Web basics

Stanislav Protasov

Agenda

- Internet in a nutshell
- Web 1.0 … 4.0
  - Idea of the Web
  - Web transport and formats
  - Browsers and DOM
- Crawling basics
  - Robots.txt and sitemap
  - Terms and Conditions
  - APIs

Ok, Google, what is internet?

IP protocol and address

213.159.212.4

192.34.57.61

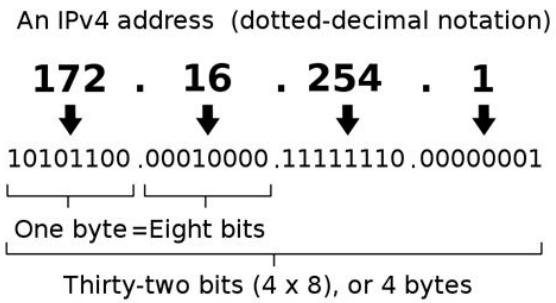
0xC022393D

3223468349

meduza.io:@3223468349

Construct an email which looks like whatever, by leads to mail.ru:

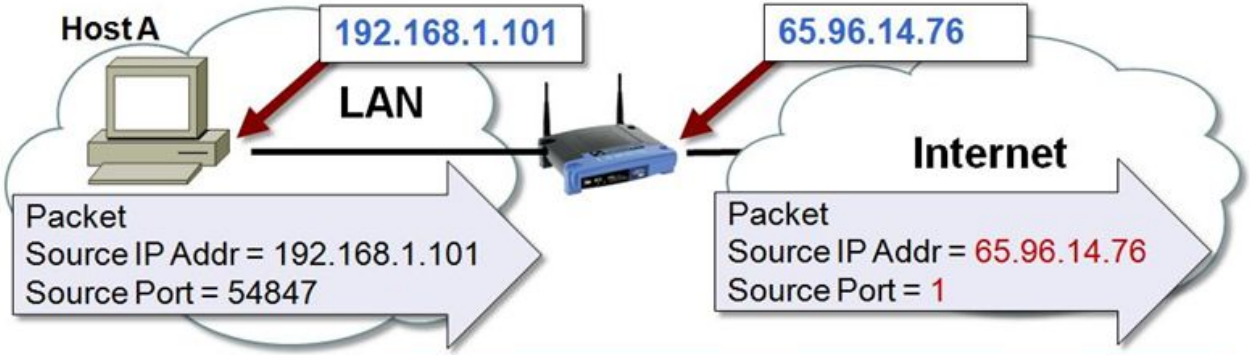
1. Obtain IP address of mail.ru
2. Convert IP to hex
3. Create **`whatever:@ip`** url



192.0.2.235
0xC0.0x00.0x02.0xEB
0300.0000.0002.0353
0xC00002EB
3221226219
030000001353

https://myip.ru/

IP != machine



NAT Translation Table

	Local IP Address	Source Port #		Internet IP Address	Source Port #
process X, Host A	192.168.1.101	54,847	=	65.96.14.76	1
Host B	192.168.1.103	24,123	=	65.96.14.76	2
process Y, Host A	192.168.1.101	42,156	=	65.96.14.76	3
Host C	192.168.1.102	33,543	=	65.96.14.76	4

Try on your machine

```
W> ipconfig.exe | findstr IPv
```

```
L/M> ifconfig | grep inet
```

How many addresses do you have?

Country and IP

Using IP is one of pretty (surprisingly) reliable ways of geo location.

- GeoIP 99% for country detection, 95% for city detection

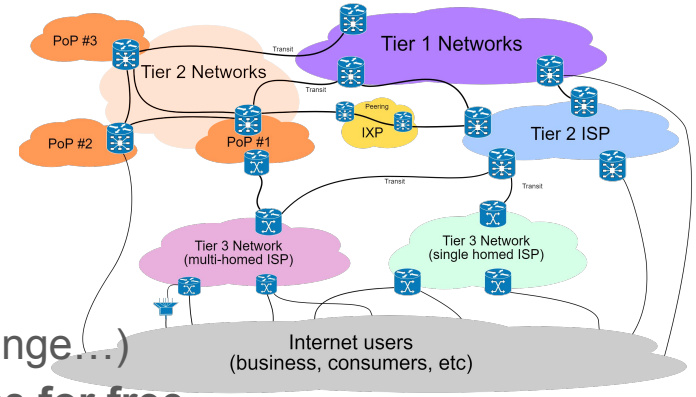
To detect country you need a database.

Service processes IP of the last visible node in a chain, so

- NAT
- Proxy
- Turbo mode

will fool the service.

Tier-X operators (podcast in rus)



- **Tier-1 operators**
  - (e.g. Rostelecom*, MTS*, ... for Russia, AT&T, Orange…)
  - These networks exchange traffic on peer conditions **for free**
  - Altogether can be considered as a **backbone** of the Internet
- **Tier-2 operators** have partially free peering with some segments but paid transit to other segments
- **Tier-3** have only paid access to internet
- Takeaways:
  - Local operators are **subjects of regulations** (RKN)
  - Backup channels, VPNs and so on are important for service quality
  - Think twice about where to attach/rent/build your DC
  - Small networks share the same IP-address for other networks
  - **IPv4 is over**

