   INT,     /*Идентификатор пользователя*/
   user_name CHAR(64)  /*ФИО сотрудника*/
);
```

Старайтесь работать с БД именно с помощью скриптов. Это упростит понимание команд и функций. Кроме того, если у вас нет доступа к среде разработки (phpMyAdmin), это вам очень поможет. Работая со скриптами, используем вкладку *SQL* (см. рис. ниже)



Рассмотрим код:

CREATE TABLE Users [Создать таблицу] с именем Users
(
Описание полей таблицы
user_id[наименование поля] as INT [тип данных целочисленный],
user_name[наименование поля] as CHAR(64)[текстовое поле длинной 64 символа],
);

ВНИМАНИЕ! При создании таблицы старайтесь называть ее в соответствии с данными, которые будут в ней храниться. Также в каждой таблице желательно создавать поле идентификатора записи (ID). Это очень пригодится.

Для удаления таблицы используется код:

```
DROP TABLE Users;
```

Для множественного удаления

```
DROP TABLES Users, GroupUsers;
```

Будьте внимательны при вызове указанных команд.

Для изменения таблицы используется код:

```
ALTER TABLE Users
  ADD [MODIFY]
    COLUMN - Колонки
    INDEX  - Индекса
;
```

Запросы

БД, как таковая, должна иметь механизмы ввода-вывода информации. Для вставки, изменения, получения или удаления необходимо использовать запросы. Рассмотрим только запросы SQL (Structured Query Language — «Структурированный язык запросов»), т.к. большинство БД работают именно с запросами такого типа.

Существует 4 вида запросов:

Для примера, поиграем с нашим справочником "Users":

1) Select - Получение данных из таблицы

```
SELECT * FROM Users;
```

Давайте разберем запрос. *SELECT[Выбрать] *[Все столбцы] FROM[Из таблицы] Users*. Данный запрос вернет все записи по всем столбцам таблицы.

Чтобы ограничить выборку (полные выборки требуются только в случае получения отчетов, и то не всегда), добавим в наш запрос условие WHERE.

```
SELECT * FROM Users WHERE user_id = 1;
```

Здесь мы получаем данные по пользователю №1. Идем далее..

2) Update - Обновление данных в таблице

```

```

```
UPDATE Users SET user_name = 'Иванов И.И.' WHERE user_id = 1;
```

Давайте разберем запрос. *UPDATE[Обновить] Users SET[Установить [поле] user_name = 'Иванов И.И.' WHERE[Где] user_id = 1*. В данном случае мы изменили ФИО пользователя №1 на "Иванов И.И.".

3) Insert - Вставка новых данных в таблицу

По - хорошему, данный запрос нужно писать в первую очередь, т.к. справочник у нас изначально пуст, но обычно разработчики создают таблицы через интерфейс (phpMyAdmin) и сразу добавляют несколько тестовых записей.

```
INSERT INTO Users (user_id, user_name) VALUES (2, 'Петров П.П.');
```

Разбираем запрос. *INSERT INTO[Вставить в] Users ([перечнь полей]user_id, user_name) VALUES ([Перечнь значений]2, 'Петров П.П.')*. Запрос вставит запись в таблицу со значениями *user_id* равному 2 и *user_name* равному 'Петров П.П.'.

Если вы заполняете все столбцы таблицы, можно немного упростить запрос и не указывать список полей для вставки.

```
INSERT INTO Users VALUES (2, 'Петров П.П.');
```

Однако при этом важно соблюдать порядок столбцов.

При вставке нескольких строк можно воспользоваться запросом множественной вставки.

```
INSERT INTO Users VALUES (2, 'Петров П.П.'), (3, 'Сидоров С.С.');
```

4) Delete - Удаление данных из таблицы

```
DELETE * FROM Users WHERE user_id = 1;
```

Тут все просто. *DELETE[Удалить] *[Все] FROM[Из] Users*.

Запросу неважно, что будет указано после команды DELETE, будь это перечень полей или одно поле, он удалит всю строку данных.

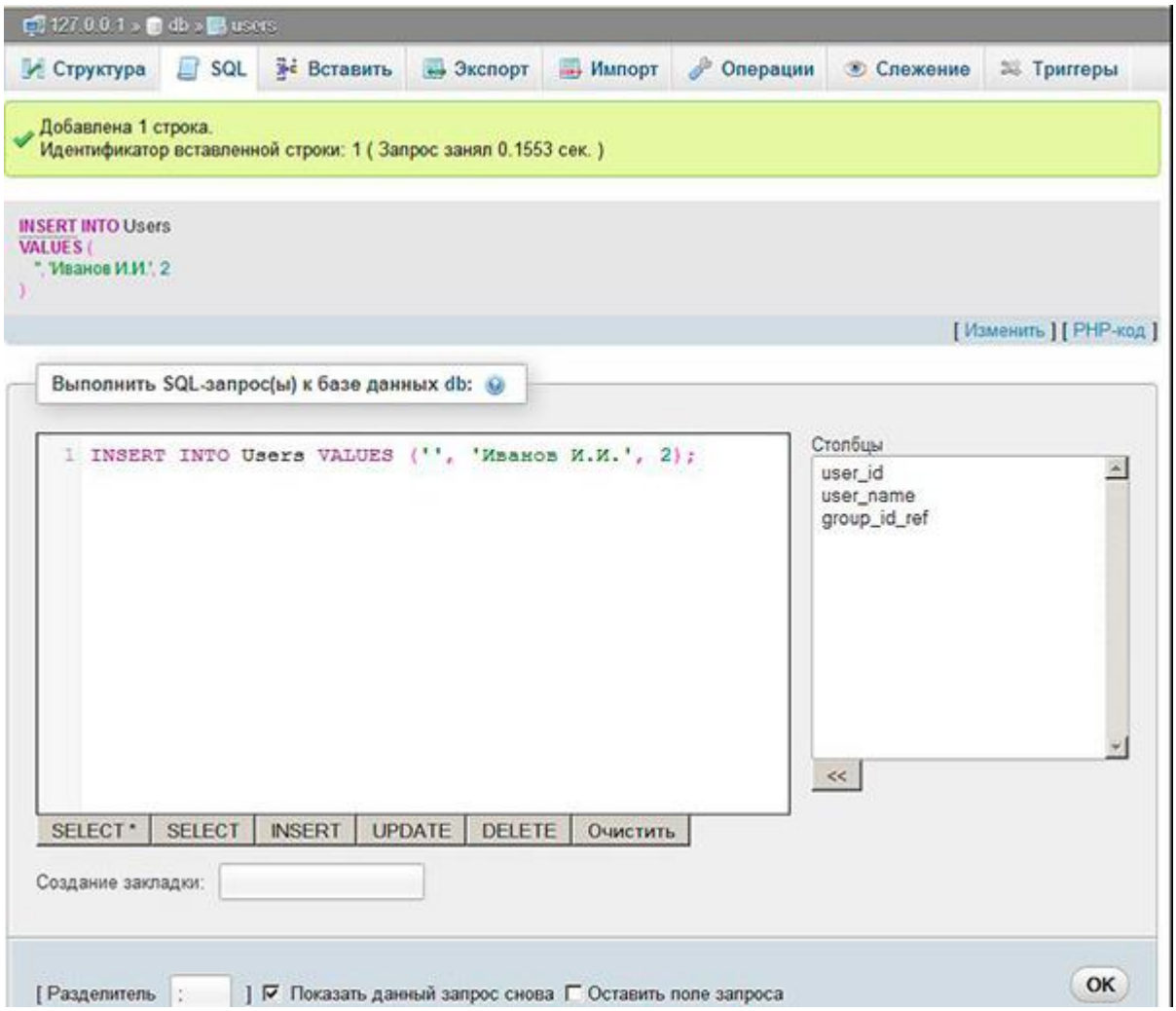
ВНИМАНИЕ! В данном запросе не забывайте писать условие WHERE (Это касается и запроса UPDATE), иначе запрос очистит всю вашу таблицу.

При отладке приложения старайтесь при написании команды DELETE изначально описывать запрос в виде SELECT. Нужно нам, например, написать запрос на удаление одного пользователя, мы пишем:

```
SELECT * FROM Users WHERE user_id = 1;
```

Проверяем. Запрос выводит одну строку, которую нам надо удалить. Заменяем теперь *SELECT* на *DELETE* и спокойно выполняем запрос, не опасаясь за данные.

```
DELETE * FROM Users WHERE user_id = 1;
```



Пример вставки строк в таблицу. Среда оповещает нас о вставке строки.

Триггеры

Триггер - процедура, выполняемая при перед/после выполнения запросов вставки, обновления или удаления.

Триггер позволяет нам сократить код приложения при написании проверок или дополнительных действий. Также триггер помогает проверять связи при редактировании записей, но об этом позже.

Рассмотрим на примере.

Допустим мы желаем хранить историю изменений в нашем приложении. Иногда это очень полезно. Создадим таблицу истории. Как правильно описать поля, мы выяснили ранее.

```
CREATE TABLE History (
  history_id   INT         /*Идентификатор записи*/
  history_type INT          /*Тип изменения*/
```

```
  history_date DATETIME     /*Дата изменения */
  history_text TEXT         /*Текст изменения*/
  history_user CHAR(64)     /*Пользователь*/
);
```

Определим типы изменений:

1 - добавление записи

2 - изменение записи

Теперь создадим триггер на нашей таблице *Users*:

```
CREATE TRIGGER users_trg     /*Создать новый триггер users_trg*/
 AFTER INSERT ON Users       /*Отрабатывать после вставки таблицы*/
FOR EACH                     /*Для каждой новой записи*/
ROW INSERT INTO history      /*Добавить запись в history*/
VALUES ('', 1, now(), 'Пользователь создан', user());
```

Теперь при добавлении нового пользователя в таблицу истории триггер будет добавлять запись о его создании. По сути, триггеры очень полезны. Проверка может осуществляться на уровне БД, не касаясь приложения.

В некоторых СУБД триггеры используются для заполнения уникального идентификатора таблиц, но в MySQL это не требуется.

Индексы

Индексы - это наборы данных в БД, используемые для оптимизации поиска записей.

В таблицах данные хранятся в случайном порядке и, иногда, запись перебором (если индекса нет) найти очень затратно по времени. Индекс содержит в себе указатели на группы данных в таблице, объединенные по какому-либо критерию.

Рассмотрим на примере нашего справочника. Допустим, мы хотим привязать сотрудников по отделам. Модифицируем наш справочник пользователей, добавив в него ссылку на группу.

```
ALTER TABLE Users ADD COLUMN group_id_ref INT;
```

Теперь у нас справочник пользователей имеет привязку к группе. Заполним пустые поля записями 1 и 2. При выполнении запроса вида:

```
SELECT * FROM Users WHERE group_id_ref = 1;
```

будут проверяться все записи в таблице на предмет равенства поля *group_id_ref*единице.

Для ускорения поиска необходимо добавить индекс.

```
ALTER TABLE Users ADD INDEX group_id_idx (group_id_ref);
```

ВИМАНИЕ! Эта процедура разовая и проводится на этапе разработки. Времени много не занимает, поэтому советую не опускать этот пункт. Кроме того, это основа быстродействия вашего приложения.

Теперь разберем, чем нам поможет данных индекс. Возьмем тот же запрос:

```
SELECT * FROM Users WHERE group_id_ref = 1;
```

Алгоритм поиска по таблице будет идти только по группе записей, group_id_ref которых равен 1.

В нашем случае, когда у пользователей только две группы, время поиска сократится незначительно, но в ситуации, когда групп пользователей много, индекс ускоряет поиск в разы и десятки раз.

Теперь, чтобы как-то определить группы, создадим справочник.

```
CREATE TABLE GroupUsers (
  group_id   INT      /*Идентификатор группы*/
  group_name CHAR(64) /*Наименование группы*/
);
```

Также нам может пригодиться первичный индекс. Для чего он нужен? Для того, чтобы при добавлении записей в справочники каждая из них имела свой идентификатор.

Это очень удобно. Оперируя данными таблицы, нам достаточно указать идентификатор конкретной строки для ее определения. Как пример, для сравнения представим ситуацию, что в нашей таблице *Users* есть три записи о пользователях.

```
user_id | user_name         | group_id_ref
--------+-------------------+-------------
1       | Иванов И.И.       | 1
2       | Иванов И.И.       | 2
3       | Петров П.П.       | 1
```

Получается, что у нас есть два пользователя с одинаковым именем, но в разных отделах. Такое встречается очень часто, особенно, если ИО заполняется инициалами.

При отсутствии уникального идентификатора запрос на изменение второй записи выглядел бы так:

```
UPDATE Users SET user_name = 'Иванов Иван Иванович'
 WHERE user_name = 'Иванов И.И'
   AND group_id_ref = 2;
```

При наличии уникального идентификатора *user_id* запрос выглядит немного проще:

```
UPDATE Users SET user_name = 'Иванов Иван Иванович'
 WHERE user_id = 2;
```

При всем при этом поиск в таблице по тексту — не самое лучшее решение. СУБД позволяет это делать, но вы сильно теряете в скорости. Недостатком такого поиска также является передача в запрос текстовых параметров. Их необходимо экранировать.

В MySQL есть очень простой способ создания уникального индекса.

```
ALTER TABLE Users ADD user_id INT NOT NULL AUTO_INCREMENT FIRST,
                  ADD PRIMARY KEY (user_id);
```

Рассмотрим подробнее код:

```
ALTER TABLE[Изменить таблицу] Users
  MODIFY COLUMN[изменить колонку]
    user_id AS INT[тип данных целочисленный]
      auto_increment[автоматическое заполнение столбца значениями по возрастанию (1,2,3 и т.д.)];
```

ВНИМАНИЕ! *ALTER* это изменение структуры БД. Необходимо минимизировать или вообще стараться избегать этой команды в скриптах. Все индексы и счетчики должны быть учтены при проектировке БД.

Вот так выглядит скрипт создания таблицы *Users* без использования *ALTER*:

```
CREATE TABLE Users (
  user_id      INT auto_increment,  /*Идентификатор */
  user_name    CHAR(64),            /*ФИО сотрудника*/
  group_id_ref INT,                 /*Группа*/
  PRIMARY KEY (user_id),       /*Первичный ключ*/
  KEY (group_id_ref)           /*Индекс по группе*/
);
```

Итог

Мы познакомились с Базами данных, узнали о таблицах, запросах, триггерах и индексах. Пройдемся еще раз по основным объектам.

1) Таблица — объект БД для хранения данных.

2) Запрос — средство ввода-вывода информации.

3) Триггер — процедура, выполняемая при изменении данных в таблице.

4) Индекс — указатель на записи в таблице.

По сути данного набора знаний будет вполне достаточно, чтобы разработать сайт-визитку.

Итак, войдите в своем любимом браузере в систему управления базами данных MySQL по ад-

ресу **http://localhost/Tools/phpMyAdmin/** . (**Внимание** перед переходом по ссылке не забудьте запустить Дэнвер). На экране вы увидите примерно такую картинку:

4.3.3 Проектирование баз данных. Нормализация таблиц.

Одна из важнейших задач проектирования баз данных - устранение из нее избыточности. Для этого используется прием, называемый *нормализацией.* Прежде чем приступить к нормализации, обсудим некоторые фундаментальные понятия реляционных баз данных. *Модель данных -это диаграмма*, показывающая конструкцию вашей базы данных. Она состоит из трех основных элементов - сущностей, атрибутов и связей. Пока остановимся на сущностях и атрибутах, а о связях поговорим позднее.

Сущности в базе данных

Сущность - это важная вещь или объект, сведения о котором нужно сохранить. Не все вещи являются сущностями, а только те, данные о которых должны быть сохранены. Сведения о сущностях имеют вид атрибутов и/или связей. Если некий кандидат на то, чтобы быть сущностью, не имеет атрибутов или связей, в действительности он не является сущностью. В модели базы данных сущности представляются в виде прямоугольника с заголовком. Заголовок является именем сущности.

Атрибуты сущности

Атрибут описывает данные о сущности, которые нужно сохранить. У каждой сущности ноль или более атрибутов, описывающих ее, и каждый атрибут описывает в точности одну сущность. Каждый экземпляр сущности (строка таблицы) имеет в точности одно значение, возможно, равное NULL, для каждого из своих атрибутов. Значение атрибута может быть числом, строкой символов, датой, временем или другим базовым значением данных. На первом этапе проектирования базы данных, логическом моделировании, нас не заботит то, каким образом будут храниться данные.

NULL лежит в основе проблемы, связанной с отсутствующей информацией. Он специально используется тогда, когда какая-то часть данных отсутствует. Рассмотрим, к примеру, ситуацию, когда на CD нет данных о длительности каждой песни. У каждой песни есть длительность, но, глядя на коробку, вы не можете ска-

зать, какова она. Хранить длительность как О нежелательно, поскольку это было бы неверно. Вместо этого вы записываете длительность как NULL. Если вы считаете, что можно сохранить ее как 0 и использовать 0 для обозначения «неизвестной длины», то можете попасть в одну из тех западней, которые привели к проблеме 2000-го года. В старых системах не только год хранится как две цифры, но и придается особое значение величине 9-9-99.

В нашем примере база данных ссылается на ряд объектов - CD, название CD, название ансамбля, песни и название фирмы звукозаписи. Какие из них являются сущностями, а какие - атрибутами?

Модель данных

Обратите внимание, что мы определяем несколько видов данных (название CD, название ансамбля и т. д.), относящихся к каждому CD, и без которых описать CD совершенно невозможно. Поэтому CD является одним из тех объектов, которые мы хотим описать, и, похоже, является сущностью. Начнем разработку модели данных с изображения CD как сущности.

По общепринятому соглашению об именовании сущностей имя сущности должно быть в единственном числе. Поэтому мы называем таблицу, в которой хранятся CD «CD», а не «CDs». Мы используем это соглашение, поскольку каждая сущность дает имя экземпляру. Например, «San Francisco 49ers» является экземпляром сущности «Футбольная команда», а не «Футбольные команды».

На первый взгляд кажется, что оставшаяся часть базы данных описывает CD. Это указывает на то, что она содержит атрибуты CD. В модели данных атрибуты представлены как имена, перечисленные в прямоугольнике сущности.

Эта диаграмма проста, но мы еще не закончили. В действительности, мы только начали. Ранее мы говорили, что целью моделирования данных является устранение избыточности с помощью приема, называемого нормализацией. У нашей базы данных прекрасная диаграмма, но мы не покончили с избыточностью, как намеревались. Пора нормализовать нашу базу данных.

Нормализация

Е. Ф. Кодд (E. F. Codd), занимавшийся исследовательской работой в IBM, впервые представил концепцию нормализации в не-

скольких важных статьях, написанных в 1970-е годы. Задача нормализации остается той же самой и сегодня: устранить из базы данных некоторые нежелательные характеристики. В частности, ставится задача устранить некоторые виды избыточности данных и благодаря этому избежать аномалий при изменении данных. Аномалии изменения данных - это сложности при операциях вставки, изменения и удаления данных, возникающие из-за структуры базы данных. Дополнительным результатом нормализации является конструкция, хорошо соответствующая реальному миру. Поэтому в результате нормализации модель данных становится более ясной.

Например, предположим, что мы ошиблись при вводе «Herbie Hancock» в нашу базу данных и хотим исправить ошибку. Нам потребовалось бы рассмотреть все диски этого исполнителя и исправить имя. Если изменения производятся с помощью приложения, позволяющего одновременно редактировать только одну запись, нам придется редактировать много строк. Было бы гораздо лучше запомнить имя «Herbie Hancock» лишь один раз и редактировать его в одном месте.

Уникальный идентификатор

Прежде чем обсуждать связи, мы должны применить к сущностям еще одно правило. У каждой сущности должен быть однозначный идентификатор, который мы будем называть ID. *ID* есть атрибут сущности, к которому применимы следующие правила:

- Он уникален для каждого экземпляра сущности.
- Для каждого экземпляра сущности он имеет значение, отличное от NULL в течение всего срока существования экземпляра.
- В течение всего времени существования экземпляра его значение не меняется.

ID очень важен, поскольку позволяет узнать, с каким из экземпляров сущности мы имеем дело. Выбор идентификатора также существенен, потому что он используется для моделирования связей. Если после выбора ID для сущности вы обнаружили, что он не удовлетворяет одному из перечисленных правил, это может повлиять на всю вашу модель данных.

Новички в моделировании данных часто делают ошибку, выбирая в качестве ID неподходящие атрибуты. Если, к примеру, у вас

есть сущность Person (человек, лицо), может возникнуть соблазн выбрать в качестве идентификатора Name (фамилию), поскольку она есть у каждого лица и не меняется. Но что если лицо вступает в брак или законным образом хочет изменить фамилию? Или вы допустили ошибку при первоначальном вводе фамилии? При каждом из этих событий нарушается третье правило для идентификаторов. Еще хуже то, что фамилия окажется не уникальной. Если вы не можете стопроцентно гарантировать, что атрибут Name уникален, вы нарушаете первое правило для идентификаторов. Наконец, вы считаете, что у каждого экземпляра Person фамилия отлична от NULL. Но вы уверены, что всякий раз, вводя первоначальные данные в базу, будете знать фамилию? Ваш процесс может быть организован так, что при начальном создании записи фамилия может быть неизвестна. Из этого следует извлечь тот урок, что при выборе неидентифицирующего атрибута в качестве идентификатора возникает много проблем.

Выход в том, чтобы изобрести идентифицирующий атрибут, не имеющий никакого иного смысла, кроме как служить идентифицирующим атрибутом. Поскольку этот атрибут искусственный и никак не связан с сущностью, мы имеем над ним полный контроль и можем обеспечить его соответствие правилам для уникальных идентификаторов. На рис. 2-4 к каждой из наших сущностей добавлен искусственный ID. На диаграмме уникальный идентификатор изображается как подчеркнутый атрибут.

ближайшем рассмотрении оказывается, что следует установить прямую связь между Artist и Song. У каждого Artist есть одна или много Song. Каждая Song исполняется одним и только одним Artist.

Это не только более разумно, чем связь между Artist и CD, но и решает проблему дисков-сборников.

Виды связей

При моделировании связей между сущностями важно определить оба направления связи. После определения обеих сторон связи мы приходим к трем основным видам связей. Если оба конца связи имеют степень «один и только один», то связь называется «один-к-одному». Как мы позднее убедимся, связи «один-к-одному» встречаются редко. В нашей модели данных их нет.

Если одна сторона имеет степень «один или много», а другая сторона имеет степень «один и только один», то это связь «один-ко-многим» или «1-к-М». Все связи в нашей модели - это связи «один-ко-многим». Этого можно было ожидать, поскольку связи «один-ко-многим» наиболее распространены.

И наконец, последний тип связей - когда обе стороны имеют степень «один-ко-многим». Такого типа связи называются «многие-ко-многим», или «М-к-М». В предыдущей версии нашей модели данных связь Artist-CD имела тип «многие-ко-многим».

Уточнение связей

Как отмечалось ранее, связи «один-к-одному» очень редки. На практике, если в процессе моделирования вы столкнетесь с такой связью, следует внимательнее изучить свой проект. Такая связь может означать, что две сущности являются на самом деле одной, и если это так, их следует объединить в одну.

Связи «многие-ко-многим» встречаются чаще, чем «один-к-одному». В этих связях часто есть некоторые данные, которыми мы хотим охарактеризовать связь. Взглянем, например, на предыдущую версию нашей модели данных на рис. 2-8, в которой была связь «многие-ко-многим» между Artist и CD. Artist имеет связь с CD, поскольку у исполнителя есть одна или несколько Song на этом CD.

Все связи «многие-ко-многим» нужно разрешать с помощью следующей технологии:

1. Создайте новую сущность, иногда называемую *сущностью-связкой.* Назовите ее подходящим образом. Если вы не можете придумать подходящее название, образуйте его из сочетания имен связываемых сущностей, например ArtistCD. В нашей модели Song является сущностью-связкой для связи Artist-CD.
2. Свяжите новую сущность с двумя исходными. Каждая из исходных сущностей должна иметь связь «один-ко-многим» с сущностью-связкой.
3. Если в новой сущности нет очевидного уникального идентификатора, введите в нее идентифицирующие атрибуты исходных сущностей и сделайте эту пару уникальным идентификатором новой сущности.

Почти всегда обнаружатся дополнительные атрибуты, принадлежащие новой сущности. Если это не так, то все равно необходимо разрешить связь «многие-ко-многим», иначе возникнут проблемы при переводе вашей модели данных в физическую схему.

Тщательное проектирование базы данных чрезвычайно важно для безупречной работы приложения.

MySQL – это **реляционная база данных**. Важная особенность реляционных систем, их отличие от одноуровневых баз данных – возможность располагать данные в нескольких таблицах. Взаимосвязанные данные можно хранить в отдельных таблицах и объединять по ключу, общему для обеих таблиц. Ключ – это *отношение (relation)*между таблицами. Выбор *первичного ключа (primary key)* – наиболее важное решение, принимаемое при разработке новой базы данных. Самое главное, что следует понимать, – вы должны гарантировать уникальность выбранного ключа. Если есть вероятность того, что значение некоторого атрибута может совпасть у двух записей, то его нельзя использовать в качестве первичного ключа. Если таблица содержит ключевые поля из другой таблицы, то между ними образуется связь – взаимоотношением *внешнего ключа (foreign key)*, например, «начальник-подчиненный» или «покупатель-покупка».

В качестве примера создадим таблицу в базе данных users (мы ее использовали в предыдущей статье) соответствующую, например, книжному интернет-магазину. Выполните следующий SQL-код для создания таблицы и вставки данных:

Код SQL для примеров

