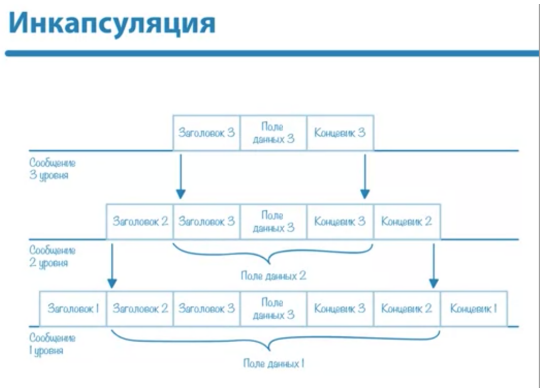
# 1 Неделя 1

## 1.1 Урок 1: основы организации компьютерных сетей; понятия интерфейса и протокола; инкапсуляция протокола.

Протокол - это правила и соглашения, используемые для связи уровня N одного компьютера с уровнем N другого компьютера.




После этого сообщение передается по каналам связи.

Основные модели организации архитектуры сети:

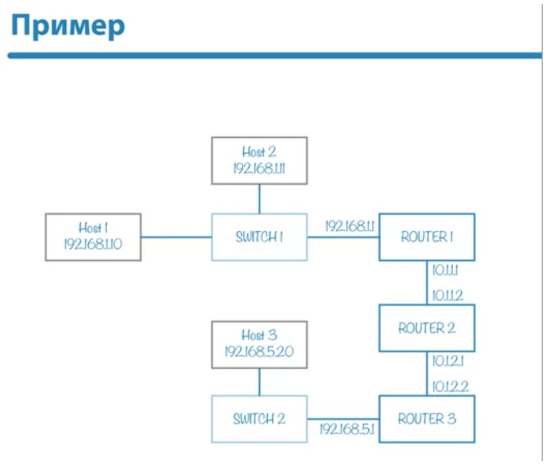
- Модель взаимодействия открытых систем OSI;
- Модель TCP/IP.

## 1.2 Урок 2:Модель TCP/IP и немного про модель OSI


### 1.2.1 Уровень сетевых интерфейсов (Network access layer)

Задачи:

- упаковка IP-пакета в единицу передаваемых данных промежуточной сети;
- преобразование сетевых адресов в адреса технологии данной промежуточной сети.




Происходит преобразование IP-адреса в адрес локальной технологии (его mac-адрес). И с помощью этой локальной технологии пакет доставится адресату.

### 1.2.2 Сетевой уровень (Network layer)

- Служит для образования единой транспортной системы, объединяющей несколько сетей, причем эти сети могут использовать совершенно разные технологии передачи данных;
- Пример протокола: IP.

Тот же пример, что и был. На этот раз необходимо доставить пакет от Host1 к Host3. Задача сетевого уровня - найти путь и доставить пакет от router 1 к router 3. Доставкой пакета от Host1 к Router1 и от Router3 к Host3 занимается локальная технология, например, «изернет».

Пример протокола сетевого уровня: IP.

### 1.2.3 Транспортный уровень (Transport layer).

Обеспечивает передачу данных между процессами (?).