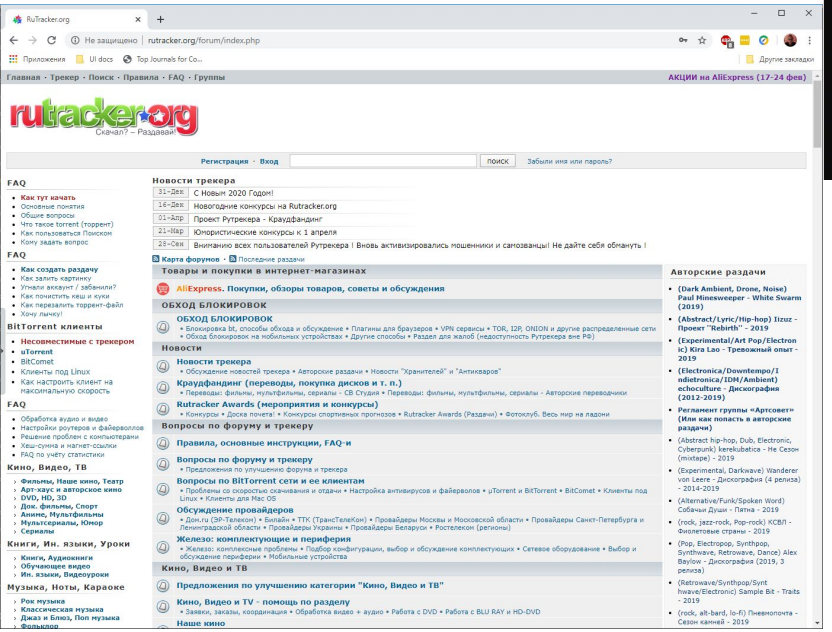
МАГИСТРАЛЬНАЯ СЕТЬ РОСТЕЛЕКОМ

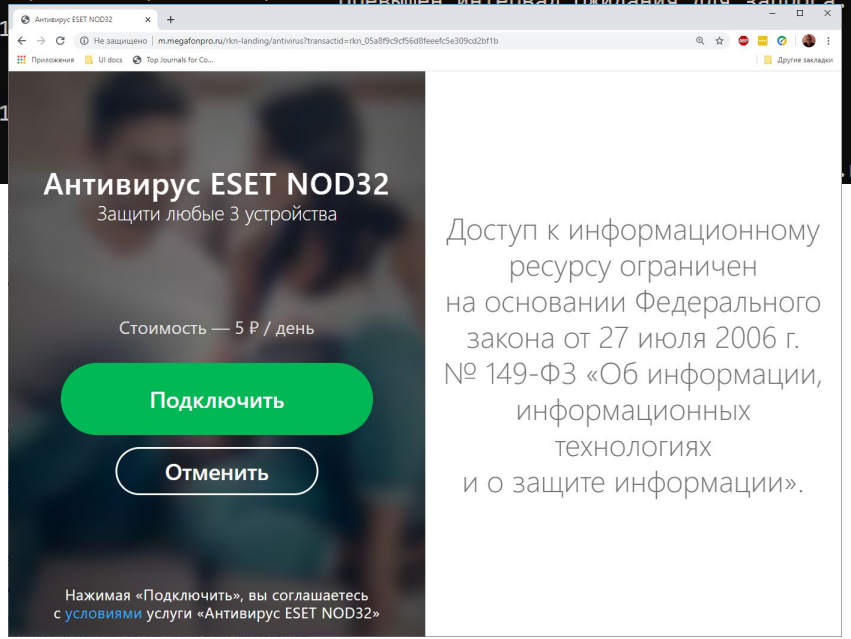
```
C:\Users\s.protasov>tracert -h 50 -w 1 rutracker.org

Трассировка маршрута к rutracker.org [195.82.146.214]
с максимальным числом прыжков 50:

  1     2 ms     1 ms     1 ms  10.91.64.1
  2    22 ms    20 ms    39 ms  87.226.217.249
  3     *       17 ms    18 ms  213.59.212.235
  4    48 ms    48 ms    48 ms  188.254.83.102
  5     *        *        *     Превышен интервал ожидания
```

RTK

```
C:\Users\s.protasov>tracert -h 50 -w 1 rutracker.org

Трассировка маршрута к rutracker.org [195.82.146.214]
с максимальным числом прыжков 50:

  1     *        *        3 ms  192.168.43.116
  2     *        *        *     Превышен интервал ожидания для запроса.
  3     *        *        *     Превышен интервал ожидания для запроса.
  4     *        *        *     Превышен интервал ожидания для запроса.
  5     *        *        *     Превышен интервал ожидания для запроса.
  6     *        *        *     Превышен интервал ожидания для запроса.
  7     *        *        *     Превышен интервал ожидания для запроса.
  8    57 ms    38 ms    47 ms  37.29.5.65
  9     *        *        *     Превышен интервал ожидания для запроса.
 10     *        *        *     Превышен интервал ожидания для запроса.
 11     *        *        *     Превышен интервал ожидания для запроса.
 12  1
 13
 14
 15  1
 16
 17                                                                  .net
```

Megafon





Доступ к информационному ресурсу ограничен на основании Федерального закона от 27 июля 2006 г. № 149-ФЗ «Об информации, информационных технологиях и о защите информации».

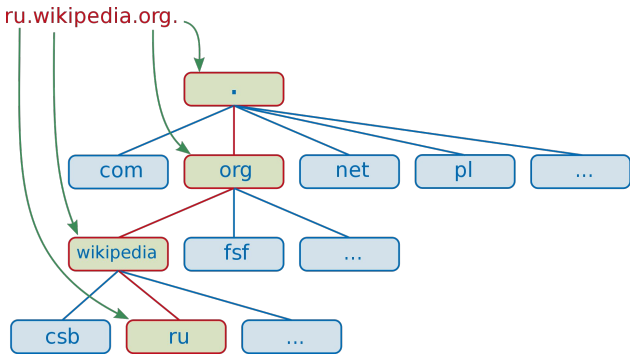
Try on your machine

```
W: tracert -w 1 ya.ru
```

```
L/M: traceroute -w 1 ya.ru
```

Where is provider, where is Yandex?

Name services



DNS — domain name system, allows to operate human-readable names instead of addresses

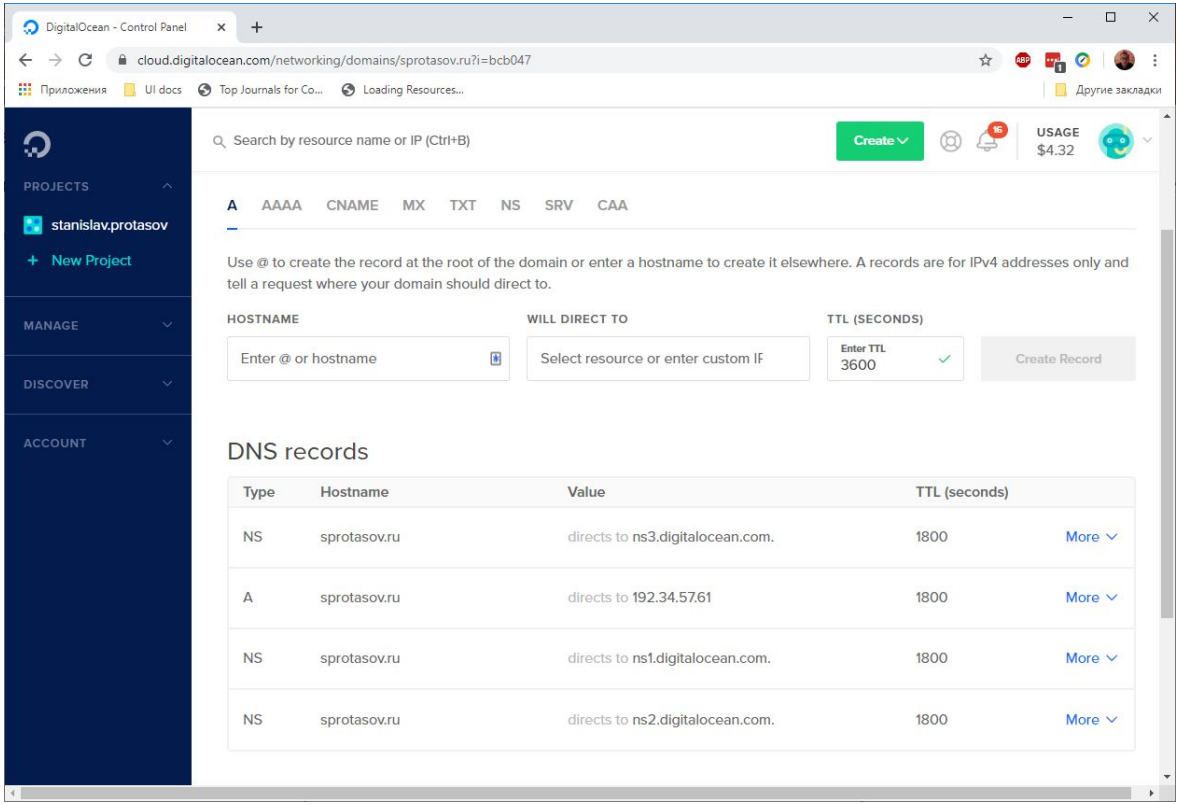
There are 13 core [Root Servers]() ([`a..m].root-servers.net`) responsible for the Internet. Lower level responsible for domains, subdomains, …