```
-- Создадим таблицу Customers (покупатели)
CREATE TABLE Customers (
   id INT NOT NULL AUTO_INCREMENT,
   firstname VARCHAR(32),
   secondname VARCHAR(50),
   adress VARCHAR(256),
   telephone VARCHAR(20),
   bookTitle VARCHAR(256),
   bookAuthor1 VARCHAR(64),
   bookAuthor2 VARCHAR(64),
   pageCount INT(4),
   dateOrder DATETIME,
```

PRIMARY KEY(id)) *CHARACTER SET* utf8;

-- Наполнить таблицу какими-то данными
INSERT INTO Customers *VALUES*
(1, 'Александр', 'Иванов', 'Ленинский проспект 68 - 34, Москва 119296', '+7-920-123-45-67', 'Золотые сказки', 'Александр Сергеевич Пушкин', '', 128, '2013-04-18 14:56:00'),
(*NULL*, 'Дмитрий', 'Петров', 'Хавская 3 - 128, Москва 115162', '+7-495-123-45-67', 'ASP.NET MVC 4', 'Джесс Чедвик', 'Тодд Снайдер', 432, '2013-02-11 09:18:00'),
(*NULL*, 'Дмитрий', 'Петров', 'Хавская 3 - 128, Москва 115162', '+7-495-123-45-67', 'LINQ. Язык интегрированных запросов', 'Адам Фримен', 'Джозеф С. Раттц', 656, '2013-02-25 19:44:00'),
(*NULL*, 'Александр', 'Иванов', 'Ленинский проспект 68 - 34, Москва 119296', '+7-920-123-45-67', 'Сказки Старого Вильнюса', 'Макс Фрай', '', 480, '2013-05-02 14:12:00'),
(*NULL*, 'Александр', 'Иванов', 'Ленинский проспект 68 - 34, Москва 119296', '+7-920-123-45-67', 'Реверс', 'Сергей Лукьяненко', 'Александр Громов', 352, '2013-03-12 08:25:00'),
(*NULL*, 'Елена', 'Козлова', 'Тамбовская - 47, Санкт-Петербург 192007', '+7-920-765-43-21', 'Золотые сказки', 'Александр Сергеевич Пушкин', '', 128, '2013-04-12 12:56:00'),
(*NULL*, 'Елена', 'Козлова', 'Тамбовская - 47, Санкт-Петербург 192007', '+7-920-765-43-21', 'ASP.NET MVC 4', 'Джесс Чедвик', 'Тодд Снайдер', 432, '2013-04-14 10:11:00');

Теперь, когда у нас есть отдельные таблицы для хранения взаимосвязанных данных, нужно подумать об элементах в каждой таблице, которые будут описывать связи с элементами в других таблицах.

Нормализация

Представление о взаимоотношениях данных и наиболее эффективном способе их организации называется ***нормализацией***. Нормализация заключается в разделении данных на основе логических взаимоотношений с целью минимизировать дублирование данных. Повторяющиеся данные понапрасну расходуют дисковое пространство сервера и затрудняют их обслуживание. При внесении изменений в повторяющиеся данные есть риск пропустить какие-то из них, что может привести к возникновению несогласованностей в базе данных.

С другой стороны, лучшее – враг хорошего: когда данные хранятся по частям в отдельных таблицах, это может потребовать слишком больших накладных расходов на их извлечение, да и запросы могут получаться чересчур замысловатыми. Главная цель – найти золотую середину.

Продолжим пример с книжным интернет-магазином. Сайт магазина должен хранить данные о покупателях, включая имя и фамилию пользователя, адрес и номер телефона, а также информацию о книгах, включая название, автора, количество страниц и дату продажи каждой книги. Изначально мы разместили всю информацию в одной таблице:

id	firstname	secondname	adress	telephone	bookTitle	bookAuthor1	bookAuthor2	pageCount	dateOrder
1	Александр	Иванов	Ленинский проспект 68 - 34, Москва 119296	+7-920-123-45-67	Золотые сказки	Александр Сергеевич Пушкин		128	2013-04-18 14:56:00
2	Дмитрий	Петров	Хавская 3 - 128, Москва 115162	+7-495-123-45-67	ASP.NET MVC 4	Джесс Чедвик	Тодд Снайдер	432	2013-02-11 09:18:00
3	Дмитрий	Петров	Хавская 3 - 128, Москва 115162	+7-495-123-45-67	LINQ. Язык интегрированных запросов	Адам Фримен	Джозеф С. Раттц	656	2013-02-25 19:44:00
4	Александр	Иванов	Ленинский проспект 68 - 34, Москва 119296	+7-920-123-45-67	Сказки Старого Вильнюса	Макс Фрай		480	2013-05-02 14:12:00
5	Александр	Иванов	Ленинский проспект 68 - 34, Москва 119296	+7-920-123-45-67	Реверс	Сергей Лукьяненко	Александр Громов	352	2013-03-12 08:25:00
6	Елена	Козлова	Тамбовская - 47, Санкт-Петербург 192007	+7-920-765-43-21	Золотые сказки	Александр Сергеевич Пушкин		128	2013-04-12 12:56:00
7	Елена	Козлова	Тамбовская - 47, Санкт-Петербург 192007	+7-920-765-43-21	ASP.NET MVC 4	Джесс Чедвик	Тодд Снайдер	432	2013-04-14 10:11:00

Размещение всех данных в одной таблице может показаться заманчивым, однако такой способ приводит к напрасному расходованию пространства в базе данных и делает утомительной операцию обновления данных. С каждой новой покупкой все сведения о покупателе записываются повторно. Для каждой книги можно указать не больше двух авторов. Кроме того, если покупатель переедет и поменяет адрес, это потребует внести изменения в каждую запись, связанную с этим покупателем.

Формы нормализации

Чтобы нормализовать базу данных, начнем с самых главных правил и будем продвигаться вперед шаг за шагом. Процесс нормализации состоит из трех этапов, называемых *формами*. Первый этап, который называется приведением к первой нормальной форме, должен быть выполнен перед приведением базы данных ко второй нормальной форме. Аналогично, невозможно привести базу данных к третьей нормальной форме, минуя вторую. Процесс нормализации приводит структуру данных в соответствие с тремя нормальными формами.

Первая нормальная форма

Необходимо, чтобы приведенная к первой нормальной форме база данных соответствовала трем требованиям. Ни одна таблица не должна иметь повторяющихся столбцов, содержащих одинаковые по смыслу значения, и все столбцы должны содержать единственное значение. Обязательно должен быть определен первичный ключ, который уникальным образом описывал бы каждую строку. Это может быть один столбец или комбинация из нескольких столбцов, в зависимости от того, сколько потребуется столбцов для обеспечения уникальной идентификации строк.

В созданной нами таблице нарушено требование, предъявляемое к повторяющимся столбцам, потому что в столбцах "bookAuthor1" и "bookAuthor2" хранятся одинаковые по смыслу данные. Это несоответствие надо устранить, в противном случае вам может потребоваться добавить много полей для хранения имен авторов (например, если у книги три автора), что приведет к неоправданному расходу пространства, или может не хватить предусмотренного количества полей для хранения всех имен, если над книгой трудились много авторов. Решение заключается в том, чтобы переместить имена всех авторов в отдельную таблицу, которая будет связана с таблицей книг.

Вторая нормальная форма

Как уже отмечалось, первая нормальная форма снижает избыточность данных в строке. Вторая нормальная форма ликвидирует избыточность данных в столбцах. Нормальные формы получаются последовательно. Для приведения ко второй нормальной форме необходимо, чтобы таблицы уже соответствовали требованиям первой.

Чтобы привести таблицу базы данных ко второй нормальной форме, нужно определить, какие из ее столбцов содержат одни и те же данные для нескольких строк. Такие столбцы нужно поместить в отдельную таблицу, связав ее с первоначальной по ключу. Другими словами, нужно отыскать поля, не зависящие от первичного ключа. Поскольку имена авторов и такие сведения о книгах, как количество страниц, никак не связаны с первичным ключом, идентифицирующем покупателя, выделим эту информацию в отдельные таблицы (это действие будет включать в себя также нормализацию по первой форме, т.к. мы выделяем в отдельную таблицу имена авторов):

Код SQL

-- Создадим новую таблицу Books

```
CREATE TABLE Books (
   bookId INT NOT NULL AUTO_INCREMENT,
   title VARCHAR(500),
   authors VARCHAR(1000),
   pageCount INT(4),
PRIMARY KEY(bookId)) CHARACTER SET utf8;


-- Заполним таблицу Books и столбец bookId таблицы customers
INSERT INTO Books VALUES(1, 'Золотые сказки', 'Александр Сергеевич Пушкин', 128),
 (NULL, 'ASP.NET MVC 4', 'Джесс Чедвик, Тодд Снайдер', 432),
 (NULL, 'LINQ. Язык интегрированных запросов', 'Адам Фримен, Джозеф С. Раттц', 656),
 (NULL, 'Сказки Старого Вильнюса', 'Макс Фрай', 480),
 (NULL, 'Реверс', 'Сергей Лукьяненко, Александр Громов', 352);

-- Видоизменим таблицу Customers удалив избыточные столбцы bookTitle, bookAuthor1, bookAuthor2, pageCount
-- и добавим столбец bookId
ALTER TABLE customers DROP bookTitle, DROP bookAuthor1, DROP bookAuthor2, DROP pageCount, ADD bookId INT(4);

UPDATE Customers SET bookId = 1 WHERE id = 1;
UPDATE Customers SET bookId = 2 WHERE id = 2;
UPDATE Customers SET bookId = 3 WHERE id = 3;
UPDATE Customers SET bookId = 4 WHERE id = 4;
UPDATE Customers SET bookId = 5 WHERE id = 5;
UPDATE Customers SET bookId = 1 WHERE id = 6;
UPDATE Customers SET bookId = 2 WHERE id = 7;
```

Этот код позволяет создать дополнительную таблицу Books хранящую данные о книгах. Однако мы еще не выполнили полную нормализацию по второй форме. Можно заметить, что в нескольких строках таблицы Customers повторяется информация о пользователях, сделавших несколько заказов. Для приведения ко второй нормальной форме мы определим новую таблицу Orders (Заказы):

Код SQL

-- Создадим новую таблицу Orders (orderId - идентификатор заказа, userId - идентификатор пользователя, который сделал заказ)

CREATE TABLE Orders (
orderId *INT NOT NULL* AUTO_INCREMENT,
userId *INT*,
bookId *INT*,
dateOrder DATETIME,
PRIMARY KEY(orderId)) *CHARACTER SET* utf8;

-- Видоизменим таблицу Customers и добавим данные Orders, обратите внимание, что проводить нормализацию заполненной базы данных является трудоемкой задачей, поэтому ее нужно проводить на этапе проектирования базы данных

INSERT INTO Orders (dateOrder, bookId) *SELECT* dateOrder, bookId *FROM* Customers;

UPDATE Orders *SET* userId = 1 *WHERE* orderId = 1;
UPDATE Orders *SET* userId = 2 *WHERE* orderId = 2;
UPDATE Orders *SET* userId = 2 *WHERE* orderId = 3;
UPDATE Orders *SET* userId = 1 *WHERE* orderId = 4;
UPDATE Orders *SET* userId = 1 *WHERE* orderId = 5;
UPDATE Orders *SET* userId = 3 *WHERE* orderId = 6;
UPDATE Orders *SET* userId = 3 *WHERE* orderId = 7;

ALTER TABLE customers *DROP* dateOrder;

-- Удалить дублирующие данные пользователей из таблицы Customers

ALTER IGNORE *TABLE* customers *ADD UNIQUE* INDEX (telephone);

ALTER TABLE customers *DROP* bookId;

Теперь данные оформлены безупречно. У нас появились отдельные таблицы со сведениями о покупателях (Customers), книгах (Books) и покупках (Orders).

Третья нормальная форма

Если вы завершили приведение к первой и второй нормальным формам, возможно, вам не потребуется ничего больше делать с базой

данных, чтобы привести ее к третьей нормальной форме. Для приведения к третьей нормальной форме нужно просмотреть таблицы и выделить данные, которые не зависят от первичного ключа, но зависят от других значений. Пока еще не совсем понятно, как применить это к нашим таблицам. В таблице Customers компоненты адреса не имеют прямого отношения к покупателю. Название улицы и номер дома связаны с почтовым индексом, почтовый индекс – с городом и, наконец, сам город – с областью или краем. Третья нормальная форма требует, чтобы каждая такая часть данных была выделена в отдельную таблицу.

Ниже показано, как можно разделить информацию об адресе создав отдельную таблицу Adresses:

Код SQL

```
-- Создадим новую таблицу Adresses
CREATE TABLE Adresses (
   userId INT NOT NULL AUTO_INCREMENT,
   city VARCHAR(30),
   street VARCHAR(50),
   postcode INT(6),
PRIMARY KEY(userId)) CHARACTER SET utf8;

