```
не структурированно
```

http -- протокол прикладного уровня, на котором упрощенно говоря общаются браузер и веб сервер. это протокол предметной области -- он знает что такое web, какие атрибуты(свойства) есть у запросов и у ответов
http пакет заворачивается в tcp пакет
протокол в более нижнего уровня
самый верхний (дня нас) уровень -- http
https = http завернутый в tls чтобы шифровать (открытым\закрытым ключом сервера)
Отец основатель веба -- Тим Бернер-Ли (CERN) , первый сайт info.cern.ch 6 августа 1991

История

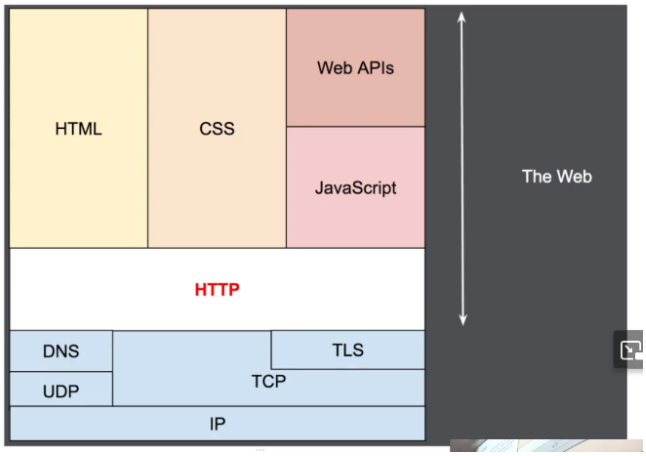
- “Отец” веба Тим Бернерс-Ли (CERN), первый сайт http://info.cern.ch 6 августа 1991 года.
- HTTP/0.9 — 1990-1992 (+язык HTML и идентификаторы URI)
- HTTP/1.0 — 1996
- HTTP/1.1 — 1999
- HTTP/2 — 2015
- HTTP/3 — ведется работа, HTTP over QUIC

сервер и клиент(пользовательский агент) обмениваются запросами и ответами на них (requests and responds)
начиная с хттп2 протокол усложнился но основная часть общения происходит по такой схеме: пользовательский user-agent посылает запрос "дай мне (или получи лот меня) такие то данные" -- и сервер отвечает(получает) этими данными

HTTP

HTTP

- протокол прикладного уровня (другие примеры ftp, ssh, dns)
- клиент-серверный, запрос-ответ
- текстовый (не совсем в HTTP 2)
- stateless
- семейство TCP/IP (до HTTP 3)
- поверх строятся другие протоколы (SOAP, XML-RPC, WebDAV)



TCP -- протокол с помощью которого сетевые интерфейсы взаимодействуют между собой устанавливая надежное (reliable-)соединение
это делается трехвазным рукопожатием во время установки соединения -- небыстрая тема

кажется
когда был хттп 1.0 предполагалось что для каждого документа который клиент хочет скачать с сервера (ну т е для каждого запроса) используется свое соединение т е устанавливается с нуля tcp соединение

мы просим сервер дает и соединение закрывается
очевидно долгая ресурсоемкая хуйня поэтому в 1.1 был придуман keep-alive концепция -- когда user-agent устанавливает соединение с сервером и может не разрывать его и уже в рамках этого соединения можно посылать много реквестов респонсов
кип-элайвы не бесконечны -- обычно 10ки-100ни секунд
TLS -- типа шифрованный TCP

когда мы говорим tcp\ip нужно понимать что протокол ip более низкий чем tcp
"пакеты tcp уложены в протокол ip"

все выше хттп -- **в основном** (#проверить) для отображения запрошенного контента