DNS supports forward (domain → IP) and reverse (IP → domain) requests

```
nslookup sprotasov.ru

nslookup 192.34.57.61

nslookup code-test.ru
```

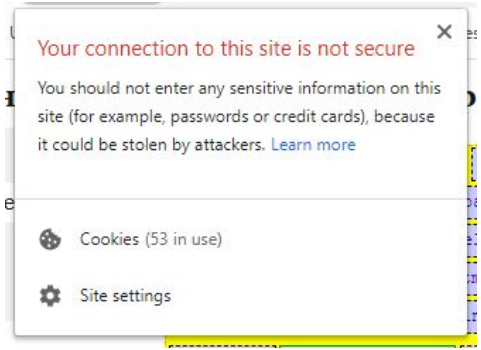
```
root@simpletrack:~# ping -c 1 yandex.ru
PING yandex.ru (77.88.55.80) 56(84) bytes of data.
64 bytes from yandex.ru (77.88.55.80): icmp_seq=1 ttl=244 time=116 ms

--- yandex.ru ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 116.697/116.697/116.697/0.000 ms
root@simpletrack:~# logout
Connection to sprotasov.ru closed.
(base) stranger@sprotasovn:~$ ping yandex.ru
PING yandex.ru (213.180.193.56) 56(84) bytes of data.
64 bytes from familysearch.yandex.ru (213.180.193.56): icmp_seq=1 ttl=56
```

Resource record



Short quiz. Test yourself

1. One website opens for me, but another — does not. Same for my neighbour in office. Why?
2. One website opens for me, but another — does not. Both work for my neighbour in office. Why?
3. I was crawling the data from marketplace with `requests` lib, it worked for 2 days, but today crawler throws exceptions… Why?
4. I clicked a link starting with *google.com* in my email. I know that all google pages are secured, but browser says that Why?



Before we continue - Internet

1. **Internet and IP**. **IP** manages (routes) how data is flowing from one machine (e.g. server) to another (e.g. smartphone). Providers are working on this level
   a. Tools: `tracert, ping, ifconfig/ipconfig`
   b. **TCP** manages how to transfer more than one packet of date preserving order and integrity
      i. Tools: `nc, telnet`
2. **DNS**. Assigning string names to IP-addresses allow to establish many-to-many relations, thus, make infrastructure faster and reliable. Domain names are objects of legal regulations (whois)
   a. Tools: `nslookup, ping`

Web and HTTP

~~KEY~~ BUZZ WORDS

Internet – network for transferring information among devices

WWW (Web) – graph of documents (hypertext), placed at web-servers, that are connected to Internet

Hypertext – text, that contain references to other texts

HTML (hypertext markup language) – standard of hypertext for Internet

Web 2.0 – everything, that is beyond static HTML documents: social networks, blogs, video-hostings, internet-marketing, web application and services. Service-oriented network.

Web 3.0 (*semantic web*) – graph of machine-readable, semantically rich documents. Content-oriented network.

HYPERTEXT EVOLUTION

Web 1.0	Web 2.0	Web 2.0, 3.0
HTML – Subset of SGML (markup language)	**xHTML** – fusion of HTML tags and XML standard	**[x]HTML5** – valuable layout changes, semantic tags were added
SGML parsers	**XML** parsers	**HTML5/XML** parsers
For presentation in browsers	For displaying interactive and media content	For creation of web-applications that support semantic markup

URI VS URL

Uniform resource identifier (URI) –machine-readable text identifier of the resource, created according specific rules

URI common syntax: *scheme:scheme-specific-part*

Uniform resource locator (URL) – subset of URI, describes location and way (protocol) to access object in the Internet

`http://www.mail.ru/`

Uniform resource name (URN) – subset of URI, identifies object, but does not locate it

`urn:isbn:0451450523` `or`
Magnet links: `magnet:?xt=urn:btih:c12fe1c06bba254a9dc9f519b335aa7c1367a88a`

URL SYNTAX

Examples:

- https://mail.google.com/
- ftp://root:qwerty@ftp.example.com/
- wss://server.name:443/method/name
- http://sprotasov.ru/index.html#author:Aleksandr%20Buyanov

Before we continue - Web

1. **Hypertext, HTML and HTTP**. Hypertext is an approach to represent **linked** documents (altogether = The Web). xHTML5 is a de-facto standard. HTTP - a protocol for transferring [hyper]text data, or text-encoded media (base64). Defines methods (GET, POST, ...), status codes (200, 403, 502), headers (metainformation), sessions (1.1+). Works over TCP (means one HTTP message can be bigger that 1 IP frame).
   a. Tools: `telnet, curl, wget, postman`
2. **URI ⊃ URL**. URL is a standard way to define together:
   a. Where is the document (domain + port + path)
   b. How to access the document (protocol, credentials)

HTTP

HTTP (hypertext transfer protocol) – application (7) level protocol to deliver text data. Created to transfer hypertext. Provide communication between *client* (usually browser) and *server* (web-server) using client requests and server responses.

HTTP v1.0 – does not support using single TCP session for multiple requests. Supports following client request methods:

- GET – get content from the server
- HEAD – get only header from the server without content ("what to expect")
- POST – sent data to the server

HTTP v1.1. – supports also PUT, DELETE, TRACE, OPTIONS, CONNECT, PATCH

HTTP/2 - SDPY (Google) based update. Binary. Header compression, Server pushes, conveyor requests, request multiplexing over single TCP

HTTP HEADER

HTTP request =
Request URI + HOST + [[headers]] + <empty_line> + body

```
POST /index.html HTTP/1.1
HOST: example.org:8080
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml
...

param1=value1&param2=value2
```

HTTP response=

Response code + headers + <empty_line> + body

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 3123
...


