

**HTML5**  
**JAVA SCRIPT**  
**JQUERY**  
**CSS3**

# **Введение в web-программирование**

**Инструктор: Максим**

1. Информация
2. Клиент-серверная архитектура
3. Web-программирование
4. Адресация
5. Доменная система имен
6. Порты и сервисы
7. Структура протокола HTTP



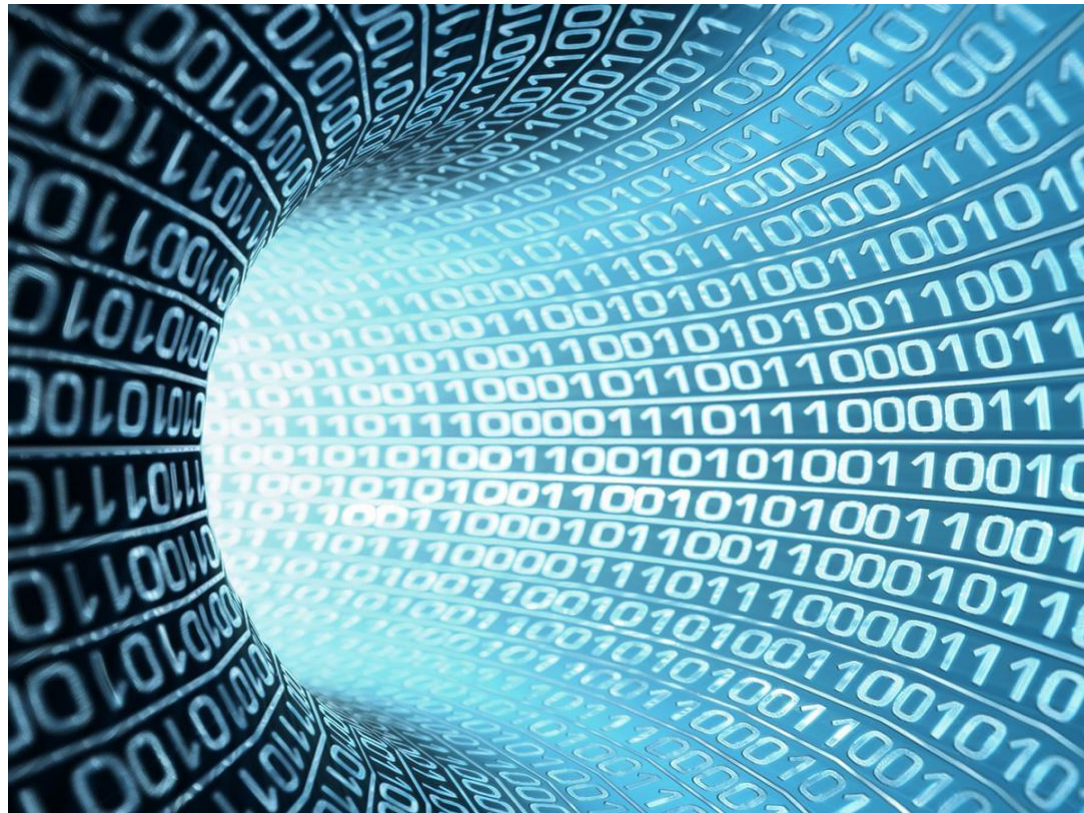
# 1. Информация

Web-программирование – это IT (Information Technology, информационные технологии)

Ключевое понятие – **Информация**

Информацию можно:

- **Хранить**
- **Обрабатывать**
- **Передавать**
- **Принимать**



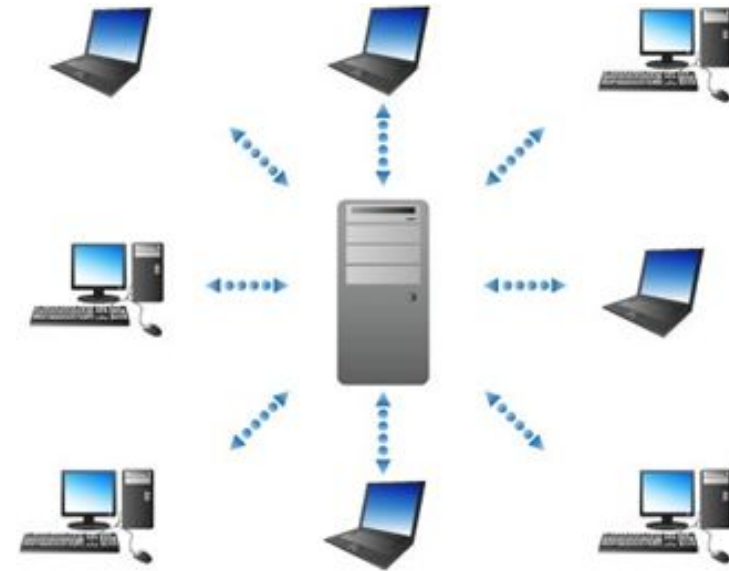
## 2. Клиент-серверная архитектура

**Клиент-серверная архитектура** – это концепция компьютерной сети, в которой основная часть ее ресурсов сосредоточена в серверах, обслуживающих своих клиентов. Рассматриваемая архитектура определяет два типа компонентов: **клиенты** и **серверы**

