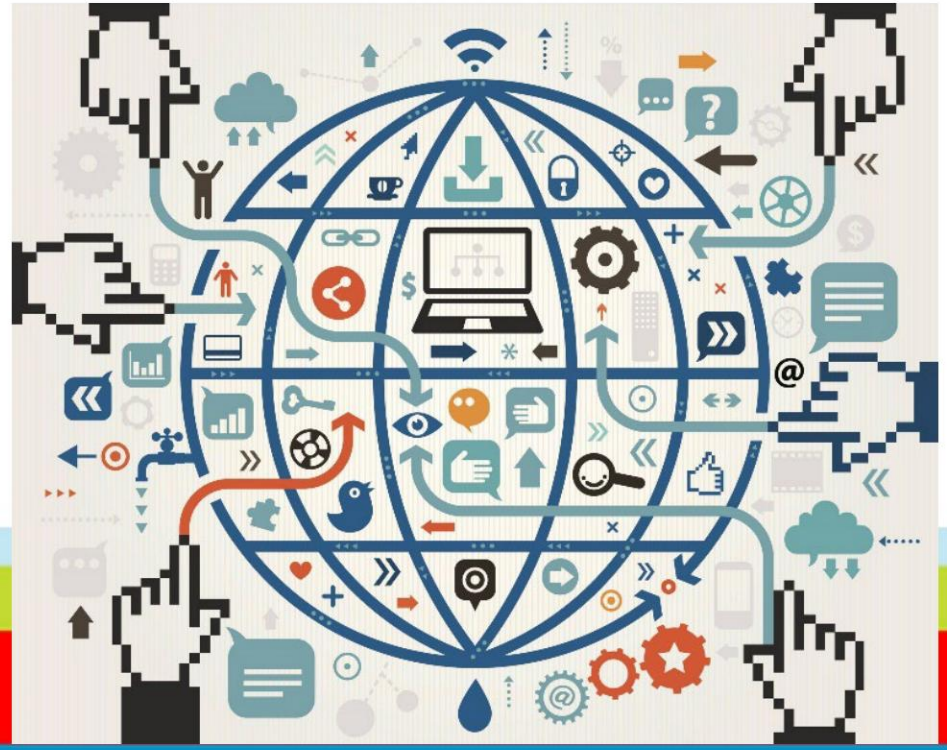
CEU | Universidad San Pablo



HTTP

Web Systems I

Degree in Information Systems Engineering Álvaro Sánchez Picot

alvaro.sanchezpicot@ceu.es

v20230918

Based on the work of: •
David González Márquez

Introduction

• Hypertext Transfer Protocol (HTTP)

– Hypertext Transfer Protocol • Application layer • Transmit hypermedia documents as HTML • Communication between clients and servers

– Can be used for other purposes

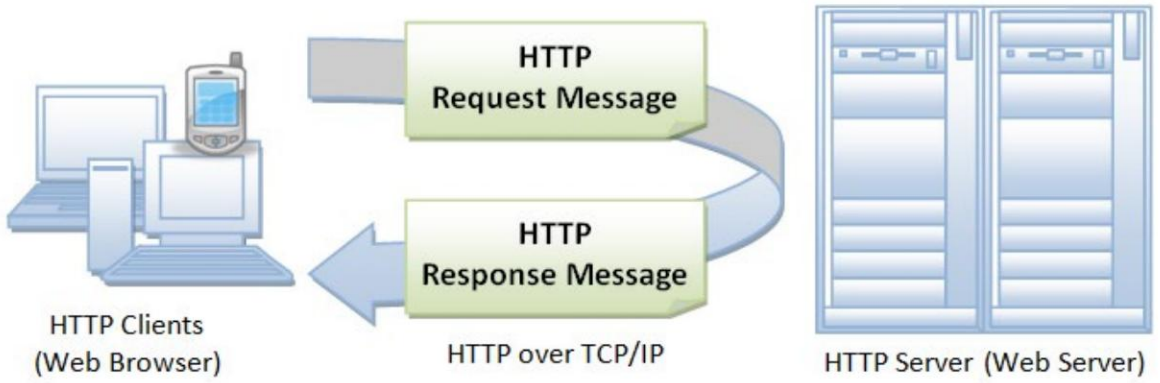
• Transaction protocol (request – response) • Stateless protocol