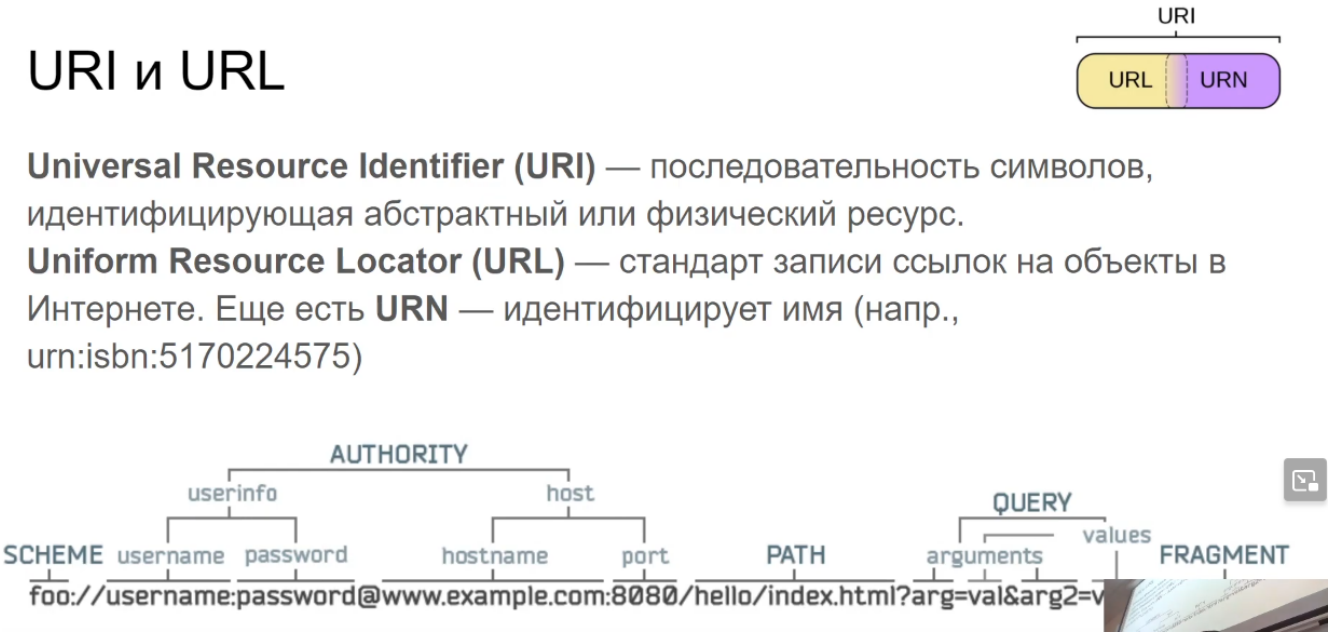
http -- протокол прикладного уровня, предметной области

примеры других протоколов прикладного уровня: ssh -- "присоединяемся к серверу" , smtp
"клиент-серверный протокол, запрос-ответ ориентированный"
разделение клиент\сервер есть не во всех протоколах

stateless -- не интересует история
состояние закодировано в запросе (вспоминаем session.getAttribute())

URI и URL

https://ru.wikipedia.org/wiki/URI



картинка -- **УРЛ**
"унифицированный локатор ресурсов" , местоположение ресурсов в интернете
будем обращаться к базам данных и вместо foo будет mysql или jdbc
может быть ssh
порт это tcp порт
для http дефолтный порт это 80
https -- 443
путь -- case-sensitive, а схема и хостнэйм **вроде** нет

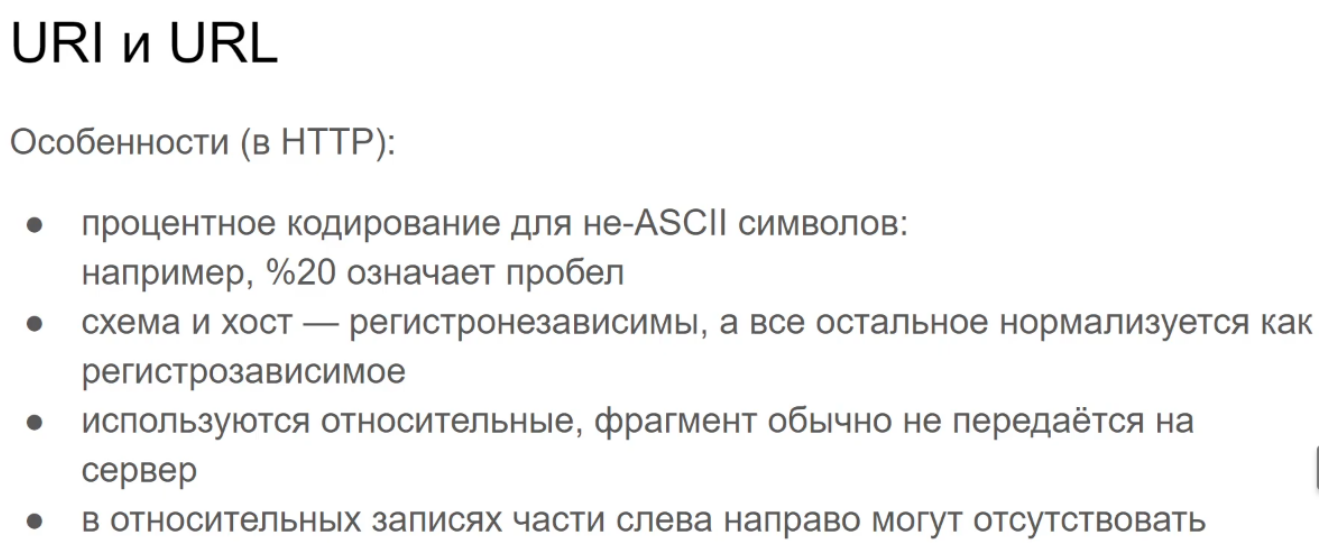
параметры у запроса (своего рода) бывают 2х видов:

1. get-параметры -- которые содержатся в get строке -- используются в основном для гет запросов -- могут быть доступны и видны промежуточным прокси-серверам, попадают в логи

2. которые находятся в самом запросе лол
   fragment: после решетки -- определенный кусок на странице куда проскроллить
   обычно не передается на сервер

URI это в принципе концепция , объединение URL и URN
URN это концепция идентификатора имени, например isbn (urn:isbn:5170224575), *впринципе это нам знать не надо*

особенности (слайд)



например русские символы будут кодироваться через "процентное" кодирование

практичим

neerc.ifmo.ru -- хост
чтобы установить tcp соединение по сокету
сокет это конечная точка соединения
нам нужны 2 вещи -- ip адресс и номер порта
ip адрес вполне определяется по хост-нейму с помощью протокола dns (это суть этого протокола)

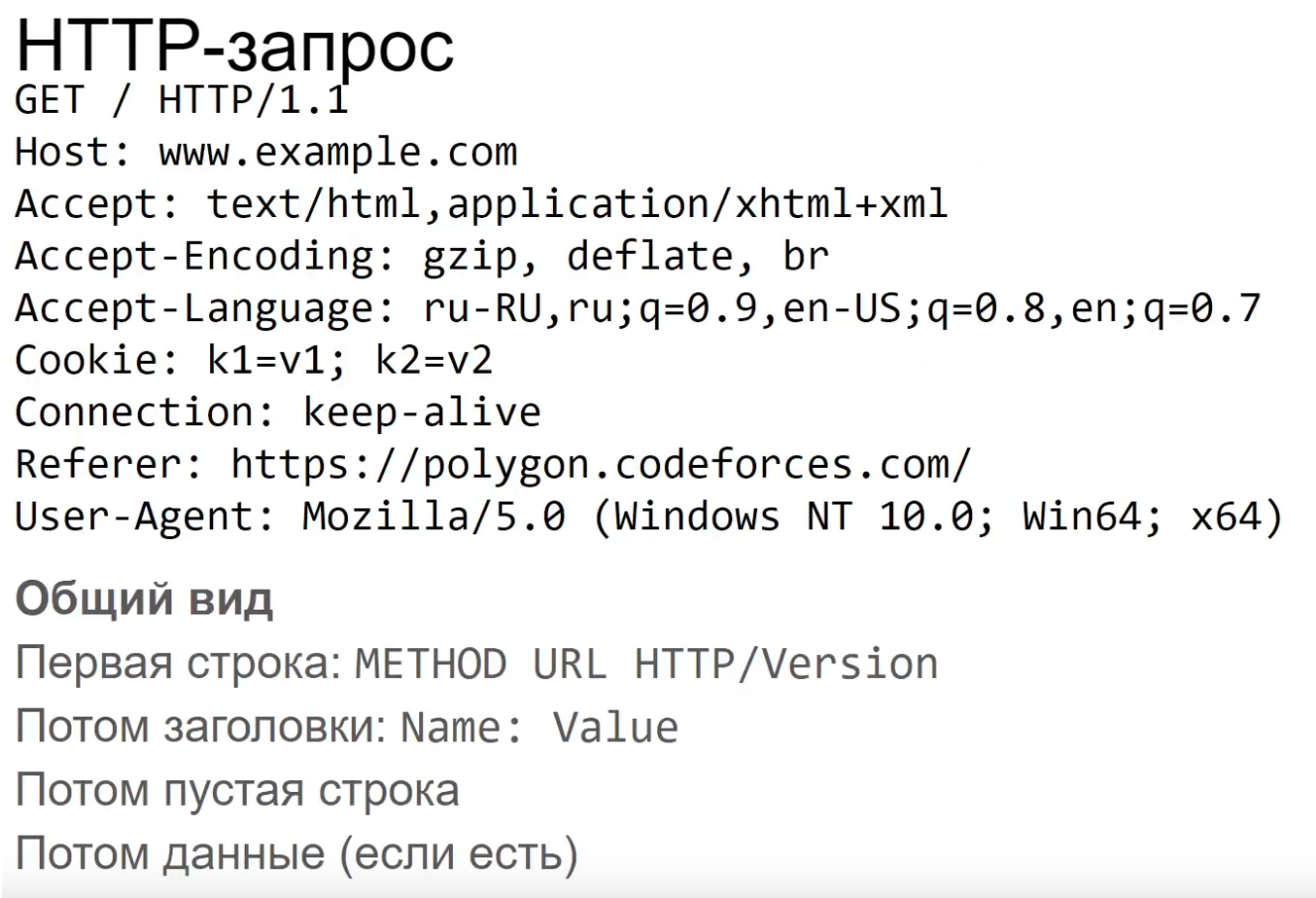
```
> ping neerc.ifmo.ru

64 bytes from neerc.ifmo.ru (77.234.215.132): icmp_seq=1 ttl=56 time=30.2 ms
64 bytes from neerc.ifmo.ru (77.234.215.132): icmp_seq=2 ttl=56 time=27.5 ms
64 bytes from neerc.ifmo.ru (77.234.215.132): icmp_seq=3 ttl=56 time=27.5 ms
^C
--- neerc.ifmo.ru ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 27.460/28.398/30.200/1.274 ms
```