-- Видоизменим таблицу Customers и добавим данные в Adresses
ALTER TABLE Customers DROP adress;
INSERT INTO Adresses (city, street, postcode) VALUES ('Москва', 'Ленинский проспект 68 - 34',  119296),
 ('Москва', 'Хавская 3 - 128',  115162),
   ('Санкт-Петербург', 'Тамбовская - 47', 192007);
UPDATE Customers SET id = 3 WHERE id = 6;
```

С практической точки зрения, вы можете обнаружить, что после приведения к третьей нормальной форме было создано больше таблиц, чем вам хотелось бы иметь в своей базе данных. Поэтому вы должны сами решать, когда остановить процесс нормализации. Хорошо, если ваши данные будут соответствовать, по крайней мере, второй нормальной форме. Цель – избежать избыточности данных, предотвратить их повреждение и минимизировать занимаемое данными пространство. Кроме того, нужно убедиться, что одни и те же значения не хранятся в нескольких местах. В противном случае, когда эти данные изменятся,

вам придется обновлять их в нескольких местах, что может привести к повреждению базы данных.

Как вы могли заметить, третья нормальная форма еще сильнее снижает избыточность данных, но ценой простоты их представления и производительности. В нашем примере не приходится ожидать, что информация об адресах будет часто изменяться. Однако третья нормальная форма позволяет снизить риск появления орфографических ошибок в названиях городов и улиц. Поскольку это ваша база данных, вам и определять соотношение между нормализацией, простотой и производительностью.

Типы связей

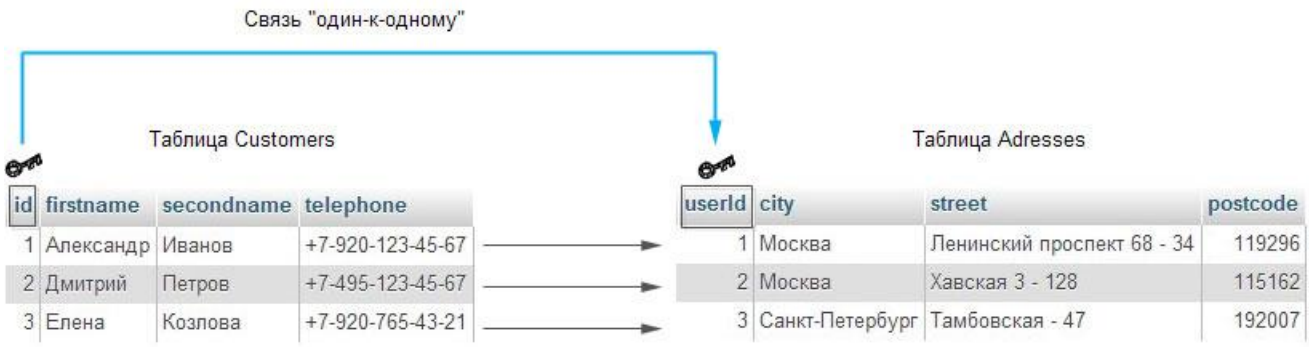
Взаимоотношения или связи, в базах данных подразделяются на следующие категории:

- связи "один-к-одному";
- связи "один-ко-многим";
- связи "многие-ко-многим".

Мы рассмотрим каждую из этих связей на примере созданной нами базы данных.

Связи "один-к-одному"

При связи "один-к-одному" каждому элементу соответствует один и только один другой элемент. Например, в контексте книжного интернет-магазина связь "один-к-одному" существует между покупателем и адресом доставки. Каждый покупатель должен иметь единственный адрес доставки. Знак ключа рядом с каждой из таблиц на рисунке ниже указывает на поле, которое является ключом для этой таблицы:



id	firstname	secondname	telephone
1	Александр	Иванов	+7-920-123-45-67
2	Дмитрий	Петров	+7-495-123-45-67
3	Елена	Козлова	+7-920-765-43-21

userId	city	street	postcode
1	Москва	Ленинский проспект 68 - 34	119296
2	Москва	Хавская 3 - 128	115162
3	Санкт-Петербург	Тамбовская - 47	192007

Связь "один-к-одному" между покупателями и их адресами

Например, чтобы вывести адрес пользователя "Александр Иванов" можно воспользоваться следующей SQL-конструкцией:

Код SQL

SELECT * *FROM* customers *JOIN* adresses *ON* (customers.id = adresses.userid) *WHERE* customers.id = 1;

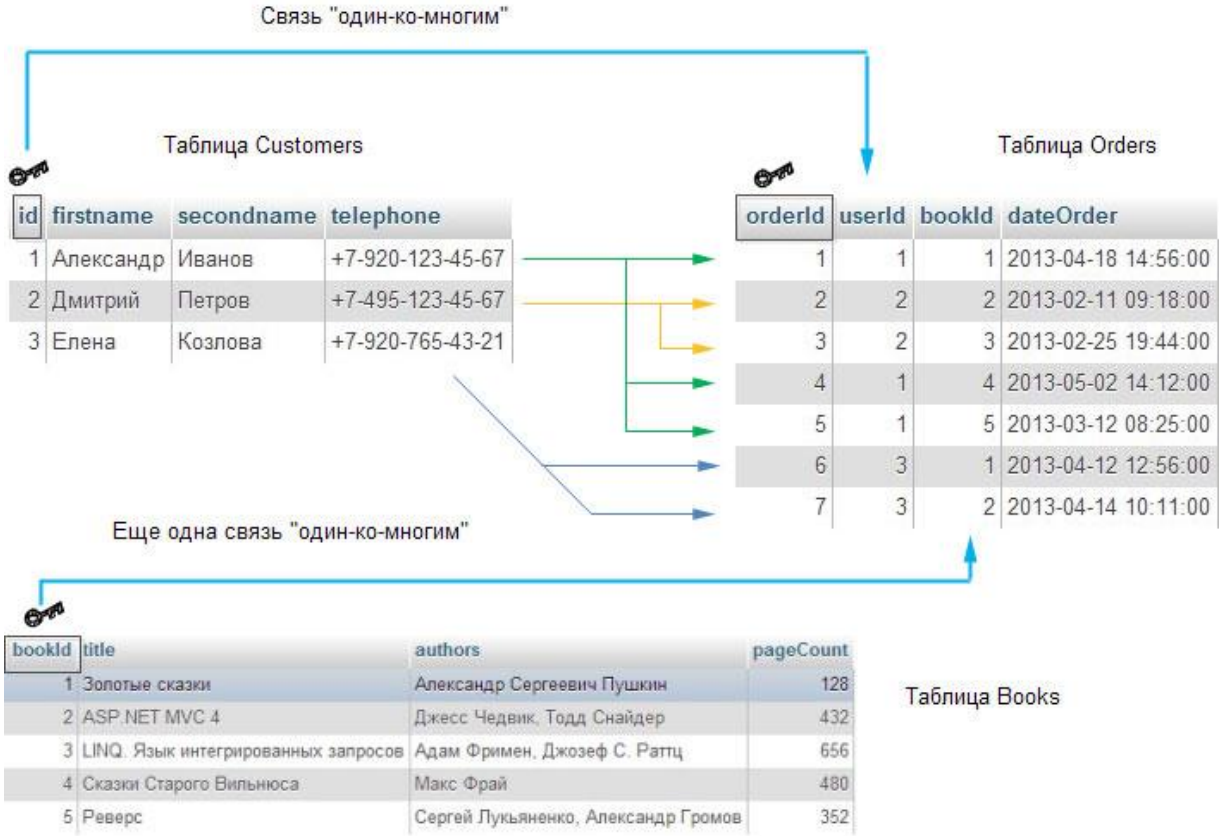
+ Параметры

id	firstname	secondname	telephone	userId	city	street	postcode
1	Александр	Иванов	+7-920-123-45-67	1	Москва	Ленинский проспект 68 - 34	119296

Пример использования связи "один-к-одному"

Связь "один-ко-многим"

В случае связи "один-ко-многим" каждый ключ из одной таблицы может встречаться несколько раз в другой таблице. Это наиболее распространенный тип связи. Например, у одного покупателя может быть несколько заказов, в то же время каждый заказ имеет свой уникальный идентификатор, но два покупателя могут заказать одну и ту же книгу:



id	firstname	secondname	telephone
1	Александр	Иванов	+7-920-123-45-67
2	Дмитрий	Петров	+7-495-123-45-67
3	Елена	Козлова	+7-920-765-43-21

orderId	userId	bookId	dateOrder
1	1	1	2013-04-18 14:56:00
2	2	2	2013-02-11 09:18:00
3	2	3	2013-02-25 19:44:00
4	1	4	2013-05-02 14:12:00
5	1	5	2013-03-12 08:25:00
6	3	1	2013-04-12 12:56:00
7	3	2	2013-04-14 10:11:00

bookId	title	authors	pageCount
1	Золотые сказки	Александр Сергеевич Пушкин	128
2	ASP.NET MVC 4	Джесс Чедвик, Тодд Снайдер	432
3	LINQ. Язык интегрированных запросов	Адам Фримен, Джозеф С. Раттц	656
4	Сказки Старого Вильнюса	Макс Фрай	480
5	Реверс	Сергей Лукьяненко, Александр Громов	352

Связь "один-ко-многим" между покупателями и заказами, заказами и книгами

Например, чтобы вывести все заказы пользователя "Александр Иванов" можно воспользоваться следующей SQL-конструкцией:

Код SQL

*SELECT * FROM* customers *JOIN* orders *ON* (customers.id = orders.userid) *WHERE* customers.id = 1;

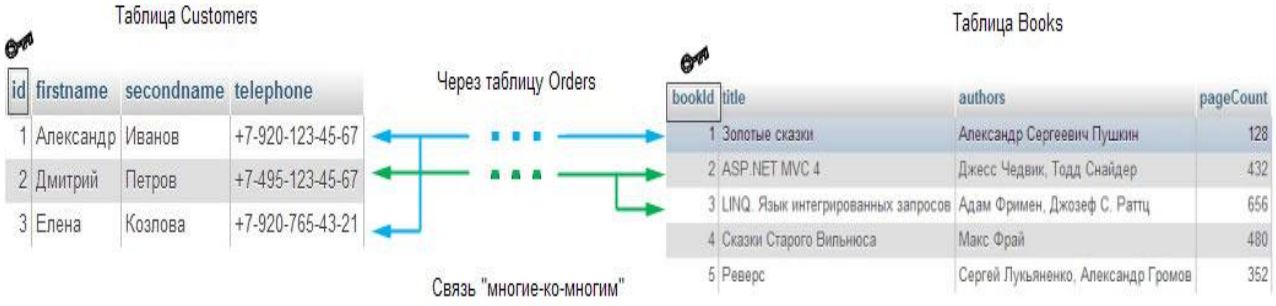
+ Параметры

id	firstname	secondname	telephone	orderId	userId	bookId	dateOrder
1	Александр	Иванов	+7-920-123-45-67	1	1	1	2013-04-18 14:56:00
1	Александр	Иванов	+7-920-123-45-67	4	1	4	2013-05-02 14:12:00
1	Александр	Иванов	+7-920-123-45-67	5	1	5	2013-03-12 08:25:00

Пример использования связи "один-ко-многим"

Связь "многие-ко-многим"

Связь "многие-ко-многим" возникает между двумя таблицами, когда в каждой из них может присутствовать несколько ключей другой таблицы. Например, покупатель приобретает в интернет-магазине сразу несколько книг. Или одну и ту же книгу приобретают несколько покупателей. На рисунке ниже показана связь "многие-ко-многим" между покупателями и приобретенными книгами:



Связь "многие-ко-многим" между покупателями и приобретенными книгами

Чтобы данные со связью "многие-ко-многим" могли быть представлены в базе данных, этот тип связи преобразуется в две связи "один-ко-многим" с помощью **таблицы отображения (mapping table)**. В нашем случае такой таблицей является Orders.

Вопросы для самоконтроля:

1. Перечислите варианты хранения информации в сети Internet.
2. Опишите принципы хранения информации в базах данных MySQL.
3. Опишите архитектуру базы данных MySQL.
4. Как происходит проектирование баз данных?
5. Опишите фундаментальные понятия проектирования баз данных-сущность, связи, атрибуты, модель данных.

Тема 4. Межплатформенный язык запросов SQL (диалект MySQL)

Синтаксис запросов к базе данных. Механизм работы с базами данных — PhpMyAdmin. Решение задач (сортировка, вывод с условиями и т.д.). Управление форматами даты и времени. Функция DATE_FORMAT

4.4.1 Синтаксис запросов к базе данных. Решение задач (сортировка, вывод с условиями и т.д.). Управление форматами даты и времени. Функция DATE_FORMAT.

Действия, выполняемые над информацией, хранящейся в базе данных, называются манипулированием данными. К ним относятся выборка данных по некоторым условиям, сортировка данных, обновление, удаление и добавление данных. Выполнение этих действий производится с помощью запросов

Запрос — это команда на выполнение определенного вида манипулирования данными.

Существует универсальный язык, на котором формулируются запросы во многих СУБД. Он называется SQL (Structured Query Language) — структурированный язык запросов. Здесь мы оказываемся перед выбором, с которым часто приходится сталкиваться в информатике: обучаться ли составлению запросов на языке SQL или воспользоваться каким-то более высокоуровневым вспомогательным средством. В большинстве современных СУБД такие средства имеются.

SQL (англ. Structured Query Language — язык структурированных запросов) — универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.

Вопреки существующим заблуждениям, SQL в его чистом (базовом) виде является информационно-логическим языком, а не языком программирования. Вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений, с учётом которых язык уже вполне может рассматриваться в качестве языка программирования.

SQL основывается на реляционной алгебре.

Обсудим основные категории команд, реализующих в SQL выполнение различных функций. Среди таких функций — построение объектов базы данных, управление объектами, пополнение таблиц базы данных новыми данными, обновление данных, уже имеющихся в таблицах, выполнение запросов, управление доступом пользователей к базе данных, а также осуществление общего администрирования базы данных.

Такими категориями являются:

- DDL (Data Definition Language — язык определения данных);
- DML (Data Manipulation Language — язык манипуляций данными);
- DQL (Data Query Language — язык запросов к данным);
- DCL (Data Control Language — язык управления данными);
- команды администрирования данных;
- команды управления транзакциями.

Определение структур базы данных (DDL)

Язык определения данных (DDL) является частью SQL, дающей пользователю возможность создавать различные объекты базы данных и переопределять их структуру, например, создавать или удалять таблицы.

Среди основных команд DDL:

- CREATE TABLE
- ALTER TABLE
- DROP TABLE
- CREATE INDEX
- ALTER INDEX
- DROP INDEX

Команда создания таблицы CREATE

Таблицы создаются командой **CREATE TABLE**. Эта команда создает пустую таблицу — таблицу без строк. Значения вводятся с помощью DML команды INSERT. Команда CREATE TABLE в основном определяет имя таблицы, в виде описания набора имен столбцов, указанных в определенном порядке. Она также определяет типы данных и размеры столбцов. Каждая таблица должна иметь, по крайней мере, один столбец. Синтаксис команды CREATE TABLE:

```
CREATE TABLE <table-name >
(<column name > <data type>[(<size>)],
<column name > <data type> [(<size>)] ...);
```

Эта команда будет создавать таблицу Продавцов:

```
CREATE TABLE Saleepeople
(snum  integer,
 sname char (10),
 city  char (10;
```

Порядок столбцов в таблице определяется порядком, в котором они указаны. Имя столбца не должно разделяться при переносе строки, но отделяется запятыми.

Индексы

Индекс — это упорядоченный (буквенный или числовой) список столбцов или групп столбцов в таблице. Таблицы могут иметь большое количество строк, а, так как строки не находятся в каком-нибудь определенном порядке, их поиск по указанному значению может потребовать значительного времени. Когда вы создаете индекс в поле, ваша база данных запоминает соответствующий порядок всех значений этого поля в области памяти. Предположим, что наша таблица Заказчиков имеет тысячи входов, а вы хотите найти заказчика с номером cnum=299. Так как строки не упорядочены, ваша программа будет просматривать всю таблицу, строку за строкой, проверяя каждый раз значение поля cnum на равенство значению 299. Однако если бы имелся индекс в поле cnum, то программа могла бы выйти на номер 299 прямо по индексу и дать информацию о том, как найти правильную строку таблицы. В то время как индекс значительно улучшает эффективность запросов, использование индекса несколько замедляет операции модификации DML INSERT, UPDATE и DELETE, что вполне понятно, поскольку при модификации таблицы должен модифицироваться и индекс, а сам индекс занимает объем памяти. Следовательно, каждый раз, когда вы создаете таблицу, Вы должны принять решение, индексировать ее или нет. Синтаксис для создания индекса — обычно следующий (помните, что это не ANSI стандарт):

```
CREATE INDEX <index name> ON <table name>
```

```
(<column name> [,<column name>]...);
```

Таблица, конечно, должна уже быть создана и должна содержать имя столбца. Имя индекса не может быть использовано для чего-то другого в базе данных (любым пользователем). Однажды созданный, индекс будет невидим пользователю. Сервер SQL сам решает, когда он необходим, чтобы ссылаться на него, и делает это автоматически. Если, например, таблица Заказчиков будет наиболее часто упоминаемой в запросах продавцов к их собственной клиентуре, было бы правильно создать такой индекс в поле snum таблицы Заказчиков.

```
CREATE INDEX Clientgroup ON Customers (snum);
```

Теперь, тот продавец, который имеет отношение к этой таблице, сможет найти собственную клиентуру очень быстро.

Удаление индексов Главным признаком индекса является его имя, поэтому он может быть удален. Обычно пользователи не знают о существовании индекса. SQL автоматически определяет, позволено ли пользователю использовать индекс, и если да, то разрешает использовать его. Однако, если вы хотите удалить индекс, вы должны знать его имя. Этот синтаксис используется для удаления индекса:

```
DROP INDEX <Index name>;
```

Удаление индекса не воздействует на содержание полей.

Изменение существующей таблицы ALTER TABLE

Команда ALTER TABLE не часть стандарта ANSI; но это — широко доступная, и довольно содержательная форма, хотя ее возможности несколько ограничены. Она используется, чтобы изменить определение существующей таблицы. Обычно, она добавляет столбцы к таблице. Иногда она может удалять столбцы или изменять их размеры, а также в некоторых программах добавлять или удалять ограничения. Типичный синтаксис, чтобы добавить столбец к таблице:

```
ALTER TABLE <table name> ADD/DROP <column name>
<data type> <size>;
```

Удаление таблиц DROP

Вы должны быть собственником (т.е. быть создателем) таблицы, чтобы иметь возможность удалить ее. Поэтому не волнуйтесь о случайном разрушении ваших данных, SQL сначала потребует, чтобы вы очистили таблицу прежде, чем удалит ее из базы данных. Таблица с находящимися в ней строками, не может быть удалена.

```
DROP TABLE <table name>;
```

При подаче этой команды, имя таблицы больше не распознается, и нет такой команды, которая могла бы быть дана этому объекту. Вы должны убедиться, что эта таблица не ссылается внешним ключом к другой таблице, и что она не используется в определении Представления.

реализациях SQL, поддерживаема и полезна. К счастью, она более проста, и, следовательно, более непротиворечива, чем ALTER TABLE. ANSI просто не имеет способа для определения разрушенных или неправильных таблиц. Примечание. Не все SQL-серверы требуют очистки таблицы перед ее удалением. Здесь нужно обратиться к документации по Вашей системе.

Язык манипулирования данными (DML)

DML - Data Manipulation Language. Язык манипулирования данными. Используется для работы с информацией, хранимой в базе данных. Основными командами этой группы являются:

- Select - вычитка информации.
- Insert - добавление информации.
- Update - обновление информации.
- Delete - удаление информации.

Добавить новую запись в таблицу INSERT

```
INSERT INTO <имя_таблицы> [
(<имя_столбца>,<имя_столбца>,...) ]
                                    VALUES (<значе-
ние>,<значение>,..)
```

Список столбцов в данной команде не является обязательным параметром. В этом случае должны быть указаны значения для всех полей таблицы в том порядке, как эти столбцы были перечислены в команде CREATE TABLE, например:

```
INSERT INTO publishers VALUES (16,"Microsoft
Press","http://www.microsoft.com");
```

Пример с указанием списка столбцов:

```
INSERT INTO publishers (publisher,pub_id)
            VALUES ("Super Computer Publish-
ing",17);
```

Модификация записей UPDATE

```
UPDATE <имя_таблицы> SET <имя_столбца>=<значе-
ние>,...
          [WHERE <условие>]
```

Если задано ключевое слово WHERE и условие, то команда UPDATE применяется только к тем записям, для которых оно выполняется. Если условие не задано, UPDATE применяется ко всем записям. Пример:

```
UPDATE publishers SET url="http://www.super-
pub.com" WHERE pub_id=17;
```

В качестве условия используются логические выражения над константами и полями. В условиях допускаются: операции сравнения: > , < , >= , <= , = , <> , != . В SQL эти операции могут применяться не только к числовым значениям, но и к строкам ("<" означает раньше, а ">" позже в алфавитном порядке) и датам ("<" раньше и ">" позже в хронологическом порядке).

- оперции проверки поля на значение NULL: IS NULL, IS NOT NULL
- операции проверки на вхождение в диапазон: BETWEEN и NOT BETWEEN.
- операции проверки на вхождение в список: IN и NOT IN
- операции проверки на вхождение подстроки: LIKE и NOT LIKE
- отдельные операции соединяются связями AND, OR, NOT и группируются с помощью скобок.

```
UPDATE publishers SET url="url not defined" WHERE
url IS NULL;
```

Эта команда находит в таблице publishers все неопределенные значения столбца url и заменяет их строкой "url not defined".

Удаление записей DELETE

```
 DELETE FROM <имя_таблицы> [ WHERE <условие> ]
```

Удаляются все записи, удовлетворяющие указанному условию. Если ключевое слово WHERE и условие отстутствуют, из таблицы удаляются все записи. Пример:

```
DELETE FROM publishers WHERE publisher = "Super
Computer Publishing";
```

Эта команда удаляет запись об издательстве Super Computer Publishing.

Выборка данных SELECT

Для извлечения записей из таблиц в SQL определен оператор SELECT. С помощью этой команды осуществляется не только операция реляционной алгебры "выборка" (горизонтальное подмножество), но и предварительное соединение (join) двух и более таблиц. Это наиболее сложное и мощное средство SQL, полный синтаксис оператора SELECT имеет вид:

```
  SELECT [ALL | DISTINCT] <список_выбора>
           FROM <имя_таблицы>, ...
           [ WHERE <условие> ]
           [ GROUP BY <имя_столбца>,... ]
              [ HAVING <условие> ]
           [ORDER BY <имя_столбца> [ASC |
DESC],... ]
```

Порядок предложений в операторе SELECT должен строго соблюдаться (например, GROUP BY должно всегда предшествовать ORDER BY), иначе это приведет к появлению ошибок. Этот оператор всегда начинается с ключевого слова SELECT. В конструкции <спи-

сок_выбора> определяется столбец или столбцы, включаемые в результат. Он может состоять из имен одного или нескольких столбцов, или из одного символа * (звездочка), определяющего все столбцы. Элементы списка разделяются запятыми.

Пример: получить список всех авторов

```
SELECT author FROM authors;
```

Выборка из нескольких таблиц.

Очень часто возникает ситуация, когда выборку данных надо производить из отношения, которое является результатом слияния (join) двух других отношений. Например, нам нужно получить из базы данных publications информацию о всех печатных изданиях в виде следующей таблицы:

```
-----------------------------------------------
|название_книги | год_выпуска | издательство   |
-----------------------------------------------
|                |             |                |
|                |             |                |
```

Для этого СУБД предварительно должна выполнить слияние таблиц titles и publishers, а только затем произвести выборку из полученного отношения.

Для выполнения операции такого рода в операторе SELECT после ключевого слова FROM указывается список таблиц, по которым произвоится поиск данных. После ключевого слова WHERE указывается условие, по которому производится слияние. Для того, чтобы выполнить данный запрос, нужно дать команду:

```
SELECT titles.title,titles.yearpub,publishers.pub-
lisher
        FROM titles,publishers
        WHERE titles.pub_id=publishers.pub_id;
```

А вот пример, где одновременно задаются условия и слияния, и выборки (результат предыдущего запроса ограничивается изданиями после 1996 года):

```
SELECT titles.title,titles.yearpub,publishers.pub-
lisher
        FROM titles,publishers
        WHERE titles.pub_id=publishers.pub_id AND
              titles.yearpub>1996;
```

Следует обратить внимание на то, что когда в разных таблицах присутствуют одноименные поля, то для устранения неоднозначности перед именем поля указывается имя таблицы и знак "." (точка).

Вычисления внутри SELECT.

SQL позволяет выполнять различные арифметические операции над столбцами результирующего отношения. В конструкции <список_выбора> можно использовать константы, функции и их комбинации с арифметическими операциями и скобками. Например, чтобы узнать сколько лет прошло с 1992 года (год принятия стандарта SQL-92) до публикации той или иной книги можно выполнить команду:

```
 SELECT title, yearpub-1992 FROM titles WHERE
yearpub > 1992;
```

В арифметических выражениях допускаются операции сложения (+), вычитания (-), деления (/), умножения (*), а также различные функции (COS, SIN, ABS - абсолютное значение и т.д.). Также в запрос можно добавить строковую константу:

```
SELECT 'the title of the book is', title, yearpub-
1992
            FROM titles WHERE yearpub > 1992;
```

В SQL также определены так называемые агрегатные функции, которые совершают действия над совокупностью одинаковых полей в группе записей. Среди них:

- AVG(<имя поля>) - среднее по всем значениям данного поля
- COUNT(<имя поля>) или COUNT (*) - число записей
- MAX(<имя поля>) - максимальное из всех значений данного поля

- MIN(<имя поля>) - минимальное из всех значений данного поля
- SUM(<имя поля>) - сумма всех значений данного поля

Следует учитывать, что каждая агрегирующая функция возвращает единственное значение. Примеры: определить дату публикации самой "древней" книги в нашей базе данных