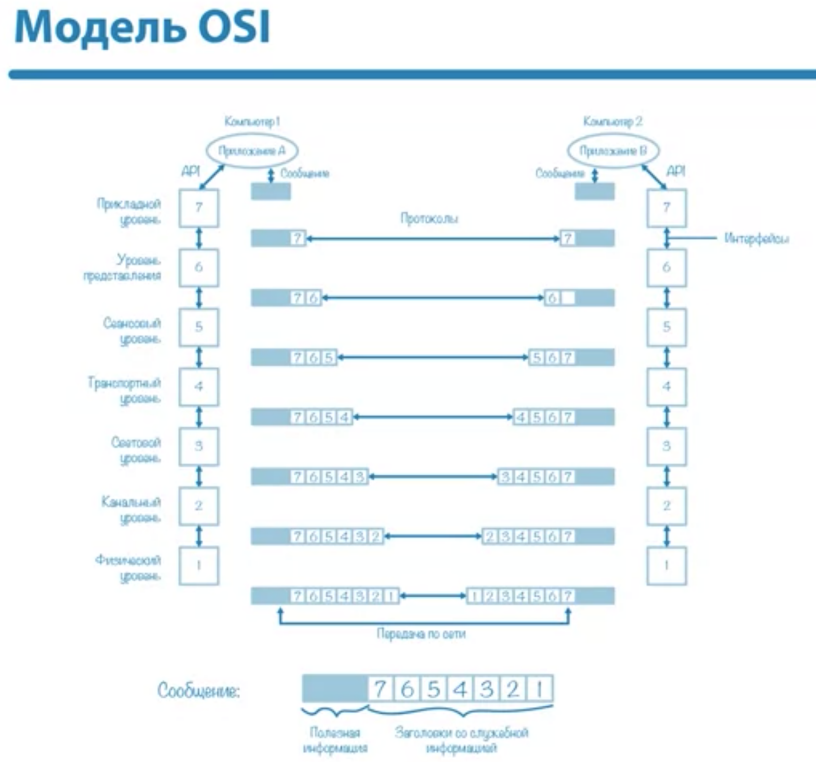
- Особенностью транспортного уровня является управление надежностью. Уровенть предоставяет приложениям или верхним уровням стека передачу данных с той степенью надежности, которая им требуется.
- Примеры протоколов: TCP и UDP.

Пример: необходимо передать большой файл по сети. Он будет передаваться кусочками. Необходимо, чтобы каждый кусок дошел, да еще чтоб в правильном порядке.

### 1.2.4 Прикладной уровень (Application layer).

- Набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к разделяемым ресурсам.
- Пример протоколов: HTTP, FTP.

### 1.2.5 Модель OSI (Open Systems Interconnection) (модель взаимодействия открытых систем).




Модель OSI и TCP/IP во многом схожи. Достоинство модели OSI - теоретически проработанна. Но на практике широко не применяется. Достоинство TCP/IP - стек протоколов. Тк он широко применяется на практике и лежит в основе интернета.

## 1.3 Урок 3: Детально транспортный уровень и его протоколы.

Задачи транспортного уровня

- передача данных между процессами в сети;
- предоставление различного уровня надежности передачи данных, независимо от надежности сети.

Для адресации на траспортном уровне используются порты.

Каждое сетевое приложение на хосте имеет свой порт. Номера портов у приложений не должны повторяться.

Формат записи вместе с IP-адресом: 192.168.0.1:8080

### 1.3.1 Протокол UDP (User Datagram Protocol).

Особенности UDP:

- Работает без установления логического соединения;
- Нет гарантии доставки данных;

- Нет гарантии сохранения исходного порядка дейтаграмм;
- Гарантирует корректность данных внутри одной дейтаграммы.


### 1.3.2 Протокол TCP (Transmission control protocol)

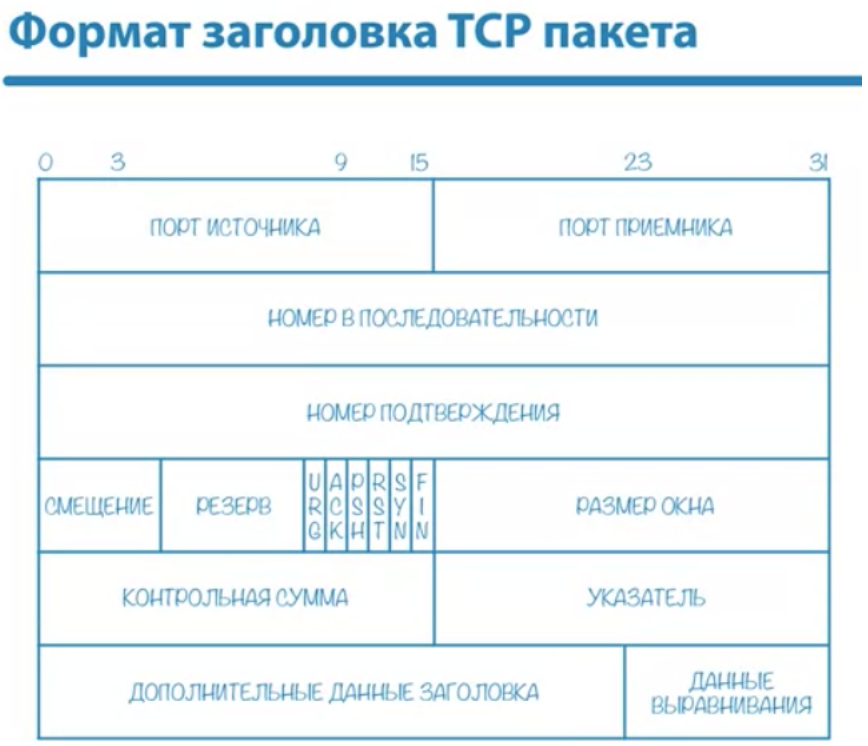
- Является надежным протоколом передачи данных.
- Особенности TCP
  - Работает с установления логического соединения
  - Гарантирует доставку данных
  - Гарантирует сохранение порядка следования пакетов