кстати на компе есть?бывает? файлик hosts где браузер может сначала поискать нужный ip для текстого представления хоста
т о на разработке для несуществующего хоста можно вбить сайт -> 127.0.0.1 и типа вот)
кажется в \etc\hosts

telnet -- tcp хуйня

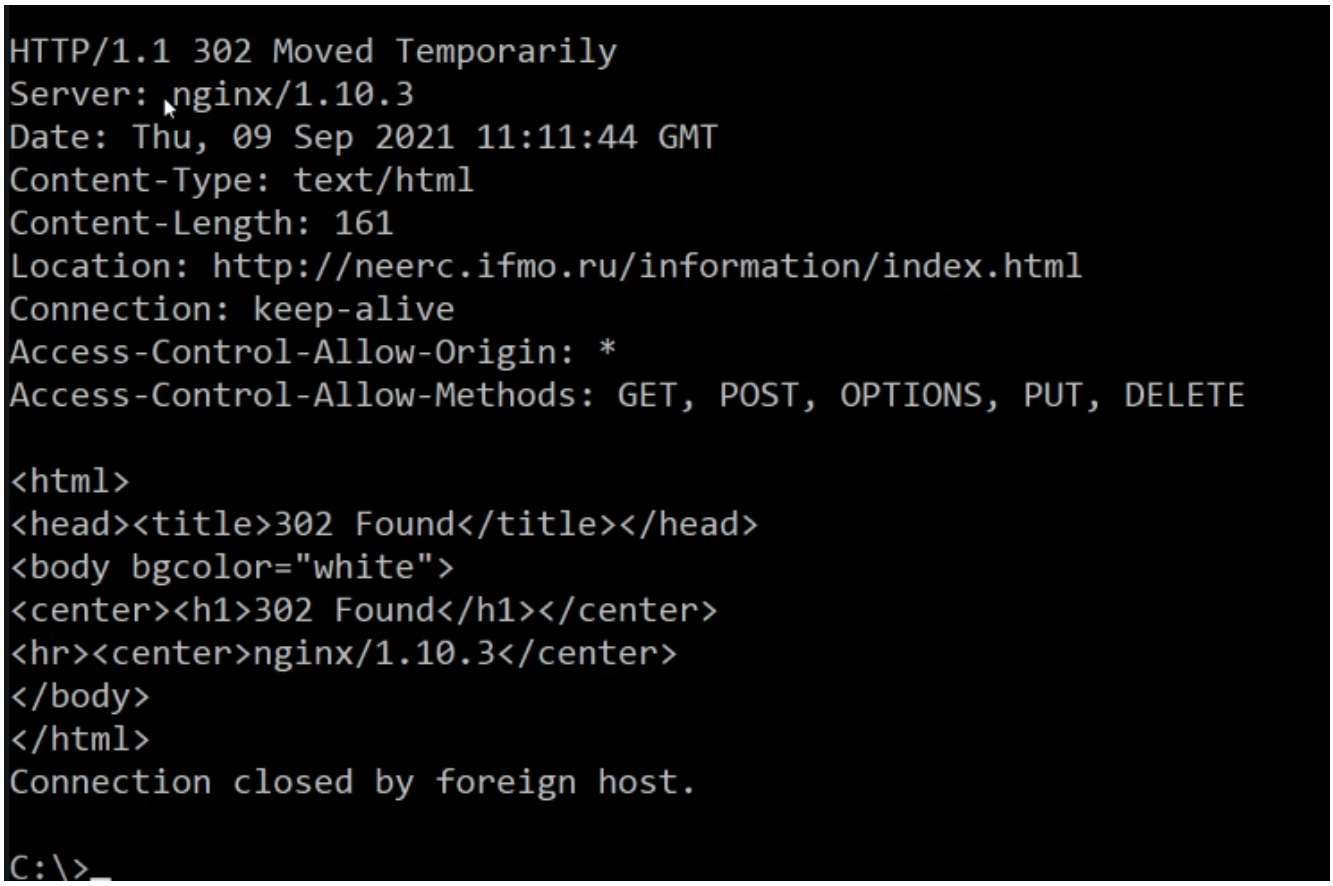
как выглядит HTTP запрос



GET POST и т п -- методы http запроса
get значит что мы хотим запросить данные -- типа read-only запрос
post -- отправить данные
версия протокола прописывается например чтобы делать обратную совместимость

сайтов в мире больше чем серверов так что один сервер может держать несколько сайтов
обязательные -- первые две строки и пустая строка как конец запроса

пример ответа



302 -- статусный код

код ответа:

первая цифра от 1 до 5

302 -- редирект

200 -- OK

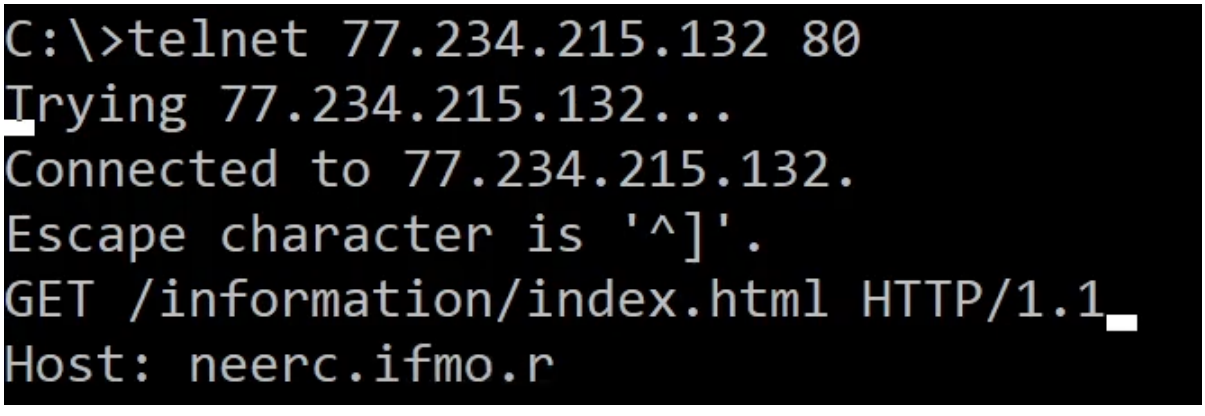
после заголовков пустая строка и тело ответа

заметка:

ответ на фото нам вернул реверс-прокси сервера, который стоит перед сервером

nginx это легковесный реверс-прокси

зачастую версию сервера(или реверс-прокси) скрывают чтобы было сложнее ломать



еще один пример запроса

поговорили про заголовки

Accept: формат "группа/подгруппа" + веса всякие... пример: text/html

Accept-Encoding: алгоритмы сжатия(компрессии) которые "браузер" использует

Accept-language понятно что

Connection: keep-alive

user-agent: например какой браузер какой версии используешь

ответ (что не видели):

Access-Control-Allow-Methods: GET, POST, OPTIONS, PUT, DELETE

Access-Control-Allow-Origin: *

Content-Length: 161 -- не всегда используется (вдруг мы что-то большое и непонятное по размеру и не сразу передаем типа видео?)