• Typically used with TCP/IP

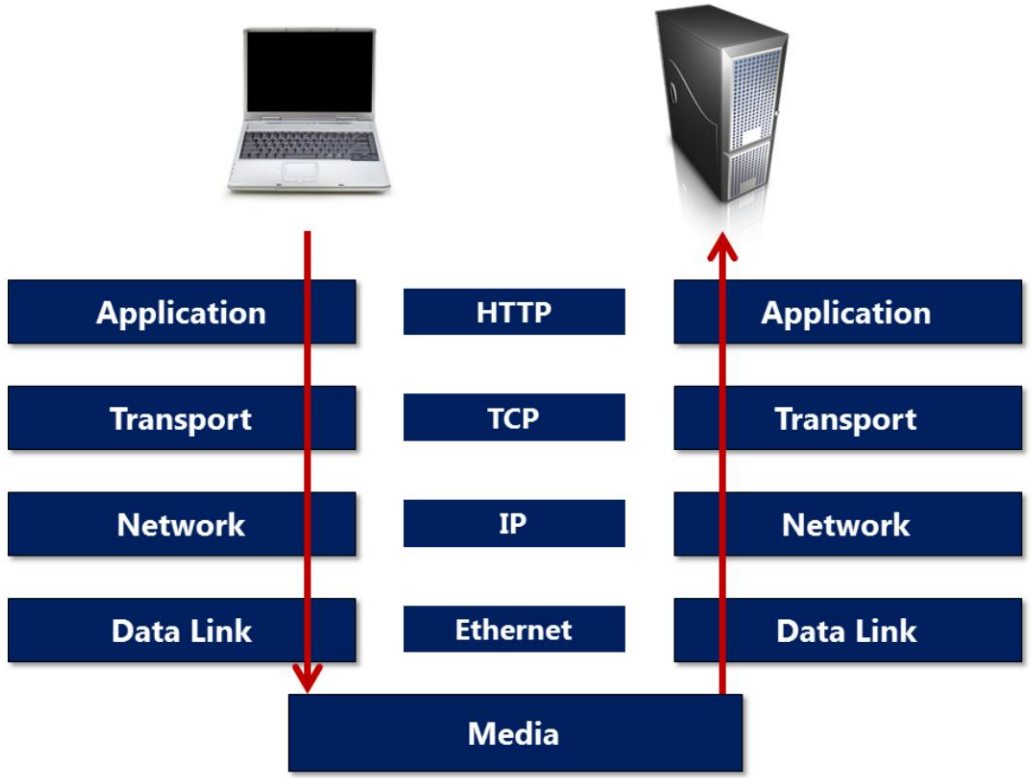
– Can be used with any reliable transport layer (e.g. RUDP)

Introduction

- The client opens a connection (e.g. TCP)
- The client initiates an HTTP request (request)
- The server returns the HTTP response (response)
- The connection is closed



Introduction



Evolution

- 1991: HTTP V0.9
- 1996: HTTP V1.0 (RFC 1945)
- 1997: HTTP/1.1 (RFC 2068)
- 1999: Improvements to HTTP/1.1 (RFC