### 1.3.3 Логическое соединение

Для надежности передачи данных между двумя процессами необходимо установить логическое соединение. «Соединение» является договоренность о параметрах между двумя процессами.

Взаимодействие партнеров с использованием протокола TCP строится в три этапа:

- установление логического соединения;
- обмен данными;
- закрытие соединения.





Флаг SYN - пакет является запросом для установления логического соединения.

TCP B отвечает в номере подтверждения на 1 больше чем в номере последовательности.

Флаг ACK - означает, что TCP пакет содержит в своем поле номера подтверждения верные данные.

На третьем шаге A подтверждает правильность приема пакета B.

### 1.3.4 Процесс обмена данными

Каждый раз когда TCP модуль принимает данные, он подтверждает их прием:

вычисляет значение поля «номер подтверждения» = номер в последовательности + длина данных.


### 1.3.5 Закрытие соединения

По инициативе A:


Флаг FIN - Tcp пакет представляет из себя запрос на закрытие логического соединения и вляется признаком конца потока данных.

B отправляет пакет, у которого в номере подтверждения на 1 больше значение чем в номере последовательности полученного от A.

После этого посылка данных от A невозможна, но B имеет возможность отправлять данные A и получать подтверждение об этом.

После того как пакет B формирует флаг FIN и получает подтверждение о его принятии, соединение считается закрытым.

## 1.4 Протоколы UDP и TCP на практике

seq - номер в последовательности

win - размер окна (?)

флаг S - SYN

флаг . - ACK

Флаги E, W - служебные флаги.

ack - номер подтверждения.

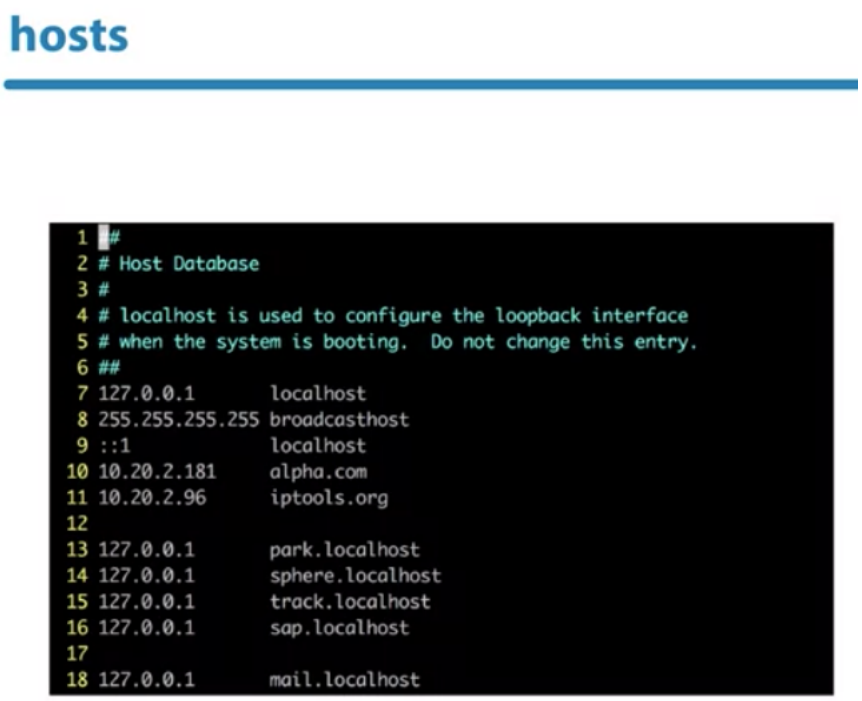
дальше идут номера последовательности в относительном режиме (от единицы) пример: 1:15.