Клиент – это объект, который использует ресурсы сервера и предоставляет удобные интерфейсы пользователя. Интерфейсы пользователя это процедуры взаимодействия пользователя с системой. Клиент является инициатором и использует сервисы сервера

Сервер – это объект, предоставляющий сервис другим объектам сети по их запросам.

**Сервис** – это процесс обслуживания клиентов. Сервер отвечает на запросы клиентов и управляет их выполнением. После выполнения каждого запроса, сервер посылает полученные результаты клиенту



## 2. Клиент-серверная архитектура. Типы Серверов

**Web-сервер** – сервер, принимающий HTTP-запросы от клиентов, обычно Web-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потокком или другими данными

**Сервер приложений** – действует как набор компонентов, доступных разработчику программного обеспечения через API (Application Programming Interface, интерфейс прикладного программирования). Для Web-приложений эти компоненты обычно работают на той же машине, где запущен Web-сервер. Их основная работа – обеспечивать создание динамических страниц

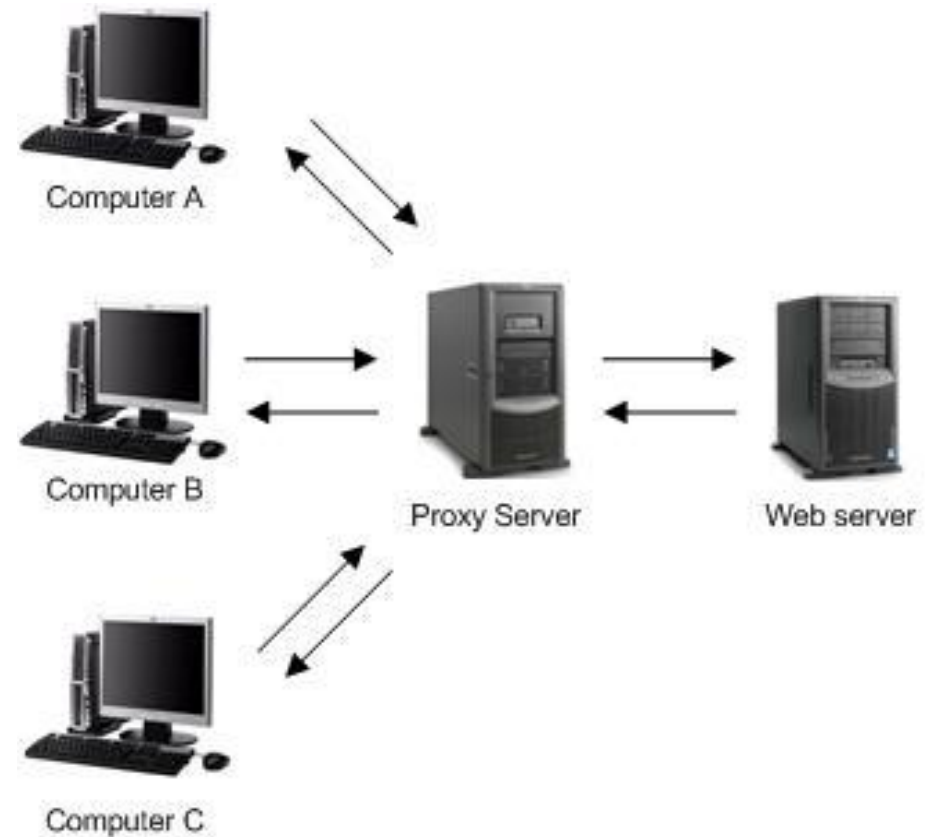
**Сервер базы данных** – выполняет обслуживание и управление базой данных и отвечает за целостность и сохранность данных, используется для обработки пользовательских запросов на языке SQL