2616) • 2007: HTTP Working

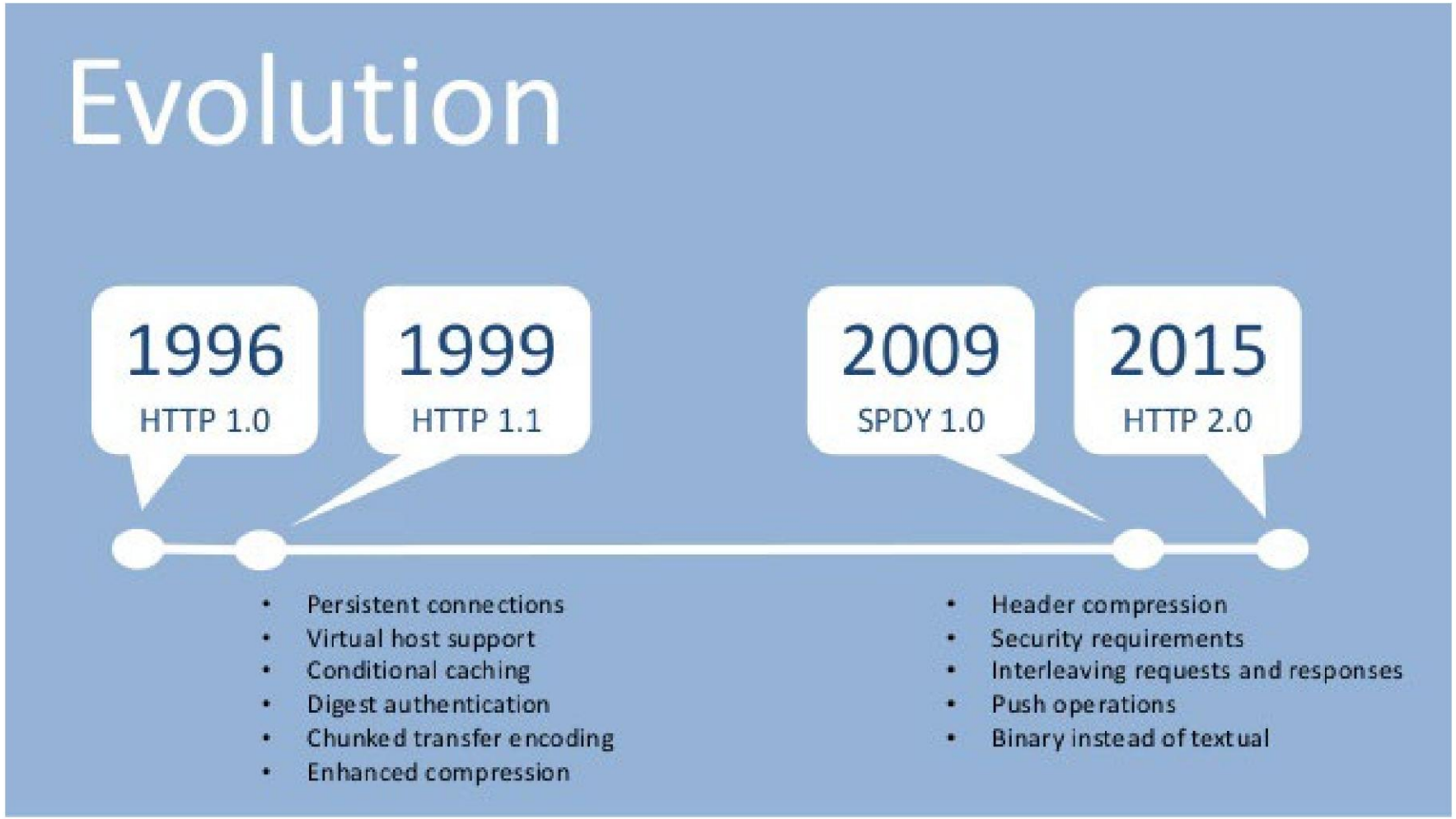
Group • 2009: SPDY protocol developed by Google to reduce the latency and would end up being HTTP/