<html>...
```

Demo with telnet

```
o sprotasov.ru 80
HEAD / HTTP/1.1
host:sprotasov.ru

HEAD / HTTP/1.1
host:code-test.ru

GET / HTTP/1.1
host:code-test.ru
```

IMPORTANT HEADERS

REQUEST

Accept, Accept-Charset, Accept-Encoding – formats, that your browser understands (text/plain, application/xml), encodings (utf-8) and supported compression algorithms (gzip, deflate)

Authorization – header that stores authentication type, credentials/keys,…

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

Content-Length – request body length (same for response)

User-Agent – browser and operating system

Upgrade – request to change communication protocol (`Upgrade:websocket`)

RESPONSE

Cache-Control – time to store document in a browser cache

Content-Encoding, Content-Language, Content-Type – content characteristics

Browser fingerprint: domain I see first time

Example Domain

Not secure | example.com

Apps | UI docs | Top Journals for Co... | Loading Resources... | Other bookmarks

Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Audits

Preserve log | Disable cache | Online

Filter | Hide data URLs | All | XHR | JS | CSS | Img | Media | Font | Doc | WS | Manifest | Other

Only show requests with SameSite issues

100 ms | 200 ms | 300 ms | 400 ms | 500 ms | 600 ms | 700 ms | 800 ms | 900 ms | 1000 ms

Name

example.com

favicon.ico

2 requests | 1.4 KB tr

Headers | Preview | Response | Initiator | Timing

```
GET / HTTP/1.1
Host: example.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.39
87.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/s
igned-exchange;v=b3;q=0.9
```

Console | What's New

RESPONSE CODES

- 1xx – information
  - 101 - Switching Protocols
- 2xx – success
  - 200 – OK
  - 201 – created (new resource)
  - 206 – partial content
- 3xx – redirection (specified header Location: addr)
  - 301 – moved permanently
  - 304 – not modified
- 4xx – client-side error
  - 401 – unauthorized
  - 404 – not found
- 5xx – server-side error
  - 503 – server unavailable

REQUESTS AND SYSTEM STATE: SAFE AND IDEMPOTENT REQUESTS

Safe request does not change server/object state. For getting some information

Idempotent request if you make 2 or more identical requests, second and other requests do not change server/object state
(*F(state) == F(F(state))*)

	SAFE	NOT SAFE
IDEMPOTENT	HEAD, GET, OPTIONS, TRACE	PUT, DELETE
NON-IDEMPOTENT	--	POST, PATCH

PARAMETERS

GET params:

- http://server.name/path?param1=value1&param2=value2

POST params:

- POST /path HTTP/1.1

  param1=value1&param2=value2

COOKIES

- Cookies – small **drive space** to store data sent by server to browser. Max – 4KB
- We need cookies for **stateful services** (e-shop cart, etc) or for storing session keys
- Cookies have life period and are **sent** to server with **each request**

COOKIES EXAMPLES

```
GET /index.html HTTP/1.1
Host: www.example.org
```

browser --------→ **server**

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT

(content of page)
```

browser ←-------- **server**

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name=value; name2=value2
Accept: */*
```

browser --------→ **server**

FTP

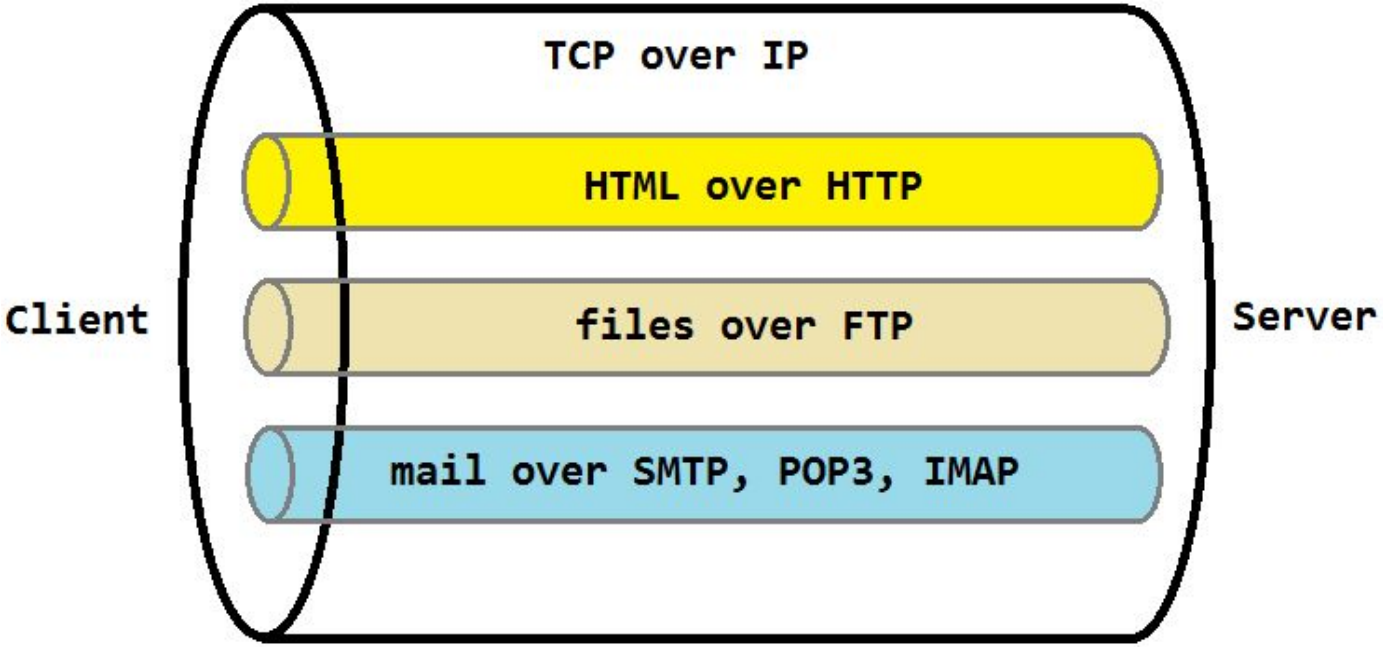
FTP (file transfer protocol), 1971. Consists of greeting session and request session (VERB param [params]\015\012), and server responses

Client:		Connects @server:21
Server:	220 Hello,...	
Client:	USER MB1234	
Server:	331 Password required to access user account MB1234.	
Client:	PASS QXJ4Z2AF	PLAIN TEXT
Server:	230 Logged in.	
Client:	CWD Bills	Change directory to "Bills."
Server:	250 "/home/MB1234/Bills" is new working directory.	
Client:	PORT 192,168,1,2,7,138	accepts data @client:1930 [=7*256 + 138]
Server:	200 PORT command successful.	
Client:	LIST	Send the list of files in "Bills."
Server:	150 Opening ASCII mode data connection for /bin/ls.	server connects out from its port 20 to port client:1930
Server:	226 Listing completed.	succeeded
Client:	PORT 192,168,1,2,7,139	
Server:	200 PORT command successful.	
Client:	RETR Yoyodyne.TXT	Download "Yoyodyne.TXT."
Server:	150 Opening ASCII mode data connection for Yoyodyne.TXT	
Server:	226 Transfer completed.	succeeded
Client:	QUIT	
Server:	221 Goodbye.	

EMAIL: SMTP, IMAP, POP3

- **SMTP** (simple mail transfer protocol, @:25) – for transferring messages between servers and for server-client communication. FTP's brother.
- **POP3** (post office protocol v3, @:110) – standard protocol for client to get messages from server
- **IMAP** (internet message access protocol, @:143) – standard protocol for client to get messages from server; has sending implementation (considered bad), keeps session, supports multiple clients for 1 mailbox.

OVERVIEW


TCP over IP
HTML over HTTP
Client
files over FTP
Server
mail over SMTP, POP3, IMAP

Web security (client side)

TERMS

Identification – assigning labels (IDs) to objects, as long as process of comparing one label with the list

<u>Authentication</u> – procedure of checking authenticity, proving match between ID and object. We can authenticate user (ID + password), machine, document (digital signature). Can be multi-factor, one-way, both-way

Authorization – granting access to perform some action

FUNCTION OF HTTP-AUTHENTICATION

- Limiting access by means of HTTP protocol
  - Rare for sites. Most sites use forms-based authentication
  - Common for **services** and **APIs**. (access not via browser UI, but server or ajax code)

COMMON FACTS ABOUT AUTHENTICATION

If server returns 401, this means it wants to authenticate you. Server must send WWW-Authenticate header to you.

