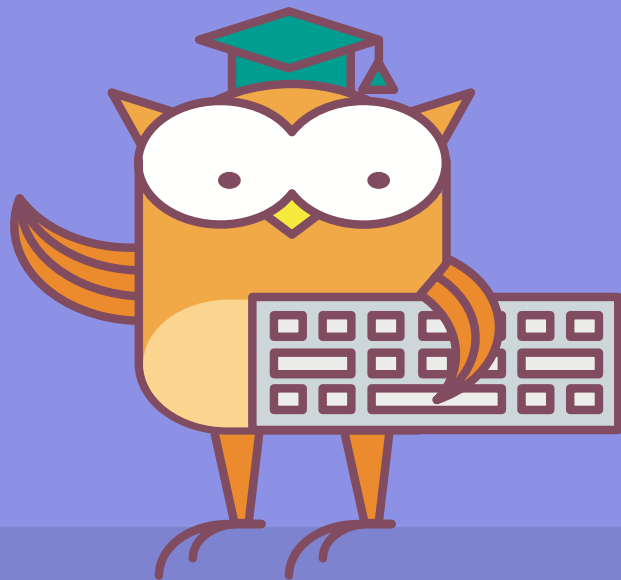




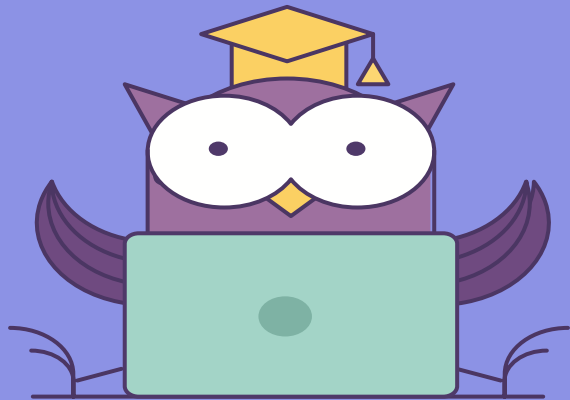
ОНЛАЙН-ОБРАЗОВАНИЕ


O T U S

# Web сервера



# Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!  
Ставьте  если все хорошо

# 01

## World Wide Web

1969 ARPANET Пол Бэран  
1983 ARPANET -> Internet  
1989 WWW Тим Бернерс-Ли  
1994 W3C

HTTP (HyperText Transport Protocol) - протокол передачи данных

- HTML (HyperText Markup Language) - язык разметки
- URI (Uniform Resource Identifier) - унифицированный идентификатор ресурса
- URL - Uniform Resource Locator, помогает найти какой либо ресурс
- URN - Uniform Resource Name, помогает этот ресурс идентифицировать

# 02

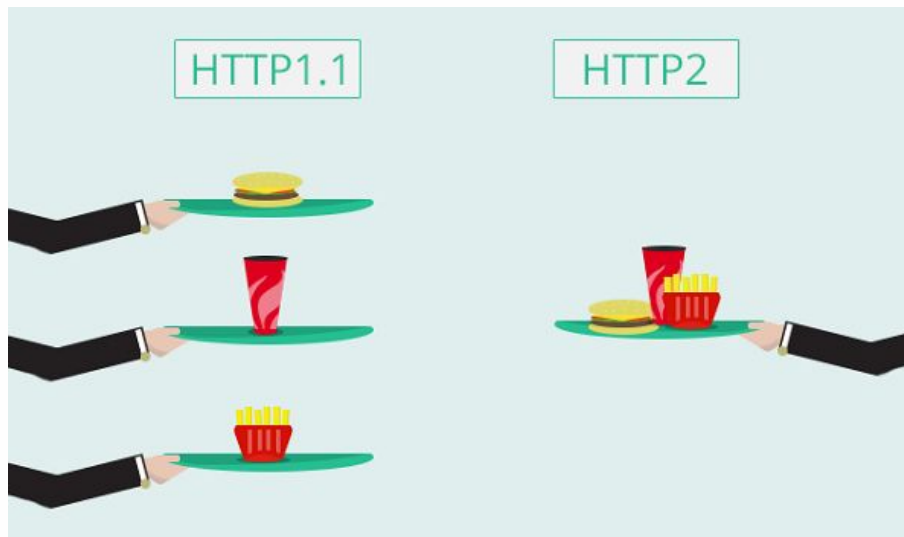
## HTTP

- Протокол предназначенный для передачи данных. Изначально только гипертекстовых.
- Определяет, как взаимодействуют между собой клиент и сервер, как запрашивается и передается контент по интернету.
- Для идентификации ресурсов HTTP использует глобальные URI



- 0.9 - первая версия, представленная в 1991 году. Один запрос - GET.
- 1.0 - 1996 год.
- 1.1 - 1999 год. Появился keep-alive механизм, заголовки.
- 2.0 - 2015 год. Абсолютно бинарный, мультиплексирование.
- 3.0 - 201X???. Обещают научить протокол работать по UDP