Location: http://neerc.ifmo.ru/information/index.html

-- куда произошел редирект

ETag: W/"612d09dd-2b03" -- для хэширования, сокращения трафика

HTTP-запрос

```
GET / HTTP/1.1
Host: www.example.com
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: k1=v1; k2=v2
Connection: keep-alive
Referer: https://polygon.codeforces.com/
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
```

Общий вид

Первая строка: `METHOD URL HTTP/Version`

Потом заголовки: `Name: Value`

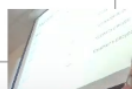
Потом пустая строка

Потом данные (если есть)

referer -- кто инициировал запрос

Методы HTTP-запросов

Метод	Тело	RO	Идемпотент	Семантика
GET	Нет	Да	Да	Прочитать ресурс
POST	Да	Нет	Нет	Отослать ресурс
HEAD	Нет	Да	Да	Получить информацию о ресурсе
PUT	Да	Нет	Да	Установить ресурс
DELETE	Нет	Нет	Да	Удалить ресурс
PATCH	Да	Нет	Нет	Изменить ресурс



то, какая семантика вкладывается в запрос

RO = **read-only**= не меняет (состояние и что либо) сервера и идемпотент самим протоколом никак не проверяются (и не могут), но лучше их как бы да

HTTP-ответ

```
HTTP/1.1 200
Server: nginx
Date: Mon, 07 Sep 2020 17:42:28 GMT
Content-Type: text/html;charset=UTF-8
Connection: close
Set-Cookie: JSESSIONID=FAA2BDB3; HttpOnly
Cache-Control: no-cache, no-store, must-revalidate

<!DOCTYPE html>
<html lang="en">
    ....
</html>
```

Некоторые заголовки

- Общие: Cache-Control, Connection, Date
- Запрос: Accept, Accept-Encoding, Accept-Language, Connection, Cookie, Host, Referer, User-Agent, If-None-Match
- Ответ: Connection, Connection-Encoding, Connection-Type, Transfer-Encoding, Set-Cookie, Location

Параметры cookie

- name, value
- expires, max-age
- path, domain
- secure, httponly, samesite

HTTP Status Codes

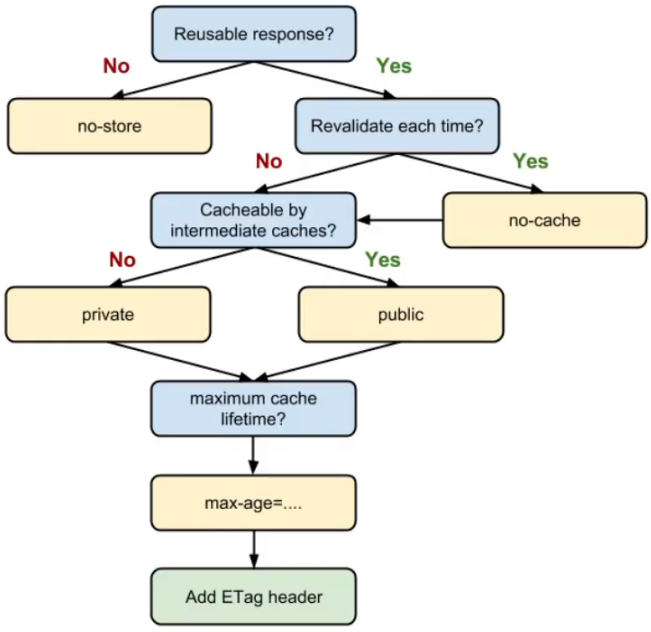
- 1xx — информационные
- 2xx — успех (200 OK, 206 Partial Content)
- 3xx — перенаправление (301 Moved Permanently, 302 Moved Temporarily, 304 Not Modified, 303 See Other, 307 Temporary Redirect, 308 Permanent Redirect)
- 4xx — ошибка клиента (403 Forbidden, 404 Not Found, 405 Method Not Allowed)
- 5xx — ошибка сервера (500 Internal Server Error, 501 Not Implemented, 502 Bad Gateway, 503 Service Unavailable, 504 Gateway Timeout)