Флаг F - FIN.

## 1.5 DNS-протокол

DNS (Domain Name System - система доменных имен) - распределенная система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства).

Раньше было так:




Потом он стал слишком большим и придумали DNS.

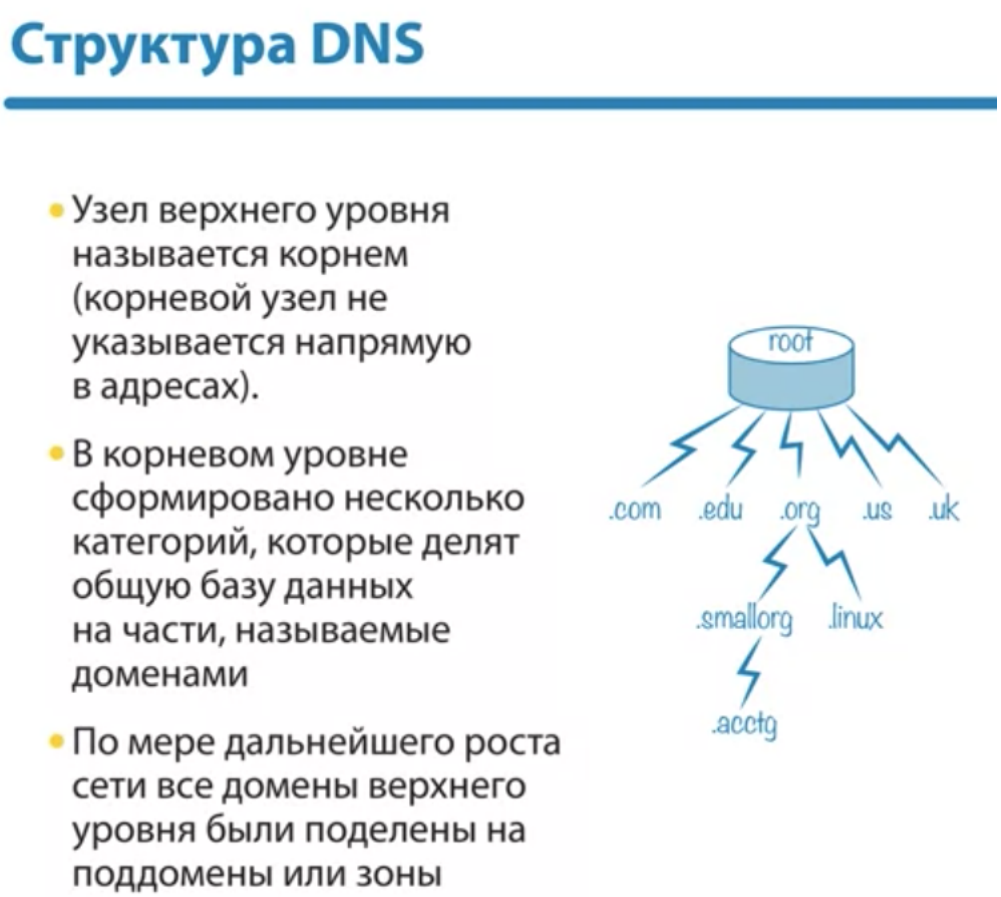
### 1.5.1 Назначение DNS

Распределенность. Ответственность за разные части иерархической структуры несут разные люди или организации; каждый узел сетив обязательном порядке должен хранить только те данные, которые входят в его зону ответственности.

Кэширование. Узел может хранить некоторое количество данных не из своей зоны ответственности для уменьшения нагрузки на сеть.

Резервирование. За хранение и обслуживание своих узлов (зон) отвечают (обычно) несколько серверов.

### 1.5.2 Структура DNS




### 1.5.3 Схемы разрешения DNS-имен

**Нерекурсивная.**

Клиент обращается к корневому DNS-серверу с указанием полного доменного имени, он отвечает клиенту, указывая адрес следующего DNS-сервера. Клиент делает запрос следующего DNS-сервера, который отсылает к DNS-серверу нужного поддомента и т.д., пока не будет найден DNS-сервер, в котором хранится соответствие запрошенного имени IP-адресу. Этот сервер дает окончательный ответ клиенту.

