**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**
**СЕВЕРО-КАВКАЗСКИЙ ФИЛИАЛ**
**ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО ОБРАЗОВАТЕЛЬНОГО БЮДЖЕТНОГО**
**УЧРЕЖДЕНИЯ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**
**МОСКОВСКОГО ТЕХНИЧЕСКОГО УНИВЕРСИТЕТА СВЯЗИ И ИНФОРМАТИКИ**

**Ткачук Е.О.**

# WEB-программирование

## Методическое пособие по выполнению лабораторных работ

**Ростов-на-Дону**
**2019**

УДК 004.75:004:425.2

Ткачук Е.О.

WEB-программирование. Методическое пособие по выполнению лабораторных работ. / Моск. техн. ун-т связи и информатики, Сев.-Кавк. филиал. – Ростов н/Д, 2019, 68 с.

В пособии даются организационно-методические указания к лабораторным вопросам и порядок выполнения и оформления лабораторных работ.

Предназначено для студентов всех специальностей обеих форм обучения, изучающих дисциплину «WEB-программирование», а также может быть полезно всем остальным студентам, желающим самостоятельно современные методы WEB-программирование.

Рецензент канд. техн. наук, доц. А.Н. Чикалов (СКФ МТУСИ)

Обсуждено и утверждено на заседании кафедры СПОИ (протокол заседания кафедры №1 от 26.08.2019).

© Московский технический университет связи и информатики, Северо-Кавказский филиал, 2019

СОДЕРЖАНИЕ

# Лабораторная работа № 1. Программное обеспечение сетей ЭВМ

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Гетерогенные системы содержат независимые компьютеры, соединенные разными сетями (например, состоять из нескольких локальных сетей, соединенных коммутируемой магистралью FDDI или ATM).

FDDI (англ. Fiber Distributed Data Interface — Волоконно-оптический интерфейс передачи данных) — стандарт передачи данных в локальной сети, протянутой на расстоянии до 200 километров. Стандарт основан на протоколе Token Ring. Кроме большой территории, сеть FDDI способна поддерживать несколько тысяч пользователей.

В качестве среды передачи данных в FDDI рекомендуется использовать волоконно-оптический кабель, однако можно использовать и медный кабель, в таком случае используется сокращение CDDI (Copper Distributed Data Interface). В качестве топологии используется схема двойного кольца, при этом данные в кольцах циркулируют в разных направлениях. Одно кольцо считается основным, по нему передаётся информация в обычном состоянии; второе — вспомогательным, по нему данные передаются в случае обрыва на первом кольце. Для контроля за состоянием кольца используется сетевой маркер, как и в технологии Token Ring.

Поскольку такое дублирование повышает надёжность системы, данный стандарт с успехом применяется в магистральных каналах связи.

ATM (англ. asynchronous transfer mode — асинхронный способ передачи данных) — сетевая высокопроизводительная технология коммутации и мультиплексирования, основанная на передаче данных в виде ячеек (англ. cell) фиксированного размера (53 байта[1]), из которых 5 байтов используется под заголовок. В отличие от синхронного способа передачи данных (STM — англ. synchronous transfer mode), ATM лучше приспособлен для предоставления услуг передачи данных с сильно различающимся или изменяющимся битрейтом.

Битре́йт (от англ. bitrate) — количество бит, используемых для хранения одной секунды мультимедийного контента[1][2]. Битрейт принято использовать при измерении эффективной скорости передачи потока данных по каналу, то есть минимального размера канала, который сможет пропустить этот поток без задержек.

Битрейт выражается битами в секунду (бит/с, bps), а также производными величинами с приставками кило- (кбит/с, kbps), мега- (Мбит/с, Mbps) и т. д.

Компьютерной сетью называют совокупность узлов (компьютеров, терминалов, периферийных устройств), имеющих возможность информационного взаимодействия друг с другом с помощью специального коммуникационного оборудования и программного обеспечения.

Средства передачи и обработки информации ориентированы в ней на коллективное использование общесетевых ресурсов – информационных, программных, аппаратных.

Компьютерные сети могут работать в различных режимах: обмена данными между абонентами сети, запроса и выдачи информации, сбора информации пакетной обработки данных по запросам пользователей с удаленных терминалов, в диалоговых режимах.

Таким образом, с появлением сетей ЭВМ разрешены две очень важные проблемы:

1) обеспечение в принципе неограниченного доступа к ЭВМ пользователей независимо от территориального расположения,

2) возможность оперативного перемещений больших массивов информации на любые расстояния, позволяющий своевременно получать данные для принятия тех или иных решений.

Использование вычислительных сетей дает предприятию следующие возможности:

1. Разделение дорогостоящих ресурсов;
2. Улучшение доступа к информации;
3. Быстрое и качественное принятие решений;
4. Совершенствование коммуникаций;
5. Свобода в территориальном размещении компьютеров.

Программное обеспечение сетей ЭВМ в расширенном варианте составляют:

1) сетевые операционные системы;

2) сетевые драйвера, протоколы, службы и другое дополнительное программное обеспечение сетевых интерфейсов;

3) прикладное сетевое программное обеспечение.

Под сетевыми операционными системами понимают такие операционные системы, которые обеспечивают пользователям распределенный доступ к сетям ЭВМ.

Для распределенных компьютеров ОС можно разделить на две категории: сильно связанные и слабо связанные.

Сильно связанные ОС обычно называются распределенными ОС (Distributed Operation System, DOS) и используются для управления мультипроцессорными и гомогенными мультикомпьютерными системами. Основная их цель – скрыть тонкости управления аппаратным обеспечением.

Слабо связанные ОС называются сетевыми ОС (Network Operation System, NOS). Они используются для управления гетерогенными мультикомпьютерными системами. Помимо традиционных функций управления ресурсами, они должны обеспечить доступ удаленных клиентов к локальным службам.

Однако, для создания РИС служб сетевой ОС недостаточно. Необходимо добавить к ним дополнительные компоненты для организации поддержки прозрачности распределения.

Эти компоненты образуют средства промежуточного уровня (middleware) и составляют основу современных РИС.

Во вторую группу входит большой круг всевозможного программного обеспечения в основном изготовителя данного интерфейса (сетевой платы, модема и т.п.) для обеспечения правильной работы сетевого устройства.

При этом под драйвером понимается программа, непосредственно взаимодействующая с интерфейсом - сетевым адаптером и операционной системой (ОС). Драйвер сетевого адаптера взаимодействует с ОС через систему протоколов и служб, которые могут находится как в самих ОС, так и поставляться вместе с устройством.

При этом под сетевым протоколом понимается набор правил поведения сетевых узлов при передаче-приеме информации.

Под сетевыми службами понимается набор программного обеспечения сетевого обеспечения узкоспециального назначения, например:

- клиенты сетей - позволяют подключаться, обозревать и пользоваться сетевыми ресурсами соответствующих сетей,
- службы контроля трафика сетей,
- службы использования доступа к разделяемым ресурсам,
- доменные службы и др.

Со́кеты (англ. socket — разъём) — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения.

Следует различать клиентские и серверные сокеты. Клиентские сокеты грубо можно сравнить с конечными аппаратами телефонной сети, а серверные — с коммутаторами. Клиентское приложение (например, браузер) использует только клиентские сокеты, а серверное (например, веб-сервер, которому браузер посылает запросы) — как клиентские, так и серверные сокеты.

Интерфейс сокетов впервые появился в BSD Unix. Программный интерфейс сокетов описан в стандарте POSIX.1 и в той или иной мере поддерживается всеми современными операционными системами.

Кроме того, приложения часто пользуются интерфейсами локальных файловых систем. Но проблема такого подхода состоит в том, что наличие распределения слишком очевидно.

Решение заключается в том, чтобы поместить между приложением и сетевой операционной системой промежуточный уровень программной поддержки, обеспечивающий дополнительное абстрагирование. Потому этот уровень и называется промежуточным. Он находится посредине между приложением и сетевой операционной системой.Таким образом, основную роль в построении РИС играют программные средства (службы) промежуточного уровня между сетевыми ОС и распределенными приложениями.

Для упрощения разработки и интеграции РИС, основная часть промежуточного ПО должна базироваться на некоторой модели, определяющей распределение и связь.

Самая простая первая ранняя модель – представление всех объектов в виде файлов (распределенная файловая система). Пример – файловая система Unix.

В операционной системе UNIX файл является хранилищем двоичных и символьных данных, хранимых как поток байтов. В UNIX символьные данные кодируются с помощью кода ASCII,

Вторая ранняя модель основана на удаленных вызовах процедур (Remote Procedure Calls, RPC). В этой модели акцент делается на сокрытии сетевого обмена за счет того, что процессу разрешается вызывать процедуры, реализация которых находится на удаленной машине. При вызове процедуры параметры передаются на удаленную машину, где она выполняется, после чего управление передается в точку вызова процедуры. Внешне это выглядит как обычный вызов процедуры.

Более современные модели основаны на взаимодействии распределенных объектов. Пример – DCOM, COM+ (а также основанные на них технологии ActiveX). Идея распределенных объектов состоит в том, что каждый объект реализует интерфейс, скрывающий все внутренние детали реализации объекта от пользователя. Интерфейс содержит методы, реализуемые объектом, и все, что видит процесс – это интерфейсы.

В Web 1.0. применяется модель распределенных документов. В этой модели информация организована в виде документов, каждый из которых размещен в каком-то месте, причем физическое размещение документа скрыто от пользователя (прозрачно). Документы могут содержать ссылки на другие документы, которые могут быть извлечены

и отображены на экран. Однако, с развитием Web 2.0 эта модель претерпела значительные изменения, поскольку Web все больше наполняется программным содержимым.

Круг прикладного сетевого программного обеспечения составляют всевозможные сетевые приложения.

Каждый компьютер работает под управлением собственной операционной системы, а какая-либо «общая» операционная система, распределяющая работу между компьютерами сети, отсутствует. Взаимодействие между компьютерами сети про-исходит за счет передачи сообщений через сетевые адаптеры и каналы связи. С помощью этих сообщений один компьютер обычно запрашивает доступ к локальным ресурсам другого компьютера. Такими ресурсами могут быть как данные, хранящиеся на диске, так и разнообразные периферийные устройства — принтеры, модемы, факс-аппараты и т.д. Разделение локальных ресурсов каждого компьютера между всеми пользователями сети — основная цель создания вычислительной сети.

Каким же образом сказывается на пользователе тот факт, что его компьютер подключен к сети? Прежде всего, он может пользоваться не только файлами, дисками, принтерами и другими ресурсами своего компьютера, но и аналогичными ресурсами других компьютеров, подключенных к той же сети. Правда, для этого недостаточно снабдить компьютеры сетевыми адаптерами и соединить их кабельной системой. Необходимы еще некоторые добавления к операционным системам этих компьютеров. На тех компьютерах, ресурсы которых должны быть доступны всем пользователям сети, необходимо добавить модули, которые постоянно будут находиться в режиме ожидания запросов, поступающих по сети от других компьютеров. Обычно такие модули называются программными серверами (server), так как их главная задача — обслуживать (serve) запросы на доступ к ресурсам своего компьютера. На компьютерах, пользователи которых хотят получать доступ к ресурсам других компьютеров, также нужно добавить к операционной системе некоторые специальные программные модули, которые должны вырабатывать запросы на доступ к удаленным ресурсам и передавать их по сети на нужный компьютер. Такие модули обычно называют программными клиентами (client). Собственно же сетевые адаптеры и каналы связи решают в сети достаточно простую задачу — они передают сообщения с запросами и ответами от одного компьютера к другому, а основную работу по организации совместного использования ресурсов выполняют клиентские и серверные части операционных систем.

Пара модулей «клиент – сервер» обеспечивает совместный доступ пользователей к определенному типу ресурсов, например к файлам. В этом случае говорит, что пользователь имеет дело с файловой службой (service). Обычно сетевая операционная система поддерживает несколько видов сетевых служб для своих пользователей — файловую службу, службу печати, службу электронной почты, службу удаленного доступа и т. п.

Термины «клиент» и «сервер» используются не только для обозначения про-граммных модулей, но и компьютеров, подключенных к сети. Если компьютер предоставляет свои ресурсы другим компьютерам сети, то он называется сервером, а если он их потребляет — клиентом. Иногда один и тот же компьютер может одновременно играть роли и сервера, и клиента.

Сетевые службы всегда представляют собой распределенные программы, состоящие из нескольких взаимодействующих частей, причем каждая часть, как правило, выполняется на отдельном компьютере сети.

До сих пор речь шла о системных распределенных программах. Однако в сети могут выполняться и распределенные пользовательские программы - приложения.

Распределенное приложение также состоит из нескольких частей, каждая из которых выполняет какую-то определенную законченную работу по решению прикладной задачи. Например, одна часть приложения, выполняющаяся на компьютере пользователя, может поддерживать специализированный графический интерфейс, вторая - работать на мощном выделенном компьютере и заниматься статистической обработкой введенных пользователем данных, а третья - заносить полученные результаты в базу данных на компьютере с установленной стандартной СУБД. Распределенные приложения в полной мере используют потенциальные возможности распределенной обработки, предоставляемые вычислительной сетью, и поэтому часто называются сетевыми приложениями.

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

1. Охарактеризовать сетевые операционные системы по следующей схеме:

1) платность,
2) доступ к исходному коду,
3) многоплатформенность,
4) мультизадачность,
5) количество пользователей,
6) функции управления сетью,
7) интерфейс работы,
8) потребляемые ресурсы.

2. Ответьте на контрольные вопросы

**Контрольные вопросы**

1. Что понимают под программным обеспечением сетей ЭВМ? ( ПК-2)
2. Что дает предприятию использование компьютерных сетей? ( ПК-2)
3. Классификация сетевого программного обеспечения. ( ПК-2)
4. Что называют операционной системой? ( ПК-2)
5. Что входит в группу прикладного программного обеспечения? ( ПК-2)
6. По каким критериям можно охарактеризовать сетевую операционную систему? ( ПК-2)
7. Что называют сетевым драйвером? ( ПК-2)
8. Что называют сетевым протоколом? ( ПК-2)
9. Перечислить сетевые операционные системы. ( ПК-2)
10. Что такое сетевые службы? ( ПК-2)
11. Что называют стандартным программным обеспечением ЭВМ? ( ПК-2)
12. Что такое технология «клиент-сервер»? ( ПК-2)

# Лабораторная работа № 2 Изучение утилит протокола TCP/IP

В состав TCP/IP входят диагностические утилиты, предназначенные для проверки конфигурации стека и тестирования сетевого соединения.

| **Утилита** | **Применение** |
|---|---|
| hostname | Выводит имя локального хоста. Используется без параметров. |
| ipconfig | Выводит значения для текущей конфигурации стека TCP/IP: IP-адрес, маску подсети, адрес шлюза по умолчанию, адреса WINS (Windows Internet Naming Service) и DNS (Domain Name System) |
| ping | Осуществляет проверку правильности конфигурирования TCP/IP и проверку связи с удаленным хостом. |
| tracert | Осуществляет проверку маршрута к удаленному компьютеру путем отправки эхо-пакетов протокола ICMP (Internet Control Message Protocol). Выводит маршрут прохождения пакетов на удаленный компьютер. |
| arp | Выводит для просмотра и изменения таблицу трансляции адресов, используемую протоколом разрешения адресов ARP (Address Resolution Protocol - определяет локальный адрес по IP-адресу) |
| route | Модифицирует таблицы маршрутизации IP. Отображает содержимое таблицы, добавляет и удаляет маршруты IP. |
| netstat | Выводит статистику и текущую информацию по соединению TCP/IP. |
| nslookup | Осуществляет проверку записей и доменных псевдонимов хостов, доменных сервисов хостов, а также информации операционной системы, путем запросов к серверам DNS. |

**1. Проверка правильности конфигурации TCP/IP с помощью ipconfig.**

При устранении неисправностей и проблем в сети TCP/IP следует сначала проверить правильность конфигурации TCP/IP. Для этого используется утилита ipconfig.

Эта команда полезна на компьютерах, работающих с DHCP (Dynamic Host Configuration Protocol), так как дает пользователям возможность определить, какая конфигурация сети TCP/IP и какие величины были установлены с помощью DHCP.

**Синтаксис:**

ipconfig [/all | /renew[adapter] | /release]

**Параметры:**

| | |
|---|---|
| all | выдает весь список параметров. Без этого ключа отображается только IP-адрес, маска и шлюз по умолчанию; |
| renew[adapter] | обновляет параметры конфигурации DHCP для указанного сетевого адаптера; |
| release[adapter] | освобождает выделенный DHCP IP-адрес; adapter – имя сетевого адаптера; |
| displaydns | выводит информацию о содержимом локального кэша клиента DNS, используемого для разрешения доменных имен. |

Таким образом, утилита ipconfig позволяет выяснить, инициализирована ли конфигурация и не дублируются ли IP-адреса:

- если конфигурация инициализирована, то появляется IP-адрес, маска, шлюз;
- если IP-адреса дублируются, то маска сети будет 0.0.0.0;
- если при использовании DHCP компьютер не смог получить IP-адрес, то он будет равен 0.0.0.0 .