## 2. Клиент-серверная архитектура. Типы Серверов

**Почтовый сервер** – представляет услуги по отправке и получению электронных почтовых сообщений

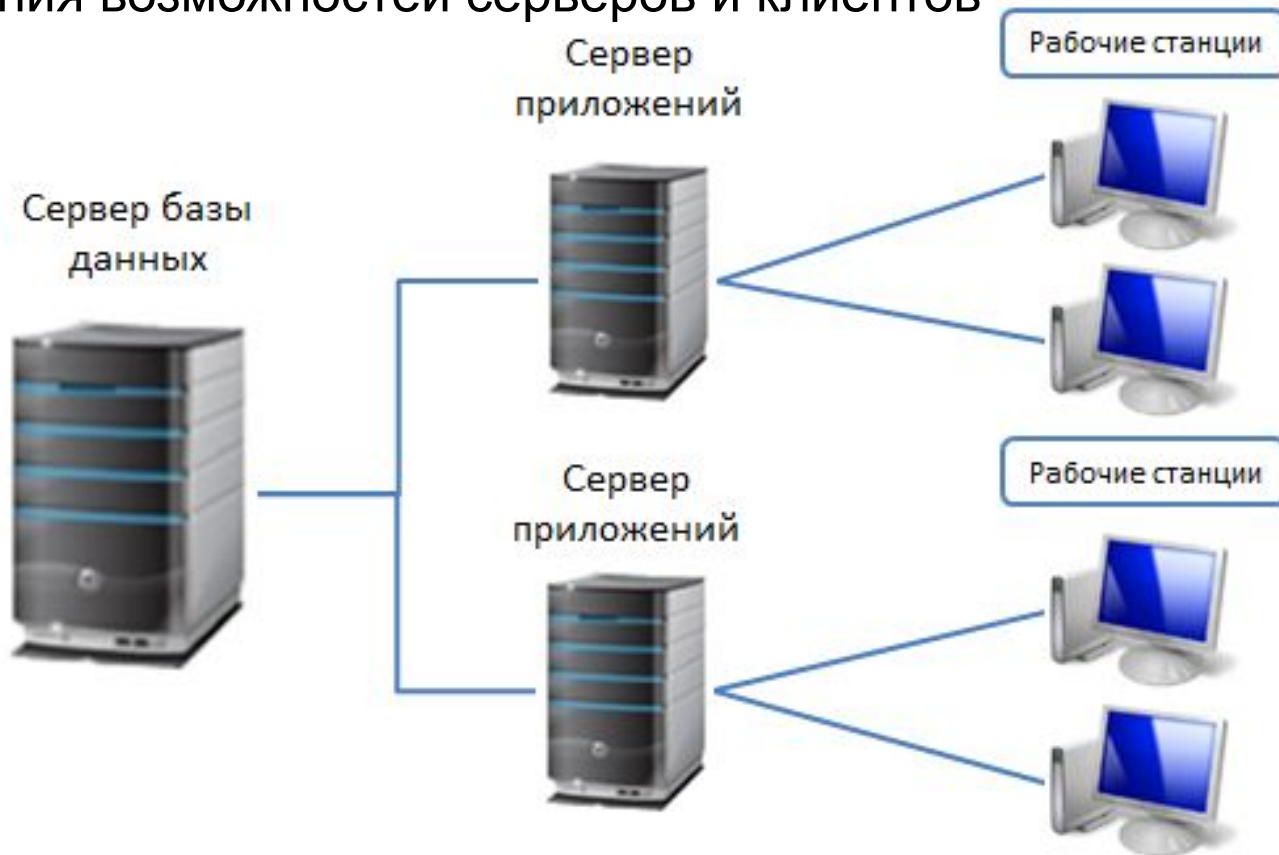
**Файл-сервер** – хранит информацию в виде файлов и представляет пользователям доступ к ней

**Прокси-сервер** – во-первых, действует как посредник, помогая пользователям получить информацию из Интернета и при этом обеспечивая защиту сети. Во-вторых, сохраняет часто запрашиваемую информацию в кэш-памяти на локальном диске, быстро доставляя ее пользователям без повторного обращения к Интернету



## 2. Клиент-серверная архитектура. Трёхуровневая архитектура

**Трёхуровневая архитектура** – разновидность архитектуры клиент-сервер, в которой функция обработки данных вынесена на один или несколько отдельных серверов. Это позволяет разделить функции хранения, обработки и представления данных для более эффективного использования возможностей серверов и клиентов



## 3. Web-программирование

**Web-программирование** – раздел программирования, ориентированный на разработку Web-приложений

**Web-приложение** – клиент-серверное приложение, в котором клиентом выступает **браузер**, а сервером – **Web-сервер**. Логика Web-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому Web-приложения являются межплатформенным