**Рекурсивная.**

Клиент запрашивает локальный DNS-сервер. Далее возможны два варианта действий: если локальный DNS-сервер знает ответ, то он сразу же возвращает его клиенту; если локальный сервер не знает ответ, то он выполняет итеративные запросы к корневому серверу и т.д. точно так же, как это делал клиент в предыдущем варианте, а подучив ответ, передает его клиенту, который все это время просто эжет его от своего DNS-сервера.

### 1.5.4 Типы записей в DNS

DNS-сервер, отвечающий за имена хостов в своей зоне, должен хранить информацию о хостах в базе данных и выдавать е по запросу с удаленных компьютеров. Обычно, база дан-

ных DNS представляет собой текстовый файл, состоящий из исходных записей RR (resource records).

Записи:

- A-запись.

  Указывает адрес хоста. Она отображает хост-имя на адрес и может выглядеть следующим образом: myhost.mycompany.com IN A 192.168.0.1

- AAAA-запись.

  Указывает адрес хоста. Она отображает хост-имя на адрес IPv6 и может выглядеть следующим образом:

  myhost.mycompany.com IN AAAA 1234:1:2:3:4:567:89cd

- CNAME-запись

  Это каноническая запись, обычно используемая для алиасов (псевдонимов). Она позволяет отображать несколько хост-имен на заданный IP-адрес.

- NS-запись.

  В каждой зоне должно быть по крайней мере два сервера DNS. Записи NS служат для их идентификации другими DNS-серверами, которые пытаются преобразовать имена хостов, относящихся к данной зоне.

- SOA-запись.

  Запись, которая определяет авторитетную информацию о DNS-зоне.

  Например, указываются параметры:

  - Primary Name Server - имя первичного DNS-сервера зоны
  - Hostmaster - контактный адрес лица, ответственного за администрирование файла зоны
  - Serial number - серийный номер файла зоны. 32-разрядное целое число, меняющееся при каждом обновлении зоны.

- PTR-запись

  Запись позволяет быстро выполнять обратный поиск с помощью зоны обратного поиска (inaddr.arpa). Она считается обратной A-записью, но не похожа на нее ввиду использования inaddr.arpa в реальной записи.

  Такая запись для хоста syscrusher.skillet.com с IP-адресом 100.200.252.1 имеет следующий вид: 1.252.200.100.in-addr.arpa IN PTR syscrusher.skillet.com.

- MX-запись

  С их помощью удаленные серверы электронной почты узнают, куда отправлять почту для вашей зоны.

- TXT-запись

  Запись TXT содержит текстовые данные любого вида.

### 1.5.5 Утилита dig

-x - ищет PTR записи по IP-адресу.

mx - mx записи дополнительно

+trace - то, как происходит поиск нужного IP-адреса (его путь).

## 1.6 HTTP-протокол

### 1.6.1 HTTP-протокол

HTTP (HyperText Transfer Protocol - RFC 1945, RFC 2616) - протокол прикладного уровня для передачи гипертекста.

Все ПО для работы с протоколом HTTP разделяется на:

- Сервер - поставщики услуг хранения и обработки информации (обработка запросов).
- Клиент - конечные потребители услуг сервера (отправка запросов).

### 1.6.2 Схема HTTP-сеанса

- Установление TCP-соединения.
- Запрос клиента.
- Ответ сервера.
- Разрыв TCP-соединения.

Таким образом, клиент посылает серверу запрос, получает от него ответ, после чего взаимодействие прекращается. Из этого следует, что протокол не хранит информацию о предыдущих запросах клиентов и ответах сервера.

### 1.6.3 URI/URL

*Для того, чтобы получить какие-то данные, необходимо КУДА-ТО за ними сходить*

Единообразный идентификатор ресурса URI (Uniform Resource Identifier) представляет собой короткую последовательность символов, идентифицирующую абстрактный или физический ресурс. URI не указывает на то, как получить ресурс, а только идентифицирует его. Один из самых известных примеров URI - URL.

Для однозначной идентификации ресурсов в сети Веб используются уникальные идентификаторы URL (Uniform Resource Locator). *Указывает однозначно - где находится ресурс и как его получить.*

Имеет следующую структуру:

<схема>://<логин>:<пароль>@<хост>:<порт>/<URL-путь>.