**2. Тестирование связи с использованием утилиты *ping*.**

Утилита ping (Packet Internet Grouper) используется для проверки конфигурирования TCP/IP и диагностики ошибок соединения. Она определяет доступность и функционирование конкретного хоста. Использование ping лучший способ проверки того, что между локальным компьютером и сетевым хостом существует маршрут. Хостом называется любое сетевое устройство (компьютер, маршрутизатор), обменивающееся информацией с другими сетевыми устройствами по TCP/IP.

Команда ping проверяет соединение с удаленным хостом путем отправки к этому хосту эхо-пакетов ICMP и прослушивания эхо-ответов. Ping ожидает каждый посланный пакет и печатает количество переданных и принятых пакетов. Каждый принятый пакет проверяется в соответствии с переданным сообщением. Если связь между хостами плохая, из сообщений ping станет ясно, сколько пакетов потеряно.

По умолчанию передается 4 эхо-пакета длиной 32 байта (возможны и другие варианты значения по умолчанию) - периодическая последовательность символов

алфавита в верхнем регистре. Ping позволяет изменить размер и количество пакетов, указать, следует ли записывать маршрут, который она использует, какую величину времени жизни (ttl) устанавливать, можно ли фрагментировать пакет и т.д.. При получении ответа в поле time указывается, за какое время (в миллисекундах) отправленный пакет доходит до удаленного хоста и возвращается назад. Так как значение по умолчанию для ожидания отклика равно 1 секунде, то все значения данного поля будут меньше 1000 миллисекунд. Если вы получаете сообщение «Request time out» (Превышен интервал ожидания), то, возможно, если увеличить время ожидания отклика, пакет дойдет до удаленного хоста. Это можно сделать с помощью ключа –w.

Ping можно использовать для тестирования как имени хоста (DNS или NetBIOS), так и его IP-адреса. Если ping с IP-адресом выполнилась успешно, а с именем – неудачно, это значит, что проблема заключается в распознавании соответствия адреса и имени, а не в сетевом соединении.

Утилита ping используется следующими способами:

1) Для проверки того, что TCP/IP установлен и правильно сконфигурирован на локальном компьютере, в команде ping задается адрес петли обратной связи (loopback address):

ping 127.0.0.1

Если тест успешно пройден, то вы получите следующий ответ:

Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128

Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128

Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128

Ответ от 127.0.0.1: число байт=32 время<1мс TTL=128

2) Чтобы убедиться в том, что компьютер правильно добавлен в сеть и IP-адрес не дублируется, используется IP-адрес локального компьютера:

ping IP-адрес_локального_хоста

3) Чтобы проверить, что шлюз по умолчанию функционирует и что можно установить соединение с любым локальным хостом в локальной сети, задается IP-адрес шлюза по умолчанию:

ping IP-адрес_шлюза

4) Для проверки возможности установления соединения через маршрутизатор в команде ping задается IP-адрес удаленного хоста:

ping IP-адрес_удаленного хоста

**Синтаксис:**

ping [-t] [-a] [-n count] [-l length] [-f] [-i ttl] [-v tos] [-r count] [-s count] [ [-j host-list] | [-k host-list] ] [-w timeout] destination-list

**Параметры:**

-t выполняет команду ping до прерывания. Control-Break - посмотреть статистику и продолжить. Control-C - прервать выполнение команды;

-a позволяет определить доменное имя удаленного компьютера по его IP-адресу;

-n count посылает количество пакетов ECHO, указанное параметром count;

-l length посылает пакеты длиной length байт (максимальная длина 8192 байта);

-f посылает пакет с установленным флагом «не фрагментировать». Этот пакет не будет фрагментироваться на маршрутизаторах по пути своего следования;

-i ttl устанавливает время жизни пакета в величину ttl (каждый маршрутизатор уменьшает ttl на единицу);

-v tos устанавливает тип поля «сервис» в величину tos;

-r count записывает путь выходящего пакета и возвращающегося пакета в поле записи пути. Count - от 1 до 9 хостов;

-s count позволяет ограничить количество переходов из одной подсети в другую (хопов). Count задает максимально возможное количество хопов;

-j host-list направляет пакеты с помощью списка хостов, определенного параметром host-list. Последовательные хосты могут быть отделены промежуточными маршрутизаторами (гибкая статическая маршрутизация). Максимальное количество хостов в списке, позволенное IP, равно 9;

-k host-list направляет пакеты через список хостов, определенный в host-list. Последовательные хосты не могут быть разделены промежуточными маршрутизаторами (жесткая статическая маршрутизация). Максимальное количество хостов – 9;

-w timeout указывает время ожидания (timeout) ответа от удаленного хоста в миллисекундах (по умолчанию – 1сек);

destination-list указывает удаленный хост, к которому надо направить пакеты ping.

***Пример использования утилиты ping:***

C:\WINDOWS>ping –n 10 www.netscape.com

Обмен пакетами с www.netscape.com [205.188.247.65] по 32 байт:

Ответ от 205.188.247.65: число байт=32 время=194мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=240мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=173мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=250мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=187мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=239мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=263мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=230мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=185мс TTL=48
Ответ от 205.188.247.65: число байт=32 время=406мс TTL=48
Статистика Ping для 205.188.247.65:
Пакетов: послано = 10, получено = 10, потеряно = 0 (0% потерь)
Приблизительное время передачи и приема:
Наименьшее = 173мс, наибольшее = 406мс, среднее =236мс

В случае невозможности проверить доступность хоста утилита выводит информацию об ошибке. Ниже приведен пример ответа утилиты ping при попытке послать запрос на несуществующий хост.

Обмен пакетами с 172.16.6.21 по 32 байт:

Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.
Превышен интервал ожидания для запроса.

Статистика Ping для 172.16.6.21:
Пакетов: отправлено = 4, получено = 0, потеряно = 4 (100% потерь),

Приблизительное время передачи и приема:
наименьшее = 0мс, наибольшее = 0мс, среднее = 0мс

Утилита сообщает не об отсутствии хоста, а о том, что за отведенное время не был

получен ответ на посланный запрос. Причиной этого не обязательно является отсутствие хоста в сети. Проблема может крыться в сбоях связи, перегрузке или неправильной настройке маршрутизаторов и т. п. Ошибка «сеть недоступна» (network unreachable) прямо указывает на проблемы маршрутизации.

**3. Изучение маршрута между сетевыми соединениями с помощью утилиты *tracert.***

Tracert - это утилита трассировки маршрута. Она использует поле TTL (time-to-live, время жизни) пакета IP и сообщения об ошибках ICMP для определения маршрута от одного хоста до другого.

Утилита tracert может быть более содержательной и удобной, чем ping, особенно в тех случаях, когда удаленный хост недостижим. С помощью нее можно определить район проблем со связью (у Internet-провайдера, в опорной сети, в сети удаленного хоста) по тому, насколько далеко будет отслежен маршрут. Если возникли проблемы, то утилита выводит на экран звездочки (*), либо сообщения типа «Destination net unreachable», «Destination host unreachable», «Request time out», «Time Exeeded».

Утилита tracert работает следующим образом: посылается по 3 пробных эхо-пакета на каждый хост, через который проходит маршрут до удаленного хоста. На экран при этом выводится время ожидания ответа на каждый пакет (Его можно изменить с помощью параметра -w). Пакеты посылаются с различными величинами времени жизни. Каждый маршрутизатор, встречающийся по пути, перед перенаправлением пакета уменьшает величину TTL на единицу. Таким образом, время жизни является счетчиком точек промежуточной доставки (хопов). Когда время жизни пакета достигнет нуля, предполагается, что маршрутизатор пошлет в компьютер-источник сообщение ICMP “Time Exeeded” (Время истекло). Маршрут определяется путем посылки первого эхо-пакета с TTL=1. Затем TTL увеличивается на 1 в каждом последующем пакете до тех пор, пока пакет не достигнет удаленного хоста, либо будет достигнута максимально возможная величина TTL (по умолчанию 30, задается с помощью параметра -h).

Маршрут определяется путем изучения сообщений ICMP, которые присылаются обратно промежуточными маршрутизаторами.

Примечание: некоторые маршрутизаторы просто уничтожают пакеты с истекшим TTL и не будут видны утилите tracert.

**Синтаксис:**

tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout] имя_целевого_хоста

**Параметры:**

-d указывает, что не нужно распознавать адреса для имен хостов;

-h maximum_hops указывает максимальное число хопов для того, чтобы искать цель;

-j host-list указывает нежесткую статическую маршрутизацию в соответствии с host-list;

-w timeout указывает, что нужно ожидать ответ на каждый эхо-пакет заданное число мсек.

**4. Утилита *arp*.**

Основная задача протокола ARP – трансляция IP-адресов в соответствующие локальные адреса. Для этого ARP-протокол использует информацию из ARP-таблицы (ARP-кэша). Если необходимая запись в таблице не найдена, то протокол ARP отправляет широковещательный запрос ко всем компьютерам локальной подсети, пытаясь найти владельца данного IP-адреса. В кэше могут содержаться два типа записей: статические и динамические. Статические записи вводятся вручную и хранятся в кэше постоянно. Динамические записи помещаются в кэш в результате выполнения широковещательных запросов. Для них существует понятие времени жизни. Если в течение определенного времени (по умолчанию 2 мин.) запись не была востребована, то она удаляется из кэша.

**Синтаксис:**

arp [-s inet_addr eth_addr] | [-d inet_addr] | [-a]

**Параметры:**

-s занесение в кэш статических записей;

-d удаление из кэша записи для определенного IP-адреса;

-a просмотр содержимого кэша для всех сетевых адаптеров локального компьютера;

inet_addr - IP-адрес;

eth_addr - MAC-адрес.

**5. Утилита route.**

Утилита **route** предназначена для работы с локальной таблицей маршрутизации. Она имеет следующий **синтаксис:**

route [-f] [-p] [команда [узел] [MASK маска] [шлюз] [METRIC метрика] [IF интерфейс]]

**Параметры:**

-f Очистка таблицы маршрутизации.

-p При указании совместно с командой ADD создает постоянную запись, которая сохраняется после перезагрузки компьютера. По умолчанию записи таблицы маршрутов не сохраняются при перезагрузке.

*команда* одна из четырех команд:
PRINT - вывод информации о маршруте;
ADD - добавление маршрута;
DELETE - удаление маршрута;
CHANGE - изменение маршрута.

*узел* адресуемый узел

*маска* маска подсети; по умолчанию используется маска 255.255.255.255

*шлюз* адрес шлюза

*метрика* метрика маршрута;

*интерфейс* идентификатор интерфейса, который будет использован для пересылки пакета

Для команд PRINT и DELETE возможно использование символов подстановки при указании адресуемого узла или шлюза. Параметр шлюза для этих команд может быть опущен.

При добавлении и изменении маршрутов утилита route осуществляет проверку введенной информации на соответствие условию (УЗЕЛ & МАСКА) == УЗЕЛ. Если это условие не выполняется, то утилита выдает сообщение об ошибке и не добавляет или не изменяет маршрут.

Утилита осуществляет поиск имен сетей в файле networks. Поиск имен шлюзов осуществляется в файле hosts. Оба файла расположены в папке %systemroot%\system32\drivers\etc. Наличие и заполнение этих файлов не обязательно для нормального функционирования утилиты route и работы маршрутизации.

Хотя в большинстве случаев на рабочей станции это не требуется, можно вручную редактировать таблицы маршрутизации.

***Пример использования утилиты route:***

Добавление статического маршрута:

route add 172.16.6.0 MASK 255.255.255.0 172.16.11.1 METRIC 1 IF 0x1000003

**6. Утилита *netstat*.**

Утилита netstat позволяет получить статическую информацию по некоторым из протоколов стека (TCP, UDP, IP, ICMP), а также выводит сведения о текущих сетевых соединениях. Особенно она полезна на брандмауэрах, с ее помощью можно обнаружить нарушения безопасности периметра сети.

**Синтаксис:**

netstat [-a] [-e] [-n] [-s] [-p protocol] [-r]

**Параметры:**

-a выводит перечень всех сетевых соединений и прослушивающихся портов локального компьютера;

-e выводит статистику для Ethernet-интерфейсов (например, количество полученных и отправленных байт);

-n выводит информацию по всем текущим соединениям (например, TCP) для всех сетевых интерфейсов локального компьютера. Для каждого соединения выводится информация об IP-адресах локального и удаленного интерфейсов вместе с номерами используемых портов;

-s выводит статистическую информацию для протоколов UDP, TCP, ICMP, IP. Ключ «/more» позволяет просмотреть информацию постранично;

-r выводит содержимое таблицы маршрутизации.

**7. Утилита *nslookup*.**

Утилита **nslookup** предназначена для диагностики службы DNS, в простейшем случае - для выполнения запросов к DNS-серверам на разрешение имен в IP-адреса. В общем случае утилита позволяет просмотреть любые записи DNS-сервера:

*A* – каноническое имя узла, устанавливает соответствие доменного имени ip-адресу.

*SOA* – начало полномочий, начальная запись, единственная для зоны;

*MX* – почтовые серверы (хосты, принимающие почту для заданного домена);

*NS* – серверы имен (содержит авторитетные DNS-серверы для зоны);

*PTR* – указатель (служит для обратного преобразования ip-адреса в символьное имя хоста) и т. д.

Утилита nslookup достаточно сложна и содержит свой собственный командный интерпретатор.

В простейшем случае (без входа в командный режим) утилита **nslookup** имеет следующий

**Синтаксис:**

nslookup хост [сервер]

**Параметры:**

*Хост* DNS-имя хоста, которое должно быть преобразовано в IP-адрес.

*Сервер* Адрес DNS-сервера, который будет использоваться для разрешения имени. Если этот параметр опущен, то будут последовательно использованы адреса DNS-серверов из параметров настройки протокола TCP/IP.

***Примеры использования утилиты nslookup:***

1. Получение списка серверов имен для домена yandex.ru без входа в командный режим (с использованием ключей).

C:\> nslookup -type=ns yandex.ru

Server: dns01.catv.ext.ru

Address: 217.10.44.35

Non-authoritative answer:

yandex.ru nameserver = ns4.yandex.ru

yandex.ru nameserver = ns5.yandex.ru

yandex.ru nameserver = ns2.yandex.ru

yandex.ru nameserver = ns1.yandex.ru

ns2.yandex.ru internet address = 213.180.199.34

ns5.yandex.ru internet address = 213.180.204.1

2. Получение записи SOA домена yandex.ru с авторитетного сервера с использование командного интерпретатора nslookup.

C:\>nslookup

Default Server: dns04.catv.ext.ru

Address: 217.10.39.4

> set type=SOA

> server ns2.yandex.ru

Default Server: ns2.yandex.ru

Address: 213.180.199.34

> yandex.ru

Server:  ns1.yandex.ru

Address:  213.180.193.1

>yandex.ru

primary name server = ns1.yandex.ru

responsible mail addr = sysadmin.yandex-team.r

serial  = 2009022707

refresh = 1800 (30 mins)

retry   = 900 (15 mins)

expire  = 2592000 (30 days)

default TTL = 900 (15 mins)

yandex.ru nameserver = ns5.yandex.ru

yandex.ru nameserver = ns1.yandex.ru

yandex.ru nameserver = ns4.yandex.ru

yandex.ru nameserver = ns2.yandex.ru

ns1.yandex.ru internet address = 213.180.193.1

ns2.yandex.ru internet address = 213.180.199.34

ns4.yandex.ru internet address = 77.88.19.60

ns5.yandex.ru internet address = 213.180.204.1

> exit

3. Получение адреса почтового сервера для домена yandex.ru.

C:\ >nslookup

Default Server:  dns01.catv.ext.ru

Address:  217.10.44.35

> set q=mx

> yandex.ru

Server:  dns01.catv.ext.ru

Address:  217.10.44.35

Non-authoritative answer:

yandex.ru MX preference = 10, mail exchanger = mx2.yandex.ru

yandex.ru MX preference = 10, mail exchanger = mx3.yandex.ru

yandex.ru MX preference = 10, mail exchanger = mx1.yandex.ru

yandex.ru nameserver = ns2.yandex.ru

yandex.ru nameserver = ns1.yandex.ru

yandex.ru nameserver = ns4.yandex.ru

yandex.ru nameserver = ns5.yandex.ru

mx1.yandex.ru internet address = 77.88.21.89

mx2.yandex.ru internet address = 93.158.134.89

mx3.yandex.ru internet address = 213.180.204.89

ns2.yandex.ru internet address = 213.180.199.34

ns4.yandex.ru internet address = 77.88.19.60

ns5.yandex.ru internet address = 213.180.204.1

>

Указав ключ type=any, можно получить все записи о узле или домене. Ключи querytype, t, q эквивалентны type.

**Задания на лабораторную работу**

1. Изучите методические указания к лабораторной работе.
2. Выполните упражнения.
3. Оформите отчет по лабораторной работе, описав выполнение упражнений и дав краткие ответы на контрольные вопросы.

**Упражнение 1. Получение справочной информации по командам.**

Выведите на экран справочную информацию по всем рассмотренным утилитам (см. таблицу п.1). Для этого в командной строке введите имя утилиты без параметров и дополните **/?**.

Сохраните справочную информацию в отдельном файле.

Изучите ключи, используемые при запуске утилит.

**Упражнение 2. Получение имени хоста.**

Выведите на экран имя локального хоста с помощью команды hostname. Сохраните результат в отдельном файле.