```
SELECT MIN(yearpub) FROM titles;
```

подсчитать количество книг в нашей базе данных:

```
SELECT COUNT(*) FROM titles;
```

Область действия данных функции можно ограничить с помощью логического условия. Например, количество книг, в названии которых есть слово "SQL":

```
SELECT COUNT(*) FROM titles WHERE title LIKE
'%SQL%';
```

Групировка данных.

Группировка данных в операторе SELECT осуществляется с помощью ключевого слова GROUP BY и ключевого слова HAVING, с помощью которого задаются условия разбиения записей на группы.

GROUP BY неразрывно связано с агрегирующими функциями, без них оно практически не используется. GROUP BY разделяет таблицу на группы, а агрегирующая функция вычисляет для каждой из них итоговое значение. Определим для примера количество книг каждого издательства в нашей базе данных:

```
SELECT publishers.publisher, count(titles.title)
          FROM titles,publishers
          WHERE titles.pub_id=publishers.pub_id
          GROUP BY publisher;
```

Ключевое слово HAVING работает следующим образом: сначала GROUP BY разбивает строки на группы, затем на полученные наборы накладываются условия HAVING. Например, устраним из предыдущего запроса те издательства, которые имеют только одну книгу:

```
SELECT publishers.publisher, count(titles.title)
         FROM titles,publishers
         WHERE titles.pub_id=publishers.pub_id
         GROUP BY publisher
           HAVING COUNT(*)>1;
```

Сортировка данных.

Для сортировки данных, получаемых при помощи оператора SELECT служит ключевое слово ORDER BY. С его помощью можно сортировать результаты по любому столбцу или выражению, указанному в <списке_выбора>. Данные могут быть упорядочены как по возрастанию, так и по убыванию. Пример: сортировать список авторов по алфавиту: SELECT author FROM authors ORDER BY author; Более сложный пример: получить список авторов, отсортированный по алфавиту, и список их публикаций, причем для каждого автора список книг сортируется по времени издания в обратном порядке (т.е. сначала более "свежие" книги, затем все более "древние"):

```
 SELECT authors.author,titles.title,ti-
tles.yearpub,publishers.publisher
       FROM authors,titles,publishers,titleauthors
       WHERE titleauthors.au_id=authors.au_id AND
             titleauthors.title_id=titles.title_id
AND
              titles.pub_id=publishers.pub_id
       ORDER BY authors.author ASC, titles.yearpub
DESC;
```

Ключевое слово DESC задает здесь обратный порядок сортировки по полю yearpub, ключевое слов ASC (его можно опускать) - прямой порядок сортировки по полю author.

Когда пользователь работает с большим количеством запросов или пишет их для третьих лиц, либо запросы имеют достаточно сложную логику, то в SQL-коде желательно использовать комментарии, вставляя их непосредственно в запрос.

Комментарии могут быть двух типов: однострочные и многострочные. В зависимости от типа применяется различный синтаксис. Однострочные начинаются с сочетания двух тире (--) и продолжаются

до конца строки. Многострочные комментарии начинаются с сочетания символов слеша и звездочки (**/***) и заканчиваются ими же, но в другой последовательности (***/***).

```
1 -- Пример однострочного комментария
2
3 /* Комментарий
4 на
5 несколько
6 строк  */
7
8 /* Многострочный комментарий в одну строку */
```

Примечание:

Синтаксис комментариев зависит от системы, к которой выполняется запрос. Приведенные выше примеры подходят для систем MS SQL Server и Oracle, являющимися самыми распространенными.

В подавляющем большинстве случаев, записи в таблицах баз данных уникальны, но в отдельных столбцах допускаются повторяющиеся значения. Если запрос выгружает столбцы не являющиеся ключевыми, то полученные строки могут дублироваться.

4.4.2 Механизм работы с базами данных — PhpMyAdmin

Мы начинаем знакомиться с **возможностями ПО PHPMyAdmin**. И давайте с Вами научимся **управлять пользователями в PHPMyAdmin**, так как эта первая вещь, которую необходимо уметь делать.

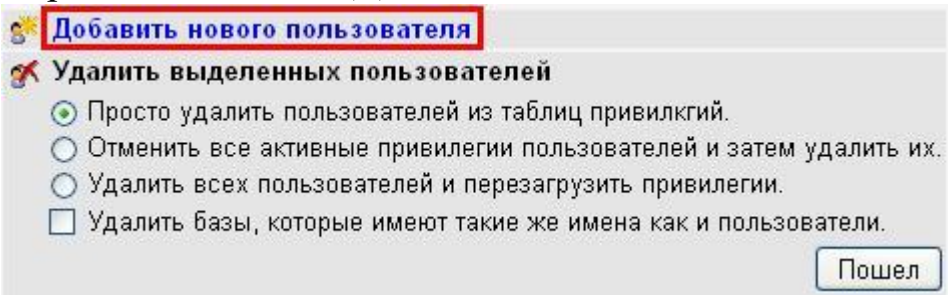
Давайте подробно разберём следующие пункты:

1. **Создание нового пользователя в PHPMyAdmin.**
2. **Редактирование пользователя в PHPMyAdmin.**
3. **Удаление пользователя в PHPMyAdmin.**

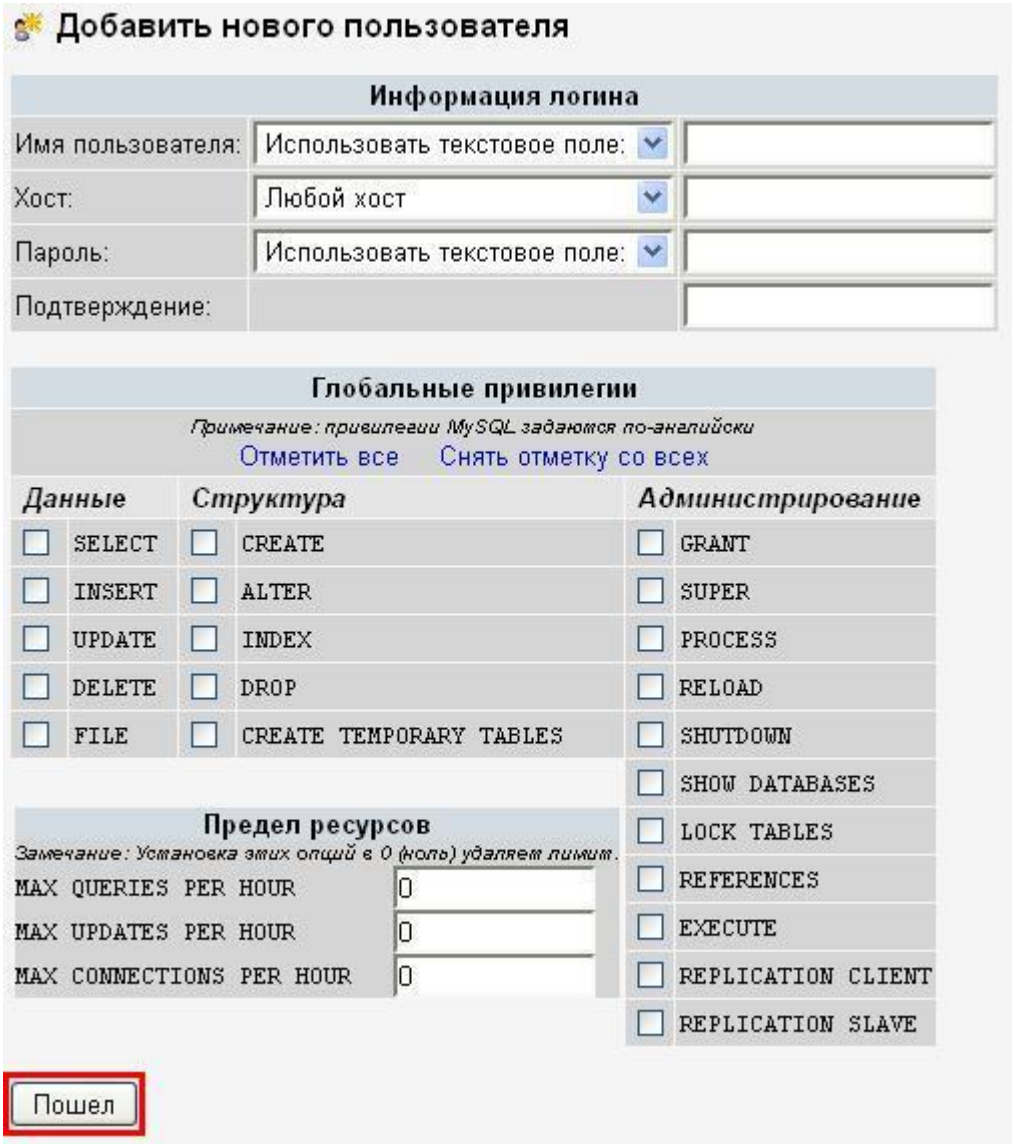
Начнём с **создания нового пользователя**:

1. Зайти на главную страницу **PHPMyAdmin**.
2. Выбрать пункт "**Привилегии**".

3. Перейти по ссылке "**Добавить нового пользователя**".



4. Настроить параметры нового пользователя и нажать на кнопку "**Пошёл**".



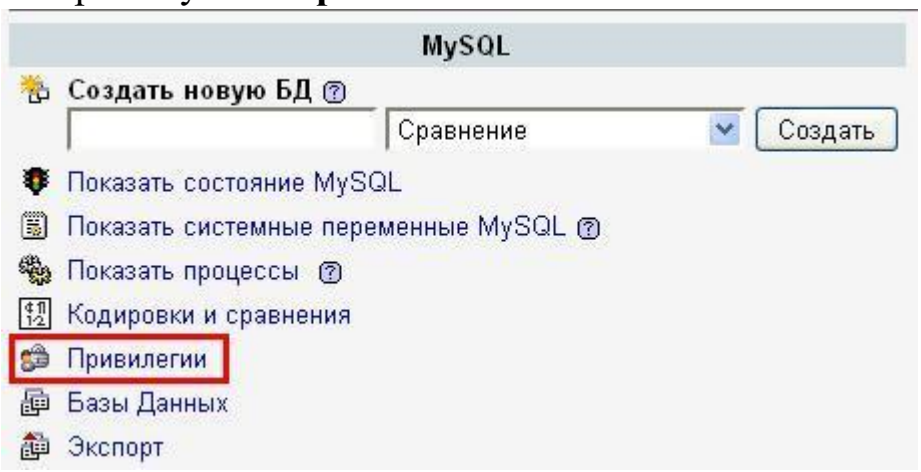
Теперь поговорим о параметрах нового пользователя подробнее:

- **Имя пользователя** - либо задавайте обычный логин, либо, если укажите в выпадающем списке "Любой пользователь", логин задавать не нужно.
- **Хост** - тот адрес, с которого данный пользователь может подключаться. Как правило, пишут "localhost", чтобы подключаться можно было только с этого же хоста, однако, иногда требуется подключение с других хостов. В таком случае нужно выбрать в выпадающем списке "**Любой хост**".
- **Пароль** - вводите пароль. Если не хотите использовать пароль, то можете выбрать в выпадающем списке "**Без пароля**". Обратите внимание: "**Без пароля**" - это не то же самое, что "**Любой пароль**". То есть если при подключении Вы укажите пароль для пользователя, у которого его нет, то будет ошибка авторизации.

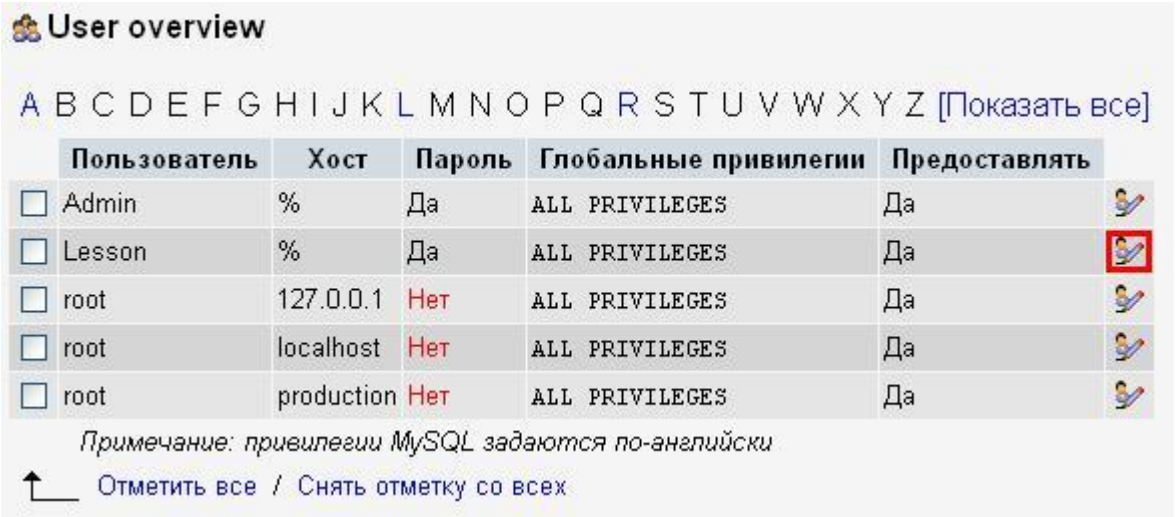
- **Подтверждение** - если указывали пароль, то повторите его ввод.
- **Глобальные привилегии** - подробно о каждой привилегии мы говорили в статье: Права пользователей в PHPMyAdmin

Теперь давайте подробно разберём, **как редактировать пользователей в PHPMyAdmin**:

1. Зайти на главную страницу **PHPMyAdmin**.
2. Выбрать пункт "**Привилегии**".



3. Выбрать пользователя, которого Вы хотите отредактировать.



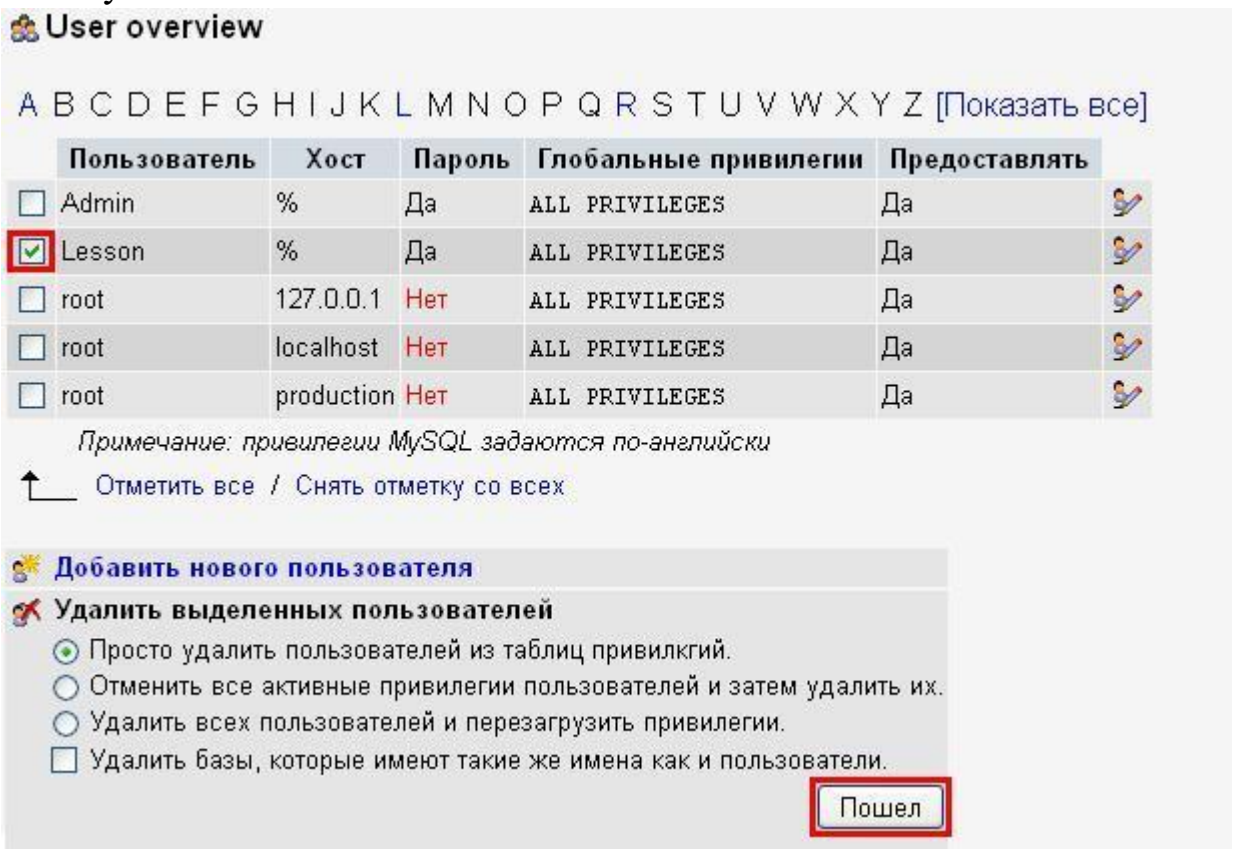
4. Изменить настройки пользователя и нажать на кнопку "**Пошёл**".

При изменении настроек существующих пользователей принцип тот же, что и при настройке новых пользователей.

И, наконец, разберём процесс **удаления пользователей в PHPMyAdmin**:

1. Зайти на главную страницу **PHPMyAdmin**.
2. Выбрать пункт "**Привилегии**".

3. Выбрать пользователей, которых Вы хотите удалить и нажать на кнопку "**Пошёл**".



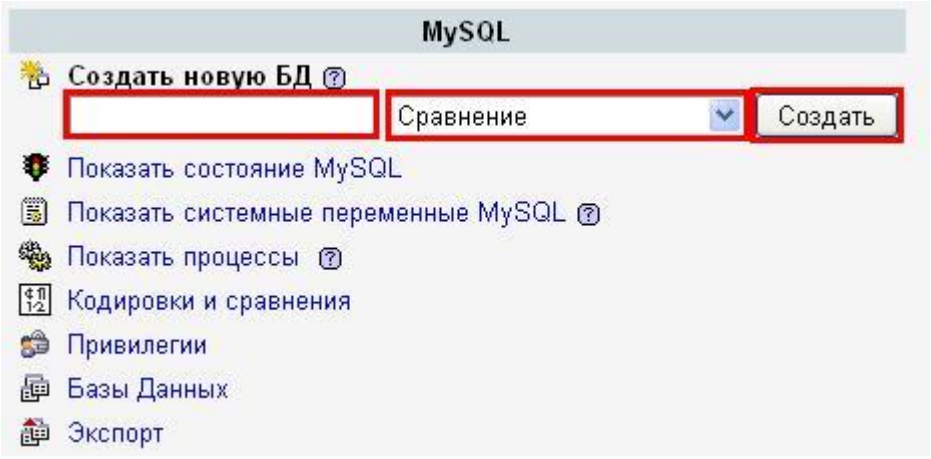
Как управлять базами данных в PHPMyAdmin.

С базами данных в PHPMyAdmin можно проводить следующие операции:

- Создавать.
- Редактировать.
- Удалять.

Последовательность действий при создании базы данных в PHPMyAdmin:

1. Зайти на главную страницу PHPMyAdmin.
2. Задайте имя для базы данных, выберите кодировку и нажмите на кнопку "Создать".

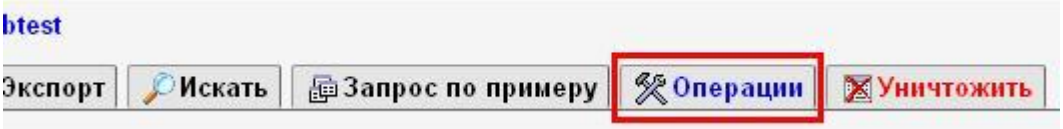


Здесь хочется остановиться на кодировке. Если у Вас на сайте будет только русские и латинские буквы, то ставьте кирилицу (cp1251_general_ci). А если у Вас будет мультиязычный сайт, то ставьте unicode (utf8_general_ci). Другие кодировки Вам вряд ли потребуются.

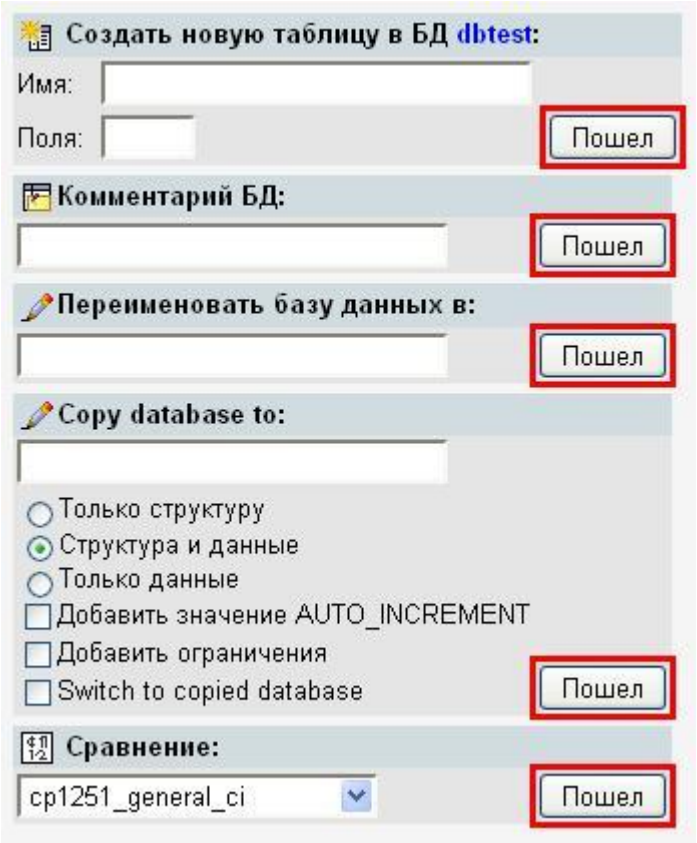
Теперь научимся редактировать базы данных в PHPMyAdmin. Для этого надо сделать следующее:

1. Зайти на главную страницу PHPMyAdmin.
2. Выбрать из выпадающего списка имя базы данных, которую Вы хотите отредактировать.

3. Перейти в пункт "Операции".

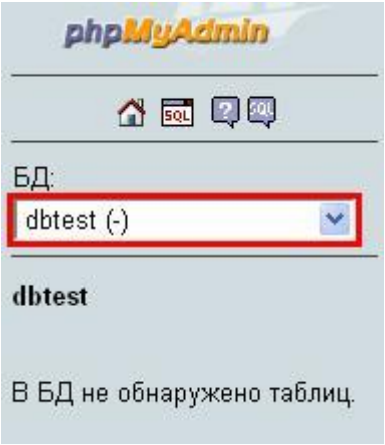


4. Отредактировать базу данных и нажать на соответствующую настройке кнопку "Пошёл".



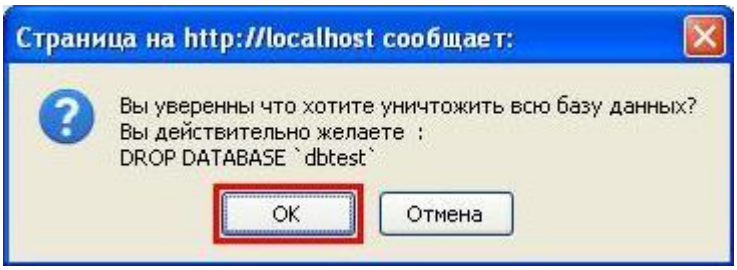
И, наконец, удаление баз данных в PHPMyAdmin:

1. Зайти на главную страницу PHPMyAdmin.
2. Выбрать из выпадающего списка имя базы данных, которую Вы хотите удалить.



3. Перейти в пункт "Уничтожить".

4. Подтвердить удаление базы данных.



Вопросы для самоконтроля:

1. а) Что входит в понятие манипулирования данными в БД?
   б) Какова цель запроса на выборку?
2. Напишите на языке запросов команду, формирующую таблицу расшифровки кодов специальностей. Строки должны быть упорядочены по возрастанию кодов.
3. Придумайте серию запросов к базе данных, построенной по индивидуальному заданию в практикуме.

Тема 5. Взаимодействие скриптов на языке PHP и базы данных MySQL

Подключение к базе данных из PHP файла. Вывод данных на PHP-страницу, попавших в выборку по SQL запросу. Передача параметров в запрос

4.5.1 Подключение к базе данных из PHP файла.

Неважно, насколько простые или сложные у вас сценарии, если они общаются с базой данных, они начинаются с одних и тех же нескольких действий:

1. Подключение к установленной базе данных MySQL.
2. Использование команды USE в отношении нужной базы данных MySQL.
3. Отправка SQL базе данных.
4. Получение результатов.
5. Обработка результатов.

Действия 3, 4 и 5 будут изменяться в зависимости от характера выполняемой работы. Сценарий, создающий таблицы, немного отличается от сценария, который ведет поиск в существующих таблицах. Но первые два действия — подключение к MySQL и использование нужной базы данных — всегда одни и те же, независимо от предназначения сценария.

При подключении к MySQL необходимо указывать сервер, пользователя и его пароль, а также базу данных, с которой требуется работать. Синтаксис подключения к MySQL имеет следующий вид:

```
<?php
mysql_connect("сервер","имя пользователя","пароль пользователя");
mysql_select_db("имя базы данных",идентификатор подключения к серверу)
?>
```

Рассмотрим подробно этот алгоритм подключения:

1. Соединяемся с MySQL сервером и получаем идентификатор.

Для того, чтобы подключиться к базе данных необходимо сначала соединиться с MySQL сервером. Для этого существует функция «mysql_connect», в которой указывается место нахождение сервера,

пользователь, который имеет право работать с сервером и пароль пользователя. Результат соединения можно занести в переменную, которая будет идентификатором подключения к MySQL серверу.

2. Выбираем базу данных, с которой будем работать.

На сервере может быть сразу несколько баз данных. За выбор БД отвечает функция «mysql_select_db». В качестве параметров этой функции указываются: имя базы данных и идентификатор подключения к серверу.

Рассмотрим реальный примеры соединения с сервером и БД:

```
<?php
$db = mysql_connect("localhost","admin","12345"); /*Подключение к серверу */
mysql_select_db("baza",$db); /*Подключение к базе данных на сервере*/
?>
```

Другой вариант соединения:

```
<?php
$host="localhost";/*Имя сервера*/
$user="admin";/*Имя пользователя*/
$password="12345";/*Пароль пользователя*/
$db="baza";/*Имя базы данных*/
mysql_connect($host, $user, $password); /*Подключение к серверу*/
mysql_select_db($db); /*Подключение к базе данных на сервере*/
?>
```

В данных примерах мы подключаемся к локальному серверу (localhost), пользователь у нас «admin», пароль пользователя «12345». Информация о подключении к серверу MySQL помещается в переменную «db», которая будет служить идентификатором подключения к MySQL. На втором шаге мы подключаемся к базе данных на сервере, которая имеет имя «baza», указываем идентификатор подключения к MySQL серверу (переменная «db»).

Обработка ошибок при подключении к MySQL

Иногда при подключении к MySQL могут возникать ошибки и соединиться с БД не получится. Поэтому нужно использовать обработчик ошибок, который будет выводить текстовое сообщение при ошибке.

Рассмотрим пример обработчика ошибок при подключении к серверу и БД.

<?php
$host="localhost";
$user="admin";
$password="12345";
$db="baza";
mysql_connect($host, $user, $password) or die("MySQL сервер недоступен!".mysql_error());
mysql_select_db($db) or die("Нет соединения с БД".mysql_error());
?>

4.5.2 Вывод данных на PHP-страницу, попавших в выборку по SQL запросу. Передача параметров в запрос.

Чаще всего для вывода данных из таблицы используется цикл и функция mysqli_fetch_array(). Она принимает в качестве входного параметра переменную, в которую записывается результат работы mysqli_query(). Пример:

view sourceprint?

```
01.<?php
02.$str= mysqli_connect('lo-
calhost', 'root', '', 'world');
03.$select= mysqli_query($str, "SELECT region,
name, continent FROM `world`.`country`;");
04.while ($r= mysqli_fetch_array($select)) {
05.echo $r['region'] . " ";
06.echo $r['name'] . " ";
07.echo $r['continent'] . "<br />";
08.}
09.mysqli_close($str);
10.?>
```

В этом примере вывода сортированных данных из базы MySQL PHP функция mysqli_fetch_array() преобразует результаты выборки в ассоциативный массив. В качестве ключей используются имена столбцов, указанные в запросе.

Обратите внимание на функцию mysqli_close(), вызов которой происходит в конце скрипта. Она закрывает соединение с БД. В качестве аргумента функция принимает идентификатор строки подключения.

```php
//header('Content-Type: text/html; charset=utf-8');
//error_reporting(0);
$str= mysqli_connect('localhost', 'root', '', 'world');
$select= mysqli_query($str, "SELECT region, name, continent FROM `world`.`country`;");
//$r=mysqli_fetch_array($res);
while ($r= mysqli_fetch_array($select)) {
        echo $r['region'] . " ";
        echo $r['name'] . " ";
        echo $r['continent'] . "<br />";
}