схема это - http, ftp, https и тд.

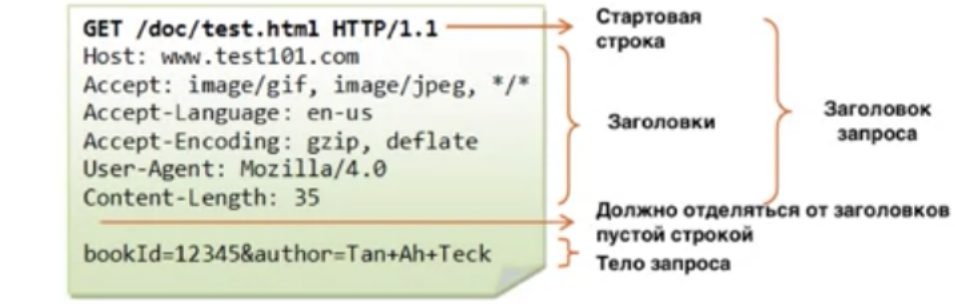
логин и пароль - необязательные поля, которые в каких-то протоколах требуются, а в каких-то нет (в http они опускаются).

хост и порт к которым нужно обратиться

### 1.6.4 Структура запроса

#### Стартовая строка HTTP

Стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа, заголовки и тело сообщения могут отсутствовать. Стартовые строки различаются для запроса и ответа.




Строка запроса выглядит так: Метод URL-Путь HTTP/Версия протокола

### 1.6.5 Виды запросов

**Методы протокола:**

- GET

  Используется для получения какой-то информации (пример: мы делаем запрос в браузере).

- HEAD

  Аналогично GET, но он как бы говорит:*выполни запрос, но не присылай тело этого запроса (?)*

- POST

  Служит, чтобы отсылать от клиента данные на сервер.

- PATCH.

  Служит для того, чтобы изменять отправленные данные. Пример: С помощью POST мы отправили какую-то информацию на сервер, она сохранилась и теперь, чтоб изменить ее мы используем PATCH.

- DELETE

  Обращаемся к серверу с необходимостью удалить какие-то данные.

- PUT

- ...

PATCH используется для частичного изменения ресурса. PUT создает новый ресурс или заменяет представление целевого ресурса, данными представленными в теле запроса.

**Поля заголовка**

Поля заголовка, следующие за строкой состояния, позволяют уточнять запрос т.е. передавать серверу дополнительную информацию. Поле заголовка имеет следующий формат:

Имя_поля:Значение.

*Различные поля заголовка:*

- Host - доменное имя или IP-адрес узла, к которому обращается клиент
- Referer - URL, откуда перешел клиент (например, если мы с google.com перешли на docs.google.com, то в Reger будет передаваться google.com)
- Accept - MIME-типы данных, обрабатываемых клиентом.
- Accept-Charset - перечень поддерживаемых кодировок
- Content-Type - MIME-тип данных, содержащихся в теле запроса
- Content-Length - число символов, содержащихся в теле запроса
- Connection - дирректива для управления TCP-соединением (например, если в Connection указать тип Live, то соединение не будет закрыто и будет использовано для дальнейших запросов)
- User-Agent - информация о клиенте (например, если мы делаем запрос с мозилы, то в User-agent отобразится это, если с сафари, то с сафари и тд)

**MIME**

*Данный спецификатор позволяет передавать от клиента к серверу различные типы данных*

Спецификация MIME (Multipurpose Internet Mail Extension - многоцелевое почтовое расширение Internet) первоначально была разработана для того, чтобы обеспечить передачу различных форматов данных в составе электронных писем.

До появления MIME-устройства, взаимодействующие по протоколу HTTP, обменивались исключительно текстовой информацией.

Для описания формата данных используются тип и подтип. Тип определяет. к какому классу относится формат содержимого HTTP-запроса или HTTP-ответа. Подтип уточняет формат (text/html, image/png)

### 1.6.6 Структура ответа

Строка состояния состоит из версии протокола и статуса HTTP-ответа и расшифровки этого ответа.