**Упражнение 3. Изучение утилиты ipconfig.**

Проверьте конфигурацию TCP/IP с помощью утилиты ipconfig. Заполните таблицу:

| | |
|---|---|
| Имя хоста | |
| IP-адрес | |
| Маска подсети | |
| Основной шлюз | |
| Используется ли DHCP (адрес DHCP-сервера) | |
| Описание адаптера | |
| Физический адрес сетевого | |

| | |
|---|---|
| адаптера | |
| Адрес DNS-сервера | |
| Адрес WINS-сервера | |

**Упражнение 4. Тестирование связи с помощью утилиты ping.**

1. Проверьте правильность установки и конфигурирования TCP/IP на локальном компьютере.
2. Проверьте функционирование основного шлюза, послав 5 эхо-пакетов длиной 64 байта.
3. Проверьте возможность установления соединения с удаленным хостом.
4. С помощью команды ping проверьте адреса (взять из списка локальных ресурсов на сайте aspu.ru) и для каждого из них отметьте время отклика. Попробуйте изменить параметры команды ping таким образом, чтобы увеличилось время отклика. Определите IP-адреса узлов.

**Упражнение 5. Определение пути IP-пакета.**

С помощью команды tracert проверьте для перечисленных ниже адресов, через какие промежуточные узлы идет сигнал. Изучите ключи команды.

a) aspu.ru

b) mathmod.aspu.ru

c) yarus.aspu.ru

**Упражнение 6: Просмотр ARP-кэша.**

С помощью утилиты arp просмотрите ARP-таблицу локального компьютера.

Внести в кэш локального компьютера любую статическую запись.

**Упражнение 7: Просмотр локальной таблицы маршрутизации.**

С помощью утилиты route просмотреть локальную таблицу маршрутизации.

**Упражнение 8. Получение информации о текущих сетевых соединениях и протоколах стека TCP/IP.**

С помощью утилиты netstat выведите перечень сетевых соединений и статистическую информацию для протоколов UDP, TCP, ICMP, IP.

**Упражнение 9. Получение DNS-информации с помощью nslookup.**

1) Узнайте ip-адреса узлов, сайтов ростовских предприятий.

2) Узнайте авторитетные (компетентные) сервера для этих узлов.

3) Получите запись SOA с одного из этих серверов для домена выбранного вами домена.

## Контрольные вопросы

1. Раскрыть термины: хост, шлюз, хоп, время жизни пакета, маршрут, маска сети, авторитетный/неавторитетный (компетентный) DNS-сервер, порт TCP, петля обратной связи, время отклика.
2. Какие утилиты можно использовать для проверки правильности конфигурирования TCP/IP?
3. Каким образом команда ping проверяет соединение с удаленным хостом?
4. Каково назначение протокола ARP?
5. Как утилита ping разрешает имена узлов в ip-адреса (и наоборот)?
6. Какие могут быть причины неудачного завершения ping и tracert? (превышен интервал ожидания для запроса, сеть недоступна, превышен срок жизни при передаче пакета).
7. Всегда ли можно узнать символьное имя узла по его ip-адресу?
8. Какой тип записи запрашивает у DNS-сервера простейшая форма nslookup?

# Лабораторная работа № 3 Адресация узлов в сети. MAC-адрес, IP-адрес, доменное имя

.

**Цель работы**: Рассмотреть схему адресации узлов в ip-сетях. Получить представление о порядке разрешения адресов, используемых на различных уровнях стека TCP/IP.

Задания к работе

1. Определить физический и сетевой адреса локального хоста и его доменное имя.
2. Просмотреть таблицу преобразования физических адресов. Сохранить полученную информацию в файле.
3. Командой ping проверить доступность следующих узлов:
   - 127.0.0.1;
   - locahost;
   - example.com
   - трех-четырех соседних компьютеров.
4. Просмотреть таблицу преобразования адресов и сравнить ее с результатами, полученными в задании 1.
5. Сделать перерыв в сетевой активности на несколько минут, после которого повторить предыдущий пункт. Пояснить причины изменений (или отсутствия таковых) в таблице arp за время перерыва.
6. Добавить в таблицу *статическую* запись (действительные аппаратный и сетевой адреса одной из соседних машин)
7. Выполнить ping добавленного в предыдущем пункте сетевого адреса
8. Добавить в таблицу преобразований следующие записи (пары "mac-адрес — ip-адрес"):
   - действительный mac-адрес — недействительный сетевой адрес;
   - недействительный mac-адрес — действительный сетевой адрес;
9. Проверить доступность добавленных узлов. Объяснить полученные результаты.
10. Просмотреть таблицу arp и сохранить ее в файле для дальнейшего использования.
11. Перезагрузить компьютер и снова просмотреть кэш arp. Сравнить с результатами задания 9. Что стало с записями, добавленными вами в заданиях 5 и 7?
12. Добавить в файл hosts (путь к файлу в ОС Windows: %systemroot%\System32\Drivers\etc\hosts, в UNIX: /etc/hosts, в обоих случаях нужны привилегии администратора) следующую запись:
    194.188.210.1 edu.asoiu
    Если такая запись уже имеется, то перейти к следующему заданию
13. Выполнить ping узла edu.asoiu
14. Определить по таблице arp mac-адрес узла edu.asoiu.
15. Определить все ip-адреса (публичные) одного из указанных сервисов: mail.ru, ya.ru, google.com или подобного.
16. Определите имя и ip-адрес первичного DNS-сервера зоны ru.
17. Ответить на контрольные вопросы

Указания к работе

1.1.1 Преобразование адресов

В ip-сетях используется три типа сетевых адресов: mac-адрес, сетевой адрес и доменное имя. Они используются на разных уровнях сетевой модели для идентификации хостов.

Для сопоставления сетевого адреса с аппаратным адресом интерфейса в стеке TCP/IP имеются специализированные протоколы типа *arp* (address resolution protocol, RFC-826). Это позволяет использовать сетевые протоколы стека поверх различных протоколов канального уровня. Все операции преобразования выполняются прозрачно для протоколов верхних уровней. Результаты преобразований кэшируются и сохраняются на некоторый интервал времени, что позволяет не выполнять преобразование при повторном обращении к ранее взаимодействовавшим узлам.

Кэш arp представлен в виде таблицы, заполненной записями примерно такого вида:

"сетевой адрес — MAC-адрес — интерфейс — способ назначения"

Эта таблица формируется *динамически*, при любом сетевом взаимодействии узла. Для просмотра кэша arp используется одноименная команда — arp. Эта же команда позволяет формировать таблицу MAC-адресов *статически*, передавая записи через список аргументов. Команда arp используется как в UNIX, так и в Windows-системах.

Основной способ заполнения таблицы преобразований — динамический, при котором записи добавляются по мере участия узла в сетевом обмене. Это означает, что в отсутствие сетевой активности кэш arp пуст (если не задано статических записей). Для выполнения заданий к этой работе вам необходимо организовать некоторое сетевое взаимодействие. Пожалуй, самым доступным способом для этого является использование команды *ping*.

Команда ping использует протокол ICMP (Internet Control Message Protocol; RFC-792, RFC-1256) для отправки запросов датаграммного типа (ECHO_REQUEST) и ожидает ответ (ECHO_RESPONSE) от запрашиваемого хоста или шлюза.

ECHO_REQUEST — это датаграмма, имеющая заголовок IP и ICMP. Поле данных заполнено некоторым количеством произвольной информации. Для анализа сети выполняется отправка определенного количество таких датаграмм. По результатам анализа можно судить о доступности запрашиваемого хоста и некоторых аспектах работы сети в целом.

Обязательным параметром команды ping является сетевой адрес узла, заданный в числовом виде:

```
ping 192.0.32.10
```

или в символьном представлении:

```
ping example.com
```

Если задан символьный адрес, то ping попытается выполнить преобразование символьного имени в сетевой адрес. Для этого сначала будет перечитываться содержимое файла *hosts*, который является своего рода сервером DNS в масштабе отдельно взятого сетевого узла. Содержательно файл hosts — обычный текстовый файл, где прописано соответствие ip-адресов доменным именам. Его основное назначение — ускорить преобразование имен компьютеров в сетевые адреса. Формат файла приведен ниже:

```
#ip-address    hostname        aliaces
x.x.x.x        hostname        [aliace1 [aliace2 [...[aliaceN]]]]
```

Обычно в этом файле содержится единственная запись:

```
127.0.0.1      localhost
```

Если требуемое имя узла найдено в файле hosts, то возвращается соответствующий ему сетевой адрес. Иначе — выполняется запрос к внешнему серверу DNS, указанному в настройках сетевого интерфейса.

### 1.1.2 Как узнать MAC-адрес и ip-адрес?

Чтобы узнать физический адрес локального хоста и его ip-адрес нужно выполнить команду ifconfig (в ОС Windows - ipconfig). Запущенная без параметров, команда ifconfig

отображает информацию об имеющихся в системе сетевых интерфейсах и их физических и сетевых адресах:

```
aag@localhost:~> sudo /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:1D:92:A2:90:E7
          inet addr:192.168.1.250  Bcast:192.168.255.255  Mask:255.255.0.0
          inet6 addr: fe80::21d:92ff:fea2:90e7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:811957 errors:0 dropped:0 overruns:0 frame:0
          TX packets:446207 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:596559482 (568.9 Mb)  TX bytes:114698114 (109.3 Mb)
          Interrupt:28 Base address:0xe000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:24226 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24226 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:50861906 (48.5 Mb)  TX bytes:50861906 (48.5 Mb)
```

### 1.1.3 Как узнать доменное имя?

Узнать доменное имя хоста можно командой hostname.

### 1.1.4 Как узнать адрес сервера DNS?

Узнать адрес сервера DNS можно разными способами, самый простой — посмотреть содержимое файла resolv.conf:

```
cat /etc/resolv.conf
```

Расширенную информацию о сервере DNS можно получить используя специальные команды, такие как dig (man 1 dig) или host (man 1 host). В ОС Windows можно использовать утилиту nslookup.

В Листинге 1 приведен простой пример использования утилиты dig. Шрифтом выделено имя отвечающего сервера имен (сравните с записью в resolv.conf).

Листинг 1. Пример использования команды dig.

```
aag@localhost:~> dig example.com

; <<>> DiG 9.7.3 <<>> example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34206
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;example.com.                   IN      A

;; ANSWER SECTION:
example.com.            1095    IN      A       192.0.32.10

;; AUTHORITY SECTION:
example.com.            172792          IN      NS      a.iana-servers.net.
example.com.            172792          IN      NS      b.iana-servers.net.
```

```
;; ADDITIONAL SECTION:
b.iana-servers.net.     28792  IN      A       193.0.0.236
b.iana-servers.net.     28792  IN      AAAA    2001:610:240:2::c100:ec

;; Query time: 1 msec
;; SERVER: 192.168.3.1#53(192.168.3.1)
;; WHEN: Thu Apr  7 10:34:46 2011
;; MSG SIZE  rcvd: 137
```

## Контрольные вопросы

1. Mac адрес и его структура.
2. ip- адрес и его структура.
3. Для чего применяется маска подсети.
4. Какие есть специальные ip – адреса.
5. Что произойдет, если в таблицу arp добавить две или более записей, в которых одному mac-адресу сопоставлены разные сетевые адреса?
6. Что произойдет, если в таблицу arp добавить две или более записей, в которых одному сетевому адресу сопоставлены разные аппаратные адреса?
7. Как отличается "время жизни" динамических и статических записей в таблице arp?
8. Почему в ip-сетях не используется прямое сопоставление символьного адреса физическому адресу?
9. Что произойдет, если в файл hosts записать два (или более) узла с одинаковыми именами (например, myhost.mydomain), но разными сетевыми адресами, а затем обратиться к ним по имени (например так: ping myhost.mydomain

# Лабораторная работа № 4 Практическое создание web страницы с применением html5 и CSS3

**Цели занятия**: Научиться создавать файлы – интернет - страницы, содержащие элементы форматирования, написанные на языке HTML.

**Теоретический материал**

HyperTextMarkupLanguage (HTML) – это язык разметки документов во Всемирной паутине, принятый за стандартный. Большая доля всех Web-страниц в Интернете создана при помощи языка HTML (или XHTML), поэтому мы рассмотрим его подробно.

Язык HTML позволяет форматировать текст и другие элемента Web-страницы:

*Цвет, жирность, стиль, название шрифта* для текста.

*Позволяет выделять фрагменты и символы* например: ударение в слове, заголовок страницы или абзаца, сам абзац, пункт списка.

*Гипертекстовые ссылки*, позволяют переходить между документами и между фрагментами одного документа.

*Формы* для введения данных, как правило, данные из форм обрабатываются с помощью скриптов на языках программирования, ориентированных на Web,например PHP.

*Отображение мультимедийных файлов*, их может отображать сам браузер – изображения, аудиофайлы, или внешние приложения, взаимодействующие с браузером, например Flash-ролики, Java-апплеты.

HTML – язык разметки документов основанный на тэгах. Документ на языке HTML представляет собой набор элементов, при этом начало и конец каждого элемента обозначается служебными символами – тегами. Все тэги HTML начинаются с «<« (левой угловой скобки) и заканчиваются символом «>« (правой угловой скобки). Завершающий тег выглядит также, как начальный, и отличается от него прямым слэшем перед текстом внутри угловых скобок.

**<HTML></HTML>**

HTML регистронезависимый язык, теги могут быть написаны как строчными, так и заглавными буквами (в отличие от XHTML). Теги могут быть вложенными друг в друга.

<**HTML>**
**<HEAD>**
**<TITLE>**
Заголовок страницы
**</TITLE>**
**</HEAD>**
**<BODY>**
*Всё содержание документа – web страницы*
**</BODY>**
**</HTML>**

В общем виде синтаксис записи тега <BODY> со всеми допустимыми атрибутами выглядит так:

<BODY BACKGROUND="URL" BGCOLOR="значение1" TEXT="значение2" LINK="значение3" VLINK="значение4" ALINK="значение5">

тело документа HTML

<BODY>

Атрибут BACKGROUND позволяет дизайнеру поместить на web-страницу некий фоновый рисунок, записав в качестве параметра атрибута URL этого рисунка. URL можно задавать либо в виде полного адреса Интернета (например,"http://www.server.ru/imaes/имя_файла.gif"), либо в виде сокращенного адреса с указанием пути к директории на текущем сервере, в которой хранится данное изображение (например, ".. images/имя_файла.gif"). Допускается просто указывать имя графического файла, если он хранится в той же директории, что и использующий его файл HTML. Данное изображение может иметь любой размер, поскольку при интерпретации кода оно многократно повторяется, заполняя все доступное пространство в окне броузера. Подробно о правилах включения графических файлов в html-документ мы поговорим в ходе следующего урока.

**СОВЕТ** Для того чтобы избежать неадекватности отображения того или иного цвета различными броузерами, например, когда web-дизайнер решил применить на странице заливку какого-либо редко используемого оттенка, рекомендуется следующий подход: создайте в любом подходящем редакторе графический файл нужного цвета размерами 1x1 пиксел, после чего укажите его в качестве фонового изображения, включив URL этого рисунка в параметр атрибута BACKGROUND тега <BODY>.

Атрибут TEXT позволяет задать цвет текста для всего документа в целом. Но не забывайте, что параметр, назначенный данному атрибуту в теге <BODY>, можетбыть изменен в определенном участке текста путем использования команды<FONT> с атрибутом COLOR.

Для того чтобы назначить фоновый цвет всему документу, используется атрибут BGCOLOR. В этом случае web-страница будет целиком заполнена равномерной заливкой указанного цвета.

**ПРИМЕЧАНИЕ** Атрибуты BGCOLOR и BACKGROUND не исключают друг друга, однако у последнего имеется приоритет. Это означает, что в случае, когда заданы оба эти атрибута, сначала выполняется заливка web-страницыцветом, назначенным в атрибуте BGCOLOR, поверх которой размещается изображение, заданное атрибутом BACKGROUND.

**ВНИМАНИЕ** Если вы не используете графических изображений в качестве фонового рисунка, а основным цветом html-документа приняли белый, использование атрибута BGCOLOR с параметром "#FFFFFF" в составе тега<BODY> обязательно. Обусловлено это правило следующей причиной: некоторые броузеры (например, Microsoft Internet Explorer) позволяют пользователям произвольно изменять фоновый цвет загружаемых web-страниц, если он не задан явно. Учтите, что пренебрежение явным указанием фонового цвета может вызвать полное нарушение разработанного вами дизайна.

Атрибут LINK дает web-мастеру возможность назначить цвет, которым отображается не посещенная гиперссылка, то есть ссылка, к которой посетитель данного web-сайта еще не обращался. По умолчанию ей присваивается значение"blue" (IOOOOFF). В свою очередь, атрибут VLINK указывает на цвет посещенной ссылки, значение которой по умолчанию — "purple" (#800080). И наконец, атрибут ALINK определяет цвет активной гиперссылки, то есть цвет, который гиперссылка принимает с момента нажатия на нее курсором мыши до момента загрузки вызываемого ею ресурса. По умолчанию данный атрибут также имеет значение "purple".

Очевидно, что значения всех атрибутов тега <BODY>, кроме атрибута BACKGROUND, представляют собой обозначения цветов символьными метками или шестнадцатеричным цифровым кодом.