2 • 2014: HTTP/1.1 improvements (RFCs

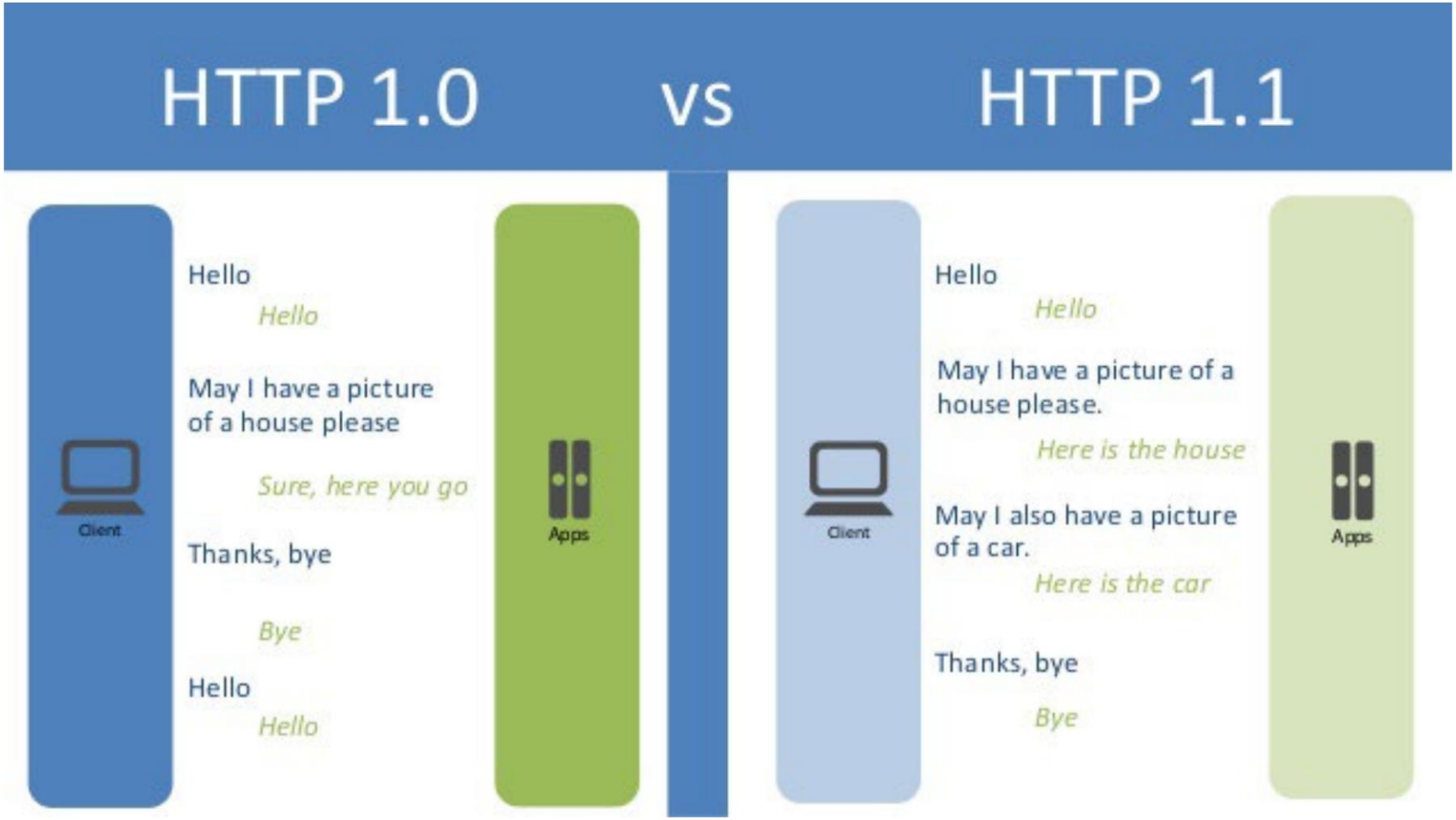
7230-7235) • 2015: HTTP/2 (RFC 7540)