```
HTTP/1.0 401 Unauthorized
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 58
Content-Type: text/html
Expires: -1
Server: Microsoft-IIS/8.0
WWW-Authenticate: Basic realm="area to be accessed"
```

BASIC AUTHENTICATION

Easiest way to setup authentication

```
GET /sometail.aspx HTTP/1.1
Host: somehost
Authorization: Basic bG9naW46cGFzc3cwcmQ=
```

where

“bG9naW46cGFzc3cwcmQ=” == base64(“login:passw0rd”)

NB:

- Login and password are not secured in fact! Only way to use – over HTTPS
- You can send this without challenge
- With each request

DIGEST AUTHENTICATION

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="testrealm@host.com",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
opaque="5ccc069c403ebaf9f0171e9517f40e41"

… … …

Authorization: Digest username="Mufasa",
         realm="testrealm@host.com",
         nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
         uri="/dir/index.html",
         response="e966c932a9242554e42c8ee200cec7f6",
         opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

$$\mathrm{HA1} = \mathrm{MD5}\left(\mathrm{A1}\right) = \mathrm{MD5}\left(\mathrm{username} : \mathrm{realm} : \mathrm{password}\right)$$

$$\mathrm{HA2} = \mathrm{MD5}\left(\mathrm{A2}\right) = \mathrm{MD5}\left(\mathrm{method} : \mathrm{digestURI}\right)$$

$$\mathrm{response} = \mathrm{MD5}\left(\mathrm{HA1} : \mathrm{nonce} : \mathrm{HA2}\right)$$

RFC 2617:

$$\mathrm{HA1} = \mathrm{MD5}\left(\mathrm{A1}\right) = \mathrm{MD5}\left(\mathrm{username} : \mathrm{realm} : \mathrm{password}\right)$$

$$\mathrm{HA2} = \mathrm{MD5}\left(\mathrm{A2}\right) = \mathrm{MD5}\left(\mathrm{method} : \mathrm{digestURI}\right)$$

Если значение директивы QOP равно «auth-int», то HA2 равняется:

$$\mathrm{HA2} = \mathrm{MD5}\left(\mathrm{A2}\right) = \mathrm{MD5}\left(\mathrm{method} : \mathrm{digestURI} : \mathrm{MD5}(\mathit{entityBody})\right)$$

Если значение директивы QOP равно «auth» или «auth-int»:

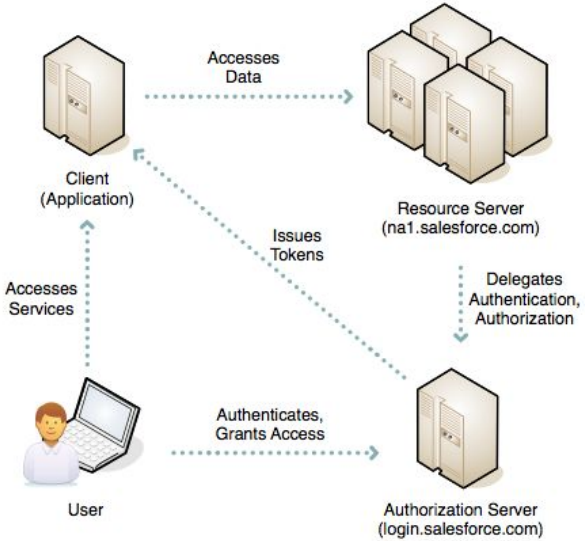
$$\mathrm{response} = \mathrm{MD5}\left(\mathrm{HA1} : \mathrm{nonce} : \mathrm{nonceCount} : \mathrm{clientNonce} : \mathrm{qop} : \mathrm{HA2}\right)$$

Если директива QOP не определена, то ответ вычисляется так:

$$\mathrm{response} = \mathrm{MD5}\left(\mathrm{HA1} : \mathrm{nonce} : \mathrm{HA2}\right)$$

OAUTH

- Authenticates application on behalf of user (or anonymously)
- Based on Access Tokens



FORMS AUTHENTICATION

- Not a part of HTTP protocol
- Based on HTML <FORM>-tag and request parameters mechanism