**ПРИМЕЧАНИЕ** При выборе цветов документа, текста и гиперссылок следует соблюдать определенную осторожность, поскольку цвета, контрастно отображаемые на цветном мониторе, могут быть неразличимы на черно-белом. Избежать подобных ошибок можно при помощи несложного приема: сделайте снимок экрана («скриншот») в момент, когда ваша web-страницазагружена в броузер (для этого необходимо нажать кнопку Print Screenна клавиатуре компьютера), загрузите полученное изображение в графический редактор и просмотрите его в режиме grayscale (256 оттенков серого). Если выбранная вами цветовая схема по-прежнему выглядит контрастно, ее можно смело применять, если нет — придется использовать другие цвета.

**СОВЕТ** Подбирая цвета для web-страницы, помните, что используемая вами цветовая схема должна быть «корректной». Чтение текста не должно вызывать затруднений, глаза посетителей вашей странички не должны уставать. Пожалейте зрение пользователей, не пишите оранжевым по зеленому.

**ВНИМАНИЕ** Установив одно из значений цветовых параметров вашей страницы, жестко задавайте и остальные значения. Некоторые броузеры позволяют пользователю произвольно менять цвет фона документа или выводимого на экран текста, если они не указаны явно. В этом случае заданный вами цвет текста может совпасть с цветом фона, установленным в броузере пользователя по умолчанию, в результате чего текст станет нечитаемым. Изменение одного из цветовых параметров без изменения остальных недопустимо. Исключение можно сделать лишь в том случае, когда в качестве фонового цвета web-страницы используется белый.

Вот пример использования тега <BODY> со всеми допустимыми атрибутами:

<BODY BACKGROUND="http://www.myserver.com/images/back.jpg" BGCOLOR="#000000" TEXT="#FFFFFF" LINK="#008000" VLINK="#800080" ALINK="#FF0000">

Тэги могут быть пустыми, то есть не содержать текста или других вложенных конструкций (например, <**br**> который переводит строку). Закрывающий тег в таком случае не указывается.

Также, элементы разметки могут иметь атрибуты, задающие их свойства (например, размер шрифта, цвет, расположение). Атрибуты задаются в начале тега.

**<ahref=«**http://www.yandex.ru**»>**Пример элемента с атрибутом **href.</a>**

Теги можно разделить на следующие группы.

*Гиперссылки*

**<AHREF=«**filename**»target=«_self»>**текст ссылки**</A>**

где filename – имя файла (может быть и локальным) или адрес страницы вInternet, на который нужно совершить переход.

текст ссылки – текст гипертекстовой ссылки, который будет отображаться в браузере, как правило, выделяется подчеркиванием.

target – задает окно или фрейм, в котором будет открыт документ, при переходе по ссылке. Он может принимать значения:

_top – документ откроется в текущем окне

_blank – документ откроется в новом окне

_self – документ откроется в текущем фрейме

_parent – документ откроется в родительском фрейме

По умолчанию принимает значение _self.

*Текстовые ссылки.*

**<H1></H1>,<H2></H2>**, … ,**<H6></H6>** – заголовки 1-6 уровней. Применяются для выделения частей выводимого текста (заголовок 1 – будет выведен очень большим, 6 – будет размером сопоставимым с обычным текстом).

**<P>** – обозначает начало нового абзаца. Закрывающий тег**</P>**, не является необязательным.

**<BR>** – переход на новую строку. Закрывающий тег**</br>**, отсутствует.

**<HR>** – горизонтальная линия.

**<BLOCKQUOTE></BLOCKQUOTE>** – цитата. Выделение заданного текста как цитаты.

**<PRE></PRE>** – режим предпросмотра. При этом текст заключается в рамку и выводиться не форматированным (то есть все теги, кроме </PRE>, игнорируются, но при этом переводы строки ставятся там, где они присутствуют в исходном документе).

**<DIV></DIV>** – блок текста (как правило, применяется для использования каскадных стилей CSS).

**<SPAN></SPAN>** – строка (как правило, применяется для использования каскадных стилей CSS).

*Теги форматирования текста*

**<EM></EM>** – выделение символа, на который падает ударение (обычно отображается курсивом).

**<STRONG></STRONG>** – выделение символа, на который падает усиленное ударение (обычно отображается жирным текстом).

**<I></I>** – выделение текста курсивом.

**<B></B>** – выделение текста жирным шрифтом.

**<U></U>** – подчёркивание текста

**<S></S>** – зачёркивание текста.

**<STRIKE></STRIKE>** – то же самое, что <S> … </S>

**<BIG></BIG>** – увеличение шрифта.

**<SMALL></SMALL>** – уменьшение шрифта.

**<BLINK></BLINK>** – мигающий текст.

**<MARQUEE></MARQUEE>** – сдвигающийся по экрану текст.

**<SUB></SUB>** – вывод текста под строкой. Например, **H<SUB>2</SUB> O**тобразиться в виде текста H2O.

**<SUP></SUP>** – вывод текста над строкой. Например, **E=mc<SUP>2</SUP>**отобразиться в виде текста E=mc2.

**<FONT атрибуты></FONT>** – задание атрибутов у используемого шрифта. Атрибуты могут быть следующими:

**COLOR=color** – указание цвета. Цвет может быть указан шестнадцатеричным числом в формате #rrggbb (первые 2 шестнадцатеричные цифры указывают интенсивность красного, следующие 2 – зелёного, последние 2 – синего) или названием самого цвета.

FACE= указываем имя шрифта.

SIZE= позволяет изменить размер шрифта. Размеры могут быть от 1 до 7, по умолчанию размер 3.

SIZE=+ размер или SIZE=-размер – размер больше или меньше стандартного. Например,SIZE=+2 указывает размер на 2 больше стандарта, то есть размер 5.

***Списки.***

Данная конструкция

**<UL>**

<LI> первый элемент списка </LI>

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

создаёт список вида:
- первый элемент;
- второй элемент;
- третий элемент.

Также стоит отметить, что тегов есть параметры, позволяющие менять вид списка.

*Объекты.*

EMBED – вставка объектов различных типов

APPLET – вставка Java-апплетов

SCRIPT – вставка различных скриптов, например JavaScript

*Изображения.*

IMG – тег для вставки изображения. Это не закрывающийся тег.

SRC – имя локального файла или путь к нему в виде URL

ALT – текст картинки (отобразится, в виде текста, если не удалось отобразить картинку)

TITLE – подсказка (показывается при попадании курсора в область картинки)

WIDTH, HEIGHT – размеры изображения (выводимое изображение будет масштабировано до указанных размеров)

ALIGN – обтекание текста

*Таблицы.*

TABLE – тег создание таблицы. Тег имеет следующие параметры:

BORDER – задает толщину границу таблицы

CELLSPACING – задает расстояние от ячейки до ячейки

CAPTION – задает заголовок таблицы (необязательный тег)

TR – добавление строки в таблицы

TH – задает заголовок столбца (необязательный тег)

TD – добавление ячейки таблицы

WIDTH, HEIGHT – размеры таблицы

*Формы.* Формы ввода данных могут быть самыми разнообразными. Поэтому рассмотрим только основные теги:

FORM – тег для создания формы

INPUT – добавление элемента ввода

TEXTAREA – добавление текстового поля

SELECT – добавление списка (как правило, это выпадающее меню)

OPTION – пункт списка

*Символы.* Некоторые символы не могут быть выведены напрямую. Для их вывода требуется использовать их определения, например, символ апмерсанд& в коде HTML будет иметь вид **&amp**, символ меньше < будет иметь вид **&lt**, символ больше > будет **&gt**. Это ограничение введено, так как эти символы уже используются в языке HTML как служебные.

Любая HTML-страница должна иметь обозначение начала и конца документа обрамленные тегами**<html>** и **</html>** соответственно. Внутри них должны находиться теги заголовка **<head>** и **</head>**, итеги, обозначающие тело документа **<body>** и **</body>**. А внутри них могут быть произвольные комбинации из групп тегов описанных ранее.

Также рассмотрим ExtensibleHypertextMarkupLanguage (XHTML) это расширяемый язык разметки гипертекста. Стоит отметить, что язык XHTML это ни описание самого языка, а список отличий XHTML от HTML. Рассмотрим основные отличия HTML и

XHTML. В XHTML все используемые теги должны иметь закрывающий тег. Теги, не имеющие закрывающего тега должны оканчиваться символом /. Например тег **<br>**, должен иметь закрывающий его тег **<br />**. В XHTML допускается писать теги и их атрибуты только строчными буквами. В XHTML очень строгая проверка синтаксиса не допускается использовать< и &, даже в URL, вместо них должны быть **&lt** и **&amp**. Браузеры, обнаружив ошибку синтаксиса в XHTML, должны прекратить его обработку и вывести ошибку на экран. В стандарте HTML браузер должен попытаться отобразить запрашиваемый документ. Стоит отметить, что XHTML расширяемый язык – за счет указания типа документа и возможности использовать свои теги.

Как мы видим, язык разметки HTML предоставляет широкие возможности для отображения информации, для этого в нем содержится большое количество тегов для различного форматирования выводимой информации. Язык XHTML очень похож на HTML, но более строгий, грамматические правила в XHTML менее сложные, и как следствие при созданий Web-страниц будет меньше ошибок.

**Ход работы**:

I. Создание простейших файлов HTML.

1. Создаём папку, в которой будем сохранять созданные Web-страницы.
2. Запускаем программу Notepad++ и набираем следующий текст с элементами форматирования:

```
<HTML>
<HEAD>
<TITLE> Учебный файл HTML </TITLE>
</HEAD>
<BODY>
Расписание занятий на вторник
</BODY>
</HTML>
```

3. Сохраняем файл под именем *schedule.html*.
4. Для просмотра созданной Web-страницы используем браузер (Рис.1).

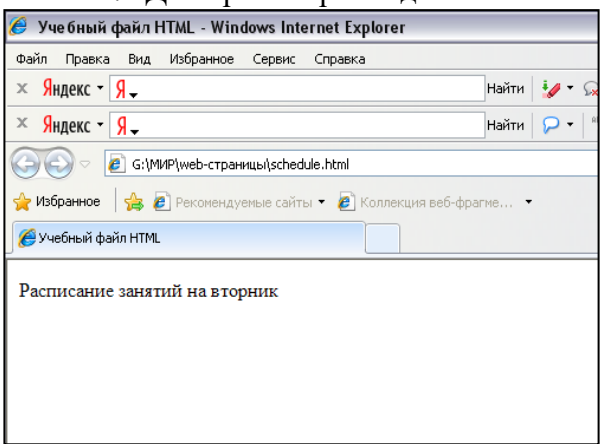



Рис.1 «Учебный файл HTML»

II. Управление расположением текста на экране.

1. Вносим изменения в текст, расположив слова "Расписание", "занятий", "на вторник" на разных строках:

```
<HTML>
<HEAD>
```

<TITLE> Учебный файл HTML </TITLE>
</HEAD>
<BODY>
Расписание
занятий
на вторник
</BODY>
</HTML>

2. Сохраняем внесенные изменения.
3. Новая полученная Web-страница не изменилась (см. Рис.1). Для того, чтобы переносить текст используют теги. Тег перевода строки <BR> отделяет строку от последующего текста или графики. Тег абзаца <P> тоже отделяет строку, но еще добавляет пустую строку..
4. Вносим изменения в текст файла HTML:

<HTML>
<HEAD>
<TITLE> Учебный файл HTML </TITLE>
</HEAD>
<BODY>
Расписание <P>занятий <BR>на вторник
</BODY>
</HTML>

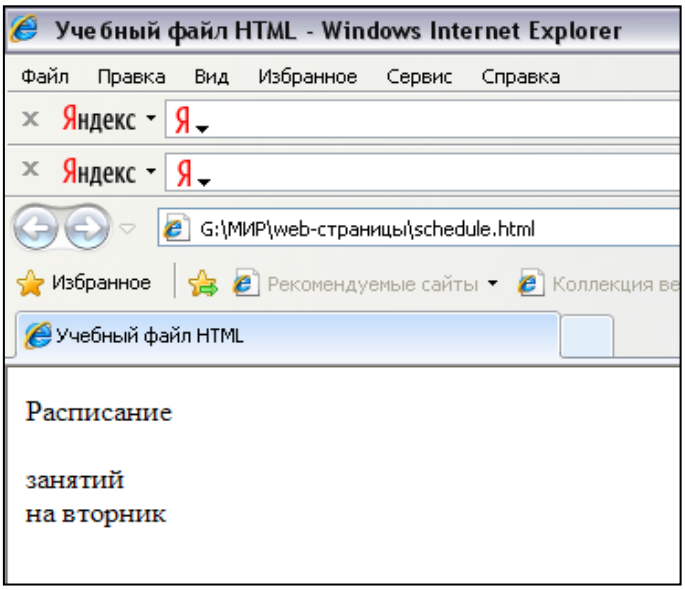



Рис.2 «Изменённый файл»

III. Выделение фрагментов текста.

Существует три тега выделения фрагментов текста: <B> : </B> — для выделения полужирным, <I> : </I> — для выделения курсивом, <U> : </U> — для выделения подчеркиванием.

1. Вносим изменения в файл *schedule.html*: Рис.3

```
<HTML>
<HEAD>
<TITLE> Учебный файл HTML </TITLE>
</HEAD>
<BODY>
<B>Расписание</B> <I>занятий</I> <U> на вторник</U>
</BODY>
</HTML>
```

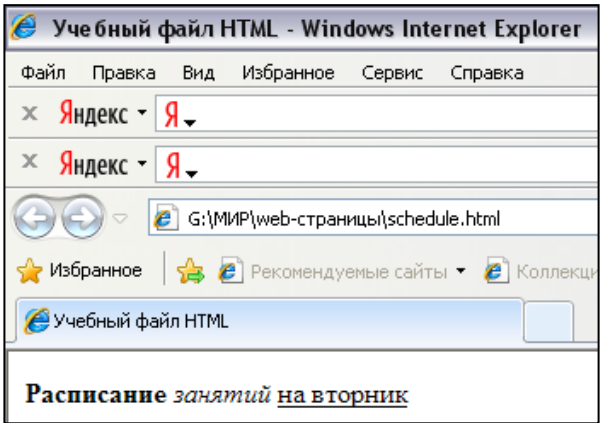



Рис. 3 «Новый вид»

2. Также используем вложение тегов:
<I><B>Расписание</B></I> <I>занятий</I> <U> на вторник</U> (Рис.4).

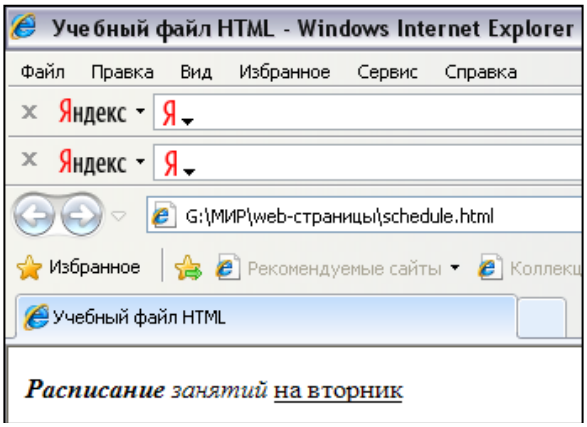



Рис.4 «Ещё один вариант»

IV. Задание для самостоятельной работы: Создайте 2 собственных HTML документа – описание двух ваших товаров или услуг, содержащие фоновый рисунок страницы, элементы форматирования текста, нумерованный и ненумерованный списки, ссылки для скачивания файлов. Для работы использовать только редактор Notepad++ и браузер.


**Часть 2**

**Ход работы**:

1. Создаём документ с именем 2019.html и вставляем туда основную часть документа:
<html>
<head>
</head>
<body>
</body>
</html>
2. В головной части задаём заголовок страницы и конструкцию использования каскадных таблиц, которые мы расположим в документе main.css.

 используется для логического выделения блока HTML-документа. Элемент группировки, как и элемент SPAN. В современном сайтостроении используется как удобный контейнер для объектов страницы, которым легко динамически манипулировать - перемещать, включать/выключать, создавать слои, регулировать отступы и т.п.

В браузеронезависимой вёрстке обычно используется для выравнивания блока html-кода в окне браузера.

Находящиеся между начальным и конечным тегами текст или HTML-элементы по умолчанию оформляются как отдельный параграф.

```
<div align="center"><a href="r/8.jpg"> <img src="r/8.jpg"> </a>

<br>
Рисунок 1. Астероид подлетающий к Земле ( художественная фантазия)</div>
<br>
   <b>Первый в этом списке</b>: астероид <b>Апофис</b> (если т
Apophis, ранее бывший просто 2004 MN4, приблизительный диаметр 270 метров, +/
учет неправильной формы), Открытый Американскими учеными ( Аризонская обсерв
```

**Рис.1 «Элемент div»**

На рисунке видно, что ссылка на картинку и подпись к ней расположены в  и расположены по центру.