mysqli_close($str);
```

СУБД может использоваться не только для хранения строк, но и текста. Например, чтобы реализовать вывод новостей из базы данных в PHP, следует лишь немного изменить предыдущий код. А для столбца таблицы, в котором будет храниться текстовый контент, задать тип данных text.

*Получение списка полей таблицы.*В PHP и на этот случай есть своя команда - mysql_list_fields.

Синтаксис mysql_list_fields

ресурс mysql_list_fields (строка database_name, строка table_name [, ресурс link_identifier])

Эта функция возвращает список полей в таблице table_name в базе данных database_name. Получается, что выбирать базу данных нам было необязательно, но это пригодится позже. Как можно заметить, результат работы этой функции - переменная типа ресурс. То есть это не совсем то, что мы хотели получить. Это ссылка, которую можно использовать для получения информации о полях таблицы, включая их названия, типы и флаги.

Функция *mysql_field_name* возвращает имя поля, полученного в результате выполнения запроса. Функция *mysql_field_len* возвращает длину поля. Функция *mysql_field_type* возвращает тип поля, а функция *mysql_field_flags* возвращает список флагов поля, записанных через пробел. Типы поля могут быть int, real, string, *blob* и т.д. Флаги могут быть not_null,primary_key, unique_key, *blob*, auto_increment и т.д.

Синтаксис у всех этих команд одинаков:

строка mysql_field_name (ресурс result, целое field_offset)строка mysql_field_type (ресурс result, целое field_offset)строка mysql_field_flags (ресурс result, целое field_offset)строка mysql_field_len (ресурс result, целое field_offset)

Здесь result - это идентификатор результата запроса (например, запроса, отправленного функциями mysql_list_fields или*mysql_query* (о ней будет рассказано позднее)), a field_offset - порядковый номер поля в результате.

Вообще говоря, то, что возвращают функции типа mysql_list_fields или *mysql_query*, представляет собой таблицу, а точнее, указатель на нее. Чтобы получить из этой таблицы конкретные значения, нужно задействовать специальные функции, которые построчно читают эту таблицу. К таким функциям и относятся *mysql_field_name* и т.п. Чтобы перебрать все строки в таблице результата выполнения запроса, нужно знать число строк в этой таблице. Команда *mysql_num_rows* (ресурс result) возвращает число строк во множестве результатов result.

А теперь попробуем получить список полей таблицы *Artifacts* (коллекция экспонатов).

<?$conn = mysql_connect("localhost","nina","123")or die("Невозможно установить соединение: ". mysql_error());echo "Соединение установлено";mysql_select_db("book");$list_f = mysql_list_fields ("book","Artifacts",$conn); $n = mysql_num_fields($list_f);for($i=0;$i<$n; $i++){ $type = mysql_field_type($list_f, $i); $name_f = mysql_field_name($list_f,$i); $len = mysql_field_len($list_f, $i); $flags_str = mysql_field_flags ($list_f, $i);echo "
Имя поля: ". $name_f;echo "
Тип поля: ". $type;echo "
Длина поля: ". $len;echo "
Строка флагов поля: ". $flags_str . "<hr>";}?>

В результате должно получиться примерно вот что (если в таблице всего два поля, конечно):

Имя поля: idТип поля: intДлина поля: 11Строка флагов поля: not_null primary_key auto_increment_____ _Имя поля: titleТип поля: stringДлина поля: 255Строка флагов поля:

Вопросы для самоконтроля:

1.Как подключиться к базе данных из PHP файла?

2.Как происходит вывод данных на PHP-страницу, попавших в выборку по SQL запросу?

3.Как произвести передачу параметров в запрос?

ЗАКЛЮЧЕНИЕ

Одним из крупнейших достижений Internet стала "всемирная паутина" – WWW (World Wide Web, или просто Web). WWW представляет собой множество независимых, но взаимосвязанных серверов. Работая с Web, пользователь "перемещается" между серверами, то есть последовательно соединяется с ними и получает информацию, как правило, в виде гипертекста.

В содержание пособия были изложены основы программирования для Web с применением различных технологий.

В результате изучения курса студенты познакомятся с технологиями и концепцией создания статических и динамических web-страниц, принципами разработки структуры web-страниц, основными элементами языка, средствами оформления страниц, способами передачи данных от удаленного пользователя на сервер.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

Основная литература

1.А.А. Дуванов. HTML-конструирование (материалы Роботландского университета). //Информатика, №21-22, 2000.

2.А.А. Дуванов. Web-конструирование. HTML. — СПб.: БХВ-Петербург, 2003. — 325 с.

3.А.А. Дуванов. Web-конструирование. DHTML. — СПб.: БХВ-Петербург, 2003. — 512 с.

4.Молли Э. Хольцшлаг. Использование HTML 4: Пер. с англ.: Уч. пос. — М: Издательский дом «Вильямс», 2000. — 1008 с.

Дополнительная литература

1.А. Матросов, А. Сергеев, М. Чаунин. HTML 4.0. Наиболее полное руководство.

2.М. Браун, Д. Ханикат. HTML 3.2 в подлиннике.

3.В.А. Острейковский. Информатика. — М.: ВШ, 2000. — 319 с.

4.В. Холмогоров. Основы Web–мастерства. Учебный курс. — СПб: Питер, 2001. — 352 с.

5.Использование HTML 4: Пер. с англ. / Луиза Паттерсон, Сью Шарльворс, Джоди Корнелиус и др.: Уч. пос. — М.: Издательский дом «Вильямс», 2000. — 400 с.

6.С.Н. Коржинский. Настольная книга Web–мастера: эффективное применение HTML, CSS и JavaScript. М.: Издательский дом «КноРус», 2000. — 320 с.

ГЛОССАРИЙ

API (Application Programming Interface) – интерфейс прикладного программирования – совокупность открытых данных и методов класса.

AJAX (Asynchronous JavaSCript and XML) – это технология разработки web-приложений, в которой используются языки JavaScript, XML и другие механизмы; обычно сводится к применению объекта XMLHttp для обращения к серверу и изменения некоторых элементов страницы без ее полного обновления

const – это ключевое слово, появившееся в PHP 5, которое употребляется для определения константных (постоянных) членов класса

DOM (Document Object Model – объектная модель документа) – это представление HTML или XML-документа в объектно-ориентированном виде

extends – это ключевое слово, обозначающее наследование классу, например,: class Canary extends Bird

final – это модификатор, применяемый к методам или классам с целью ограничить возможность наследования. Класс с таким модификатором вообще нельзя наследовать, а метод нельзя переопределять в производных классах.

HTML (HyperTExt Markup Language) – это простой язык разметки, берущий начало от SGML и применяемый для составления web-страниц

implements – это ключевое слово, обозначающее наследование интерфейсу

iterator – это встроенный в PHP интерфейс, который позволяет обходить (перебирать) объект

javadoc-формат – это формат внутренних комментариев; метод getDocComment в различных классах отражения позволяет извлекать отформатированные таким образом комментарии из исходного текста

MIME (Multipurpose Internet Mail Extensions) – это стандарт электронной почты в Интернете. Более широко – спецификация типа содержимого, например, image/jpeg

PEAR (PHP Extension and Application Repository – репозитарий расширений и приложений PHP) – это архив программ с открытыми

исходными текстами, собранными в пакеты, которые легко установить с помощью интсаллятора PEAR

PDO (PHP Data Object) – это группа классов, представляющая собой абстрагированный доступ к базам данных. По умолчанию включается в версии PHP 5.1 и старше; имеются драйверы для всех распространенных баз данных, часто используемых совместно с PHP

private (закрытый) – это ключевое слово, применяемое к методам и данным класса. Закрытые элементы доступны только внутри класса. Извне класса для доступа к ним применяются специальные методы. Они не наследуются производными классами.

protected (защищенный) – это ключевое слово, применяемое к методам и данным класса. Как и закрытые, защищенные члены класса недоступны извне класса, но, в отличие от закрытых, наследуются производными классами

public (открытый) - это ключевое слово, применяемое к методам и данным класса. Открытые методы класса, составляющие его интерфейс, могут вызываться от имени экземпляров класса и наследуются производными классами.

RSS (Really Simple Syndication или Reach Site Summary) – это новостной канал, который представляет собой XML-документ строго определенного формата

SGML (Standard Generalized Markup Language) – международный стандарт представления документов; HTML и XML являются частными случаями SGML

SPL (Standard PHP Library) – набор классов, встроенных в PHP

static – это модификатор доступа применяемый к методу или свойству класса и позволяющий обращаться к этому элементу без создания экземпляра класса. Статические свойства являются общими для всех экземпляров класса.

W3C (WorldWideWebConsortium) – организация, ответственная за разработку стандартов для World Wide Web

WSDL (Web Services Definition language) – это язык на основе XML, на котором описываются web-сервисы

XHTML (eXtensible HyperTExt Markup Language) – это вариант HTML совместимый с XML

XML (eXtensible Markup Language) – это язык разметки, подобный HTML, который берет начало от SGML. XML-совместимые документы должны удовлетворять более строгим требованиям, чем предъявляются в HTML.

Zend – это языковой интерпретатор, лежащий в основе PHP; в PHP 5 был полностью переписан для поддержки объектно-ориентированных механизмов

Абстрактный метод – это метод, который объявлен, но не определен. Если в классе есть хотя бы один абстрактный метод, то ключевое слово abstract должно присутствовать и в определении класса. Все методы интерфейса абстрактны. Класс, в котором есть только абстрактные методы, называется абстрактным.

Агрегатный класс – это класс, содержащий хотя бы один член данных, также являющихся объектом.

Базовый класс – это класс, от которого произведены другие классы. Он также может называться родительским классом или суперклассом.

Будущая совместимость (forward compatibility) – написание кода с учетом будущих модернизаций языка, например, отказ от использования механизмов, объявленных устаревшими.

Верблюжья нотация – это соглашение о написании имен, согласно которому каждое слово в составном имени начинается с заглавной буквы. Имена классов при этом начинаются с заглавной буквы (DirectoryItems), а имена методов – со строчной (getName).

Волшебный метод – это метод, имя которого начинается с двух символов подчеркивания. Обычно вызывается неявно. Самые важные: __construct, __destruct, __clone.

Время вызова – это момент, в который вызывается функция или метод.

Данные-члены – это переменные, объявленные внутри класса, но вне любого метода; называются также свойствами или переменными экземпляра.

Деструктор – это противоположность конструктору. Автоматически вызывается, когда объект покидает область видимости. Обычно применяется для того, чтобы гарантировать освобождение ресурсов, например описателей файлов.

Инкапсуляция – это процедура, позволяющая скрывать детали реализации, несущественные для программиста-клиента, то есть не раскрывать их в виде открытых методов и данных-членов. Соотносится с сокрытием данных, но более всеобъемлюща.

Интерфейс – это понятие имеет несколько значений. 1) ключевое слово в PHP, обозначающее класс, в котором методы объявлены, но не определены. В PHP разрешено наследовать нескольким интерфейсам. 2) совокупность открытых методов класса.

Класс – это составной тип, обычно включающий данные и методы; самая фундаментальная концепция ООП.

Класс-потомок – см. производный класс.

Конструктор – это специальный метод, который вызывается в момент создания объекта = __construct

Метаданные – это данные, с помощью которых описываются другие данные; например, информация о структуре базы данных.

Метод доступа (геттеры и сеттеры) – это открытый метод, служащий для получения или изменения данных-членов. Методы доступа называются также методами get и set. Считается хорошим стилем делать данные-члены закрытыми, а обращаться к ним для чтения или изменения только с помощью методов доступа.

Метод – это функция, определенная внутри класса.

Наследование – это способность объектно-ориентированного языка передавать методы и данные существующего класса новому, то есть от родителя к потомку.

Обертка, обертывающий метод – это метод, который просто заключает в себе вызов другого метода или функции.

Область видимости – это контекст, в котором возможен доступ к переменной. Переменная, определенная внутри метода, видима только в этом методе, а областью видимости переменной экземпляра/объекта является весь класс.

Обратная совместимость – это свойство версии языка программирования или приложения, позволяющая работать с предыдущими версиями или файлами, созданными предыдущими версиями программы.

Оператор разрешения области видимости – это двойное двоеточие. Оператор:: употребляемый вместе с именем класса для ссылки на константы или статические методы класса.

Пара имя/значение – это формат строки запроса, передаваемой web-странице; любая строка запроса состоит из таких пар. Доступ к ним можно получить с помощью глобальных массивов $_POST и $_GET, причем имя выступает в качестве ключа.

Паттерн проектирования – это общее описание решения проблемы; нечто напоминающее абстрактный класс или интерфейс, но еще менее конкретное.

Перегруженный метод – это характеристика метода, когда речь идет о различном поведении в зависимости от параметров. В PHP этот термин обычно употребляется в отношении методов __call, __set, __get в том смысле, что один метод может обрабатывать различные свойства и методы. Поскольку PHP – слабо типизированный язык программирования, то в нем невозможна перегрузка в традиционном для других ОО-языков смысле – когда методы могут иметь одно имя, но разные сигнатуры.

Переменная класса – это статический член данных, принадлежащий классу в целом, а не его конкретному экземпляру.

Переопределение – это изменение определения метода базового класса в производном.

Поверхностное копирование – это побитовое копирование объекта. Следует избегать при копировании агрегатных объектов.

Полиморфизм – в строгом смысле, это возможность скопировать объект производного класса в переменную, принадлежащую базовому классу, так что при этом будут вызываться методы производного класса. В PHP применяется ослабленный вариант.

Программист-клиент – это пользователь класса в отличие от его автора; иногда называется программистом-пользователем.

Производный класс – это любой класс, у которого есть родительский или базовый класс. Также называется классом-потомком или подклассом.

Прототип – это объявление функции/метода, предшествующее ее определению (в некоторых языках это обязательно, но PHP к ним не относиться); может применяться и к объявлению методов, особенно методов интерфейса.

Процедурный – это вид языка программирования, в котором широко используются процедуры; например, язык C является процедурным, а PHP может рассматриваться и как процедурный, и как объектно-ориентированный язык.

Родительский класс – см. базовый класс.

Сбор мусора – это схема автоматического управления памятью, позволяющая освобождать неиспользуемые участки памяти без вмешательства программиста.

Свойство – это синоним переменной экземпляра или члена данных.

Сигнатура – это свойство, уникально характеризующее функцию или метод; состоит из имени метода, а также числа и типов его параметров. В PHP применяется в ослабленном варианте. В строго типизированных языках программирования возможно наличие методов с одинаковыми именами, но при условии, что число или типы их параметров/аргументов различны.

Слабо типизированный – этот термин употребляется для характеристики языка, подобного PHP, в котором при объявлении переменной не обязательно указывать ее тип.

Сокрытие данных – это возможность ограничить доступ к данным-членам. Также это называется защитой данных.

Сопоставление типов – это механизм, позволяющий в определении функции или метода указать перед именем его тип и тем самым запретить передачу параметров, принадлежащих типам, не совместимым с объявленным. В PHP 5.1 можно указывать также, что передается массив.

Строка запроса – это одна или несколько пар имя/значение, передаваемых web-странице в составе URL

Устаревший (deprecated) – это означает “более не рекомендуемый”. Устаревшие механизмы рано или поздно будут исключены из языка.

Экземпляр – это конкретное воплощение класса.

ТЕСТОВЫЕ ЗАДАНИЯ

Модуль 1 Введение в Web-конструирование

Задания А

Однозначный выбор

S: Какай тэг определяет заголовок документа HTML?
: HTML
: ISINDEX
: BODY
: HEAD

S: Какой тег существует?
: <CODE>
: <PR>
: <COLOR>
: <QUOTE>

S: Найдите неправильное определение гиперссылки:
: <A HREF="nextpage.htm" TITLE="nextpage "> nextpage </A>
: <A TITLE ="nextpage.htm" HREF="nextpage"> nextpage </A>
: <A HREF="nextpage.htm" TITLE =" nextpage"> nextpage </A>
: <A HREF="nextpage.htm"> nextpage </A>

S: Какой атрибут тега BODY позволяет изменять цвет "активных" гиперссылок?
: COLOR
: ALINK
: TEXT
: VLINK

S: Какой из вариантов содержит ошибку?
: < A HREF ="nextpage.html#top"> Ссылка </A>
: < A HREF ="nextpage.html#17"> Ссылка </A>
: < A HREF ="nextpage.html"> Ссылка </A>
: < A HREF ="nextpage.html"#top> Ссылка </A>

S: Какой атрибут тэга BODY позволяет задать цвет фона страницы?
: COLOR

: BGCOLOR
: BACKGROUND
: SET

S: Какая ошибка в следующем коде: <A HREF="nextpage.html"><B><I>Ссылка</I></A>?
: не закрыт тег <B>
: внутри тега <A> не может быть тега <B> или <I>
: не указан обязательный атрибут TITLE у тега <A>
: не указан обязательный атрибут ALT у тега <A>

S: Какой тег надо использовать чтобы внутри тега <P> несколько подряд идущих пробелов в тексте не вырезались?
: <CODE>
: <PRE>
: <SPAN>
: <ADDRESS>
S: Какой атрибут тега <IMG> задает горизонтальное расстояние между границей страницы и изображением?
: BORDER
: HSPACE
: VSPACE
: WIDTH

S: Выберите вариант при котором при наведении мыши на изображение, появляется всплывающая подсказка с текстом “Подсказка":
: <IMG SRC="photo.jpg" ALT ="Подсказка" TITLE="Изображение">
: <IMG SRC ="photo.jpg" ALT="Подсказка">
: <IMG SRC ="photo.jpg" TITLE ="Подсказка">
: <IMG SRC ="photo.jpg" ALT ="Изображение" TITLE ="Подсказка">

S: Что определяет атрибут CELLSPACING у элемента разметки TABLE?
: расстояние от содержания до границы ячейки
: расстояние между ячейками
: ширину границы
: ширину ячейки

S: Как сделать ширину таблицы на всю страницу?
: <TABLE>
: <TABLE WIDTH="100%">
: <TABLE WIDTH="auto">
: <TABLE WIDTH="100">

S: В какой таблице ширина промежутков между ячейками составит 20 пикселей?
: <TABLE CELLSPACING="20">
: <TABLE GRIDSPACING="20">
: <TABLE CELLPADDING="20">
: <TABLE BORDER="20">

S: Какой из тегов позволяет создавать нумерованные списки?
: OL
: DL
: UL
: DT

S: Какой вариант написан с ошибкой?
: <OL TYPE="A">
: <OL TYPE ="I">
: <OL TYPE ="N">
: <OL TYPE ="1">

S: Как правильно создать текстовое поле для ввода информации?
: <INPUT TYPE="TEXTFIELD">
: <TEXTINPUT TYPE="TEXT">
: <TEXTFIELD>
: <INPUT TYPE="TEXT">

S: Где будет расположен текст относительно рисунка в следующем коде?
<IMG src="image.jpg" align="bottom"> Text
: слева и снизу
: справа и снизу

: слева вверху
: под рисунком

S: В какой таблице текст выровнен по центру ячеек?
: <TABLE ALIGN=""CENTER"" WIDTH=""300"">
: <TABLE ALIGN=""LEFT"">
: нет правильного ответа
: <TABLE ALIGN=""LEFT"">

S: С помощью какого тега задается фреймовая структура документа?
: FRAME
: BODY
: FRAMESET
: IFRAME

S: Каким образом горизонтальную линию можно разместить на одной строке с текстом?
: <HR WIDTH=50>ТЕКСТ
: <HR ALIGN="LEFT" WIDTH="50">ТЕКСТ
: <NOBR><HR ALIGN="LEFT" WIDTH=50>ТЕКСТ</NOBR>
: нет правильного ответа
V3: **Наиболее правильный выбор.**
S: Web-страница - это:
+: [] документ, который предназначен для открытия в специализированной программе — браузере, и который содержит данные для отображения на экране компьютера с помощью соответствующего типа ПО различного полезного контента — текстов, ссылок, графики, видео, музыки и т. д
+: [] гипертекстовый ресурс Всемирной паутины, обычно написанный на языке ссылки для быстрого перехода на другие страницы, а также статические и динамические изображения.
+: [] документ специального формата, опубликованный в Internet
+: [] Компонент веб сайта, файл, расположенный на сервере.
S: Интернет- это :
+: [100] глобальная компьютерная сеть, которая объединяет в единое целое множество компьютерных сетей и отдельных компьютеров,

предоставляющих обширную информацию в общее пользование и не является коммерческой организацией ;
+: [] метасеть, состоящая из многих сетей, которые работают согласно протоколам семейства TCP/IP, объединены через шлюзы, используют единое адресное пространство и пространство имен.
+:[] протяженная коммуникационная сеть связи, работа в которой обеспечивается с помощью телекоммуникационных компаний.
+:[]информационная система связи, объединяющая множество компьютеров во всём мире
S: DNS(*Domain Name System* - доменная система имен)- это:
+: [] база данных, обеспечивающая преобразование доменных имен компьютеров, подключенных к Интернет, в числовые IP-адреса ;
+: [] используемые в Интернете протокол и система обозначений для сопоставления адресов IP и имен, понятных пользователю .
+: [] распределенная база данных, которая содержит информацию о компьютерах, включенных в сеть Internet.
+: [] компьютерная распределённая система для получения информации о доменах.
S:Домен-это :
+: [] отдельный уровень в многоуровневой системе имен Интернета, несущий определенную информационную нагрузку;
+: [] адрес ресурса, который будут вводить пользователи при обращении к сайту.
+: [] название сайта, которое используется вместо IP..
+: [] адрес сайта
S: URL (*Uniform Resource Locator*) -это:
+: [] адрес любого ресурса в Интернете с указанием того, с помощью какого протокола следует к нему обратиться, какую программу запустить на сервере и какой конкретно файл следует открыть ;
+: [] стандартный формат представления логического адреса информационных ресурсов в Internet.
+: [] единообразный локатор (определитель местонахождения) ресурса
+: [] унифицированный указатель местонахождения ресурсов
S: TCP (*Transmission Control Protocol)*- это:
+:[] один из основных протоколов передачи данных интернета, предназначенный для управления передачей данных ;

+: [] протокол транспортного уровня, поддерживающий надежную передачу потока данных с предварительным установлением связи между источником информации и ее получателем.
+: [] протокол, обеспечивающий сквозную доставку данных между прикладными процессами, запущенными на узлах, взаимодействующих по сети.
+: [] протокол, обеспечивающий сквозную доставку данных между прикладными процессами, запущенными на узлах, взаимодействующих по сети.
S: IP (Internet Protocol) -это:
+: [] является главным протоколом семейства, он реализует распространение информации в IP-сети и выполняется на третьем (сетевом) уровне модели ВОС;
+:[] протокол сетевого уровня сетевой модели OSI (Open Systems Interconnection), относится к протоколам, которые организуют соединения на основе коммутации каналов..
+: [] маршрутизируемый протокол сетевого уровня стекаTCP/IP.
+: [] уникальный числовой идентификатор принтера, ПК и прочего устройства, который является часть компьютерной сети, которая построена применением протокола TCP/IP.

S:HyperText Markup Language(HTML) - это:
+: [] язык разметки гипертекста – предназначен для написания гипертекстовых документов, публикуемых в World Wide Web. ;
+: [] язык разметки гипертекстовых документов.
+: [] стандартизированный язык разметки документов во Всемирной паутине.
+: [] основной способ хранения и передачи документов в Internet
S:Гипертекстовый документ -это:
+: [] текстовый файл, имеющий специальные метки, называемые тегами, которые впоследствии опознаются браузером и используются им для отображения содержимого файла на экране компьютера;
+: [] документ, который представляет собой связанную систему документов, в котором есть основной документ, который Вы читаете, и дополнительные, которые содержат информацию по вопросам, поднятым в тексте.
+: [] Текстовый документ, содержащий гиперссылки.

+: [] документ, который содержит видимые ссылки на другой документ.

S: Гиперссылка -это :

+: [] специальная конструкция языка HTML, которая позволяют щелчком мыши перейти к просмотру другого документа;

+: [] часть гипертекстового документа, ссылающаяся на другой элемент (команда, текст, заголовок, примечание, изображение) в самом документе, на другой объект (файл, каталог, приложение), расположенный на локальном диске или в компьютерной сети, либо на элементы этого объекта .

+: [] текстовый или графический условный элемент (код) гипертекстового документа.

+: [] связь между веб-страницами или файлами

V3: Множественный выбор

S: Какого тега НЕ существует?

: <PRE>

: <OL>

: <ADRESS>

: <H8>

S: Как указать выравнивание текста в ячейке таблицы?

: с помощью атрибута CELLPADDING

: с помощью атрибута CELLSPACING

: с помощью атрибута VALIGN

: с помощью атрибута ALIGN

S: Как объединить несколько ячеек таблицы?

: с помощью атрибута ROWSPAN

: с помощью атрибута CELLPADDING

: с помощью атрибута CELLSPACING

: с помощью атрибута COLSPAN

S: Какие из указанных тегов HTML позволяют изменять параметры шрифта?

: HEAD

: FONT
: H1
: IMG

S: Какие из указанных тегов позволяют создавать списки определений?
: DT
: DL
: UL
: OL

S: Какие теги можно использовать для создания текстовых полей ввода в форме?
: <INPUT TYPE=text>
: <SELECT>
: <OPTION>
: <TEXTAREA>

S: Какие методы можно использовать для отправки формы?
: GET
: POST
: HEAD
: MAILTO

S: С помощью чего можно запретить кэширование документа?
: <META HTTP-EQUIV="Pragma" CONTENT="no-cache">
: <META HTTP-EQUIV="Pragma" CONTENT="cache">
: <META HTTP-EQUIPT="Pragma" CONTENT="no-cache">
: <META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">

S: В каких примерах данные формы будут переданы обработчику как часть URL?
: <FORM METHOD=""GET"" ACTION=""http://www.ru/"">
: <FORM METHOD=""POST"" ACTION=""http://www.ru/doc.pl"">
: <FORM METHOD=""GET"" ACTION=""http://www.ru/cgi"">
: <FORM METHOD=""TRY""

ACTION=""http://www.ru/script.php?param=test"">

S: В каком примере кода определяется вывод горизонтальных фреймов?

: <FRAMESET cols="30%, 65%">
: <FRAMESET rows="20%, 85%">
: <FRAMESET rows="35%, 70%">
: <FRAMESET cols="50%, 90%">

S: Какие теги существуют в языке HTML?
: <A>
: <B>
: <C>
: <I>
: <P>
: <T>

S: Какой или какие из следующих фрагментов HTML-кода содержат ошибки?
: <A> Ссылка </A>
: <HREF> page.htm </HREF>
: <A HREF=page.htm> Ссылка </A>
: <A TARGET=page. htm> Ссылка </A>
: <A HREF="page.htm"> Ссылка </A>

S: В каких тегах может содержаться тег SCRIPT?
: <A>
: <P>
: <DIV>
: <HEAD>
: <TABLE>

S: Какие значения может принимать атрибут TYPE тега <INPUT>?
: CHECKBOX
: FILE
: TEXT
: RADIO
: SOURCE
: BUTTON

S: Какие из следующих фрагментов HTML-кода кнопки отправки данных содержат ошибки?
: <INPUT type=button value=submit>
: <INPUT type=submit value=OK>

: <INPUT type=submit name=OK>OK</INPUT>
: <INPUT type=submit name=OK>
: <INPUT type=image src=ok.gif value=OK>

S: Какие значения атрибута ALIGN используются для определения положения изображения относительно окружающего текста?
: LEFT
: RIGHT
: TOP
: BOTTOM
: BASELINE

S: Какие фрагменты HTML-кода содержат ошибки?
: <FORM action=mailto:mail@mail.ru method=post enctype=application/x-wwtf-form-urlencoded> </FORM>
: <FORM action=mailto: mail@mail.ru method=get enctype=text/plain> </FORM>
: <FORM action=mailto: mail@mail.ru method=post enctype=text/plain> </FORM>
: <FORM action=mailto: mail@mail.ru method=post enctype=plain> </FORM>
: <FORM action=mailto: mail@mail.ru method=post enctype=text> </FORM>

V2: Задания B

V3: Вписать правильный ответ

J: Тег _____ определяет текстовый абзац
+:

J: Тег _____ определяет одну ячейку таблицы.
+:

J: Какой атрибут тега table задает фоновый рисунок в таблице?
_____

+:
+:
J: Сколько столбцов в следующей таблице фреймов <FRAMESET rows="15%,85%" cols="33%,34%,33%">? _____
+:
+:

J: Размер окна браузера 800 пикселей. На страницу добавили блок с шириной 50%. Затем в этот блок добавили таблицу с шириной 40%. Какова будет ширина таблица в пикселях? _____
+:
+:

V3: Установить соответствие

Q: Сопоставьте атрибут соответствующему тегу:
L: A
L: BODY
L: IMG
L: FONT
R: TARGET
R: TEXT
R: SRC
R: SIZE

Q: Сопоставьте атрибут соответствующему тегу:
L: TABLE
L: TD
L: TR
L: COL
R: SUMMARY
R: ROWSPAN
R: CHAR
R: SPAN

Q: Сопоставьте формулу и подученный результат:

L: H²SO⁴
L: H₂SO₄
L: H2SO4
L: H<pow>2</pow>SO<pow>4</pow>
R: H^2SO^4
R: H_2SO_4
R: H*2*SO*4*
R: H2SO4

Q: Сопоставьте выражение и его результат:
L: Text
L: Text
L: <u>Text</u>
L: <s>Text</s>
R: **Text**
R: *Text*
R: <u>Text</u>
R: ~~Text~~

Q: Сопоставьте тег и выполняемую им функцию:
L: задает нумерованный список
L: задает маркированный список
L: задает список определений
L: задает определение термина в списке определений
R: OL
R: UL
R: DL
R: DD
V3: Установить последовательность
Q: Установить последовательность в доменном имени сервера
L: www,
L: microsoft
L: com
R: 1
R: 2
R: 3

Q: Установить последовательность протоколов в соответствии с их уровнем
L1: прикладной,
L2: транспортный.
L3: сетевой
L4: канальный
R1:
R2:
R3:
R4:
Q: Установить последоватебльность этапов технология подготовки и проведения поиска информации в Интернете
L1: Определение общей направленности запроса, его содержания ,
L2: Определение географических регионов поиска. .
L3: Отбор поисковых машин.
L4: Составление запросов к поисковым машинам
L5: Выполнение запроса и его уточнение.
R1:
R2:
R3:
R4:
R5:

V2: Задания С
V3: Задача
Укажите номер правильного ответа
S: Какой элемент формы является необходимым для передачи формы на сервер?
1): <TEXTAREA NAME=next>
2): <INPUT TYPE=button NAME=next>
3): <INPUT TYPE=submit NAME=next>
4): <INPUT TYPE=reset>

:
S: Как правильно записать химическую формулу серной кислоты (H2SO4)?
Укажите номер правильного ответа

1): H²SO⁴
2): H₂SO₄
3): H<pow>2</pow>SO<pow>4</pow>
4): H<sup>2SO<sup>4
:
S: Какой CSS-код необходимо задать, чтобы цвет посещённых и непосещённых ссылок был одним и тем же?
Укажите номер правильного ответа

1): A:LINK, A:VISITED {COLOR: YELLOW;}
2): A:ACTIVE, A:VISITED {COLOR: YELLOW;}
3): A:LINK {COLOR: YELLOW;}
4): A:LINK, A:ACTIVE {COLOR: YELLOW;}
:

S: Сколько ячеек содержит следующая таблица:
Укажите номер правильного ответа

<TABLE WIDTH="100" HEIGHT="50" BORDER="2">
<TR><TD> </TD><TD> </TD><TD> </TD></TR>
<TR COLSPAN="5"> <TD> </TD></TR>
</TABLE>
1): 2
2): 3
3): 4
4): 5
:
S: Каким образом можно поменять цвет одной ссылки, не меняя цвет остальных?
Укажите все номера правильных вариантов ответа

1): <A HREF="1.HTML"><FONT COLOR=RED>ССЫЛКА 1</FONT></A>
2): <FONT COLOR=RED><A HREF="1.HTML">ССЫЛКА 1</A></FONT>

3): <A HREF="1.HTML" STYLE="COLOR: RED">ССЫЛКА 1</A>
4): <BODY LINK=RED>
5): через CSS, определив для данной ссылки свой класс
:

Модуль 2. Инструментальные средства создания web-сайтов
Задания А
Однозначный выбор

S: Какое свойство задает шрифт для вывода текста в CSS?
: font-font
: font-family
: font-style
: font-type

S: Какое свойство задает цвет фона в CSS?
: color
: bgcolor
: back-color
: background-color

S: Как сделать жирным текст во всех элементах <p> в CSS?
: p (font-weight:bold)
: p {font-weight:bold}
: p (font-size:bold)
: p {font-size:bold}

S: Как изменить цвет фона для всех элементов h4 на странице в CSS?
: h4 {background-color: green;}
: h4:all {background-color: green;}
: h4.all {background-color: green;}
: h4[all] {background-color: green;}

S: Выберите правильный селектор в CSS при
Text :
: doc1 { background: green }
:.doc1 { background: green }
: #doc1 { background: green }
: div.doc1 { background: green }

S: С помощью какого свойства можно выровнять содержимое блочного элемента по горизонтали в CSS?

: align:center
: text-align:center
: content-align: middle
: horizontal-align:center

S: С помощью какого свойства можно задать расстояние между областью содержания элемента и его рамкой?
: padding
: margin
: max-width
: pacing

S: Какое свойство в CSS используется для определения стиля для ссылки, когда на нее наведен курсор мыши, но ссылка еще не активирована?
: hover
: visited
: link
: vlink

S: Какое свойство в CSS используется для определения стиля самой ссылки?
: hover
: visited
: vlink
: link

S: Какое событие используется для определения полной загрузки страницы?
: onloadend
: onload
: endload
: loadpage

S: Какое событие используется для определения щелчка левой кнопки мыши?
: onclick

: ondblclick
: onmouseup
: onmouseover

S: Выберите правильный вариант создания новой переменной в VBScript:
: name = ‘Joe’;
: string name = "Joe";
: dim name = "Joe";
: var name = ‘Joe’;

S: Каждое выражение в VBScript должно заканчиваться:
: двоеточие (:)
: точкой (.)
: точкой с запятой (;)
: нет правильного ответа

S: Какой из следующих операторов используется в VBScript для объединения строк?
: .
: &
: ->
: =>

S: Какой оператор обозначает равенство значений в языке VBScript?
: ===
: ==
: =
: :=
S: Как определить константу в языке VBScript?
: Constant("Name", "Joe");
: Static Name="Joe";
: Const Name="Joe";
: Dim Name=’Joe’;

S: Какой оператор цикла существует в языке VBScript?

: While … EndWhile
: Do …While
: Do … Loop
: Do … Until

S: Какую функцию в языке VBScript нужно использовать чтобы получить только текущую дату?
: Date
: DateNow
: CurrentDate
: Now

S: Какая функция в языке VBScript возвращает случайное число от 0 до 1?
: Rnd
: Random
: Randomize
: GetRandomDigit

S: Что нужно использовать в языке VBScript, чтобы функция вернула значение?
: присвоить значение имени функции
: использовать оператор return
: передать значение в последней строке функции
: нет правильного ответа

S: Какая функция в языке VBScript приводит строку к нижнему регистру?
: LCase
: LString
: UCase
: UString
V3: **Наиболее правильный выбор.**
S: Структура сайта - это:
+: [] логическая разметка и физическая связка страниц сайта, а так же, расположение видимых элементов дизайна, обусловленная стандартами разработки сайтов;

+: [] основа для выстраивания последовательности и формы отображения имеющихся данных на сайте.
+: [] некий порядок, определяющий логическое расположение страниц ресурса и отдельных его частей при отражении на экране.
+: [] описание простыми словами списка разделов сайта, сервисов и содержания всех страниц сайта, а так же связи между разделами.

S: Веб-дизайн-это :

+: [] проектирование внешнего вида, структуры и способа функционирования веб-сайта, а также результат этого проектирования;
+: [] отрасль веб-разработки и разновидность дизайна, в задачи которой входит проектирование пользовательских веб-интерфейсов для сайтов или веб-приложений..
+:[] это процесс производства веб-сайтов, который включает техническую разработку, структурирование информации, визуальный (графический) дизайн и доставку по сети
+:[] разработка сайта, которая включает в себя все этапы, начиная от идеи, и заканчивая готовым информационным продуктом.

S: Веб-сервер-это:

+: [] сервер, отвечающий за прием и обработку запросов (HTTP-запросов) от клиентов к веб-сайту;
+: [] сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.
+:[] программное обеспечение, выполняющее обслуживание запросов HTTP, и физический компьютер, на котором работает это программное обеспечение.
+:[] программный продукт, устанавливаемый под управлением операционной системы сервера.

S: Хостинг -это:

+: [] услуга, которую предоставляет хостинговая компания, позволяющая вам размешать свои веб-сайты на её серверах ;
+: [] услуга по предоставлению ресурсов для размещения информации на сервере, постоянно находящемся в сети.
+:[] услуга по предоставлению дискового пространства для физического размещения информации на сервере, который находится в сети беспрерывно и круглосуточно

+:[] способ размещения сайта в сети интернет.

S: FTP(англ. *File Transfer Protocol* — протокол передачи файлов) - это:

+: [] стандартный протокол, предназначенный для передачи файлов по TCP-сетям ;

+: [] протокол передачи файлов и в то же время это сервис, позволяющий организовывать доступ к файловым архивам.

+:[] протокол, предназначенный для передачи файлов в компьютерных сетях.

+:[] протокол, обеспечивающий пересылку файлов между двумя, возможно, разнородными машинами

S: Системы управления контентом (CMS) – это:

+: [] информационная система или компьютерная программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления содержимым.

+: [] система управления контентом сайта, либо иначе ее еще называют «движок сайта»;

+:[] комплекс решений, на базе которого создается и администрируется ресурс

+:[] программная оболочка, позволяющая пользователям, не знающим основ HTML, CSS и PHP размещать свои материалы на сайте и, впоследствии, управлять ими.

S: Каталог -это:

: [] разбитые по тематикам ссылки на различные ресурсы Интернета;

: [] способ организации *файлов*, при котором каждому каталогу соответствуют свои группы файлов, как правило созданные при установке различных пакетов программ или самим пользователем.

:[] группа файлов на одном носителе, объединенных по какому-либо критерию.

:[] печатный или электронный список, перечень товаров и услуг, предлагаемых компанией.

S: Поисковые системы-это:

: [] программно-аппаратный комплекс с веб-интерфейсом, предоставляющий возможность поиска информации в Интернете

: [] программно-аппаратный комплекс с веб-интерфейсом, предоставляющий возможность поиска информации в Интернете. .

:[] компьютерная система, предназначенная для поиска информации

+:[] база данных по определенной информации в интернете

S: HTTP (HyperText Transfer Protocol) -это:

+: [] сетевой протокол прикладного уровня передачи данных;

+: [] протокол передачи гипертекстовой информации – транспортный протокол передачи гипертекста.

+:[] Протокол передачи файлов, подобный FTP, но со встроенным идентификатором типа передаваемой информации

+:[] протокол передачи гипертекстовых файлов

S: Веб-браузер-это:

+: [] прикладное программное обеспечение для просмотра веб-страниц, содержания веб-документов, компьютерных файлов и их каталогов; управления веб-приложениями; а также для решения других задач ;

+: []. приложение, с помощью которого те, кто имеет доступ к сети Интернет, могли бы обозревать веб-сайты с текстовой, ссылочной, графической, анимационной и прочей информацией, которая присутствует как на страницах веб-сайтов, так и локальных сетках.

+:[] программа, благодаря которой человек имеет доступ к веб страницам в глобальной сети – интернет.

+:[] специальное приложение, позволяющее просматривать **веб**-страницы в интернете

V3: Множественный выбор

S: Выберите правильные варианты задания цвета:

: color: #000

: color: #aaa

: color: #hhh

: color: #aaaaaa

S: Какое свойство в CSS устанавливает 50% прозрачность элемента?

: opacity: 50%

: opacity: 0.5

: opacity: .5

: opacity: 1/2

S: Выберите все фрагменты CSS содержащие синтаксические ошибки:
: {body;color:black}
: {body:color=black}
: body { color: black }
: body (color: black)

S: Какие из перечисленных свойств может быть задано в процентах?
: border
: padding
: height
: width

S: Какие из перечисленных вариантов написания свойства border, выведут тонкую черную сплошную рамку?
: border: 1px solid #000
: border: solid 1px #000
: border: #000 solid 1px
: border: #000 1px solid
: border: solid #000 1px

S: Какие фрагменты кода позволяет убрать рамку вокруг графической ссылки
:
:
:
:

S: Как задаются комментарии в языке VBScript? -: // это комментарий
: /* это комментарий */
: ‘это комментарий
: REM это комментарий

S: В имени какой из этих переменных в языке VBScript допущена ошибка?
: dim _myvariable
: dim my_variable
: dim 1myvariable
: dim my-variable

S: Какие операторы используются в языке VBScript для объявления переменных?
: dim
: private
: public
: local

S: Выберите правильные варианты написания оператора IF:
: If i<10 Then i++