```
<form action="Default.aspx" method="get">
    Login: <input type="text" name="username" />
    <br/><br/>
    Password: <input type="password" name="password" />
    <br/><br/>
    <input  type="submit"
            value="Log me in"/>
</form>
```

request destination

request method (get/post)

"username=..." param will be created

"password=..." param will be created

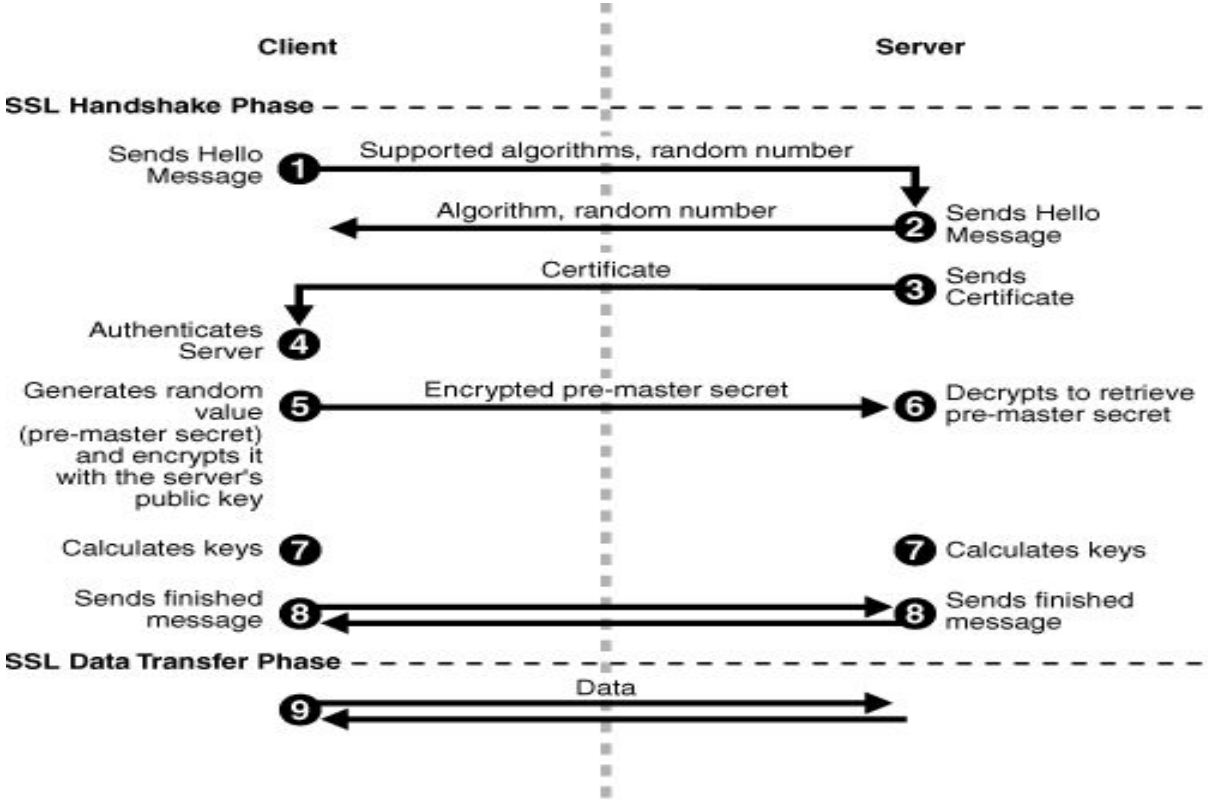
covers field with "stars"

button that triggers submission (request)

HTTP SECURE

- **HTTPS** = HTTP over SSL/TLS
  - SSL – protocol with asymmetric cryptography and symmetric encoding
  - TLS = SSL v3
- **HTTP** (FTP, telnet) work transparently over SSL/TLS
  - Firstly client's application (browser) performs "handshake".
  - Then the channel is created and data is sent over this channel using standard protocol (e.g. HTTP)

HTTPS HANDSHAKE



CERTIFICATES

Digital certificate – electronic document (file), ensuring that public key belongs to bearer. Certificate must be signed by certification authority.

- Mandatory cert parts:
  - resource ID (**Subject**)
  - **public** key
  - certification authority (**Issuer**)
- Optional cert parts
  - private key
  - usage restrictions

Before we continue - Security

1. HTTP supports **Basic and Digest *authentication*** *of a user* from the box (defined in standard). Mostly used for service-to-service interaction. **Data is still plaintext**.
2. **OAuth** is a new way to grant access to the service. Access to APPLICATION on behalf of a user. 3-sided:
   a. **User** passes login/pass to **authentication service**
   b. **Auth service** issues a token for **an app** to act on behalf of a user.
   c. **Application** uses token to interact with **a service**.

   Data is still plaintext.

3. **TLS/SSL** is used to establish **secure channel over TCP**. Uses asymmetrics keys to build end-to-end encrypted communication (session save overhead!). Certificates are used as containers for keys + validation tool. HTTP over SSL = https://….:443/….

HTML and DOM

LANGUAGE TYPOLOGY

- SGML – meta language for description of markup languages. It defines
  - Allowed symbol alphabet (SGML declaration)
  - DTD (data type definition) – markup syntax + semantics
- XML – simplified subset of SGML
  - XML Schema languages (DTD, W3C XSD)
- HTML – <u>application</u> of SGML (initially)
  - xHTML – application of XML

SCHEMA LANGUAGES

- **DTD, XML Schema** – define document structure and node constraints
- Used for
  - Defining semantic rules (for values, number of children…)
  - Document pre-validation
- Document can be:
  - **type-valid** – meet all DTD constraints
  - **tag-valid** – meet all [SGML/XML] **tag** constraints

RSS - Rich Site Summary

sprotasov.ru/podcast/rss.xml

Not secure | sprotasov.ru/podcast/rss.xml

Apps | UI docs | Top Journals for Co... | Loading Resources... | https://en.wikipedia... | Other bookmarks

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<rss xmlns:atom="http://www.w3.org/2005/Atom"
 xmlns:content="http://purl.org/rss/1.0/modules/content/"
 xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd"
 version="2.0">
  ▼<channel>
     <atom:link href="http://sprotasov.ru/podcast/rss.xml"
     rel="self" type="application/rss+xml"/>

     <title>Происхождение видов</title>

    ▼<description>
       Подкаст о том, как и зачем люди создают компьютеры,
```

DTD EXAMPLE

```
<!ELEMENT people_list (person*)>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA) >
<!ELEMENT birthdate (#PCDATA) >
<!ELEMENT gender (#PCDATA) >
<!ELEMENT socialsecuritynumber (#PCDATA) >
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
   <person>
      <name>
         Fred Bloggs
      </name>
      <birthdate>
         27/11/2008
      </birthdate>
      <gender>
         Male
      </gender>
      <socialsecuritynumber>
         1234567890
      </socialsecuritynumber>
   </person>
</people_list>
```

```
<!-- Валидация простого HTML 4.01 -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
```