3. Текст заключаем в параграф . Для пробелов используем специальный символ « ». Для перехода на другую строку необходимо ставить в том месте <br>. Также для использования css для разных элементов документа нужно задавать им имена, например см.Рис.2.

```
    <div id="div1">
    <div id="div2">
    <p>  <H2>Астероидная атака. Реальные обьек
  Из всех астероидов, известных астрономам, вы
Земли. Таких, представляющих некую угрозу выявлено около
   Но какие именно, ожидаются вблизи наше
```

**Рис.2 «Имя элемента»**

Имя задаётся при помощи конструкции id=”имя”, заключённой в скобках элемента <>.

4. Для вставки картинки используется конструкция
<img src="адрес/имя.расширение">.

Если же требуется, чтобы при нажатии картинка открывалась отдельно в новом окне, то мы используем следующую конструкцию (Рис.3).

```
Земли. Таких, представляющих некую угрозу выявлено около 500 объектов.<br>
   Но какие именно, ожидаются вблизи нашей планеты, и главное, когда?<br><br>

<div align="center"><a href="r/8.jpg"> <img src="r/8.jpg"> </a>

<br>
Рисунок 1. Астероид подлетающий к Земле ( художественная фантазия)</div>
<br>
```

**Рис.3 «Ссылка на картинку»**

5. Для вставки ссылки в документ на скачивание видео тоже используется конструкция:
(<A HREF="имя.расширение">"текст"</A> )
Как выглядит нажатая ссылка для скачивания можно увидеть на рисунке 4.

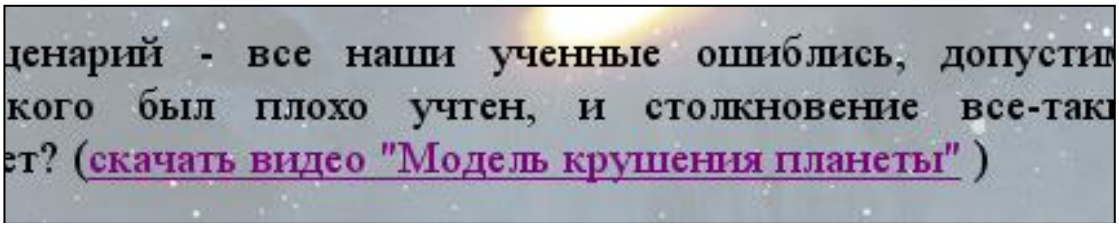



**Рис.4 «Нажатая ссылка»**

6. Таблица задается тэгом: <table></table>

таблица состоит из строк и столбцов (ячеек), надо еще указать и их: <tr></tr> - строчка таблицы; <td></td> - столбец (ячейка) таблицы.

фоновый цвет и виды рамочек мы задали в main.css (Рис.5)

```
table{
border-style:double;
border-color:white;
border-collapse:separate;
text-align: center;
background-color: silver;
border-spacing: 3px 3px;
font-size: 110%;
}
```

**Рис.5 «Параметры таблицы»**

Вид ячеек также задаём в том же документе:

```
td{
border-style:double;
border-color:white;
border-collapse:separate;
background-color: silver;
}
```

**Рис.6 «Параметры ячеек»**

7. Вид готовой страницы с таблицей

Как видно на схеме более 70% вероятности падения в океан и трудно доступные районы Сибири - 20%.

Таблица прогнозируемых последствий падения.

| Последствия падения на расстоянии от эпицентра(км) | 10 | 50 | 120 |
|---|---|---|---|
| Сила землетрясения (по шкале Рихтера) | 6,5 | 5,6 | 4,9 |
| Скорость ветра(ударная волна) м/с | 792 | 77,8 | 44,7 |
| Разрушения | Обрушение укреплённых зданий, обрушение туннелей в метро, трещины в земле | Разрушение неукреплённых зданий, разрывы трубопроводов | Падение мебели, штукатурки, незначительные последствия |
| Состояние населения(жертвы,%) | 99,0 | 70,0 | 5,0 |

Анализируя таблицу, видим, только попав в центр большого города, астероид может вызвать серьезные людские жертвы, а это весьма маловероятно. Взрыв термоядерной бомбы намного страшнее. И даже причинив огромнейшие разрушения,

**Рис.7 «Таблица»**

V. Задание для самостоятельной работы: Создайте собственный HTML документ – прайс - лист ваших товаров или услуг, содержащий фоновый рисунок страницы, элементы форматирования текста, таблицу, рисунки в ячейках таблицы, ссылки для перехода на страницы, созданные в первой части лабораторной работы, на этих страницах добавить ссылки для возврата на страницу с прайс листом.

VI. Для работы использовать только редактор Notepad++ и браузер.

VII. Созданные файлы вместе с файлами рисунков запаковать в архив с именем «ПЗ1 *Ваша Фамилия»* и отправить для проверки преподавателем

## Контрольные вопросы

1. В чем отличие CSS3?
2. Что такое селекторы?
3. Какими свойствами обладают селекторы потомков?
4. Что такое селекторы дочерних элементов?
5. Как проявляют себя селекторы элементов одного уровня?
6. Что такое псевдоклассы?
7. Как проявляют себя псевдоклассы дочерних элементов?
8. Что такое псевдо-классы форм?
9. Что такое псевдоэлементы?
10. Для чего применяют селекторы атрибутов?
11. В чем сущность наследования стилей?
12. В чем сущность каскадности стилей?

# Лабораторная работа № 5 Работа с JavaScript. Размещение JavaScript на HTML странице

**Тема:** Работа с JavaScript. Размещение JavaScript на HTML странице
**Цель занятия:** Ознакомить с основами языка JavaScript
**Задания:**

1 Ознакомьтесь с теоретическими аспектами темы.
2 Создайте простую веб-страницу с использованием JavaScript согласно методическим указаниям.
3 Создайте веб-страницу с формой и кнопкой на основе JavaScript согласно методическим указаниям.
4 Создайте интерактивную страницу с формой и диалогом, производящую расчёты для определения стоимости или физического объёма ваших товаров(услуг) и подключите её по ссылкам к созданным ранее страницам.

**Необходимые приборы:** ПК, текстовый редактор Блокнот, браузер

**Методические рекомендации к практическому занятию**

## 1. Введение

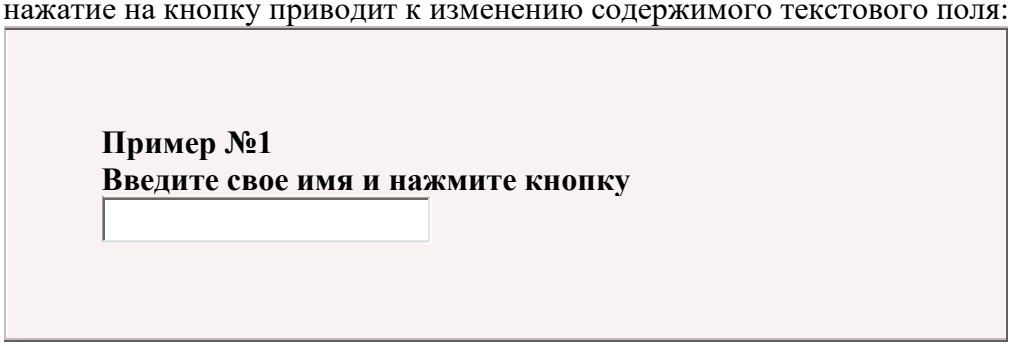
Язык программирования JavaScript был разработан Бренданом Эйком (Brendan Eich) в Netscape Communications как язык сценариев для обозревателей Netscape Navigator, начиная с версии 2.0. В дальнейшем к развитию этого языка подключилась корпорация Microsoft, чьи обозреватели Internet Explorer поддерживают JavaScript, начиная с версии 3.0. Версия Microsoft получила название JScipt, поскольку JavaScript является зарегистрированной маркой фирмы Netscape.

Код скрипта JavaScript размещется непосредственно на HTML-странице. Все, что стоит между тэгами <script> и </script>, интерпретируется как код на языке JavaScript. Инструкция*document.write()* - одна из наиболее важных команд, используемых при программировании на языке JavaScript. Команда *document.write()* используется, когда необходимо что-либо написать в текущем документе (в данном случае таком является наш HTML-документ).

События и обработчики событий являются очень важной частью для программирования на языке JavaScript. События, главным образом, инициируются теми или иными действиями пользователя. Если он щелкает по некоторой кнопке, происходит событие *"Click"*. Если указатель мыши пересекает какую-либо ссылку гипертекста - происходит событие *MouseOver*. Существует несколько различных типов событий. Мы можем заставить нашу JavaScript-программу реагировать на некоторые из них. И это может быть выполнено с помощью специальных программ обработки событий. Так, в результате щелчка по кнопке может создаваться выпадающее окно. Это означает, что создание окна должно быть реакцией на событие щелка - *Click*. Программа - обработчик событий, которую мы должны использовать в данном случае, называется *onClick*. И она сообщает компьютеру, что нужно делать, если произойдет данное событие.

Вы можете использовать в скрипте множество различных типов функций обработки событий. В большинстве случаев функции представляют собой лишь способ связать вместе нескольких команд. Функции могут также использоваться в совместно с процедурами обработки событий.

JavaScript - это объектно-ориентированный язык программирования (ООП), основан не на обработке команд кода, а на присвоении отдельным элементам программы конкретных событий и выполнении их, если данное событие имело место. Например, событие нажатие на кнопку приводит к изменению содержимого текстового поля:




Основными понятиями любого объектно-ориентированного языка являются объекты, классы, методы и свойства. Разберём основные понятия на конкретных примерах:

```
<script type="text/javascript">
document.write("Введите свое имя и нажмите кнопку")
</script>
```

В результате при просмотре данной страницы в браузере появится текст: "Введите свое имя и нажмите кнопку".

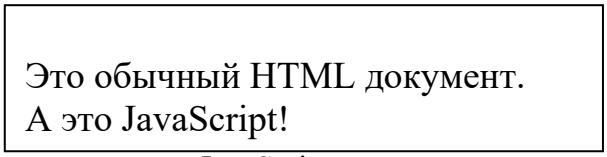
**ЗАДАНИЕ 1**

1. Запустите Notepad++ . Установите кодировку UTF-8,
   Синтаксисы – Н – HTML .
2. Введите текст

```
<html>
<body>
<br>
Это обычный HTML документ.
<br>
<script language="JavaScript">
  document.write("Аэто JavaScript!")
</script>
<br>
Вновь документ HTML.
</body>
</html>
```

3. Сохраните документ в формате html
4. Запустите страницу в окне браузера.

Результат выполнения файла в случае, если используемый браузер поддерживает JavaScript:




Если браузер не поддерживает JavaScript, то он проигнорирует тег <script>. В этом случае измените исходный текст:

```
<html>
<body>
<br>
```

```
Это обычный HTML документ.
<br>
<script language="JavaScript">
<!-- hide from old browsers
   document.write("Аэто JavaScript!")
// -->
</script><br>
Вновь документ HTML.
</body>
</html>
```

В этом случае использован тег комментария из HTML - <!-- -->. В результате новый вариант нашего исходного кода будет выглядеть как:

```
Это обычный HTML документ.
Вновь документ HTML.
```

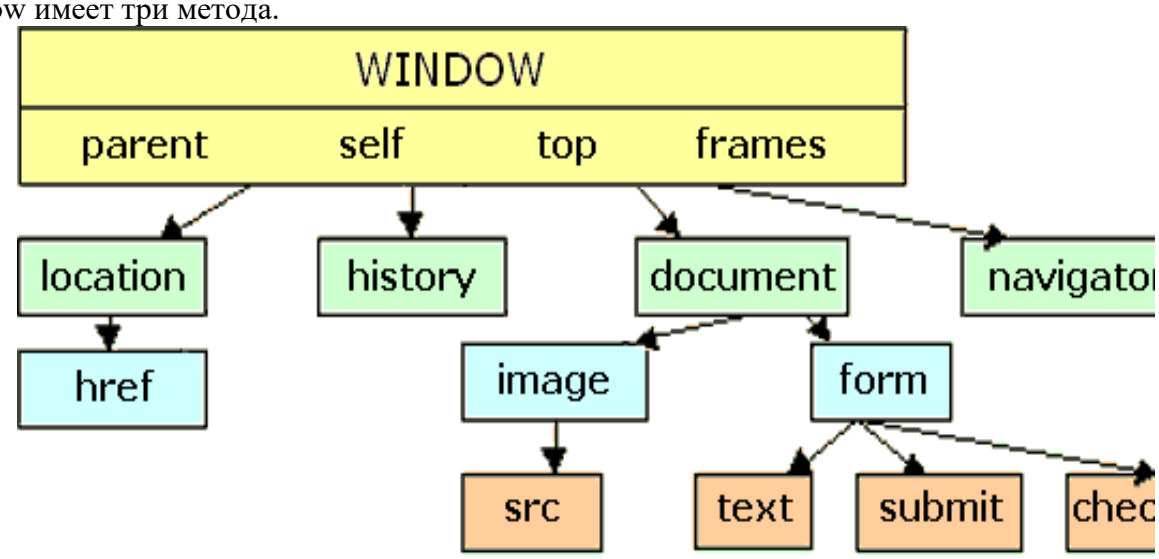
## 2. Основные понятия JavaScript: объект, метод, свойства, события.

**OBJECT** *(объект)* - это то, с чем производится действие, событие. Это может быть документ, открываемый в окне браузера или само окно браузера, или какая-то часть документа, теги. Объект должен иметь уникальное имя (ID), чтобы к нему можно было обратиться.

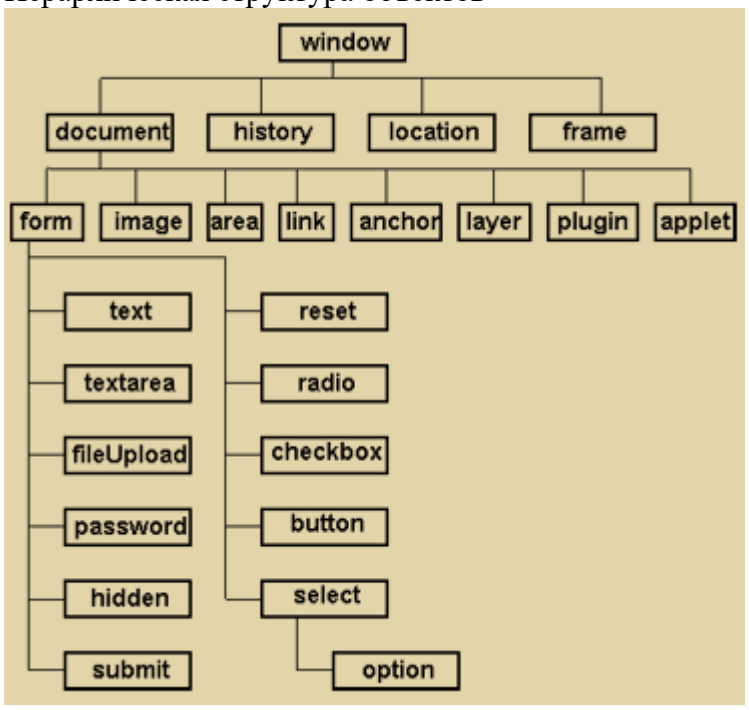
В нашем случае объектом является документ HTML и к нему можно просто обратиться по имени: **document**.

**Иерархия объектов в JavaScript**

В языке JavaScript определены объекты, которые называются объектами браузера. Каждый объект соответствует некоторому элементу Web- страницы: окну, документу, изображению, ссылке и т.п. Управлять частями документа можно с помощью методов браузера. Объекты браузера имеют иерархическую структуру. На самом верхнем уровне располагается объект window. Он является родителем остальных объектов. Объект window представляет окно браузера. Свойство window.status можно использовать для изменения вида строки состояния. Для отображения диалоговых сообщений объект window имеет три метода.




Иерархическая структура объектов



другие объекты JavaScript

Каждый объект обладает своими методами.

**METHOD** *(метод объекта)* - это действия, которые можно выполнять над объектом такого типа, или которые сам объект может выполнять.

Синтаксис кода: между именем объект и методом обязательно ставят разделительный оператор точка, после метода в скобках параметры метода.

```
Объект.Метод("параметры метода")
```

Параметры метода относятся к типу данных - строки символов. Строки символов нужно обязательно взять в кавычки либо в одинарные, либо в двойные.

Каждый объект обладает своими свойствами.

**PROPERTY (свойство)** - каждый объект имеет свои свойства. Один и тот же объект может обладать многими свойствами. Часто эти свойства необходимо изменить, при возникновении некоторого события.

Для изменения свойства объекта необходимо соблюдать следующий синтаксис:

```
Объект.свойство объекта = "новое значение свойства "
```

Например, для изменения фонового цвета документа HTML (имя данного свойства bgColor) следует написать следующее:

```
<script type="text/javascript">
document.bgColor='gold'
</script>
```

И при просмотре в окне браузера фоновый цвет HTML документа будет золотым.Коды цветов можно посмотреть в Википедии

Обратите внимание на то, что значение свойства gold пишется в кавычках (одинарных или двойных), т.к. значение свойства относится к типу данных строки символов.

**Задание №2**

Добавьте в созданный ранее файл страницы скрипт, задающий свойство документа - фон красного цвета

Разумеется, нас будет интересовать возможность изменения свойства при возникновении какого-либо события.

**EVENT (событие)** - это все, что случилось: открытие окна, загрузка в него документа, клик клавишей мышки или просто перемещение курсора по экрану, нажатие клавиши на клавиатуре - это все события, и они могут инициировать запуск больших и маленьких программ.