Web-приложения стали широко популярными в конце 1990-х – начале 2000-х годов



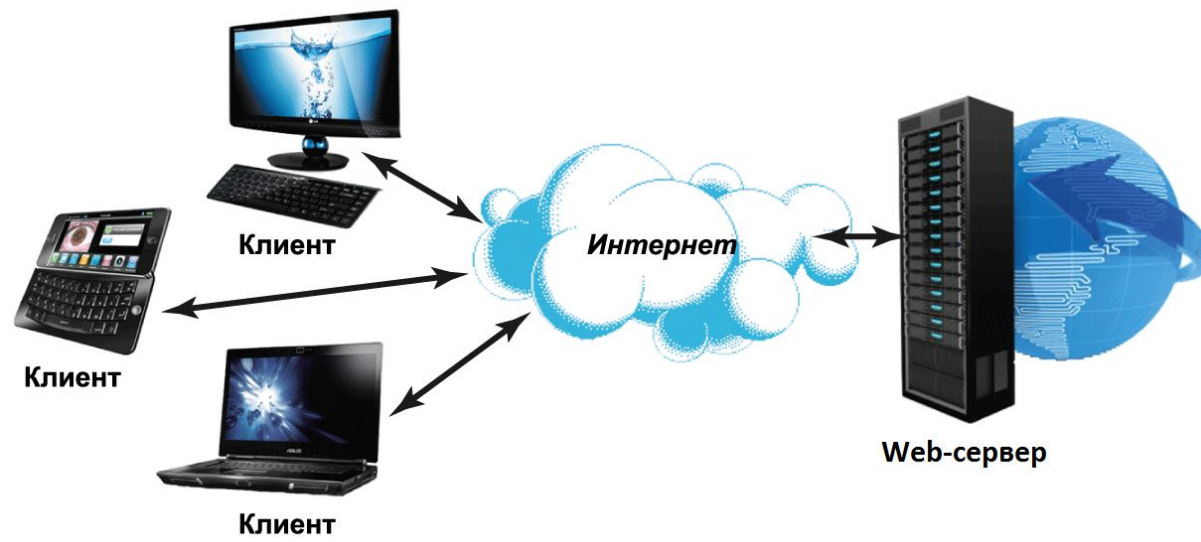


### 3. Web-программирование. Клиентская и серверная части

**Клиентская часть** реализует пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него

**Серверная часть** получает запрос от клиента, выполняет вычисления, после этого формирует Web-страницу и отправляет её клиенту по сети с использованием протокола HTTP

**Обратите внимание**, серверная часть может выступать в качестве клиента других сервисов, например, базы данных или другого Web-сервера, расположенного на другом сервере



### 3. Web-программирование. Языки и технологии

**Языки и технологии** Web-программирования можно условно разделить на две пересекающиеся группы: **клиентские** и **серверные**

Клиентские:

- **HTML** – стандартный язык разметки Web-страниц
- **CSS** – средство описания и оформления внешнего вида Web-страниц
- **JavaScript** – применяется в браузерах как язык сценариев для придания интерактивности Web-страницам

Серверные:

- **ASP.NET (C#)** – технология создания Web-приложений от компании Microsoft
- **Java EE (Java)** – технология создания Web-приложений от компании Oracle
- **PHP** – язык программирования, интенсивно применяемый для разработки Web-приложений
- **Node.js** – технология, основанная на JavaScript

## 4. Адресация. IP-адрес

Адресация в сети Интернет устроена таким образом, что каждой точке подключения к сети присваивается уникальный номер, который называется – **IP-адрес** (Internet Protocol, межсетевой протокол)

Необходимо подчеркнуть, что IP-адрес присваивается не устройству (компьютеру или маршрутизатору), а именно **интерфейсу**, поскольку многие устройства могут иметь несколько точек подключения к сети, а следовательно и несколько различных IP-адресов

Компьютеры и маршрутизаторы "знают" свои IP-адреса, и адреса своих "соседей в сети", а маршрутизаторы еще и могут определять с помощью таблиц маршрутизации, куда направлять пакеты со всеми прочими IP-адресами

Протокол – набор правил и действий (очередности действий), позволяющий осуществлять соединение и обмен данными между двумя и более включёнными в сеть устройствами

## 4. Адресация. IP-адрес

Для программно-аппаратных устройств IP-адрес это просто целое число для хранения которого выделяется ровно 4 байта (32 бита) памяти. Т.е. число в диапазоне от 0 до 4\_294\_967\_295. Человеку запоминать такие громоздкие числа сложно. Поэтому для наглядности, IP-адрес записывается в виде последовательности четырех чисел разделенных точками в диапазоне от 0.0.0.0 до 255.255.255.255. Каждое из этих четырех чисел соответствует значению отдельно каждого байта из тех четырех, в котором хранится все число

216.122.167.55 – стандартная запись IP-адреса

11011000 01111010 10100111 00110111 – двоичное представление

1\_819\_977\_527 – представление IP-адреса одним числом

## 4. Адресация. Маска подсети

С каждым IP-адресом связана 32-разрядная **маска подсети**, разбивающая адрес на две части – на уникальный **идентификатор сети**, к которой принадлежит компьютер, и на уникальный **идентификатор узла** в пределах этой сети