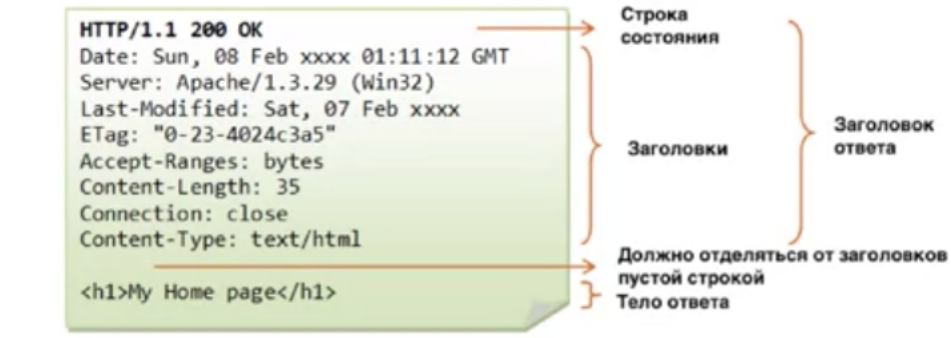
### 1.6.7 Виды кодов (статусов) ответа

Существует 5 классов ответов:

- 1xx - специальный класс сообщений, называемых информационными.

  Означает, что сервер продолжает обработку запроса. (101 Switching Protocols) (например, мы работали по протоколу HTTP, а сервер нас переключил на работу по протоколу WEB-socket и для этого он отправляет статус 101)

- 2xx - успешная обработка запроса клиента. (пример 200 Ok - получаем этот статус. когда обращаемся к google.com и нам выдается страница; пример 201 Created - мы отправили на сервер какие-то данные, он их принял, сохранил и написал Created).




- 3xx - перенаправление запроса (301 Moved Permanently - в случае когда мы обращаемся к какому-то ресурсу, но этот ресурс переехал на другой адрес и нам сервер ответит 301; 302 Found - в случае если он также переехал но на какое-то время или же мы находились на /login, ввели правильно логин и пароль, он нас зарегестрировал и отправил на главную).
- 4xx - ошибка клиента. (400 Bad Request - некорректный HTTP-запрос; 403 Forbidden - недостаточно прав для доступа к ресурсу; 404 Not Found - обращение к ресурсу, которого не существует)
- 5xx - ошибка сервера. (500 Inernal Server Error - в том случае, когда логика сервера как-то некорректно отработала; 502 Bad Gateway - в случае, если вы, например, создали такую систему, в которой вначале клиент обращается к промежуточному серверу, а потом к вашему сервер-приложению и ващ сервер-приложение по какой-то причине не отвечает, то промежуточный сервер вернет 502)