**Стандартные события в HTML**

| **имя события** | **происходит** |
|---|---|
| onclick | при щелчке кнопки мыши на элементе |
| ondblclick | при двойном щелчке кнопки мыши на элементе |
| onmousedown | при нажатии кнопки мыши на элементе |
| onmouseup | при отпускании кнопки мыши на элементе |
| onmouseove | при попадании курсора мыши на элемент |

| | |
|---|---|
| r | |
| onmousemove | при движении курсора мыши по элементу |
| onmouseout | при попадании курсора мыши за пределы элемента |
| onkeypress | при нажатии и отпускании клавиши на элементе |
| onkeydown | при нажатии клавиши на элементе |
| onkeyup | при отпускании клавиши на элементе |

Здесь следует пояснить, что события (event) и обработчики событий (event handler) относятся к JavaScript, но они скорее «встроены» в HTML-код. Они входят в структуру документа HTML и не требуют тегов <script> и </script>. Среди разнообразных обработчиков событий для начала мы выберем один, самый популярный, — **onmouseover** (навести мышь).

**Пример №2**
*...пример смотрите здесь* - *в другом окне*

Код выглядит следующим образом:

Наведи мышь на этот текст .... 

Как уже говорилось для написания этого кода не требуются теги <script> </script>. Событие встраивается в HTML код, т.е является описанием, атрибутом тега (в данном случае тега ) при выполнении данного события - наведении мышкой на текст данного абзаца - изменяется свойство объекта - фон документа HTML.

И здесь есть еще одна важная особенность: **document.bgColor ='red'** нужно также записать в кавычках - одинарных или двойных. Вы можете использовать любой тип кавычек. Однако если Вы вынуждены как в данном случае ставить кавычки дважды, то можно использовать только вложенные кавычки. Не имеет значения, в каком порядке Вы использовали кавычки - сначала двойные, а затем одинарные или наоборот.

МОЖНО:

onmouseover="document.bgColor ='red' " или onmouseover='document.bgColor ="red" '

Но НЕЛЬЗЯ:

onmouseover="document.bgColor ="red" " onmouseover='document.bgColor ='red' ' onmouseover="document.bgColor ='red" '

Если значение HTML-атрибута обработчика события состоит из нескольких JavaScript-инструкций, они должны отделяться точками с запятой либо значение атрибута должно располагаться в нескольких строках.

**Задание №3**
Измените скрипт, созданный ранее так, чтобы - фон документа при наведении курсором на какой-то текст менял цвет, а при уходе с него курсора возвращался к первоначальному.

А если мы хотим изменить не свойство всего документа, а только свойство какого-то абзаца? Как в данном случае мы можем изменить свойство данного объекта? Есть несколько способов.

**ПРИМЕР**

Этот абзац меняет цвет при наведении на него мышкой с синего на красный!

ВАЖНО! Код должен быть записан в одну строчку

Разберем код.

1. style="color:blue" определяется стиль текста в данном абзаце
2. onmouseover= событие которое может произойти с этим абзацем, в кавычках надо указать что при этом делать
3. this.style.color='red' изменить стиль абзаца: цвет текста на красный:
   - слово this используется для доступа к элементу (объекту), вызвавшему событие (к данному абзацу),
   - через точку указывается его свойство style, дающее доступ к стилям,
   - ещё через точку указывается **конкретное свойство**, значение которого мы хотим изменить (color - цвет текста)
   - затем идет знак присвоить =,
   - и затем значение свойства "цвет текста" - красный ('red').

**Рассмотрим еще один пример. Изменение цвета фона текста**:

Цвет фона текста меняется на красный при наведении мышкой на него!

Разберем код.

1. style="background-color:blue" определяется стиль текста в данном абзаце (цвет фона)
2. onmouseover= (навести мышь) и onmouseout= (увести мышь) события которые могут произойти с этим абзацем, в кавычках надо указать что при этом делать
3. this.style.backgroundColor='red" изменить стиль абзаца: цвет фона на красный.

**Задание №4**

1) Измените созданный ранее файл следующим образом.
2) Создайте три абзаца
3) Напишите скрипт, изменяющий цвет текста первого абзаца при наведении мышкой
4) Напишите скрипт, изменяющий цвет фона текста второго абзаца при наведении мышкой
5) Напишите скрипт, изменяющий цвет фона текста третьего абзаца только при наведении мышкой

**Метод *alert***

Метод window.alert отображает диалоговое окно, в которое помещается сообщение для пользователя. Этот метод часто используется при обработке полей формы. Если в какое-либо из полей формы введено неверное значение, то пользователю посылается сообщение, которое изображено на рис. 1.

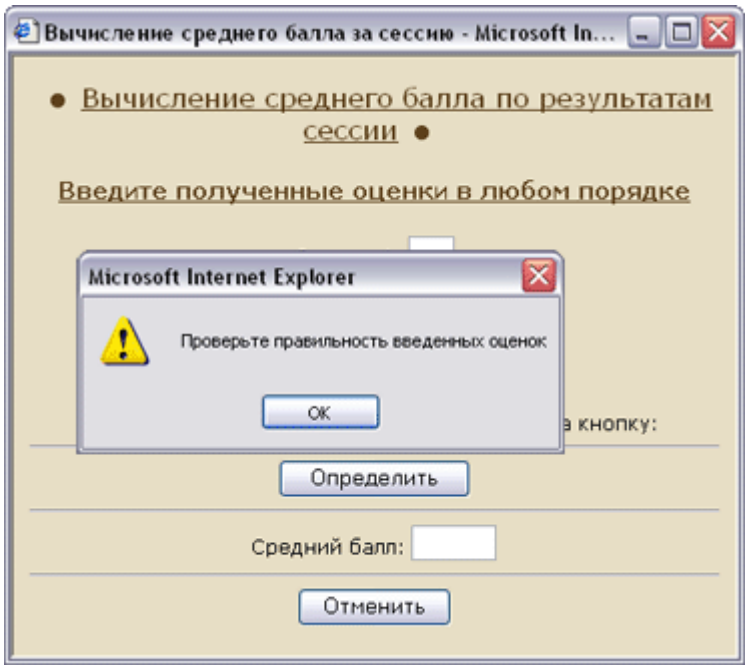



Рис. 1. Метод alert при анализе введенных данных

Документ, HTML- код которого представлен в листинге 1, содержит сценарий, при работе которого проверяется правильность введенных данных.

**Листинг 1. Метод alert для проверки введенных данных**

```
        <HTML>
< HEAD>
< TITLE>Вычисление среднего балла за сессию</TITLE>
< SCRIPT LANGUAGE=”JavaScript">
< !-- //
function msgot()
{ window.status = "неверно введены данные";
  alert ('Проверьте правильность введенных оценок')
  window.status = ""}
function st (obj )
{ var a = Number(obj.num1.value);
  var b = Number(obj.num2.value);
  var c = Number(obj.num3.value);
  var d = Number(obj.num4.value);
  if ((a < 2 ) || ( a > 5) || (b < 2 ) || ( b > 5)||
     (c < 2 ) || ( c > 5) || (d < 2 ) || ( d > 5))
       {msgot()}
  else
    obj.ball.value=(a+b+c+d)/4
}
//-->
< /SCRIPT>
< /HEAD>
<BODY>
< h4>Вычисление среднего балла по результатам сессии</h4>
< h4>Введите полученные оценки в любом порядке</h4>
< FORM name="form1">
```

```
Оценка 1: <INPUT type="text" size=1 name="num1"><BR>
Оценка 2: <INPUT type="text" size=1 name="num2"><BR>
Оценка 3: <INPUT type="text" size=1 name="num3"><BR>
Оценка 4: <INPUT type="text" size=1 name="num4"><BR>
Для определения среднего балла нажмите на кнопку:<HR>
< INPUT type="button" value=Определить onClick="st(form1)"><HR>
Средний балл: <INPUT type="text" size=5 name="ball"><HR>
<INPUT type="reset" value=Отменить>
< /FORM></BODY></HTML>
```

**Задание №5**

1) Отладьте страницу вычисления среднего балла.
2) Измените скрипт таким образом, чтобы при среднем балле менее 3 выдавалось сообщение: «Стыдно, вы двоечник!»

**Метод *confirm***

Метод confirm отображает диалоговое окно подтверждения выполнения операции. Оно содержит две кнопки **OK** и **Cancel (Отмена)**, позволяющие выбрать один из вариантов. Если пользователь щелкнул по кнопке **OK**, то возвращается значение true, после щелчка по кнопке **Отмена** возвращается значение false. В функции quest() листинга 2 анализируется возвращаемое методом confim значение и выполняются соответствующие действия.

**Листинг 2. Метод confirm объекта window**

```
        <HTML>
< HEAD>
< TITLE>Отгадка задуманного числа</TITLE>
< SCRIPT LANGUAGE=”JavaScript">
< !-- //
function test (obj)
{ var i=Number(obj.num1.value)
  var j=Number(obj.num2.value)
  var k
  var s=" "
  var p
  while ( i < j )
  { if ((i+j)%2== 0)
      k=(i+j)/2
    else
      k=(i+j-1)/2
    s="Задуманное число меньше или равно "+k+"?"
    p=confirm(s)
    if (p)
      j=k
    else
      i=k+1
  }
  obj.res.value=i
}
```

```
function cont (obj)
{ if (obj.res.value != obj.num.value )
   alert ("Кто-то обманывает")
  else
   alert ("Все хорошо!")
}
//-->
< /SCRIPT>
< /HEAD>
< BODY>
< h4>Задумайте число в интервале</h4>
< FORM name="form1">
<input type="text" name="num1" size=4> и
<input type="text" name="num2" size=4><br>
Для проверки введите задуманное число
<input type="password" name="num" size=4><br>
< HR>
Ответьте на вопросы после нажатия кнопки <b>Отгадай</b><br>
<input type="button" value="Отгадай" onclick="test(form1)"><br>
Задуманное число: <input type="text" name="res" size=10><br>
< input type="button" value="Проверка" onclick="cont(form1)">
<input type="reset" value="Очистить">
< /FORM></BODY></HTML>
```

На рис. 2 документ, HTML-код которого хранится в листинге 2.

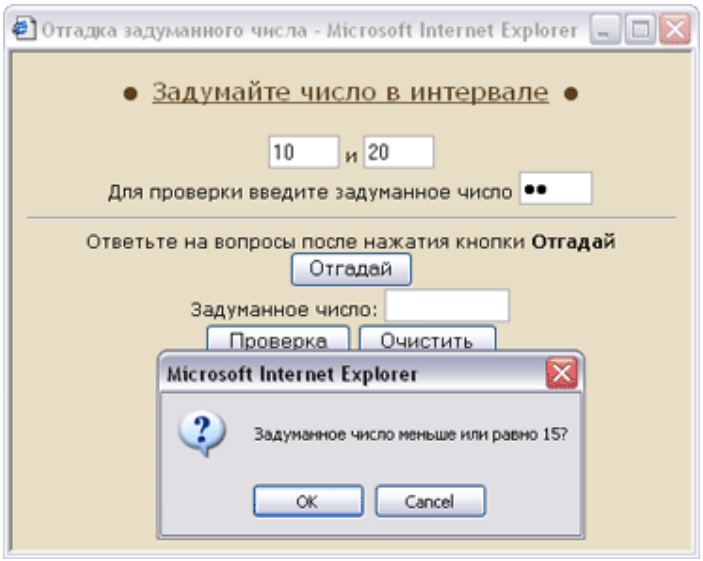



Рис. 2. Диалог с пользователем с помощью метода confim.

**Задание №6**

1) Отладьте страницу из листинга 2

**Метод *prompt***

Метод promt используется для вывода диалогового окна запроса данных. При щелчке по кнопке **OK** введенные пользователем в текстовое поле данные отображаются в документе. Если выбрано значение Cancel, то возвращается значение Null. Метод promt имеет второй параметр, с помощью которого задается значение по умолчанию.

**Листинг 3. Диалоговое окно запроса данных**

```
<HTML>
< HEAD>
< TITLE>Метод prompt объекта window</TITLE>
< SCRIPT>
function reg (obj)
{ var s= window.prompt ("Введите Ваше имя", "")
  obj.regname.value = s
}
< /SCRIPT>
< /HEAD>
< BODY>
< form name="form1">
< input type=text name=regname>
< input type="button" value="Регистрация" onClick= "reg(form1)">
< /form></BODY></HTML>
```

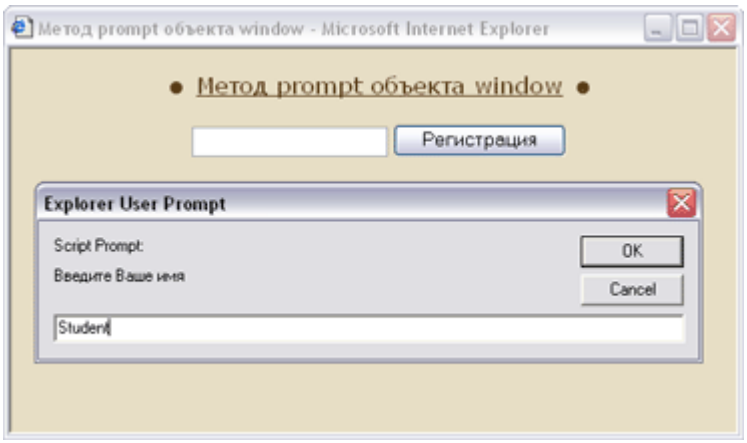



Рис. 3. Диалоговое окно запроса данных

**Задание №7**

1) Отладьте страницу из листинга 3

**Задание для самостоятельной работы**

Создайте интерактивную страницу с формой и диалогом, производящую расчёты для определения стоимости или физического объёма ваших товаров(услуг) и подключите её по ссылкам к созданным ранее страницам.

## Контрольные вопросы

1. Где могут быть размещены выражения *JavaScript ?*
2. Перечислите основные узлы дерева HTML документа.
3. Как применять программный интерфейс HTML DOM?
4. Как изменить свойство узлов?
5. Как применять регулярные выражения в *JavaScript ?*
6. В чем сущность объектной модели браузера?
7. Для чего применяется библиотека *jQuery ?*
8. Что такое DOM (Document Object Model)?
9. Что такое куки?
10. Как получить доступ к информации о текущей сессии?
11. Как записать информацию в куки?
12. Что такое Ajax – приложение?

# Лабораторная работа № 6 Программное взаимодействие с HTML документами на основе DOM API

**Цель работы:**

а) ознакомление с каскадными таблицами стилей (CSS);
б) ознакомление с базовым синтаксисом, основными элементами CSS - документа;
в) приобретение навыков создания HTML – документов с использованием CSS.

**Теоретические основы**

Расшифровывается CSS (англ. *Cascading Style Sheets)* как каскадные таблицы стилей и является технологией оформления веб-страниц.

Основным понятием CSS является стиль – т. е. набор правил оформления и форматирования, который может быть применен к различным элементам документа. В стандартном HTML для присвоения какому-либо элементу определенных свойств (таких, как цвет, размер, положение на странице и т. п.) приходилось каждый раз описывать эти свойства, увеличивая размер файла и время загрузки на компьютер просматривающего ее пользователя.

CSS действует более удобным и экономичным способом. Для присвоения какому-либо элементу определенных характеристик необходимо один раз описать этот элемент и определить это описание как стиль, а в дальнейшем просто указывать, что элемент, который нужно оформить соответствующим образом, должен принять свойства указанного стиля.

Более того, можно сохранить описание стиля не в тексте кода документа, а в отдельном файле – это позволит использовать описание стиля на любом количестве Web-страниц, а также изменить оформление любого количества страниц, исправив лишь описание стиля в одном (отдельном) файле.

Кроме того, CSS позволяет работать со шрифтовым оформлением страниц на гораздо более высоком уровне, чем стандартный HTML, избегая излишнего утяжеления страниц графикой.

**Задание на занятие:**

**Отладьте страницы со всеми приведёнными примерами**

```
<html>
<head>
<style type="text/css">
        .newfont{font-size:24px; color:#CC9933}
</style>
<title>Классы для создания тэгов.</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<blockquote class=" newfont ">Заголовок</blockquote>
</body>
</html>
```

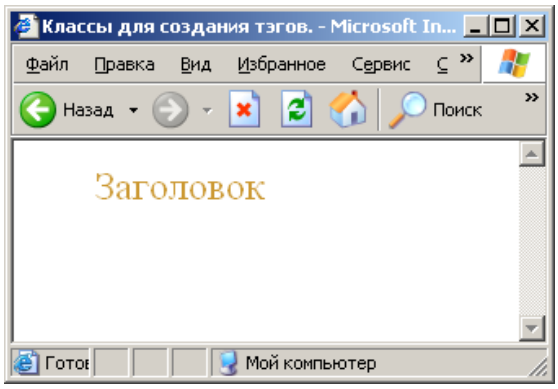



Рисунок 1

Данный пример иллюстрирует вариант объявления нового стиля в документе и потом его использования.