216.122.167.55 – IP-адрес

255.255.255.0 – Маска подсети

11011000 01111010 10100111 00110111 – IP-адрес

11111111 11111111 11111111 00000000 – Маска подсети

После побитового умножения получим:

11011000 01111010 10100111 (216.122.167) – адрес сети

00110111 (55) – адрес узла

Разбиение адреса на две части обеспечивает большую управляемость сети. Компании, имеющей парк из нескольких сотен машин, нет необходимости регистрировать адрес для каждой из них – достаточно зарегистрировать на себя отдельную подсеть и раздавать адреса внутри нее уже самостоятельно

## 4. Адресация. DHCP

**DHCP** (Dynamic Host Configuration Protocol, протокол динамической настройки узла) – сетевой протокол, позволяющий компьютерам **автоматически получать IP-адрес** и другие параметры, необходимые для работы в сети. Данный протокол работает по модели "клиент-сервер". Для автоматической конфигурации компьютер-клиент на этапе конфигурации сетевого устройства обращается к так называемому серверу DHCP и получает от него нужные параметры. Сетевой администратор может задать диапазон адресов, распределяемых сервером среди компьютеров. Это позволяет избежать ручной настройки компьютеров сети и уменьшает количество ошибок. Протокол DHCP используется в большинстве сетей

## 5. Доменная система имен. Домен

IP-адресация позволяет точно идентифицировать компьютеры, подключенные к сети. Однако, запоминать адреса вида 216.122.167.55 не слишком удобно. Поэтому с самого начала развития сети каждый узел помимо цифрового IP-адреса имел еще и символьное имя. Вначале все узлы были перечислены в одном текстовом файле, но по мере роста сети возникла необходимость в механизме, обеспечивающем, во-первых, уникальность имен, во-вторых, средства извещения всех подключенных узлов об изменениях

В настоящее время соответствие между цифровыми и символьными адресами обеспечивается серверами имен **DNS** (Domain Name Servers, система доменных имён). Под **доменом** понимается множество машин, которые администрируются и поддерживаются как единое целое. Вся сеть представляет собой одну большую иерархию доменов, позволяющую разграничить полномочия между администраторами разных сетей

## 5. Доменная система имен. Доменное имя

**Доменное имя** – это последовательность из двух и более доменов, разделенных точками. Последний домен в доменном имени называется "доменом первого уровня", второй от конца – "доменом второго уровня" и т.д.

Примеры доменов верхнего уровня:

- **.com** – предназначенный для коммерческих организация
- **.edu** – образовательные учреждения
- **.net** – сетевые провайдеры, узловые компьютеры
- **.org** – организации, не попадающие ни в одну из прочих категорий
- **.int** – международные организации
- **.fr** – Франция
- **.us** – США
- **.ua** – Украина