XML SCHEMA (XSD) EXAMPLE

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="country">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="country_name" type="xs:string"/>
        <xs:element name="population" type="xs:decimal"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<country>
    <country_name>France</country_name>
    <population>59.7</population>
</country>
```

*ML-DOCUMENT PARSING METHODS

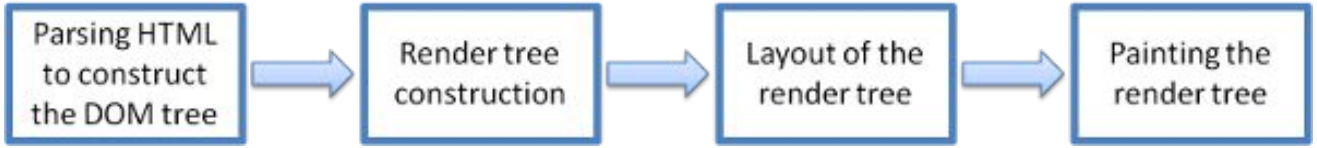
SAX (Simple API for XML)

- Raises an event/error when new element (token) appears (considering document as stream of tokens and errors)
- Works either as
  - callback-methods (push) or
  - cursor (pull, StAX)
- Requires constant memory
  - Good for embedded systems
- Does not know anything about document's model

DOM (**Document object model**, DOM tree)

- Creates full document model
- Used in browsers
- Unpredictable memory usage
  - XML-bombs using DTD
- Query languages (CSS-selectors, xpath, xquery)

BROWSER ENGINES = LAYOUT ENGINE + JS +…



- Good article about browser architecture
- Browser Layout Engine (html + css)
  - ~~**Trident**~~ (IE), "~~Edge~~" (Spartan) → Chromium (2019)
  - **Gecko** (Mozilla)
  - ~~**WebKit**~~ (Safari, Chromium-family), WebCore
    - **Blink** (Chrome 28+, Opera 15+, Chrome for Android)
  - Others (KHTML, ~~Presto~~)

HTML DOCUMENT STRUCTURE

```html
<!doctype html>
<html>
    <head>
        <title>my title</title>
    </head>
    <body>
        body
        <footer>
        <!-- html5 specific -->
        footer
        </footer>
    </body>
</html>
```

HTML TAGS

- Tag
  - **tag name** from HTML Schema - mandatory
    - `<td>..</td>`
  - **Closing tag** - mandatory
    - `<div> …. </div>`
    - `<img src="kitty.png" />`
  - Attributes – define semantics
    - ```
      <div id="div1"
          style="border:1px"
          class="myDiv">

      ```
  - Can have **inner tags** or **inner content** (text)
    - `<script> console.log(text); </script>`
    - ```
      <div>
          <span> … </span>
          <input />
      </div>
      ```
  - Layout of the tag is defined by the **style**
    - **In attribute**
    - **In CSS specification**
    - **By default**

STYLE

- inline-styles

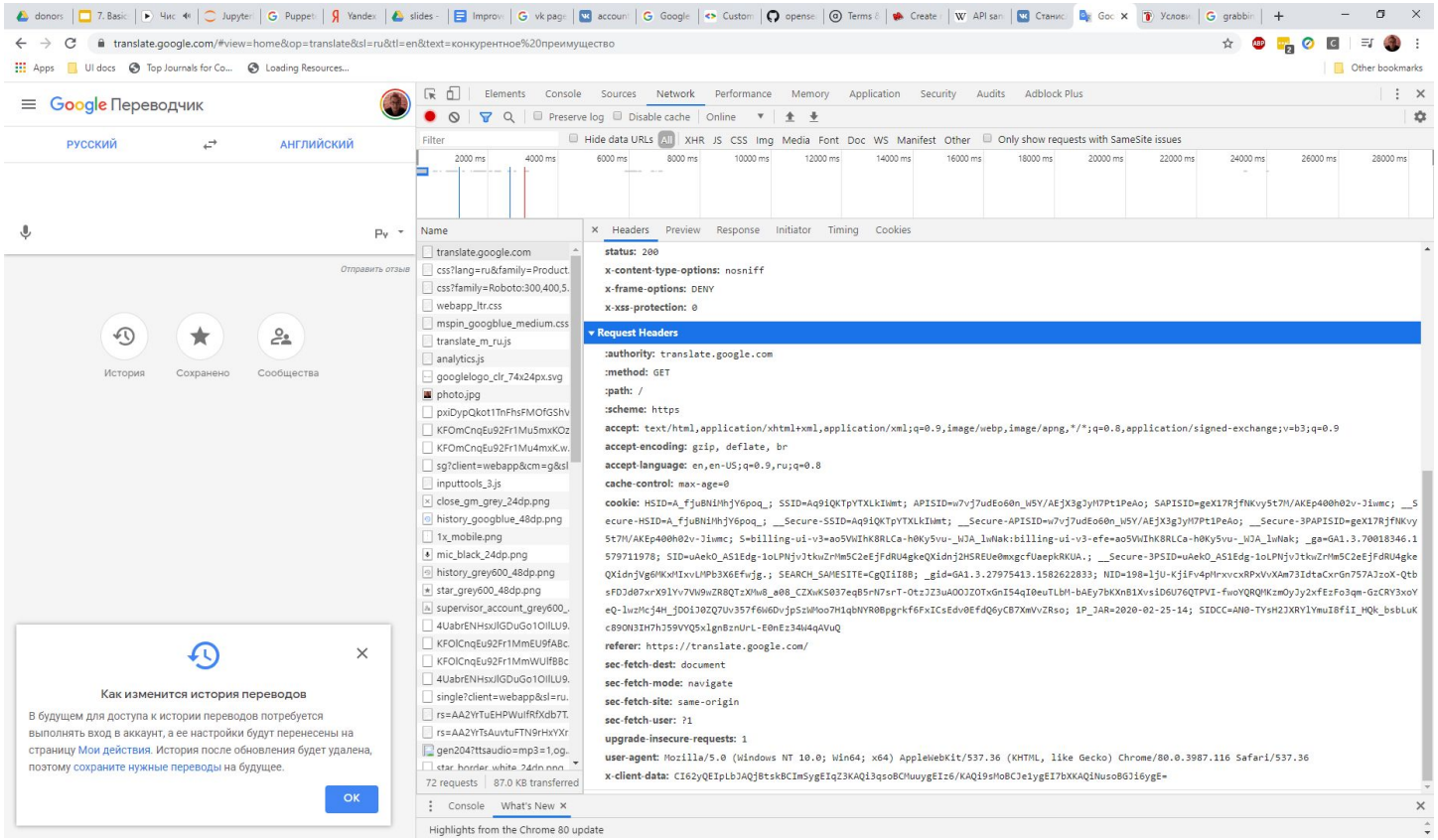
```
<div
  style="border: 1px solid gray; color: red"/>
```

- styles inside document
  - `<head><style> …… </style></head>`
- styles in separate CSS file
  -
```
<link
    rel="stylesheet" type="text/css"
  href="xxx.css"/>
```

STYLE SYNTAX