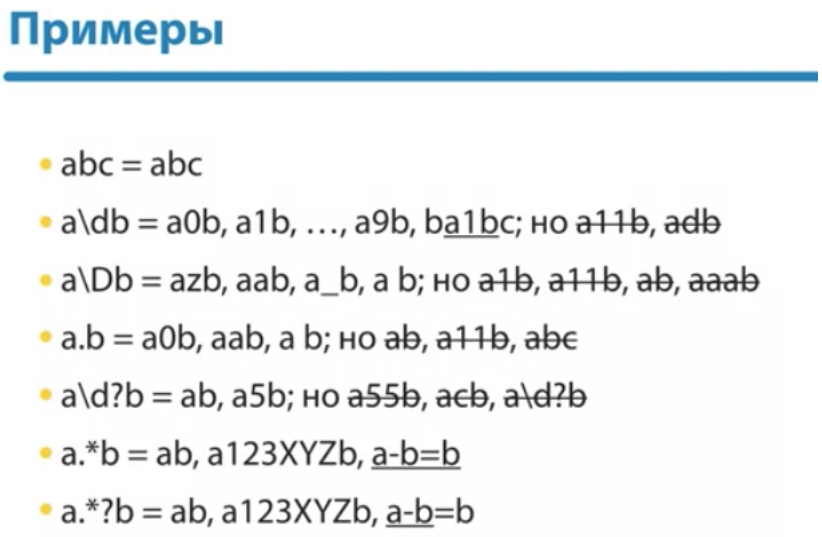
### 1.6.8 Основные заголовки

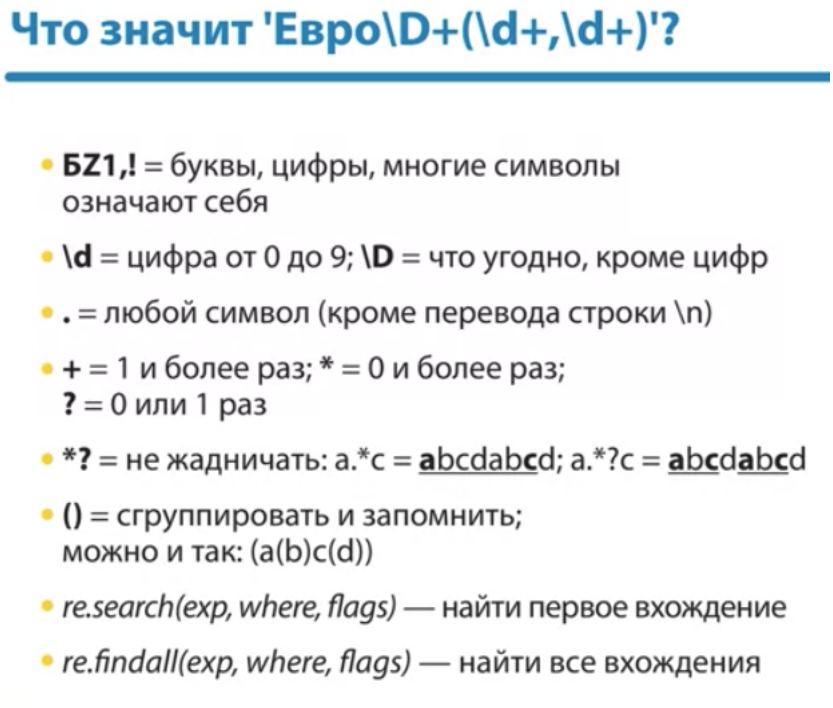
#### Поля заголовка ответа

- Server - имя и номер версии сервера (пример: NGINX, Apache)
- Allow - список методов, допустимых для данного ресурса
- Content-Type - MIME-тип данных, содержащихся в теле ответа сервера
- Content-Length - число символов, содержащихся в теле ответа сервера
- Last-Modified - дата и время последнего изменения ресурса
- Expires - дата и время, когда информация станет устаревшей
- Location - расположение ресурса (данный заголовок используется, например тогда, когда сервер нам прислал статус 302 и говорит нам переместиться на другой ресурс, соответственно, оттуда бразуер возьмет информацию и перейдет на другой ресурс)

- Cache-Control - директива управления кэшированием. (например, если указать в Cache-Control - NoCache, то данные которые приходят с сервера, никогда не будут сохранены или закэшированы).

## 1.7 Регулярные выражения




### 1.7.1 Символьные классы и квантификаторы

### 1.7.2 Сложный поиск и замена

**Специальные символьные классы** - наиболее часто используемые символьные клас-

## Специальные символьные классы

- **\w** = буква, цифра и _ = [a-zA-Z_0-9] + юникод
- **\W** = **[^\w]** = не "буквоцифра"
- **\s** = пробел [\f\n\r\t\v]
- **\S** = не пробел
- **\b** = граница между \w и \W (пустая строка)
- **\B** = позиция внутри слова
- ^ = начало строки; **$** = конец строки

- Квантификаторы: {3,5} * + ?
- Символьные классы: [a-f] и исключающие [^a-f]

сы.

## Символьные классы и квантификаторы

- **[abcd1234]** = **[a-d1-4]** = один символ из множества
- **\d** = [0123456789] = [0-9]; **\D** = [^\d] = [^0-9]
- **{1, 2}** = от 1 до 2 раз; ab{1,2}c = ac, abc, abbc, abbbc
- **{2}** = {2, 2}; a\d{2}c = abbc, a5c, a00c, a15c, a991c
- **{, 2}** = {0, 2}; ab{,2}c = ac, abc, abbc, abbbc
- **{2, }** = {2, ∞}; ab{2,}c = abc, abbc, abbbc, …
- * = {0, ∞}; + = {1, ∞}; **?** = {0, 1}

Чтобы были только буквы латинского алфавита, необходимо установить флаг re.ASCII.
Если указан флаг re.MULTILINE - то
и $ означают начало и конец КАЖДОЙ строки.