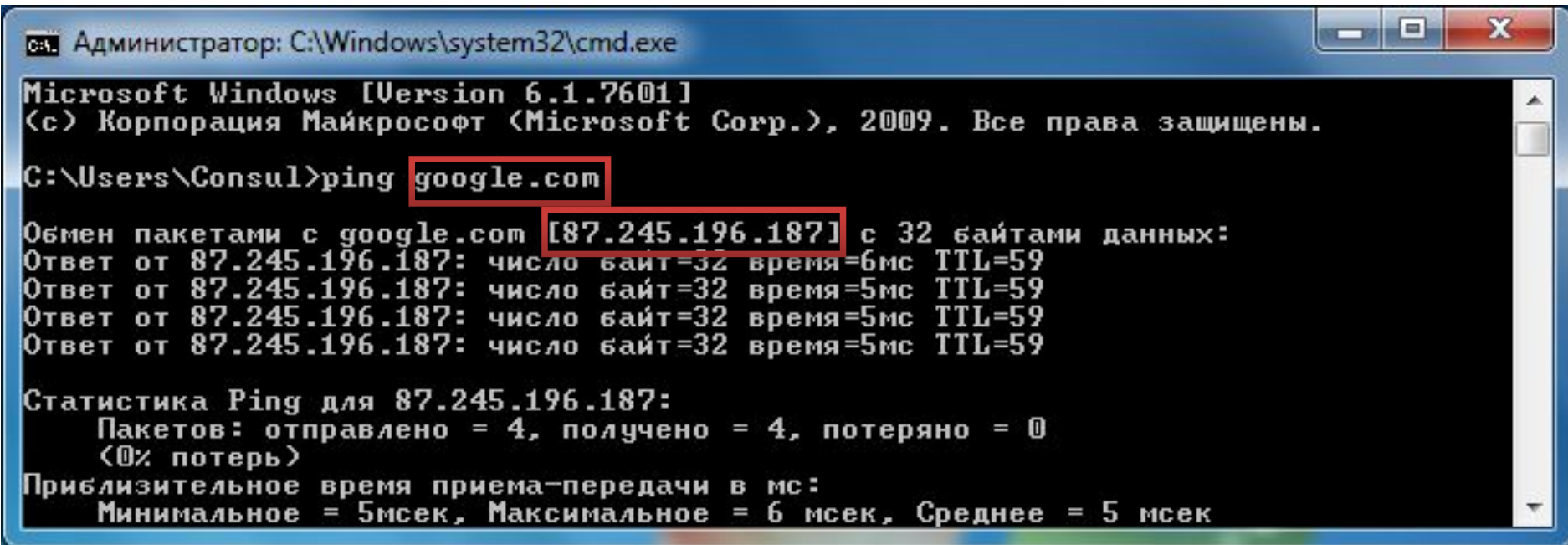
## 5. Доменная система имен. ICANN

**ICANN** (Internet Corporation for Assigned Names and Numbers, корпорация по управлению доменными именами и IP-адресами) – международная некоммерческая организация, созданная 18 сентября 1998 года при участии правительства США для регулирования вопросов, связанных с доменными именами, IP-адресами и прочими аспектами функционирования Интернета

До 1998 года регистрацией имён в доменах общего пользования занималась только одна компания. Подобный монополизм обуславливал высокую стоимость регистрации – каждый домен в зонах .com, .net и .org ежегодно обходился его владельцу в \$50. Это, в свою очередь, было одной из причин, препятствовавших росту количества зарегистрированных доменных имен: в 1998 году в мире их насчитывалось всего три миллиона

Корпорация ICANN начала использовать распределённую систему регистрации доменов, которая основана на принципе свободного доступа аккредитованных регистраторов к реестрам доменных имен. Этот шаг положил начало формированию конкурентного доменного рынка. Сегодня в доменных зонах общего пользования работает более 900 аккредитованных регистраторов, благодаря чему количество зарегистрированных доменов значительно возросло и уже превышает 270 миллионов

## 5. Доменная система имен. Пример



```
Администратор: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.
C:\Users\Consul>ping google.com
Обмен пакетами с google.com [87.245.196.187] с 32 байтами данных:
Ответ от 87.245.196.187: число байт=32 время=6мс TTL=59
Ответ от 87.245.196.187: число байт=32 время=5мс TTL=59
Ответ от 87.245.196.187: число байт=32 время=5мс TTL=59
Ответ от 87.245.196.187: число байт=32 время=5мс TTL=59

Статистика Ping для 87.245.196.187:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
    (0% потерь)
Приблизительное время приема-передачи в мс:
    Минимальное = 5мсек, Максимальное = 6 мсек, Среднее = 5 мсек
```

## 6. Порты и сервисы

IP-адрес позволяет точно идентифицировать компьютер, но этого недостаточно. Дело в том, что на каждом узле могут быть запущены самые разные **сервисы**, обеспечивающие передачу информации: электронной почты, файлов, гипертекстовой информации и т.п. Каждый сервис использует в своей работе тот или иной протокол прикладного уровня:

- для передачи файлов протокол FTP
- для передачи Web-страниц протокол передачи гипертекстовой информации HTTP
- для работы с электронной почтой протоколы SMTP, POP3 и др.

Для каждой сервиса отведен отдельный **порт**, представляющий собой число от 0 до 65534. Для наиболее популярных сервисов зарезервированы стандартные номера портов. Так, для FTP это 21, для HTTP – 80, SMTP – 25, POP3 – 110. Это значения по умолчанию, при необходимости можно изменить порт

## 7. Структура протокола HTTP. Модель OSI

Взаимодействие между узлами в Интернете построено по многоуровневому принципу, от физического уровня, связанного с физическими аспектами передачи двоичной информации, и до прикладного уровня, обеспечивающего интерфейс между пользователем и сетью



## 7. Структура протокола HTTP. Определение

**HTTP** (HyperText Transfer Protocol, протокол передачи гипертекста) – это протокол прикладного уровня, разработанный для обмена гипертекстовой информацией в Интернете

HTTP предоставляет набор методов для указания целей запроса, отправляемого серверу. Эти методы основаны на дисциплине ссылок, где для указания ресурса, к которому должен быть применен данный метод, используется универсальный идентификатор ресурсов **URI** в виде местонахождения ресурса **URL** и/или в виде его универсального имени **URN**

## 7. Структура протокола HTTP. URL. URN. URI

URL (Uniform Resource Locator, унифицированный определитель местонахождения ресурса) – адрес некоторого ресурса в Интернете. URL определяет местонахождение ресурса и способ обращения к нему