```
some css selector1
{
    property1: value1 ;
    property2: value2 ;
}
some css selector2
{
    property3: value3 ;
    property4: value4 ;
}
```

Browser fingerprint: visited site



FingerprintJS: 99.5% accuracy

- navigator.userAgent, navigator.language
- new Date().getTimezoneOffset()
- screen.height, screen.width, screen.colorDepth
- HTML5 features support (yes/no)
- doNotTrack flag (ЫЫЫЫ), cpuClass, platform
- Installed extensions
- canvas fingerprint (draw on canvas and toDataURL()) — fonts depend on platform
- WebGL fingerprint (for iOS)
- Installed fonts

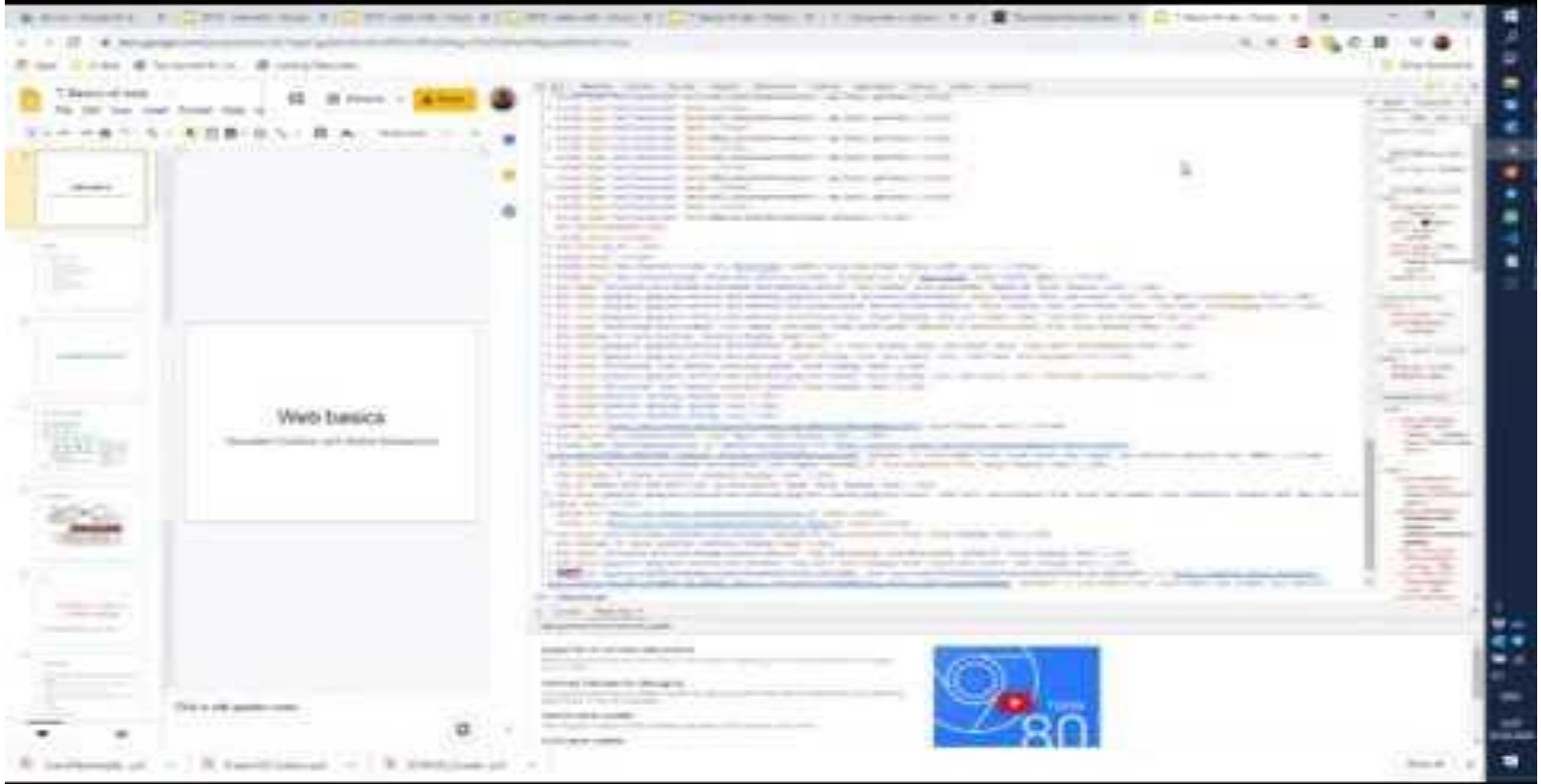
Crawling problems
...
and solutions

Problem 1: SaaS vs Documents

On your *nix machines run:

```
wget -O doc.txt http://tiny.cc/00dhkz
cat doc.txt | sed -e "s/;/;\n/g" | grep "QuadTree search"
```

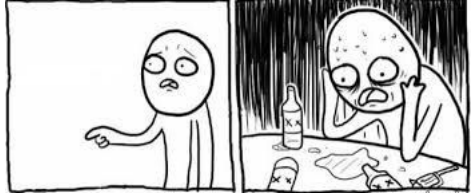
What’s wrong?


Web basics

How to?

You need a browser engine + JS

1. [Headless] browsers
2. Drivers to manage browsers
3. Automation software:
    a. Selenium
    b. Puppeteer



Problem 2: But wait… software engineering?!

How would you parse all links from the Wikipedia article page?

```
import requests
from bs4 import BeautifulSoup
…
```

Stop here! Every big **company knows** that you will parse it's data. It wants to minimize harm you can do. **APIs**! Free anonymous, free authenticated, paid.

1. Wikipedia API
2. VK API
3. Yandex Search API
4. Google Open Search API
5. ...

Problem 3. I was downloading … but it stopped working

1. Company **considers** the data (and service) it has as:
   a. A property
   b. A competitive advantage
2. Thus, company **protects** its data from grabbing (and services from proxying):
   a. With API regulations (Ya, ...) — what is the allowed rate
   b. Etiquette (Wiki)
   c. Access keys to control grabbing and proxying speed
   d. Special legal statements that prohibit grabbing (ASOS)
3. To **enforce** you to obey
   a. Access key restriction
   b. IP [range] blocking
   c. Browser **fingerprint** blocking
4. So, to speed or just enable you crawling …

Problem 4: The last but not the least...

Allow and Disallow

- robots.txt **prohibits**
  - http://innopolis.ru/robots.txt
  - https://yandex.ru/robots.txt
  - Wiki :)

```
# Sorry, wget in its recursive mode is a frequent problem.
# Please read the man page and use it properly; there is a
# --wait option you can use to set the delay between hits,
# for instance.
#
User-agent: wget
Disallow: /
```

  - Also,

```
Sitemap: https://yandex.ru/znatoki/sitemap/sitemap-index.xml
```

- sitemap.xml **helps**
  - https://yandex.ru/video/sitemap.xml
    - view-source:https://www.yandex.ru/video/sitemap.0.xml

Crawl safe!