- Использует около 30% всех сайтов.  
<https://w3techs.com/technologies/details/ce-http2/all/all>
- Все свойства 1.1 перетекли в 2.0
- Полностью бинарный
- Появилось мультиплексирование запросов <http://www.http2demo.io/>
- Server Push



Основная операция над ресурсом

GET - получить содержимое указанного ресурса

HEAD - получить только заголовки

POST - отправить данные

DELETE - удалить указанный ресурс

OPTIONS - определить параметры сервера

HEAD / HTTP/1.0

host: otus.ru

OPTIONS / HTTP/1.0

host: otus.ru

Метаданные сопровождающие “общение” по протоколу HTTP.

- connection: keep-alive (close)
- cache-control: no-cache, max-age, no-store
- accept: text/html, image/\*
- user-agent: <product> / <product-version> <comment>

- Код ответа (состояния) HTTP показывает, был ли успешно выполнен определённый HTTP запрос.
- Коды сгруппированы в 5 классов:
- 1xx: Informational
- 2xx: Success
- 3xx: Redirection
- 4xx: Client Error
- 5xx: Server Error

100: Continue

200: OK

204: No Content

301,302: Moved Permanently  
(Temporary)

403: Forbidden

500: Internal Error

504: Gateway Timeout

```
vagrant@otus:~$ telnet 192.168.11.246 80
```

```
Trying 192.168.11.246...
```

```
Connected to 192.168.11.246.
```

```
Escape character is '^['.
```

```
GET / HTTP/1.1
```

```
host:192.168.11.246
```

```
HTTP/1.1 200 OK
```

```
Date: Tue, 16 Oct 2018 11:40:21 GMT
```

```
Server: Apache/2.4.18 (Ubuntu)
```

```
Last-Modified: Mon, 15 Oct 2018 10:06:47 GMT
```

```
ETag: "2c39-57841985a7304"
```

```
Accept-Ranges: bytes
```

```
Content-Length: 11321
```

```
Vary: Accept-Encoding
```

```
Content-Type: text/html
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

- Заголовки ответа сервера:

Expires - время, после которого контент будет неактуальным

Etag - метка контента

Cache-Control: max-age=, no-cache , no-store

- Заголовки запроса:

If-Modified-Since - дай, если изменилось, возвращает 200 или 304

If-None-Match: Etag - дай, если есть у ресурса такая метка

<https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching?hl=ru>

- Избавляемся от необходимости терять время и ресурсы CPU на установку соединения.
- Используется между клиентом и сервером, но с осторожностью между сервером и бекендом.



**Content-Length** - заголовок, содержащий размер данных, начиная с первого байта после `\r\n` заголовков. Используется и сервером, и клиентом.

## **Content-Encoding chunked**

Используется для передачи данных с неизвестным конечным размером. При этом Content-Length не используется, но в начале каждого чанка в шестнадцатеричном указывается размер чанка, сопровождающегося пустой строкой. Конец чанка так-же отмечается пустой строкой.

## **gzip, compress, deflate**

Указывает способ сжатия

Клиент может со своей стороны указать, какие схемы он поддерживает с помощью заголовка Accept-Encoding, например Accept-Encoding: \*

Вместо данных иногда в ответ можно получить код 301/302 и заголовок Location с указанием URL  
Конфигурируется на стороне сервера.

Способ обработки нескольких доменных имён на одном сервере.

- Размещение более чем одного сайта на одном хосте
- Name-based: несколько сайтов на одном IP адресе
- IP-based: отдельный адрес для каждого сайта

# Пример конфига IP-based

```
# Ensure that Apache listens on port 80
Listen 80
```

```
<VirtualHost 192.168.0.1:80>
DocumentRoot /groups/smallco/www
ServerName smallco.example.com
...
</VirtualHost>
```

```
<VirtualHost 192.168.0.2:80>
DocumentRoot /groups/baygroup/www
ServerName baygroup.example.com
...
</VirtualHost>
```

# Пример конфига Name-based

```
# Ensure that Apache listens on port 80
Listen 80
<VirtualHost *:80>
    DocumentRoot "/www/example1"
    ServerName www.example.com

    # Other directives here
</VirtualHost>

<VirtualHost *:80>
    DocumentRoot "/www/example2"
    ServerName www.example.org

    # Other directives here
</VirtualHost>
```

```
server {  
    listen ip_address:port;  
    server_name ololo.trololo;  
}
```

```
server {  
    listen 192.168.1.55:8080;  
    server_name otus.ru;  
}
```

# 03

## NGINX

- Масштабирование
- Простое распределение нагрузки

```
location / {  
    proxy_pass http://IP-or-domain-name;  
    proxy_set_header Host $host;  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    proxy_set_header X-Real-IP $remote_addr;  
}
```



- round-robin
- least-connected
- ip-hash