- 2018: HTTP/3

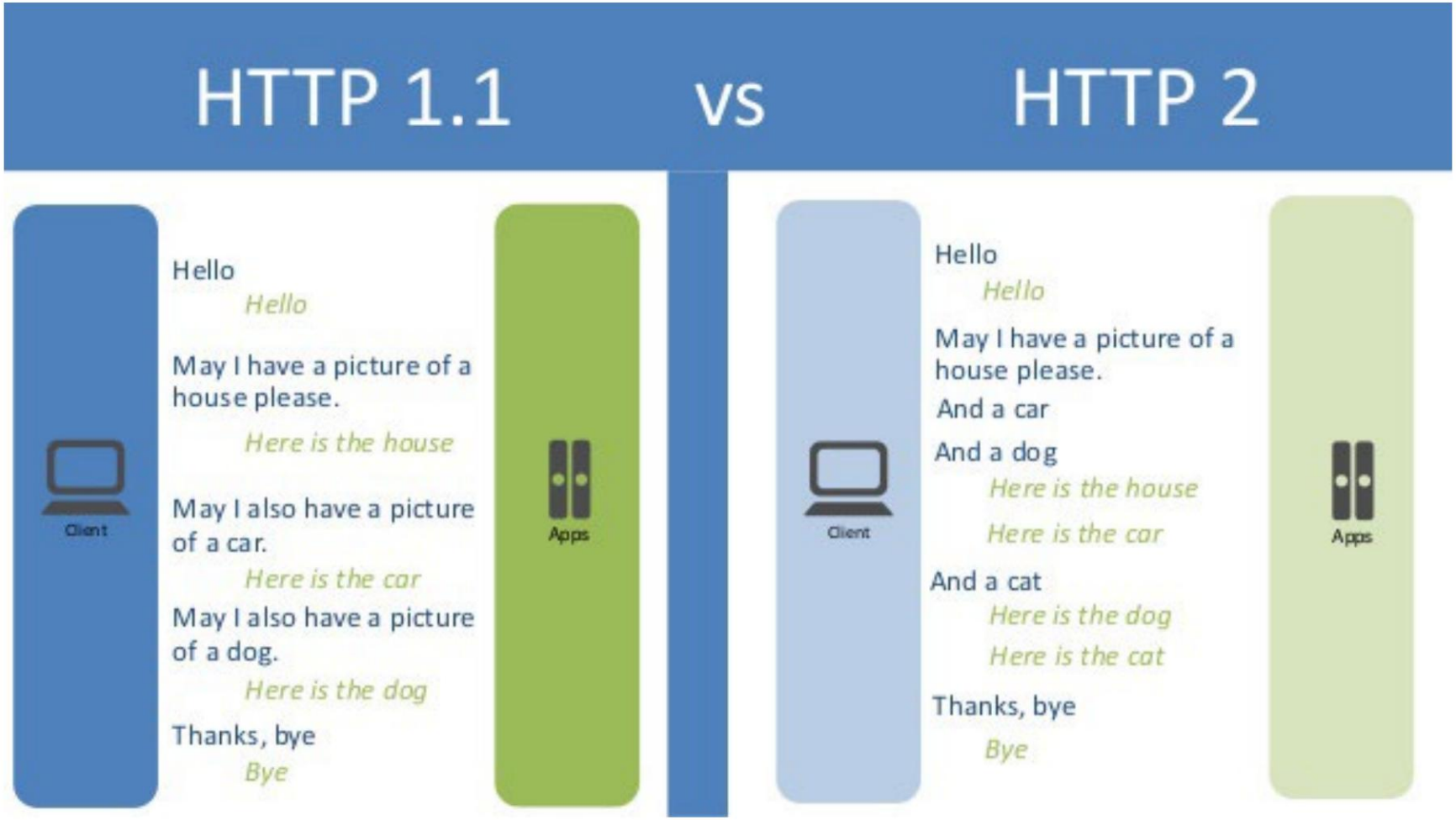
Evolution



Evolution



Evolution

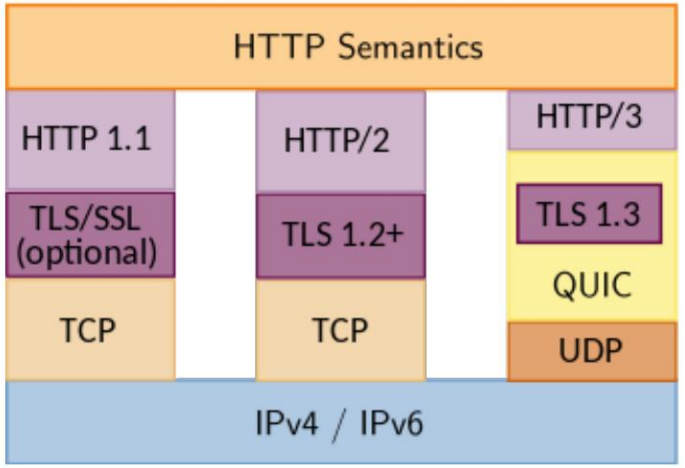


Evolution

- HTTP/1.1 vs HTTP/2 demo (https://http2.akamai.com/demo)
- Current use of versions:
  - HTTP/2 – 35.6% (September 2023)
  - HTTP/3 – 26.8% (September 2023)

    source: https://w3techs.com/
- More information on the history of HTTP

HTTP/3



- Draft version
- Previously known as “HTTP over QUIC”
- QUIC
  - Initially known as Quick UDP Internet Connections
  - Transport layer
  - Use UDP instead of TCP
  - Sometimes known as TCP/2
  - Fixes a TCP (head of line blocking) blocking issue
- Supported by default in most browsers
  - In some not activated by default, but can be activated
  - More information

CONCEPTS

User-agent

• User agent •

Program that represents a person, the client (eg the browser) https://www.whatismybrowser.com/detect/what-is-my-user-agent

User-agent

http://www.useragentstring.com

Chrome 86.0.4240.111	
Mozilla	MozillaProductSlice. Claims to be a Mozilla based user agent, which is only true for Gecko browsers like Firefox and Netscape. For all other user agents it means 'Mozilla-compatible'. In modern browsers, this is only used for historical reasons. It has no real meaning anymore