-- статусные коды
307, 308 -- браузер обязан при редиректе повторять полностью свой запрос
503 -- пока ниче не работает но так и надо

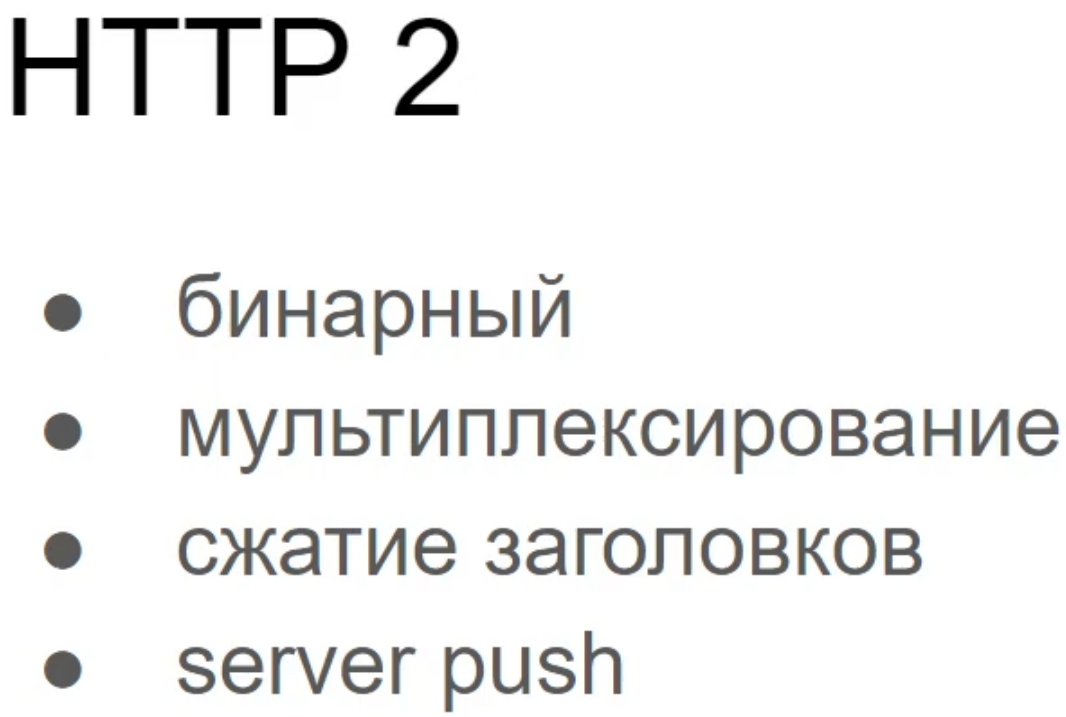
хранящиеся куки отсылаются при всех запросах на данный хост
по сути куки как бы хранят сессию
можно указывать skope? -- где кука применима (например мы хотим чтобы codeforces.com и polygon.codeforces.com были разными)
httponly -- ну типа что javascript код не имеет доступа к этой куке
secure -- запрещаем передавать по http (понятно почему)

cache-control (слайд)

Cache-Control



прокси-кэширование (не на user-agent-е) возможно (в основном?) только при http по понятным причинам



server push -- передача документов в клиент, инициированная сервером?
мультиплексирование -- в рмаках одного tcp соединения параллельная скачка? теперь нет очереди запрос-ответов...

погуглить про:

реверс-прокси, прокси-сервер
https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BA%D1%81%D0%B8-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80#%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5
TCP,
TLS -- протокол обеспечивающий защищенную (шифрованную) передачу данных между узлами в интернете
см сертификаты, центры сертификации

трехфазное рукопожатие
https://ru.wikipedia.org/wiki/TCP#%D0%A3%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0_%D1%81%D0%BE%D0%B5%D0%B4%D0%B8%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F
UDP, DNS
отличия TCP и UDP
keep-alive
ssh, **smtp**

http3

The user initiates a remote shell connection and the target system listens for such connections. With a reverse shell, the roles are opposite. It is the target machine that initiates the connection to the user, and the user's computer listens for incoming connections on a specified port.26 авг. 2019 г.

[

What Is a Reverse Shell - Acunetix

acunetix.com

https://www.acunetix.com › blog › web-security-zone
](https://www.acunetix.com/blog/web-security-zone/what-is-reverse-shell/)

Искать: What is the difference between remote shell and reverse shell?