*Синтаксис и элементы CSS*

*1 Добавление стилей CSS в HTML-документ*

Существует несколько способов связывания документа и таблицы стилей:

- Связывание - позволяет использовать одну таблицу стилей для форматирования многих страниц HTML
- Внедрение - позволяет задавать все правила таблицы стилей непосредственно в самом документе
- Встраивание в теги документа - позволяет изменять форматирование конкретных элементов страницы
- Импортирование - позволяет встраивать в документ таблицу стилей, расположенную на сервере

Остановимся на каждом из этих способов более подробно.

*Связывание.* Напомним, что информация о стилях может располагаться либо в отдельном файле, либо непосредственно в коде документа. Расположение описания стилей в отдельном файле целесообразно при применении стилей при количестве страниц более 1. Для этого необходимо создать текстовый файл, описать необходимые стили и в коде документов, которые будут использовать эти стили необходимо создать ссылку на данный файл. Отметим, что данный файл может располагаться где угодно, необходимым условием является только то, чтобы броузер клиента мог его загрузить на свою сторону. Осуществляется это с помощью тега LINK, располагающегося внутри тега HEAD документов:

<LINK REL=STYLESHEET TYPE="text/css" HREF="URL">

Первые два параметра этого тега являются зарезервированными именами, требующимися для того, чтобы сообщить броузеру, что на этой страничке будет использоваться CSS. Третий параметр – HREF= «URL» – указывает на файл, который содержит описания стилей. Этот параметр должен содержать либо относительный путь к файлу – в случае, если он находится на том же сервере, что и документ, из которого к нему обращаются – или полный URL («http://...») в случае, если файл стилей находится на другом сервере.

```
<head>
<title></title>
<meta http-equiv="content-type" content="text/html; charset=windows-1251">
<link rel="stylesheet" href="css/default.css">
</head>
```

*Внедрение.*Второй вариант, при котором описание стилей располагается в коде Web-странички, внутри тега HEAD, в теге <STYLE type="text/css">... </STYLE. В этом случае вы можете использовать эти стили для элементов, располагающихся в пределах странички. Параметр type="text/css" является обязательным и служит для указания броузеру использовать CSS.

```
<head>
<style type="text/css" >
        .el_cl_1{display:inline; z-index:1};
        .el_lst{display:list-item; margin:-1%; background:#ff0000 url("bc.jpg") no-repeat};
</style>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
```

*Встраивание в теги документа.*Данный, третий по счету, метод позволяет располагать описания стилей непосредственно внутри тега элемента, который описывает. Это осуществляется при помощи параметра STYLE, используемого при применении CSS с большинством стандартных тегов HTML. Данный метод нежелателен, т.к. приводит к потере одного из основных преимуществ CSS – возможности разделения информации и описания оформления информации.

```
<blockquote style="color:#CCFF66">Внимание!</blockquote>
```

*Импортировании.* В теге <STYLE> можно *импортировать* внешнюю таблицу стилей с помощью свойства @import таблицы стилей:

```
@import: url(styles.css);
```

Его следует задавать в начале стилевого блока или связываемой таблицы стилей перед заданием остальных правил. Значение свойства @import является URL файла таблицы стилей.

Заметим, что импортирование от связывания отличается тем, что при импортировании можно не только поместить внешнюю таблицу стилей в документ, но и поместить одну внешнюю таблицу стилей в другую.

```
<head>
<title>Untitled </title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<style type="text/css">
@import url('css/default.css');
</style>
</head>
```

Приведем пример использования свойства текста (рисунке 2.1)

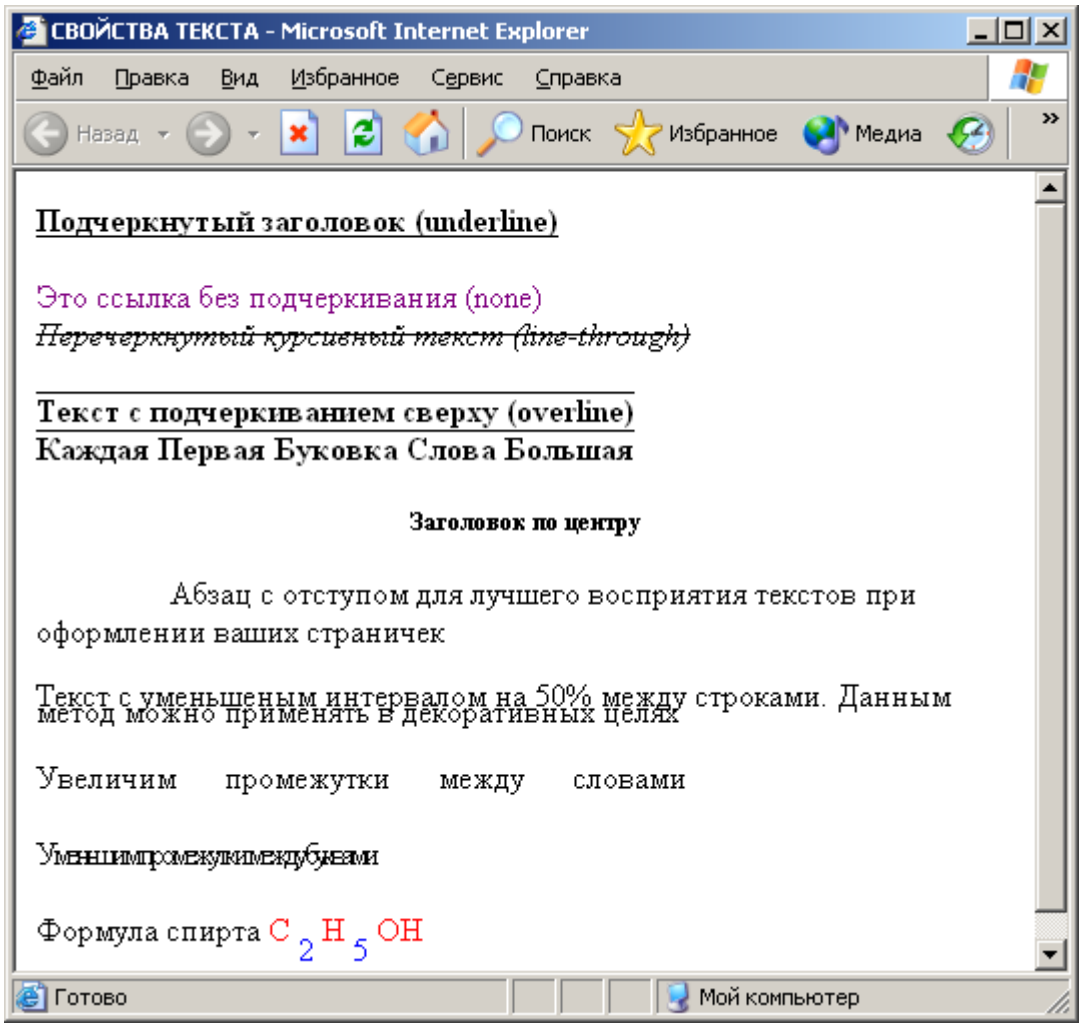



Рисунок 2.1
Ниже приведен код примера.

```
<STYLE type="text/css">
H4 {text-decoration: underline;}
A {text-decoration: none;}
i {text-decoration:line-through;}
b {text-decoration:overline;}
H5 {text-align: center}
b.cap {text-transform:capitalize;}
.otstup {text-indent: 50pt;}
.interval {line-height: 50 %}
</STYLE>
<h4>Подчеркнутый заголовок (underline)</h4>
<a href="/css/003/text.htm">Это ссылка без подчеркивания (none)</a><br>
<i>Перечеркнутый курсивный текст (line-through)</i><p>
<b>Текст с подчеркиванием сверху (overline)</b><br>
<b class=cap>каждая первая буковка слова большая</b>
<h5>Заголовок по центру</h5>
<p class=otstup>Абзац с отступом для лучшего восприятия текстов
при оформлении ваших страничек</p>
<p class=interval>Текст с уменьшеным интервалом на 50% между строками.
```

```
Данным метод можно применять в декоративных целях</p>
<p><span style="word-spacing: 15pt">Увеличим промежутки между
словами</span>
<p><span style="letter-spacing: -2pt">Уменьшим промежутки между
буквами</span>
<p>Формула спирта
<span style=color:red>C</span>
<span style= vertical-align:sub;color:blue;>2</span>
<span style=color:red>H</span>
<span style="color:blue; vertical-align:sub;">5</span>
<span style=color:red>OH</span>
```

**Задание на самостоятельную работу:**

1.Скачайте архив простого сайта «Пример сайта ПЗ 3», разберите его устройство и оформите его при помощи CSS в другом, приятном для вас стиле для выбранной Вами теме. Все стили необходимо вынести в отдельный файл.

2. Наполните полученную заготовку сайта страницами и решениями, отлаженными в предыдущих работах.

3. Добавьте страницу с интерактивной формой – опросом по вашей теме.

## Контрольные вопросы

1. Что такое DOM (Document Object Model)?
2. В чем сущность объектной модели браузера?
3. Как применять регулярные выражения в *JavaScript ?*
4. Как включить инструменты разработчика в браузерах?
5. Где могут быть размещены выражения *JavaScript ?*
6. Перечислите основные узлы дерева HTML документа.
7. Как применять программный интерфейс HTML DOM?
8. Как изменить свойство узлов?
9. Как применять регулярные выражения в *JavaScript ?*
10. В чем сущность объектной модели браузера?
11. Для чего применяется библиотека *jQuery ?*
12. Что такое профилирование сценариев?

# Лабораторная работа № 7 Динамическое создание форм

**1. Общие сведения о формах**

Некоторые WWW browser позволяют пользователю, заполнив специальную форму, возвращающую полученное значение, выполнять некоторые действия на вашем WWW-сервере. Когда форма интерпретируется WEB-броузером, создается специальные экранные элементы, такие, как поля ввода, checkboxes, radiobuttons, выпадающие меню, скроллируемые списки, кнопки и т.д. Когда пользователь заполняет форму и нажимает кнопку "Подтверждение" (SUBMIT - специальный тип кнопки, который задается при описании документа), информация, введенна пользователем в форму, посылается HTTP-серверу для обработки и передаче другим программам, работающим под сервером. Все формы начинаются тэгом <FORM> и звершаются тэгом </FORM>.

**Синтаксис:** <FORM METHOD="get | post" ACTION="URL">
Элементы формы и другие элементы HTML
</FORM>

**Атрибуты**:

| | |
|---|---|
| METHOD | Метод посылки сообщения с данными из формы. В зависимости от используемого метода вы можете посылать результаты ввода данных в форму двумя путями:<br>GET: Информация из формы добавляется в конец URL, который был указан в описании заголовка формы.<br>POST: Данный метод передает всю информацию о форме немедленно после обращения к указанному URL в теле запроса. |
| ACTION | ACTION описывает URL, который будет вызываться для обработки формы. Данный URL почти всегда указывает на программу, обрабатывающую данную форму. |

**2. Тэги Формы**

**2.1. TEXTAREA**

Тэг <TEXTAREA> используется для того, чтобы позволить пользователю вводить более одной строки информации (свободный текст). Если вы хотите, чтобы в поле ввода по умолчанию выдавался какой-либо текст, то необходимо вставить его внутри тэгов <TEXTAREA> и </TEXTAREA>.

**Синтаксис:** <TEXTAREA NAME="" ROWS= COLS= > </TEXTAREA>

**Атрибуты**:

| | |
|---|---|
| NAME | Имя поля ввода |
| ROWS | Высота поля ввода в символах |
| COLS | Ширина поля ввода в символах |

**Пример1: Example1.html**

```
<HTML>
<HEAD> <meta charset="utf-8"><TITLE>Пример 1</TITLE></HEAD>
<BODY>
    <FORM METHOD="post" ACTION="Example2.html">
      <TEXTAREA NAME="address" ROWS=10 COLS=50> Москва, Дмитровкое шоссе, д.9Б, офис
      </TEXTAREA>
```

```
    <INPUT TYPE= "SUBMIT"  VALUE=" Готово!"><br>
  </FORM>
</BODY></HTML>
```

Когда вы описываете форму, каждый элемент ввода данных имеет тэг <INPUT>. Когда пользователь помещает данные в элемент формы, инфоромация размещается в разделе VALUE данного элемента.

| | |
|---|---|
| TYPE | Определяет тип поля ввода. По умолчанию это простое поле ввода для одной строки текста. Остальные типы должны быть явно указаны: |
| TEXT | Данный тип поля ввода описывает однострочное поле ввода. Используйте атрибуты MAXLENGTH и SIZE для определения максимальной длинны вводимого значения в символах и размера отображаемого поля ввода на экране (по умолчанию принимается 20 символов). |
| PASSWORD | То же самое, что и атрибут TEXT, но вводимое пользователем значение не отображается броузером на экране. |
| RADIO | Данный атрибут позволяет вводить одно значение из нескольких альтернатив. Для создания набора альтернатив вам необходимо создать несколько полей ввода с атрибутом TYPE="RADIO" с разными значениями атрибута VALUE, но с одинаковыми значениями атрибута NAME. В CGI-программу будет передано значение типа NAME=VALUE, причем VALUE примет значение атрибута VALUE того поля ввода, которое в данный момент будет выбрано (будет активным). При выборе одного из полей ввода типа RADIO все остальные поля данного типа с тем же именем (атрибут NAME) автоматически станут невыбранными на экране. |
| CHECKBOX | Используется для простых логических (BOOLEAN) значений. Значение, ассоциированное с именем данного поля, которое будет передаваться в вызываемую программу, может принимать значение ON или OFF. |
| IMAGE | Данный тип поля ввода позволяет вам связывать графический рисунок с именем поля. При нажатии мышью на какую-либо часть рисунка будет немедленно вызвана ассоциированная форме программа. Значения, присвоенные переменной NAME будут выглядеть так - создается две новых переменных: первая имеет имя, обозначенное в поле NAME с добавлением .x в конце имени. В эту переменную будет помещена X-координата точки в пикселах ( считая началом координат левый верхний угол рисунка), на которую указывал курсор мыши в момент нажатия, а переменная с именем, содержащимся в NAME и добавленным .y, будет содержать Y-координату. Все значения атрибута VALUE игнорируются. Само описание картинки осуществляется через атрибут SRC и по синтаксису совпадает с тэгом <IMG>. |
| HIDDEN | Поля данного типа не отображаются броузером и не дают пользователю изменять присвоенные данному полю по умолчанию значение. Это поле используетс для передачи в программу статической информации, как то ID прользователя, пароля или другой информации. |
| SUBMIT | Данный тип обозначает кнопку, при нажатии которой будет вызвана CGI-программа (или URL), описанная в заголовке формы. Атрибут VALUE может содержать строку, которая будет высвечена на кнопке. |
| RESET | Данный тип обозначает кнопку, при нажатии которой все поля формы примут значения, описанные для них по умолчанию. |
| **NAME** | Имя поля ввода. Данное имя используется как уникальный идентификатор поля, по которому, впоследствии, вы сможете получить данные, помещенные |

| | |
|---|---|
| | пользователем в это поле. |
| **SIZE** | Определяет визуальный размер поля ввода на экране в символах. |
| **MAXLENGTH** | Определяет количество символов, которое пользователи могут ввести в поле ввода. При превышении количества допустимых символов броузер реагирует на попытку ввода нового символа звуковым сигналом и не дает его ввести. Не путать с атрибутом SIZE. Если MAXLENGTH больше чем SIZE, то в поле осуществляется скроллинг. По умолчанию значение MAXLENGTH не ограничено. |
| **CHECKED** | Означает, что CHECKBOX или RADIOBUTTON будет выбран. |
| **SRC** | URL,. указывающий на картинку (используется совместно с атрибутом IMAGE). |
| **VALUE** | Присваивает полю значение по умолчанию или значение, которое будет выбрано при использовании типа RADIO (для типа RADIO данный атрибут обязателен) |

**2.2. INPUT**

Тэг <INPUT> используется для ввода одной строки текста или одного слова.