URN (Uniform Resource Name, унифицированное имя ресурса) – имя некоторого ресурса в Интернете. Смысл URN в том, что он определяет только название конкретного предмета, который может находиться во множестве конкретных мест

URI (Uniform Resource Identifier, унифицированный идентификатор ресурса) – обозначает имя и адрес ресурса в сети. Как правило делится на URL и URN, поэтому URL и URN это составляющие URI

Итог:

URI = URL или URI = URN или URI = URL + URN

URL = <http://en.wikipedia.org>

URN = [/wiki/Uniform\\_Resource\\_Identifier](#)

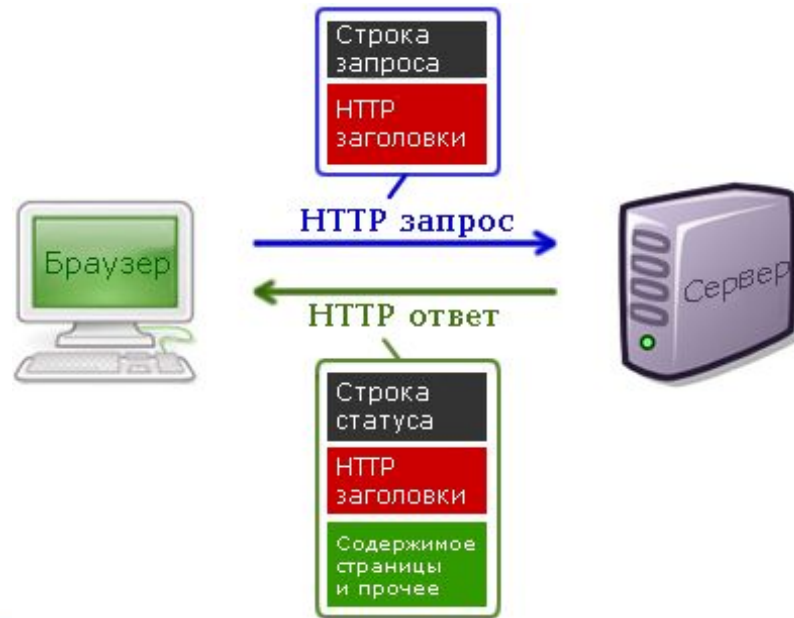
URI = [http://en.wikipedia.org/wiki/Uniform\\_Resource\\_Identifier](http://en.wikipedia.org/wiki/Uniform_Resource_Identifier)

## 7. Структура протокола HTTP

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

- **Стартовая строка** – определяет тип сообщения
- **Заголовки** – метаданные (данные о данных), характеризуют тело сообщения, параметры передачи и прочие сведения
- **Тело сообщения** – непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой

**Обратите внимание**, заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа



## 7. Структура протокола HTTP. Стартовая строка

Стартовые строки различаются для **запроса** и **ответа**

Строка запроса: Метод URI HTTP/Версия

Где:

- **Метод** – название запроса, одно слово заглавными буквами
- **URI** – определяет путь к запрашиваемому документу
- **Версия** – пара разделённых точкой цифр

```
GET www.example.com/index.html HTTP/1.1
```

Строка ответа сервера: HTTP/Версия КодСостояния Пояснение

Где:

- **Версия** – пара разделённых точкой цифр как в запросе
- **Код состояния** – три цифры. Пример: 404 Not Found – сервер понял запрос, но не нашёл соответствующего ресурса по указанному адресу
- **Пояснение** – текстовое короткое пояснение к коду ответа для пользователя. Является необязательным

```
HTTP/1.1 200 OK
```



## 7. Структура протокола HTTP. Методы

Метод HTTP – последовательность из любых символов указывающая на основную операцию над ресурсом. Обычно метод представляет собой короткое английское слово, записанное заглавными буквами

### GET

Используется для запроса содержимого указанного ресурса. Если URI указывает на статический файл, запрос GET обычно приводит к чтению файла и возврату его содержимого. Если URI указывает на программу, то в теле ответа возвращаются данные (если они имеются). Согласно стандарту HTTP, запросы типа GET считаются **идемпотентными** – многократное повторение одного и того же запроса GET должно приводить к одинаковым результатам (при условии, что сам ресурс не изменился за время между запросами). Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?»

```
GET /path/resource?param1=value1&param2=value2 HTTP/1.1
```

## 7. Структура протокола HTTP. Методы

### POST

В отличие от метода GET, который используется для извлечения информации, метод POST применяется главным образом для модификации имеющегося ресурса или **передачи данных обрабатывающему их процессу**. Тело запроса содержит данные. Исходный сервер в зависимости от URI запроса, разрешает выполнение определенных действий. Метод POST может изменять содержимое ресурса, поэтому не может считаться безопасным методом. Поскольку побочные эффекты множества идентичных запросов могут отличаться, метод POST не является идемпотентным методом

```
POST /path/resource HTTP/1.1
```

```
<Различные заголовки>
```

```
Content-Length: 27
```

```
<Различные заголовки>
```

```
param1=value1&param2=value2
```