```
http {  
    upstream backend {  
        server backend1.somesite.com weight=10;  
        server backend2.somesite.com weight=5;  
        server backend3.somesite.com;  
        server 192.0.0.1 backup;  
    }  
}
```

```
server {  
    location / {  
        proxy_pass http://backend;  
    }  
}
```

```
http {  
    server {  
        listen 80;  
        server_name trololo.ololo;  
        root /var/www/html;  
        try_files /thumb.png /greet;  
        location /greet {  
            return 200 "Hello User";  
        }  
        try_files $uri /cat.png /greet /friendly_404;  
        location /friendly_404 {  
            return 404 "Page not found";  
        }  
    }  
}
```

```
http {  
    server {  
        listen 80;  
        server_name ololo_trolo;  
        root /var/www/html;  
        location / {  
            return 200 "Hello! This is main page!";  
        }  
        location /logo {  
            return 307 /thumb.png;    }  
        rewrite ^/user/\w+ /greet;  
        location /greet {  
            return 200 "Hello User";  
        }  
        rewrite ^/user/(\w+) /greet/$1;  
    }  
}
```

Приоритеты префиксов (по порядку совпадения):

1. = Exact Match (Полное совпадение)
2. ^~ Preferential Prefix (Префикс преимущества)
3. ~ & \*~ Regex Match (Регулярные выражения)
4. без модификатора Prefix match

```
http {  
  
    server {  
  
        listen 80;  
        server_name trololo.ololo;  
        root /var/www/html;  
        index index.php index.html;  
        location / {  
            try_files $uri $uri/ =404;  
        }  
  
        location ~\.php$ {  
            include fastcgi.conf;  
            fastcgi_pass unix:/var/run/php/php7.1-fpm.sock;  
            fastcgi_cache_path /var/cache/nginx levels=1:2 keys_zone=static_cache:100m inactive=120m  
max_size=500M;  
        }  
    }  
}
```

```
server {  
    listen 80 default_server;  
  
    server_name trololo.ololo;;  
  
    return 301 https://$host$request_uri;  
}
```

```
/etc/nginx/nginx.conf
```

```
events {  
    worker_connections 4096;  
    use epoll;  
    multi_accept on;  
}
```

```
server {  
  
    gzip on;  
    gzip_static on;  
    gzip_disable "msie6";  
    gzip_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss  
text/javascript application/javascript;  
    gzip_comp_level 5;  
    gzip_buffers 16 8k;  
    gzip_http_version 1.1;  
    gzip_min_length 256;  
  
}
```



```
server {
```

```
    listen 80 ::80 443 1223;
```

```
}
```

Для location / поведение будет одинаковым.

```
server {  
    root /var/www  
    location /images/ {  
    }  
}
```

/var/www/img/1.jpg

```
server {  
    location /images/ {  
        alias /var/www  
    }  
}
```

/var/www/1.jpg

```
server {  
    location / {  
        root /data/www;  
    }  
  
    location /images/ {  
        root /data;  
    }  
  
    location /soft/ {  
        root /soft;  
    }  
  
    location /meta/ {  
        root /meta;  
    }  
}
```

# 04

## Let`s Encrypt + Certbot

HyperText Transport Protocol Secure

SSL/TLS - по сути это обертка для HTTP.

При установке безопасного соединения по HTTPS ваш компьютер и сервер сначала выбирают общий секретный ключ, а затем обмениваются информацией, шифруя её с помощью этого ключа.

Однако для полной надёжности ей кое-чего не хватает: гарантии того, что ваш собеседник именно тот, за кого себя выдаёт.

1) Лицо, которому он выдан, действительно существует и 2) Оно управляет сервером, который указан в сертификате

- `yum install epel-release -y && yum install certbot-nginx -y`
- `certbot --nginx -d example.com -d www.example.com (--dry-run certonly)`

```
server {  
    listen 443 ssl default_server;  
    server_name my-domain;  
  
    ssl_certificate /etc/letsencrypt/live/my-domain/fullchain.pem;  
    ssl_certificate_key /etc/letsencrypt/live/my-domain/privkey.pem;  
  
}
```

```
openssl x509 -noout -modulus -in server.crt| openssl md5  
openssl rsa -noout -modulus -in server.key| openssl md5
```

<https://nginxconfig.io/> - онлайн генератор конфигураций

<https://ruhighload.com/search?q=nginx&p=1> - много полезных статей про Nginx и его настройку

<https://tech.yandex.ru/tank/> - инструмент для нагрузочного тестирования

<https://www.sslforfree.com/> - бесплатные ssl сертификаты от lets encrypt

<https://www.ssllabs.com/> - тестирование установленных сертификатов

**Спасибо  
за внимание!**







**Дроздецкий Владимир**