**Атрибуты:**

**Пример 2: Example2.html**

```
<HTML>
<HEAD>
<meta charset="utf-8">
<TITLE>Пример 2</TITLE></HEAD>
<BODY>
    < FORM METHOD="post" ACTION="Example1.html">
       <INPUT TYPE= "TEXT" NAME="N1" SIZE="20" MAXLENGTH=30" VALUE=" "><br>
       <INPUT TYPE= "RADIO" NAME="N2"  VALUE=" " CHECKED> 1<br>
       <INPUT TYPE= "RADIO" NAME="N2"  VALUE=" "> 2<br>
       <INPUT TYPE= "SUBMIT"  VALUE=" Submit"><br>
     </FOFM>
<p>
 <form action="Example2.html">
  <p><b>Как по вашему мнению расшифровывается аббревиатура &amp;quot;ОС&amp;quot;?</b></p>
  <p><input type="radio" name="answer" value="a1">Офицерский состав<Br>
  <input type="radio" name="answer" value="a2">Операционная система<Br>
  <input type="radio" name="answer" value="a3">Большой полосатый мух</p>
  <p><input type="submit"></p>
 </form>
</p>
 </body>
</html>
```

**2.3. Меню выбора в формах**

Под меню выбора в формах понимают такой элемент интерфейса, как LISTBOX. Существует три типа тэгов меню выбора для форм:

**2.3.1. SELECT**

Тэг SELECT позволяет пользователю выбрать значение из фиксированного списка значений. Обычно это представлено выпадающим меню. Тэг SELECT имеет один или более параметр пежду стартовым тэгом <SELECT> и завершающим </SELECT>. По умолчанию, первый элемент отображается в строке выбора.

```
<FORM>
<SELECT NAME= >
        <OPTION >
        <OPTION>
</SELECT>
</FORM>
```

**2.3.2. SELECT SINGLE**

Тэг SELECT SINGLE - это то же самое, что и SELECT, но на экране пользователь видит одновременно несколько элементов выбора (три по умолчанию). Если их больше, то предоставляется автоматический вертикальный скроллинг. Количество одновременно отображаемых элементов определяется атрибутом SIZE.

```
<FORM>
<SELECT SINGLE NAME= SIZE= >
        <OPTION>
        <OPTION>
        <OPTIONS>
</SELECT>
</FORM>
```

**2.3.3. SELECT MULTIPLE**

Тэг SELECT MULTIPLE похож на тэг SELECT SINGLE, но пользователь может одновременно выбрать более чем один элемент списка. Атрибут SIZE определяет количество одновременно видимых на экране элементов, атрибут MULTIPLE - максимальное количество одновременно выбранных элементов. Если выбрано одновременно несколько значений, то серверу передаютс соответствующее выбранному количество параметров NAME=VALUE с одинаковыми значениями NAME, но разными VALUE.

```
<FORM>
<SELECT MULTIPLE NAME= SIZE= MULTIPLE= >
        <OPTION>
        <OPTION>
        <OPTIONS>
</SELECT>
</FORM>
```

**Пример 3: Example3.html**

```
<HTML>
<HEAD><TITLE>Пример 3</TITLE></HEAD>
<BODY>
    <FORM METHOD="post" ACTION="Example3.html">
        < b>Оставьте отзыв о посещенном сайте</b><p>
        <b> Фамилия: </b><br>
        <INPUT TYPE=”TEXT” NAME=”name” SIZE=”15” MAXLENGTH=”25” VALUE=” ”><br>
        <b> Имя: </b><br>
        <INPUT TYPE=”TEXT” NAME=”name” SIZE=”10” MAXLENGTH=”15” VALUE=” ”><br>
        <b> Отчество:</b><br>
```

```
<INPUT TYRE=”TEXT” NAME=”name” SIZE=”15” MAXLENGTH=”25” VALUE=” ”><br>
<b> Как часто Вы посещаете наш сайт:</b><br>
< SELECT NAME= SIZE=>
    <OPTION VALUE=” ”> Частота посещения сайта
    <OPTION VALUE=”0 ”>Несколько раз в месяц
    <OPTION VALUE=”1 ”> Несколько раз в неделю
    <OPTION VALUE=”2 ”> Каждый день
    <OPTION VALUE=”3 ”> Это мой первый визит
    <OPTION VALUE=”4 ”> Не посещаю
</SELECT><br>
<b> Ваши замечания:</b><br>
<TEXTAREA NAME="Comments" ROWS=3 COLS=40></TEXTAREA><br>
<INPUT TYPE= "SUBMIT"  VALUE=" Submit"><br>
</FORM></BODY></HTML>
```

**3. ЗАДАНИЕ**

1. Отладьте примеры 1 и 2 (там есть синтаксические ошибки). Убедитесь, что они могут работать совместно.
2. Отладьте пример 3.
3. Продолжите, расширьте анкету (пример 3). Используйте при создании формы все возможные атрибуты формы, максимальное число элементов.
4. Создайте страницу с формой «виртуальной торговой тележки».

**Счетчики, детекторы банкнот**

| Наименование | Цена | Фирма | Заказать |
|---|---|---|---|
| Детектор банкнот | от 500 у.е. | Дон-Электроникс | ☑ |
| Детектор банкнот "Евроскан" (проверка долларов, евро) | от 400 у.е. | Оргтехносервис ООО | ☐ |
| Детектор валют | от 500 у.е. | Спектр ООО | ☐ |

Оформить заказ

При нажатии кнопки «Оформить заказ» проверьте, выбран ли хотя бы один из предложенных заказов.

Если ни один из заказов не выбран, выдайте предупреждение : « Не выбран ни один заказ!».

Если хотя бы один заказ выбран, перейдите на страницу регистрации пользователя.

Создайте страницу содержащую сообщение «Необходима регистрация!» и форму для ввода полей “UserName” и “Password”, содержащие две кнопки: “OK”, “Cancel”.

При нажатии кнопки “Cancel” перейдите на страницу с выводом сообщения «Заказы оформляются только для зарегистрированных пользователей!».

При нажатии кнопки “OK” проверьте, введены ли поля “UserName” и “Password”, если нет – выдайте предупреждение, если да - перейдите на страницу с выводом сообщения «Ваш заказ оформлен».

## Контрольные вопросы

1. Какие версии PHP вы знаете?
2. Назовите основные особенности синтаксиса PHP.
3. Назовите основные конструкции PHP.
4. Как задаются переменные, массивы, строки PHP?
5. Как осуществляется процедурно-ориентированное программирование в PHP.
6. Для чего служат библиотеки классов PHP?
7. В чем сущность технологии AJAX в PHP?
8. Приведите примеры реализации серверных сценариев на PHP.
9. Как динамически создать web страницы с использованием PHP сценария?
10. На основе каких интерфейсов может взаимодействовать веб-сервер и веб-приложение?
11. Чем CGI отличается от ISAPI?
12. Как использовать файлы при программировании на PHP?

# Лабораторная работа №8 Заполнение БД информацией

Что такое MySQL.

MySQL – компактный многопоточный сервер баз данных. MySQL характеризуется большой скоростью, устойчивостью и легкостью в использовании.

MySQL был разработан компанией TcX для внутренних нужд, которые заключались в быстрой обработке очень больших баз данных. Компания утверждает, что использует MySQL с 1996 года на сервере с более чем 40 БД, которые содержат 10,000 таблиц, из которых более 24.01.2004 чем 500 имеют более 7 миллионов строк.

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

На текущий момент MySQL все еще в стадии разработки, хотя версии 3.22 полностью работоспособны.

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

Возможности MySQL.

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

Краткий перечень возможностей MySQL.

Поддерживается неограниченное количество пользователей, одновременно работающих с базой данных.

Количество строк в таблицах может достигать 50 млн.

Быстрое выполнение команд. Возможно MySQL самый быстрый сервер из существующих.

Простая и эффективная система безопасности.

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

Поддержка вложенных запросов, типа SELECT * FROM table1 WHERE id IN (SELECT id FROM table2).

Не реализована поддержка транзакций. Взамен предлагается использовать LOCK/UNLOCK TABLE.

Нет поддержки триггеров и хранимых процедур.

По словам создателей именно эти пункты дали возможность достичь высокого быстродействия. Их реализация существенно снижает скорость сервера. Эти возможности не являются критичными при создании Web-приложений, что в сочетании с высоким быстродействием и малой ценой позволило серверу приобрести большую популярность

**Работа с MySQL (сохранение данных в базе данных).**

Для начала создаем базу данных и таблицу. Входим в MySQL, и выполняем команды:

>CREATE DATABASE products;

>CREATE TABLE clients (name VARCHAR(25), email VARCHAR(25), choise VARCHAR(8));

Для общения с MySQL из PHP понадобятся следующие функции.

int mysql_connect(string hostname, string username, string password);

Создать соединение с MySQL.

Параметры:

Hostname – имя хоста, на котором находится база данных.

Username – имя пользователя.

Password – пароль пользователя.

Функция возвращает параметр типа int, который больше 0, если соединение прошло успешно, и равен 0 в противном случае.

int mysql_select_db(string database_name, int link_identifier);

Выбрать базу данных для работы.

Параметры:

Database_name – имя базы данных.

link_identifier – ID соединения, которое получено в функции mysql_connect. (параметр необязательный, если он не указывается, то используется ID от последнего вызова mysql_connect)

Функция возвращает значение true или false

int mysql_query(string query, int link_identifier);

Функция выполняет запрос к базе данных.

Параметры:

Query – строка, содержащая запрос

link_identifier – см. предыдущую функцию.

Функция возвращает ID результата или 0, если произошла ошибка.

int mysql_close(int link_identifier);

Функция закрывает соединение с MySQL.

Параметры:

link_identifier – см. выше.

Функция возвращает значение true или false

1. **Создадим таблицу Customer**
2. 

Поля создаваемой таблицы:

id, FirstName, LastName, Login, Password. Поле id – автоинкрементный первичный ключ.

**Код программы:**

**create.php**

```
<?php
$hostname='DOC200';
$database='________';
@mysql_connect($hostname,'_________________','____________________');
@mysql_select_db($database);
$q="CREATE TABLE `Customer_номер студенческого`
 (`id` INT (3) UNSIGNED DEFAULT '0',
`FirstName` VARCHAR (128),
PRIMARY KEY(`id`)
) ";
$r=mysql_query($q);
@mysql_free_result($r);
mysql_close();
?>
```

**2. Выведем список имеющихся таблиц в БД**

**show_tables.php**

```
<?php
$hostname='DOC200';
$database='_______';
@mysql_connect($hostname,'_______','_____________');
@mysql_select_db($database);
$r = mysql_listtables ($database);
$i = 0;
while ($i < @mysql_num_rows ($r)) {
   $ names[$i] = mysql_tablename ($r, $i);
```

```
    print $names[$i] . "<BR>";
    $i++;
}
@mysql_free_result($r);
mysql_close();
?>
```

**3. Удалим таблицу Customer**

**delete_table.php**

```
<?php
$hostname='DOC200';
$database='_____________';
@mysql_connect($hostname,'_____________','_____________');
@mysql_select_db($database);
$q="DROP TABLE `id301`.`customer`";
$r=mysql_query($q);
mysql_free_result($r);
mysql_close();
?>
```

**Упражнение**. Создайте административный скрипт для интерактивного создания и удаления таблиц БД.

**Шаг 1.** Создайте меню(**admin.html**).

- Показать список таблиц в БД
- Удалить таблицу
- Создать таблицу

Расположите на форме кнопку «Далее >>». При нажатии кнопки перейдите на **admin.php**

**Шаг 2.** Создайте обработчик формы(**admin.php**).

Используйте конструкцию switch.

1. Если выбран пункт меню 1, выполняйте **show_tables.php**. В конце расположите форму с копкой «Назад», возвращающую администратора к основному меню (**admin.html**).

2. Если выбран пункт меню 2, выполняйте **delete_table.php.** Для пункта «Удалить таблицу» предлагайте список имен всех имеющихся таблиц в БД. В конце расположите форму с копкой «Назад», возвращающую администратора к основному меню (**admin.html**).

3. Если выбран пункт меню 3, выполняйте **create.php**.

Если и флаг создания полей (имя произвольное) не установлен, то предложите форму:

Выберите имя БД список

Количество полей в таблице поле ввода

Расположите на форме кнопку «Далее >>». При нажатии кнопки перейдите на **admin.php**, установив флаг выбора меню в «3» и флаг создания полей в «1».

В скрипте проверьте, если флаг создания полей установлен в «1», то предложите пользователю таблицу:

| Имя поля 1 | Тип поля 1 | Размер | По умолчанию | Первичный ключ | Уникальный столбец | Автоинкремент |
|---|---|---|---|---|---|---|
| поле ввода | список | поле ввода | поле ввода | флажок | флажок | флажок |

.
.
.

| Имя поля N | Тип Поля N | Размер | По умолчанию | Первичный ключ | Уникальный столбец | Автоинкремент |
|---|---|---|---|---|---|---|
| поле ввода | список | поле ввода | поле ввода | флажок | флажок | флажок |

Расположите на форме кнопку «Далее >>». При нажатии кнопки перейдите на **admin.php**, установив флаг выбора меню в «3» и флаг создания полей в «2».

В скрипте проверьте, если флаг создания полей установлен в «2», то сформируйте запрос на создание таблицы и выполните его. Выведите пользователю сообщение «Таблица создана».

Выведите форму с копкой «Назад», возвращающую администратора к основному меню (**admin.html**).

**Замечание**. Типы полей списка «Тип поля»: INT, BIGINT, SMALLINT, FLOAT, DOUBLE, DATE, DATETIME, CHAR, VARCHAR

## Контрольные вопросы

1. Основные возможности СУБД MySQL и поддерживаемые платформы.
2. Типы данных, с которыми работает СУБД MySQL.
3. Команды, используемые для подключения к СУБД в консольном режиме.
4. Правила работы в консольном режиме с СУБД MySQL.
5. Особенности работы с СУБД MySQL
6. Как реализуется взаимодействие MySQLиPHP?
7. Синтаксис оператора SELEST в MySQL.
8. Связывание таблиц базы данных при выборке данных из полей нескольких таблиц.
9. Опишите технологию обеспечения поддержки интерпретатором PHP СУБД MySQL.
10. Перечислите функции PHP, необходимые для работы с СУБД MySQL.
11. Объясните различие функций MySQL_fetch_array(), MySQL_fetch_rows(), MySQL_fetch_object().
12. Какие существуют способы редактирования содержания таблиц баз данных?

## Рекомендуемая литература

**1. Основная литература**

| Код | Авторы, составители | Заглавие | Издательство, год | Кол. |
|---|---|---|---|---|
| Л1.1 | Гуриков С. Р. | Интернет-технологии: Учебное пособие/ -, 500 экз | М.: Форум, НИЦ ИНФРА-М, 2015. - 184 с.: 70x100 1/16. - (Высшее образование: Бакалавриат) ISBN 978-5-00091-001- | Э1 |
| Л1.2 | Т.И. Немцова, Т.В. Казанкова, А.В. Шнякин / под ред. Л.Г. Гагариной | Компьютерная графика и web-дизайн : учеб. пособие /. —. + Доп. материалы | М. : ИД «ФОРУМ» : ИНФРА-М, 2019. — 400 с | Э2 |
| Л1.3 | Бенкен Е.С. | PHP, MySQL, XML: программирование для Интернета. - 3-е изд., перераб. и доп. - | СПб:БХВ-Петербург, 2011. - 304 с. ISBN 978-5-9775-0724-0 | Э3 |
| Л1.4 | Будилов В.А. | Интернет-программирование на Java: Пособие / - | СПб:БХВ-Петербург, 2014. - 698 с. ISBN 978-5-9775-1931- | Э4 |
| Л1.5 | Соколова В.В. | Разработка мобильных приложений: Учебное пособие / - | Томск:Изд-во Томского политех. университета, 2014. - 176 с.: ISBN 978-5-4387-0369-3 | Э8 |

**2 Дополнительная литература**

| Код | Авторы, составители | Заглавие | Издательство, год | Кол. |
|---|---|---|---|---|
| Л2.1 | Зиангирова Л.Ф. | Сетевые технологии учебно-методическое пособие | Саратов: Вузовское образование, 2019.— 100 с | Э5 |
| Л2.2 | Будилов В.А. | Основы программирования для Интернета: Пособие / - | СПб:БХВ-Петербург, 2014. - 733 с. ISBN 978-5-9775-1917- | Э6 |
| Л2.3 | Кисленко Н.П. | Интернет-программирование на PHP учебное пособие | Новосибирск: Новосибирский государственный архитектурно- | Э7 |

| | | | | |
|---|---|---|---|---|
| | | | строительный университет (Сибстрин), | |

| **3 Методическое обеспечение для самостоятельной работы обучающихся** | | | | |
|---|---|---|---|---|
| **Код** | **Авторы, составители** | **Заглавие** | **Издательство, год** | **Кол.** |
| Л4.1 | П.Б. Храмцов [и др.]. | Основы Web-технологий учебное пособие | Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2019.— 375 с.— | Э8 |
| Л4.2 | Семенов Ю.А. | Протоколы и алгоритмы маршрутизации в «Интернет» | М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 998 с.— | Э9 |
| Л4.3 | Дубаков А.А. | Сетевое программирование учебное пособие | СПб.: Университет ИТМО, 2014.— 249 с.— | Э10 |
| Л4.3 | Зиангирова Л.Ф. | Технологии облачных вычислений учебное пособие | Саратов: Вузовское образование, 2016.— 300 с. | Э11 |

| **Электронные образовательные ресурсы** | |
|---|---|
| Э1 | http://znanium.com/bookread2.php?book=488074 |
| Э2 | http://znanium.com/bookread2.php?book=894969 |
| Э3 | http://znanium.com/bookread2.php?book=350558 |
| Э4 | http://znanium.com/bookread2.php?book=940239 |
| Э5 | http://www.iprbookshop.ru/62065.html.— ЭБС «IPRbooks» |
| Э6 | http://znanium.com/bookread2.php?book=940218 |
| Э7 | http://www.iprbookshop.ru/68769.html.— ЭБС «IPRbooks» |
| Э8 | http://www.iprbookshop.ru/67384.html.— ЭБС «IPRbooks» |
| Э9 | http://www.iprbookshop.ru/62826.html.— ЭБС «IPRbooks |
| Э10 | http://www.iprbookshop.ru/68118.html.— ЭБС «IPRbooks» |
| Э11 | http://www.iprbookshop.ru/41948.html.— ЭБС «IPRbooks» |

| **Программное обеспечение** | |
|---|---|
| П.1 | MS Windows |
| П.2 | Система визуального программирования Lazarus |
| П.3 | Пакет программ для проведения тестирования по изученным темам |