```
If (i<10) :
 i++;
```

: EndIf

```
If (i<10) Then
   i++
```

: EndIf

```
   If (i<10)
   { i++;
```

-: }

V2: Задания В

V3: Вписать правильный ответ

J: Сколько основных типов данных есть в языке VBScript?

_____
+:
+:

J: Сколько элементов в следующем массиве: public arr(5)? _____
+:
+:

J: Функция _____ в языке VBScript возвращает целую часть числа.
+:
+:

J: Ключевым словом _____ начинается объявление процедуры.
+:

J: Функция _____ возвращает аргумент, преобразованный в целочисленный тип.
+:

V3: Установить соответствие

Q: Сопоставьте логический оператор его описанию, при котором выражение истинно:
L: A Or B
L: A And B
L: A Xor B
L: Not A
R: A или B истинны
R: A и B истинны
R: A или B истинны, но не одновременно
R: A не истинно

Q: Сопоставьте предопределенную константу и ее значение:
L: vbCr
L: vbLf
L: vbCrLf
L: vbTab
R: \r
R: \f
R: \n
R: \t

Q: Сопоставьте функцию ее описанию:
L: Asc
L: Chr
L: CStr
L: CSng
R: возвращает кодовый номер ANSI первого символа в строке
R: возвращает строку символов с соответствующими номерами
R: возвращает аргумент, преобразованный в строковый тип
R: возвращает аргумент, преобразованный в числовой тип

Q: Сопоставьте выражение и результат ее действия:
L: Fix(-12.34)
L: Int(-12.34)
L: FormatNumber (-12.34,1)
L: FormatNumber (-12.34,-1)
R: -12
R: -13
R: -12.34
R: -12.3

Q: Имеется строка str=”aabbccdd”. Сопоставьте функцию и результат ее действия со строкой:
L: Mid(str,2,1)
L: Replace(str,”b”,”c”,5)
L: Replace(str,”b”,”c”,,1)
L: Right(str, 6)
R: abbccdd
R: aabbccdd
R: aacbccdd
R: bbccdd
V3: Установить последовательность
Q: Установите последовательность шагов, при определении целей и задач сайта
L1: Составление списка вопросов
L2: Определение целей по шаблонной заготовке (для коммерческих тематик)
L3: Прорисовка сценариев посещений.

L4: Вывод.
R1:
R2:
R3:
R4:
Q: Восстановите последовательность записей в протоколе FTP
L1: ftp://
L2: ftp.7bloggers.ru
L3: /FTP/
L4: humans.txt
R1:
R2:
R3:
R4:
Q: Восстановите последовательность записей в протоколе FTP

L1: ftp://
L2: UNIQUE_USER:STRONG_PASS@
L3: ftp.7bloggers.ru
L4: /FTP/humans.txt
R1:
R2:
R3:
R4:

V2: Задания С
V3: Задачи
Укажите номер правильного ответа
S: Какой будет цвет у слова "Text"?
CSS: ul li em {color: red;}
HTML: <ul> Text
-1): красный
-2): чёрный
-3): белый
+4): цвет по умолчанию
+:
S: Какой будет цвет у слова "Text"?

```
CSS:    ul li em {color: red;}
HTML: <ul> <li> <strong> <em> Text </em> </strong> </li> </ul>
```

Укажите номер правильного ответа

-1): чёрный
-2): белый
+3): красный
-4): цвет по умолчанию
+:

S: Какой будет цвет у слова "Text"?

```
CSS:    .highlight {color: red;}
HTML:  <span class="highlight"> Text </span>
```

Укажите номер правильного ответа

-1): чёрный
+2): красный
-3): белый
-4): цвет по умолчанию
+:

S: Что произойдёт в результате выполнения данной строки - A ^ B?

Укажите номер правильного ответа

-1): будет возвращен остаток от деления B на A
-2): будет возвращен остаток от деления A на B
+3): A будет возведена в степень B
-4): произойдёт сравнение A и B
+:

S: Как очистить массив Names в VBScript?

Укажите номер правильного ответа

-1): Clear Names
+2): Erase Names
-3): Bound Names
-4): Delete Names
+:

S: Каким будет результат выполнения следующего кода?

Укажите номер правильного ответа

```
a = FormatNumber(-00.13,1,-1,-1,vbTrue)
```

-1): 0.13
-2): -00.13
+3): (-0.1)
-4): (-0.13)
+:

S: Каким будет результат выполнения следующего кода? Укажите номер правильного ответа

```
Function func(a, b)
   Dim rezult
   a = a+b
   func = a
End Function
a = 0
b = 1
for i = 1 to 10
   c = func a, b
   MsgBox(c)&","
next
```

-1): 1,2,3,4,5,6,7,8,9
-2): 1,1,1,1,1,1,1,1,1,1
-3): 1,2,3,5,8,13,21,34,55,89
+4): 1,2,3,4,5,6,7,8,9,10
+:

S: Каким будет результат выполнения следующего кода? Укажите номер правильного ответа

```
Function func(a, b)
   Dim rezult
   a = a+b
   func = a
End Function
a = 0
b = 1
for i = 1 to 10  step 2
   c = func(a, b)
   MsgBox (c) &","
next
```

-1): 1,2,3,4,5,6,7,8,9,10
-2): 1,1,1,1,1,1,1,1,1,1
-3): 1, 2, 3, 4, 5
+4): 1,1,1,1,1
+:

Модуль 3. Программирование на JavaScript

Задания А

Однозначный выбор

S: Как правильно задать комментарий?

: <!-- комментарий -->

: // комментарий

: # комментарий

: /* комментарий */

S: Какой из следующих управляющих символов используется для табуляции?

: \d

: \t

: \p

: \n

S: Какой из следующих операторов заключает в себе строку с двойными кавычками?

: q{ }

: qq{ }

: qx{ }-: enclose{}

S: Какое из данных чисел является шестнадцатеричным?

: -0x0C

: 0xH1

: 0a11

: 0123

S: Какая из следующих управляющих конструкций повторяет группу операторов до тех пор, пока данное условие не станет истинным?

: while

: until

: for

: ничего из вышеперечисленного

S: Какой оператор Perl позволяет получить остаток от целочисленного деления?
: %
: //
: /
: mod

S: Какой из следующих операторов используется для объединения строк?
: .
: &
: +
: ||

S: Чем цикл while отличается от цикла do..while?
: блок действий цикла do..while гарантированно выполняется один раз
: блок действий цикла while гарантированно выполняется один раз
: while выполняет блок действий только один раз
: ничем

S: Какой из следующих операторов используется для возведения в степень?
: %
: ^
: ^^
: **

S: Начало модуля определяется следующей командой:
: include
: package
: begin
: module

S: Какая команда используется для подключения модуля?
: include
: require
: use
: usage

S: Какая функция выполняет перевод символов строки в верхний регистр?
: uc
: upper
: ^^
: qw

S: Какая функция выполняет перевод символов строки в нижний регистр?
: ls
: lc
: lr
: low

S: Как правильно определить константу?
: $const=\123
: @const =123
: $const =123
: @const =\123

S: Какой из следующих операторов проверяет равно ли значение двух переменных, и возвращает -1, 0, или 1?
: ==
: ===
: <=>
: &&

S: Какой из следующих операторов возвращает истину, если левый аргумент больше или равен правому аргументу?
: lt
: gt
: le
: ge

S: Какой из следующих операторов используется, когда текущее значение переменной должно быть видимым для вызываемых подпрограмм?

: my
: local
: state
: ничего из вышеперечисленного

S: Какой оператор используется для немедленного прерывания цикла без продолжения
: break
: last
: next
: stop

S: Какой метасимвол используется для определения хэша
: @
: %
: $
: $@
V3: **Наиболее правильный выбор.**
S: Язык программирования JavaScript -это:
+: [] объектно-ориентированный язык разработки встраиваемых приложений, выполняющихся как на стороне клиента, так и на стороне сервера.
+: [] средство и среда написания сценариев, выполняющихся на стороне клиента и сервера — программный код считывается браузером и реализуется на компьютере пользователя.
+: [] мультипарадигменный язык программирования.
+: [] это «безопасный» язык программирования общего назначения.
S: Выражение в JavaScript-это :
+: [] комбинация переменных, литералов и операторов, в результате вычисления которой получается одно единственное значение.
+: [] образец или шаблон, созданный с применением специальных символов и описывающий одну или более текстовую строку
+: [] комбинации операндов и операторов
+: [] любая строка, написанная на языке JavaScript
S: В JavaScript функцией называется:
+: [] именованная часть программного кода, которая выполняется только при обращении к ней посредством указания ее имени.

+: [] самостоятельная программа, выполняющая конкретно заданное действие.
+: [] отдельный блок кода, который состоит из одного или больше операторов.
+: [] особый вид переменных
S: Массив в JavaScript-:
+: [] упорядоченный набор однородных данных, к элементам которого можно обращаться по имени и индексу и который не имеет встроенного типа данных для создания массивов, поэтому для решения используется объект Array и его методы.
+: [] являются нетипизированными: элементы массива могут иметь любой тип, причем разные элементы одного и того же массива могут иметь разные типы. Элементы массива могут даже быть объектами или другими массивами, что позволяет создавать сложные структуры данных, такие как массивы объектов и массивы массивов.
+: [] являются динамическими: они могут увеличиваться и уменьшаться в размерах по мере необходимости
+: [] тип данных, объединяющий один или несколько элементов (значений), каждый из которых имеет свой уникальный индекс (или номер).
S: Событие в JavaScript :
+: [] совокупность определенных действий со стороны пользователя и ответных реакций встроенного в интернет-браузер интерпретатора на программный сценарий.
+: [] это определённое действие, которые вызвано либо пользователем, либо браузером
+: [] это реакция компьютера на действие пользователя.
+: [] это то, что произошло или происходит. Например, браузер генерирует событие, когда завершается загрузка документа, когда пользователь водит курсором мыши по веб-странице или нажимает клавишу на клавиатуре.
S: DHTML (динамический HTML) -это:
+: [] набор средств, которые позволяют создавать более интерактивные Web-страницы без увеличения загрузки сервера.
+: [] способ (подход) создания интерактивного веб-сайта, использующий сочетание статичного языка разметки HTML, встраиваемого (и

выполняемого на стороне клиента) скриптового языка JavaScript, CSS (каскадных таблиц стилей) и DOM (объектной модели документа).
+: [] Расширение языка HTML, позволяющее добавлять интерактивные возможности и графику на веб-страницы
+: [] Язык, являющийся расширением HTML и CSS.
S: Объектная модель документа :
+:[]Независящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получитьдоступ к содержимому HTML, XHTML и XML-документов, а также изменять содержимое, структуру иоформление таких документов
+: [] Разработанная W3C модель для представления XML документов в терминах объектной парадигмы
+: [] рассматривает части документа как объекты, а документ как иерархия таких объектов.
+: [] любой документ представляет в виде логической древовидной структуры
S: Объект event- это:
+: [] статический объект, реализующий интерфейс обращения к системным событиям программного комплекса Интеллект
+: [] это объект, описывающий изменение состояния источника, с которым оно связано.
+: [] объект, который позволяет получить доступ к информации по событию
+: [] это объект событие календаря.
S: Объекты скриптового языка JavaScript -:
+: [] это неупорядоченный набор свойств, каждое из которых имеет нуль или более атрибутов, которые определяют, как это свойство может использоваться.
+: [] это сложные сущности, позволяющие хранить сразу несколько значений разных типов данных, они представляют собой блоки, из которых строится JavaScript.
+: [] это составной тип данных, который может хранить в себе числа, строки, булев тип данных и т.д.
+: [] применяются для возвращения значений и изменения состояния форм, страниц, браузера и определенных программистом переменных. Объекты можно сопоставить с существительными.
S: Каскадные таблицы стилей -это:

+:[] одна из составляющих DHTML. с помощью которой определяется внешний вид отображаемого HTML-документа: цвет шрифта и фона документа, сам шрифт, разбивка текста и многое другое.
+:[] Технология описания внешнего вида документа, написанного языком разметки.
+: [] язык, содержащий набор свойств для описания внешнего вида любых HTML документов.
+: [] текстовый файл, обычно сохраняемый с расширением.

V3: Множественный выбор

S: В имени какой из этих переменных допущена ошибка?
: $_myvariable
: $my_variable
: $1myvariable
: $my-variable

S: Выберите варианты некорректного присваивания переменной $a строки "abcd":
: $a="abcd"
: $a='abcd'
: $a=\"abcd\"
: $a=`abcd`

S: Выберите варианты некорректного определения переменной:
: in $a
: my $a
: local $a
: private $a

S: Какие ключевые слова используются для указания области видимости переменных?
: local
: my
: your
: our
: remote

S: Что из перечисленного является числовыми литералами?
: .123
: 12_345
: "123"
: 12e+3

S: Укажите операторы, которые задают цикл:
: while
: until
: for
: foreach

S: Каким способом можно выдержать паузу длительностью три секунды?
: wait(3)
: sleep(3)
: pause(3)
: alarm(3)

S: Какие из приведенных ниже фрагментов кода позволят производить подстановку переменной $name в тексте?
: $name ='Joe'; print "My name is $name";
: $name ='Joe'; print 'My name is $name ';
: $name ='Joe'; print 'My name is “$name”';
: $lang='Php'; print "My name is ‘$name’";

S: Какими способами можно удалить концевой символ в строке?
: print chop($str);
: print chop($str,1);
: chop($str); print $str;
: print substr($str,0,length($str)-1);

V2: Задания В

V3: Вписать правильный ответ

J: Каким будет результат выполнения следующего кода print "5"; close STDOUT; print "6";? _____
+:
+:

J: Каким будет результат выполнения следующего кода $a = 010; $b = 21; print $a + $b;? _____
+:
+:

J: Какой оператор используется для сравнения строк на абсолютное равенство? _____
+:

J: Каким ключевым словом обозначается оператор безусловного перехода? _____
+:

J: Каким будет результат выполнения следующего кода
$a=0*10; $b=21; print $a+$b;_____
+:
+:

J: Какой оператор позволяет повторить выполнение тела цикла без проверки условия цикла? _____
+:

V3: Установить соответствие

Q: Заданы переменные $a=12, $b=4. Сопоставьте выражение и результат ее действия:
L: $a -=$b++
L: $a /= $b+2
L: $a %= --$a
L: $a *= $b-3
R: 8
R: 2

R: 0
R: 12

Q: Сопоставьте операцию и ее описание:
L: $a -=$b++
L: $a /= $b+2
L: $a %= --$a
L: $a *= $b-3
R: 8
R: 2
R: 0
R: 12

Q: Сопоставьте логический оператор его описанию:
L: ne
L: le
L: gt
L: eq
R: истина, если операнды не равны
R: истина, если левый операнд не больше правого
R: истина, если левый операнд больше правого
R: истина, если операнды равны

Q: Задана строка $s=”aabbccdd”. Сопоставьте выражение и результат его действия:
L: $s =~s/d//
L: $s =~s/c/d/g
L: $s =~s/d/c/g
L: $s =~s//d/
R: aabbccd
R: aabbdddd
R: aabbcccc
R: daabbccdd

Q: Задана строка $s=”aabbccdd”. Сопоставьте выражение и результат его действия:
L: $s =~tr/d/c/;

L: $s =~tr/[c-d]/c/s;
L: $s =~tr/[c-d]/a/c;
L: $s =~tr/[a-b]/[b-c]/;
R: aabbcccc
R: aabbc
R3: aabbccd
R: bbccccdd
V3: **Установить последовательность.**
Q: Пусть в документе задана форма с двумя полями ввода:

<form name="form1">
Фамилия: <input type = "text" name = "student" size = 20>
Курс: <input type = "text" name = "course" size = 2>
</form>

Установить последовательность записей в ссылке для получения фамилии студента, введенного в первом поле ввода, в программе JavaScript:

L1: document.
L2: form .
L3: student.
L4: value
R1:
R2:
R3:
R 4:

Q: : Пусть в документе задана форма с двумя полями ввода:

<form name="form1">
Фамилия: <input type = "text" name = "student" size = 20>
Курс: <input type = "text" name = "course" size = 2>
</form>

Установить последовательность записей в ссылке для определения курса, на котором обучается студент:

L1: document.
L2:form.
L3:course.
L4:value.
R1:

R2:
R3:
R 4:

Q: Установить последовательность объектов языка JavaScript в соответствии с их иерархией (сверху вниз):

L1: объект window
L2: объект frame
L3: объект document
L4: объект location
R1:
R2:
R3:
R 4:

V2: Задания С
V3: Задачи

S: Что будет присвоено переменной $b?
Укажите номер правильного ответа