### PUT

Схож с методом POST в том, что выполнение метода обычно приводит к изменению ресурса, идентифицируемого URI запроса. Если запрашиваемый через URI ресурс не существует, он создается, а если ресурс существует, то модифицируется. При использовании метода PUT в результате выполнения запроса изменяется сам идентифицируемый URI ресурс

### DELETE

Используется для удаления ресурса, идентифицируемого URI запроса. Метод предоставляет возможность дистанционного удаления ресурсов. Однако принимая во внимание суть этого действия, исходные серверы контролируют, было ли в действительности выполнено запрашиваемое действие, и когда это произошло. Сервер может отправить ответ об успешном выполнении, в действительности не удалив ресурса

## 7. Структура протокола HTTP. Заголовки

Заголовки HTTP – это строки в HTTP-сообщении, содержащие разделённую двоеточием пару параметр-значение

Формат – **Имя : Значение**

Server: Apache/2.2.11 (Win32) PHP/5.3.0

Last-Modified: Sat, 16 Jan 2010 21:16:42 GMT

Content-Type: text/plain; charset=windows-1251

Content-Language: ru

Все HTTP-заголовки разделяются на четыре группы:

- **Основные заголовки** (General Headers)
- **Заголовки запроса** (Request Headers)
- **Заголовки ответа** (Response Headers)
- **Заголовки сущности** (Entity Headers)

## 7. Структура протокола HTTP. Заголовки

**Основные заголовки** – применяются как для сообщений запросов, так и для сообщений ответов, но которые не применяются к передаваемому объекту. Эти поля заголовка применяются только к передаваемому сообщению. Некоторые из заголовков:

Заголовок	Назначение	Пример
Cache-Control	основные директивы для управления кэшированием	Cache-Control: max-age=3600
Connection	сведения о проведении соединения	Connection: close
Date	дата и время формирования сообщения	Date: Tue, 15 Nov 1994 08:12:31 GMT

## 7. Структура протокола HTTP. Заголовки

**Заголовки запроса** – позволяют клиенту передавать серверу дополнительную информацию о запросе и о самом клиенте. Некоторые из заголовков:

Заголовок	Назначение	Пример
Accept-Charset	перечень поддерживаемых кодировок для предоставления пользователю	Accept-Charset: utf-8
Authorization	данные для авторизации	Authorization: Basic QWxhZGRpbjpvvcGVuIHNIc2FtZQ==
From	адрес электронной почты ответственного лица со стороны клиента	From: user@example.com

## 7. Структура протокола HTTP. Заголовки

**Заголовки ответа** – позволяют серверу передавать дополнительную информацию, касающуюся ответа. Эти поля заголовка дают информацию о сервере и о дальнейшем доступе к ресурсу. Некоторые из заголовков:

<b>Заголовок</b>	<b>Назначение</b>	<b>Пример</b>
Age	количество секунд с момента модификации ресурса	Age: 12
Location	адрес перенаправления	Location: <a href="http://example.com/about.html">http://example.com/about.html</a>
Server	информация о программном обеспечении сервера	Server: Apache/2.2.17 (Win32) PHP/5.3.5

## 7. Структура протокола HTTP. Заголовки

**Заголовки сущности** – определяют опциональную метаинформацию о теле сообщения или, если тело не присутствует, относительно ресурса, идентифицированного запросом. Некоторые из заголовков:

<b>Заголовок</b>	<b>Назначение</b>	<b>Пример</b>
Allow	список поддерживаемых методов	Allow: GET, POST
Content-Language	один или несколько естественных языков содержимого сущности	Content-Language: en, ase, ua
Last-Modified	дата последней модификации сущности	Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT





## 7. Структура протокола HTTP. Тело сообщения. Пример

Тело HTTP сообщения, если оно присутствует, используется для передачи тела объекта, связанного с запросом или ответом

```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed
```

```
<html>
...
</html>
```

## 7. Структура протокола HTTP. HTTPS

**HTTPS** (Hypertext Transfer Protocol Secure, безопасный протокол передачи гипертекста) – расширение протокола HTTP, поддерживающее шифрование. Данные, передаваемые по протоколу HTTPS, "упаковываются" в криптографический протокол **SSL** или **TLS**, тем самым обеспечивается защита этих данных. В отличие от HTTP, для HTTPS по умолчанию используется порт 443

**Обратите внимание**, HTTPS не является отдельным протоколом. Это обычный HTTP, работающий через зашифрованные транспортные механизмы SSL и TLS

**Спасибо за внимание!**