5.0	Mozilla version
Windows NT 10.0	Operating System: Windows 10
Win64	(Win32 for 64-Bit-Windows) API implemented on 64-bit platforms of the Windows architecture - currently AMD64 and IA64
x64	64-bit windows version
AppleWebKit	The Web Kit provides a set of core classes to display web content in windows
537.36	Web Kit build
KHTML	Open Source HTML layout engine developed by the KDE project
like Gecko	like Gecko...
Chrome	Name : Chrome
86.0.4240.111	Chrome version
Safari	Based on Safari
537.36	Safari build
Description:	Free open-source web browser developed by Google. Chromium is the name of the open source project behind Google Chrome, released under the BSD license.
All Chrome user agent strings	

User-agent

http://webaim.org/blog/user-agent-string-history/

NCSA_Mosaic/2.0 (Windows 3.1)

Mozilla/1.0 (Win3.1)

Mozilla/1.22 (supported; MSIE 2.0; Windows 95)

Mozilla/5.0 (Windows; U; Windows NT 5.1; sv-SE; rv:1.7.5) Gecko/20041108 Firefox/1.0

Mozilla/5.0 (compatible; Konqueror/3.2; FreeBSD) (KHTML, like Gecko)

Mozilla/5.0 (Macintosh; U; PPC Mac OS AppleWebKit/85.7 (KHTML, like Gecko) Safari/85.5