```
$a = 43;  $b = $a++;
```

-1): 1
+2): 43
-3): 44
-4): нет правильного ответа
+

S: Чему будет равна переменная $a после выполнения кода

```
$a= (1,2,3)x3
```

?

Укажите номер правильного ответа

-1): 123123123
-2): 123, 123, 123
+3): 333
-4): 3, 3, 3
+

S: Что будет занесено в переменную $res в результате выполнения

```
$res = "aaa" cmp "bbb"
```

?

Укажите номер правильного ответа

1): 1
2): 0
3): -1
4): equal
+
S: Как преобразовать шестнадцатеричное число 0xCC в десятичное и вывести результат?
Укажите номер правильного ответа

1): print hex->dec(0xCC)
2): print dec(0xCC)
3): print hex(CC)
4): print dec(CC)
+
S: Каким будет результат выполнения следующего кода
$a = "zzz"; print ++$a;**?**
Укажите номер правильного ответа

1): 1
2): Zzz
3): aaaa
4): zzz
+
S: Каким будет результат выполнения следующего кода
@arr = (1, 3, sort 4, 2); print @arr;**?**
Укажите номер правильного ответа

1): 1342
2): 1324
3): 1234
4): 1 3 4 2
+
S: Каким будет результат выполнения следующего кода?
Укажите номер правильного ответа

```
@a = ("один", "два", "три");
print "Считаем: @{[@a]}";
```

1): Считаем:
2): Считаем:3
3): Считаем: одиндватри
4): Считаем: один два три
+

S: Как правильно заменить строчные буквы в $s на заглавные?
Укажите номер правильного ответа

1): $s =~ tr/a-z/A-Z/;
2): $s =~/upper/;
3): $s =~s///;
4): $s= upper;
+

S: Необходимо разбить строку $s по символу ","(запятая). Какое выражение необходимо для этого написать?
Укажите номер правильного ответа

1): @arr=split(",");
2): @arr=cut($s,",");
3): @arr=~s/,//g;
4): нет правильного ответа

S: Как правильно удалить из строки $s все буквы g входящие в ее состав?
Укажите номер правильного ответа

1): $s =~s/g//;
2): $s=/g//g;
3): $s=~s/g//g;
4): $s=~s/g/g/g;
+

Модуль 4. Программирование на PHP. MySQL & PHP
Задания А
Однозначный выбор

S: Выберите правильный вариант создания новой переменной:
: name = ‘Joe’;
: string name = “Joe”;
: var name = ‘Joe’;
: $name = ‘Joe’;

S: Какой тип данных недопустим в PHP?
: char
: resource
: string
: array

S: Каждое PHP выражение должно заканчиваться:
: двоеточие (:)
: запятой (,)
: точкой (.)
: точкой с запятой (;)

S: В имени какой из этих переменных допущена ошибка?
: $_myvariable
: $my_variable
: $1myvariable
: $my-variable

S: Какого типа переменных не бывает в PHP?
: целые числа
: комплексные числа
: строки
: вещественные числа

S: Какая из следующих строк содержит ошибку?
: "abcd"
: "abcd\"

: 'ab\\cd'
: "a\rb\nc\td"

S: Какой из следующих операторов используется для объединения строк?
: .
: &
: ->
: =>

S: Какой оператор обозначает равенство значений в языке PHP?
-: ===
: ==
: =
::=

S: Какой оператор PHP позволяет получить остаток от целочисленного деления?
: %
: //
: /
: mod

S: Как определить константу?
: constant("Name", "Joe");
: static("Name", "Joe");
: variable("Name", "Joe");
: define("Name", "Joe");

S: Выражение $a += 1 эквивалентно:
: $a = $a + $a;
: $a = $a + 1;
: $a = $a * $a;
: $a = $a * 1;

S: Какой будет результат выполнения следующего кода?
<?php $a = 1; ++$a; $a *= $a; echo $a--; ?>

: 1
: 4
: 5
: 3

S: Какой будет результат выполнения следующего кода?
<?php $a = 'b'; $a .=$a; echo $a; ?>
: 2b
: ab
: bb
: a2

S: Какой будет результат выполнения следующего кода?
<?php $a = 9; $b = 99; $c = 8; $c = $b++ / $a++ + --$c;
echo $c ?>
: 18
: 20
: 17
: 19

S: Чем цикл while отличается от цикла do..while?
: блок действий цикла do..while гарантированно выполняется один раз
: блок действий цикла while гарантированно выполняется один раз
: while выполняет блок действий только один раз
: ничем

S: Чем отличается оператор continue от оператора break?
: continue не может быть вызван с дополнительным аргументом, а break – может
: break заканчивает выполнение текущего цикла, а continue - текущей итерации цикла
: continue - для условных операторов, а break используется для остановки циклов

S: Какой будет результат выполнения следующего кода:

```
<?php for ($i = 0; $i < 5; $i++) {
    if ($i % 2 == 0) continue;
    echo $i;
    } ?>
```

: 01234
: 12345
: 13
: 013

S: Можно ли передавать в функцию значения, указывая их тип: func(Int, Array, String, Bool)?
: да, только для массивов
: да, для всех типов, кроме String
: да, можно для всех типов
: нет

S: Как правильно вызвать функцию func с одним параметром?
: func(11)
: func(param=11)
: invoke func(11)
: call func(11)

S: Для чего предназначена функция isset()?
: она проверяет, существует ли массив
: она проверяет, существует ли объект
: она проверяет, была ли инициализирована переменная
: ничего из вышеперечисленного

S: Какая из следующих функций в PHP не относится к файловым?
: fappend
: fopen
: fwrite
: fgets

S: Укажите правильный способ объявления массивов:
: $m = array("a"=>1, "b"=>2, "c"=>3)

: $m = array("a"->1, "b"->2, "c"->3)
: $m = array[] {"a", "b", "c"}
: $m = new array[]

V3: Наиболее правильный выбор.
S: PHP (Hypertext Preprocessor) -это:
+: [] наиболее простой скриптовый язык программирования, широко применяющийся при создании динамически генерируемых веб-страниц ;
+: [] язык написания скриптов, которые встраиваются непосредственно в гипертекстовые файлы и исполняются на Web-сервере;
+: [] язык программирования общего назначения, относящийся к скриптовому типу;
+: [] скриптовый язык программирования общего назначения .
S: Основными конструкциями языка PHP являются
+: [] Условные операторы (if, else), циклы (while, do-while, for, foreach, break, continue); конструкции выбора (switch); конструкции объявления (declare); конструкции возврата значений (return); конструкции включений (require, include).
+: [] Условные операторы (if, else), циклы (while, do-while, for, foreach, break, continue); конструкции выбора (switch); конструкции объявления (declare); конструкции включений (require, include).;
+: []Условные операторы (if, else), циклы (while, do-while, for, foreach, break, continue); конструкции возврата значений (return); конструкции включений (require, include). ;
+: [] Условные операторы (if, else), циклы (while, do-while, for, foreach, break, continue).
S:Формы HTML-это:
+: [] специальное средство, которое позволяет производить контакт между посетителями и авторами.
+: [] простые элементы управления **HTML**, которые применяются для сбора информации от посетителей веб-сайта;
+: [] важный компонент создания блоков, используемый для сбора данных о ваших посетителях и их опыте посещения сайта;
+: [] раздел документа, в котором содержатся специальные элементы, называемые управляющими (флажки, кнопки с зависимой фиксацией, меню и т.д..

S: При выполнении get- запроса :
+: [] все данные передаются в конце URL;
+: [] в адресной строке браузера будет явно виден url (адрес) страницы ;
+: [] все переменные и их значения передаются прямо через адрес;
+: [] на сервер передаются введенные в форме данные.
S: При выполнении post – запроса
+: [] все данные передаются в теле запроса ;
+: [] все названия переменных и значения будут передаваться как запрос браузера к веб-серверу;
+: [] в сохраненной браузером строке можно будет явно увидеть и логин и пароль;
+: [] на сервер передаются введенные в форме данные.
S:Определение текстового поля включает следующие атрибуты:
+: [] type - тип элемента (для текстовых полей - **text**); name - тип переменной, в которой сохраняются введённой данные;size - общий размер текстового поля в браузере; maxlength - максимальное количество символов, вводимых в текстовом поле; value - значение, отображаемое в текстовом поле по умолчанию;
+: [] type - тип элемента; name - тип переменной, в которой сохраняются введённой данные;size - общий размер текстового поля в браузере; maxlength - максимальное количество символов, вводимых в текстовом поле; value - значение, отображаемое в текстовом поле по умолчанию;
+: [] type - тип элемента (для текстовых полей - **text**); name - тип переменной, в которой сохраняются введённой данные; size - общий размер текстового поля в браузере; value - значение, отображаемое в текстовом поле по умолчанию;
+: [] type - тип элемента (для текстовых полей - **text**); в которой сохраняются введённой данные; maxlength - максимальное количество символов, вводимых в текстовом поле; value - значение, отображаемое в текстовом поле по умолчанию.
S: *Флажки (checkboxes)* используются :
+: [] в ситуациях, когда пользователь выбирает один или несколько вариантов из готового набора - по аналогии с тем, как ставятся "галочки" в анкетах;

+:[] если пользователю необходимо отметить (поставить галочку) определенную опцию;
+:[] в настройках, когда нужно выборочно выбрать определенные пункты, необходимые для комфортной работы пользователю;
+:[] в диалоговых окнах как поодиночке, так и в группе, причем все флажки устанавливаются независимо друг от друга.
S:Раскрывающиеся списки применяются:
+: [] в ситуации, когда у Вас имеется длинный перечень допустимых вариантов, из которых пользователь должен выбрать один вариант;
+: [] при работе с относительно большими наборами данных - например, при перечислении областей или стран;
+: [] для выбора оператора сравнения в данном условии, а широкие - для выбора одного из значений;
+: [] для экономии места .
S: Скрытые поля используются :
+:[] используются для передачи данных между сценариями;
+:[] для передачи в форме данных, полученных в результате выполнения сценария документа HTML;
+: [] для сохранения контекста
+: [] для передачи служебной информации.
S: Облачные хранилища данных-это :
+: [] модель онлайн-хранилища, в котором данные хранятся на многочисленных распределённых в сети серверах, предоставляемых в пользование клиентам, в основном, третьей стороной;
+: [] выделенное вам место на многочисленных серверах поставщика услуг;
+: [] такой диск в онлайне, который служит для хранения персональных данных;
+: [] место, где вы можете хранить свои данные, доступ к которым вы будете иметь в любом месте и в любое время.

V3: Множественный выбор
S: Как можно задать массив в языке PHP?
: $arr = array("a","b","c")
: $arr[0] = "a"
: $arr ("0"=> "a")
: $arr["a","b","c"] = "q"

S: Как присваивается значение константе в языке PHP?
: define("Name","Joe")
: define("Name","Joe", true)
: Name = "Joe"
: constant("Name") = "Joe"

S: Как задаются комментарии в языке PHP?
: // это комментарий
: /* это комментарий */
: / это комментарий /
: ## это комментарий ##

S: С помощью какой конструкции можно выполнять периодически блок действий до тех пор, пока верно условие?
: с помощью цикла while
: с помощью условного оператора if
: с помощью цикла for
: с помощью switch

S: Как можно задать массив в языке PHP?
: $arr = array("a","b","c");
: $arr[0] = "a";
: $arr ("0"=> "a");
: $arr["a","b","c"] = "q";

S: В каком случае на экран будет выведено слово <Bye> после выполнения кода:

```
<?php if ($var) echo "Hello"; else echo "Bye"; ?>
```

: если $var === false
: если $var == "true"
: если $var == ""
: если $var == "false"

S: Какие из циклов while записаны правильно с точки зрения синтаксиса?

```
  while ($a < $b){
        echo $a;
        $a++;
:       endwhile;
  while ($a < $b):
        echo $a;
        $a++;
:       endwhile;
  while ($a < $b){
        echo $a;
        $a++;
:       }
   while ($a < $b):
         echo $a;
         $a++;
-:       dowhile;
```

S: Какие типы наборов открывающих тегов поддерживаются в PHP?

: <?php ... ?>
: <? ... ?>
: <script language="php"> ... </script>
: <% ... %>

S: Какие скалярные типы данных есть в языке PHP?

: string
: text
: boolean
: resource
: float

S: Выберите все верные утверждения из перечисленных:

: include() подключает и вычисляет внешний файл
: require() подключает и вычисляет внешний файл
: include_once() подключает и вычисляет внешний файл только если он не был добавлен ранее
: require_once() подключает и вычисляет внешний файл только если он не был добавлен ранее

S: Какие из операторов if записаны правильно с точки зрения синтаксиса?

```
   if ($a == "") {
  echo "False";
  $a +=1;
: }
   if ($a == "") :
   echo "False";
   $a +=1;
:  endif;
  if ($a = "") {
  echo "False";
  $a +=1;
  } else : echo "True";
: endif;
   if ($a = ""):
   echo "False";
   $a +=1;
   else : echo "True";
:  endif;
```

V2: Задания
V3: Вписать правильный ответ

J: Сколько скалярных типов данных существует в языке PHP?
_____
+:
+:

J: С помощью функции _____ можно вычислить квадратный корень выражения.
+:
+:

J: Какой будет результат выполнения следующего кода?
<?php $a = 1; ++$a; $a *= $a; echo $a--; ?> _____
+:
+:

J: Какой будет результат выполнения следующего кода:

```
<?php
function func() {
   static $id = 0;
   $id++;
   echo $id;
 }
 func();
 func();
 func();
?>
```

_____

+:

+:

J: Какой будет результат выполнения следующего кода:

```
<?php for ($i = 0; $i < 5; $i++) {
      if ($i % 2 == 0) continue;
      echo $i;
     } ?>
```

_____

+:

+:

J: Размер окна браузера 800 пикселей. На страницу добавили блок с шириной 50%. Затем в этот блок добавили таблицу с шириной 40%. Какова будет ширина таблица в пикселях? _____

+:

+:

V3: Установить соответствие

Q: Сопоставьте логический оператор его описанию при котором выражение истинно:

L: a || b

L: a && b

L: a xor b

L: ! a

R: a или b истинны

R: a и b истинны

R: a или b истинны, но не одновременно

R: a не истинно

Q: Заданы переменные $a=7, $b=4. Сопоставьте выражение и результат ее действия:
L: $a -=$b++
L: $a %= $b-1
L: $a -= --$a
L: $a += $b-1
R: 3
R: 1
R: 0
R: 11

Q: Сопоставьте выражение и результат ее действия:
L: ceil(12.34)
L: floor(12.34)
L: number_format(12.34,1)
L: number_format(12.34,2)
R: 13
R: 12
R: 12.3
R: 12.34

Q: Имеется строка $str=”aBc abc”. Сопоставьте функцию и результат ее действия со строкой:
L: strchr($str, “b”)
L: stristr($str, “b”)
L: substr($str, 1, 5)
L: strtr ($str, “b”, “i”)
R: bc
R: Bc abc
R: Bc ab
R: Bc aic

V3: Установить последовательность

Q: Установить последовательность предложений полного синтаксиса оператора SELECT

L1: SELECT [ALL | DISTINCT] <список_выбора>
L2: FROM <имя_таблицы>,
L3: [GROUP BY <имя_столбца
L4: [HAVING <условие>]
L5:[ORDER BY <имя_столбца> [ASC | DESC],...]
R1:
R2:
R3:
R4:
R5:

Q: Установите последовательность действий при подключении к базе данных из PHP файла.

L1: Подключение к установленной базе данных MySQL.
L2: Использование команды USE в отношении нужной базы данных MySQL.
L3: Отправка SQL базе данных.
L4: Получение результатов.
L5: Обработка результатов.
R1:
R2:
R3:
R4:
R5:

Q:Установите последовательность записей соединения с сервером и базой данных в предложенном примере

L1:<?php
L2:$host="localhost";
L3:$user="admin";
L4:$password="12345";
L5:$db="baza";/*Имя базы данных*/
L6:mysql_connect($host, $user, $password);
L7:mysql_select_db($db);
R1:
R2:

R3:
R4:
R5:
R6:
R7:
V2: Задания С
V3: Ситуация

S: Как вывести на экран четвертый элемент массива, если он равен числу 4?
Укажите номер правильного ответа
1): `<?php if ($arr[3] == 4) echo $arr[3]; ?>`
2): `<?php if ($arr[4] == 4) echo $arr[4]; ?>`
3): `<?php if ($arr[3] = 4) echo $arr[4]; ?>`
4): `<?php if ($arr[4] = 4) echo $arr[4]; ?>`
+
S: Какой будет результат выполнения следующего кода:

```
<?php $a = array(3=>"33",2=>"22",1=>"11");
foreach ($a as $b=>$c) {echo $b;} ?>
```

: 332211
: 112233
: 123
: 321

S: Какой будет результат выполнения следующего кода:

```
<?php
function func() {
   static $id = 0;
   $id++;
   echo $id;
 }
func();
func();
func();
?>
```

: 111
: 123
: 333
: 1

S: Какая из приведенных ниже функций удаляет первый элемент из массива и возвращает его значение?
Укажите номер правильного ответа
1): array_compare()
2): array_flip()
3): array_shift()
4): array_remove_first()
+
S: Каким будет результат выполнения следующего кода?
Укажите номер правильного ответа
```
<?php  $arr = array(1=>'один',  'два', 'три', 'четыре' );
echo $arr[3];  ?>
```
1): три
2): четыре
3): возникнет ошибка выполнения
4): false
+

S: Каким будет результат выполнения следующего кода?
Укажите номер правильного ответа
```
$str1 = "Hello";
$str2 = "str1";
echo $$str2;
```
1): Hello
2): str1
3): string
4): код не скомпилируется
+
S: Каким будет результат выполнения следующего кода?
Укажите номер правильного ответа
```
<?php  $var = 1;  $str = '1 + $var';  echo $str;  ?>
```
1): 2
2): 1 + $var
3): '1 + $var'
4): 1 + 1
+
S: Каким будет результат выполнения следующего кода?

```
<?php   $a = 42;   $type = gettype(gettype($a + 0.0));  echo $type; ?>
```
: integer
: string
: mixed
: float

S: Каким будет результат выполнения следующего кода?

```
<?php  for ($i = 0; $i < 5; $i++) {
   if ($i > 2) continue;
   echo $i;
}
echo $i; ?>
```
: 0125
: 012345
: 2345
: 0123

S: Каким будет результат выполнения следующего кода?
Укажите номер правильного ответа

```
function func($a, $b)
{
   return $a + $b;
}
$a = 0;
$b = 1;
for ($i = 0; $i < 10; $i++) {
   echo func($a, $b) . ',';
}
```
1): 1,2,3,4,5,6,7,8,9
2): 1,1,1,1,1,1,1,1,1,1,
3): 1,2,3,5,8,13,21,34,55,89
4): 1,2,3,4,5,6,7,8,9,10
+

S: Каким будет результат выполнения следующего кода?
Укажите номер правильного ответа

```
<?php $a=2; $b=4; $a = &$b; $b++; echo $a.' - '.$b; ?>
```
1): 5 - 5

2): 2 - 5
3): 4 - 5
4): 5 – 4
+
S: Каким будет результат выполнения следующего кода? Укажите номер правильного ответа

```
<?php
define('AA', 10);
$array = array(10 => AA, "AA" => 20);
print $array[$array[AA]] * $array["AA"];
?>
```

1): 0
2): сообщение об ошибке
3): 100
4): 200
+

Учебное электронное издание

ТРОИЦКАЯ Елена Анатольевна
АРТЮШИНА Лариса Андреевна

WEB-ПРОГРАММИРОВАНИЕ

Учебное пособие

Издается в авторской редакции

Системные требования: Intel от 1,3 ГГц; Windows XP/7/8/10; Adobe Reader; дисковод CD-ROM.

Тираж 25 экз.

Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых
Изд-во ВлГУ
rio.vlsu@vlsu.ru

Институт информационных технологий и радиоэлектроники

Кафедра информатики и защиты информации
troickiyv@mail.ru