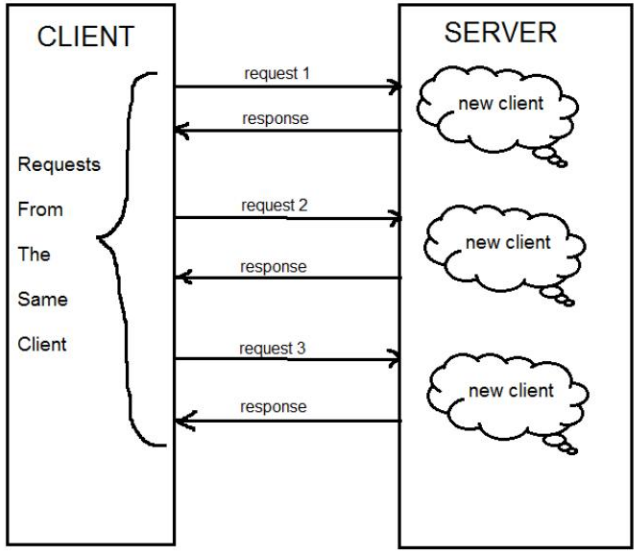
User-agent

http://webaim.org/blog/user-agent-string-history/

Mozilla/5.0 (Windows; U; Windows NT 5.1; in US) AppleWebKit/525.13 (KHTML, like Gecko) Chrome/0.2.149.27 Safari/525.13

Stateless

• HTTP is a stateless protocol • No information about previous connections is saved • The server response is the same for a previous client as for one new



Advantages and disadvantages?

Stateless

- Advantages
  - Scalability
  - Less complexity
  - Higher performance
  - Resources can be cached •

Disadvantages

  - Complicates interaction with the user
  - Extra information is needed to maintain the session
  - Susceptible to attacks such as DDoS (Distributed Denial of Service)

URL

- Uniform Resource Locator
- Uniform resource locator
- Resource
  - Information requested by the client
  - Peg. HTML document, image, video…
- Reference to a web resource specifying the location on a network and a mechanism to obtain it
- They are unique, each URL uniquely identifies a resource
- Living Standard

URL – Format

scheme://[userinfo@]host[:port]/path[?query][#fragment] • userinfo: user:password

- scheme:
  - http (HyperText Transport Protocol)
  - https (HyperText Transport Protocol Secure)
  - ftp (File Transfer Protocol)
  - And more
- host: name or IP of the server to which we want to connect

URL – Format

scheme://[userinfo@]host[:port]/path[?query][#fragment] • **port:** port to which we want to connect (optional). Default:

- http ÿ 80
- https ÿ 443
- ftp ÿ 21

• **authority:** [userinfo@]host[:port] • **path:**

path in the file system of the remote machine where the resource is located. The path is relative to the root directory of the web.

• **query:** Extra parameters, usually in the format "q=text" • **fragment:**

Extra parameters

URL – Example

http://google.es:8080/example/index.html?q=text#something • schema:

http • host:

google.es • port:

8080 •

authority: google.es:8080 •

path: example/index. html

• query: q=text

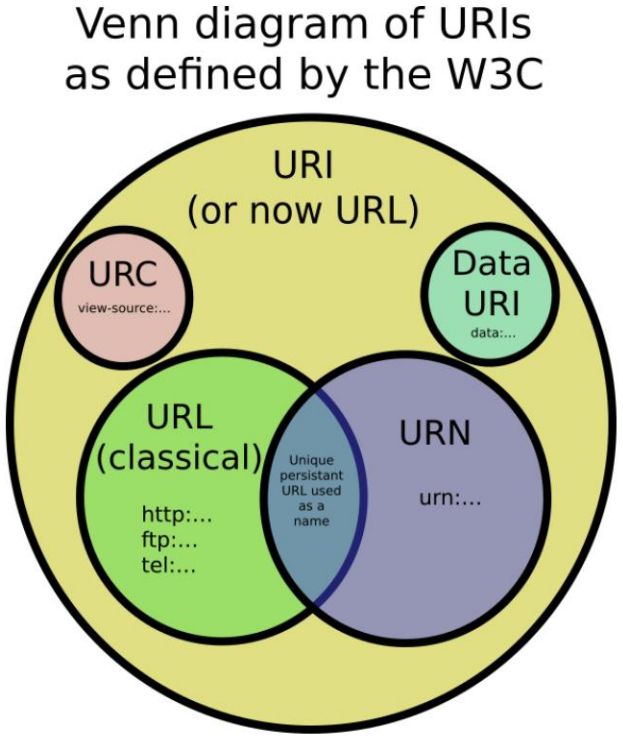
• fragment: something

URL – Example

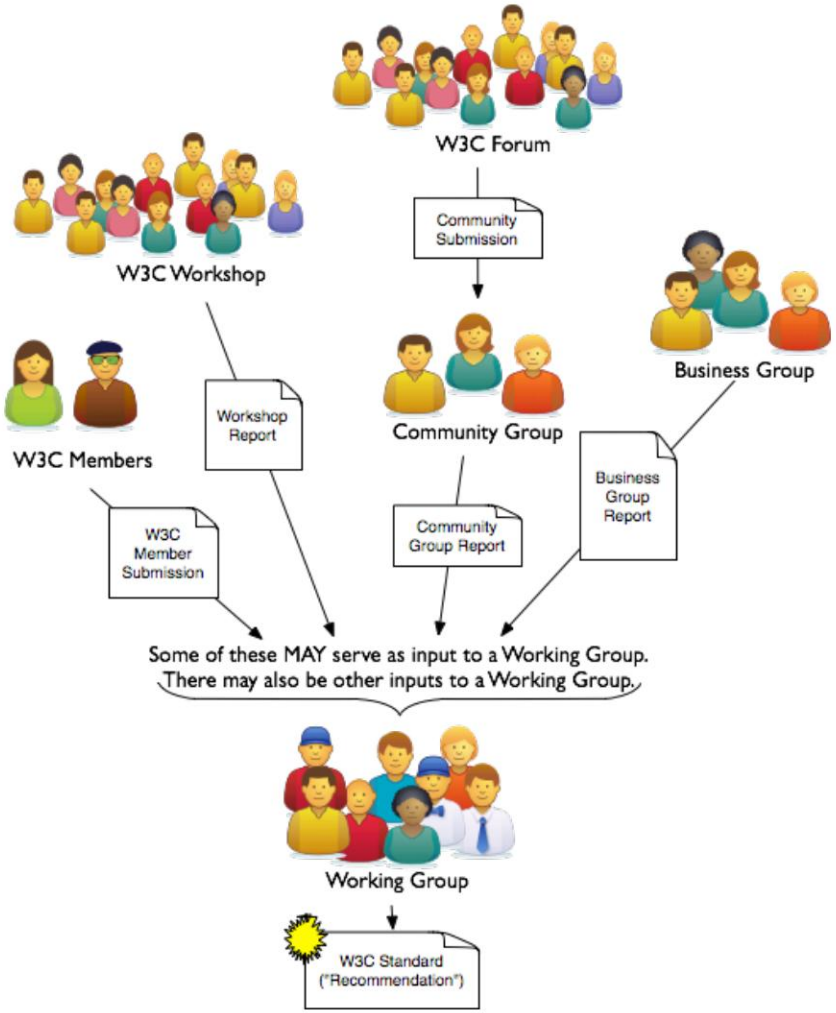
Input	Scheme	Host	Port	Path	Query	Fragment
`https://example.com/`	`"https"`	`"example.com"`	null	« the empty string »	null	null
`https://localhost:8000/search?q=text#hello`	`"https"`	`"localhost"`	8000	« `"search"` »	`"q=text"`	`"hello"`
`urn:isbn:9780307476463`	`"urn"`	null	null	`"isbn:9780307476463"`	null	null
`file:///ada/Analytical%20Engine/README.md`	`"file"`	null	null	« `"ada"`, `"Analytical%20Engine"`, `"README.md"` »	null	null

URL

- URI: Uniform Resource Identifier
  - RFC
  - Schemes
- URL: Uniform Resource Locator
- URN: Uniform Resource Name
  - urn:isbn:0451450523
  - urn:ISSN:0167-6423
- URC: Uniform Resource Characteristic
  - Metadata
- Difference between URI and URL



URL

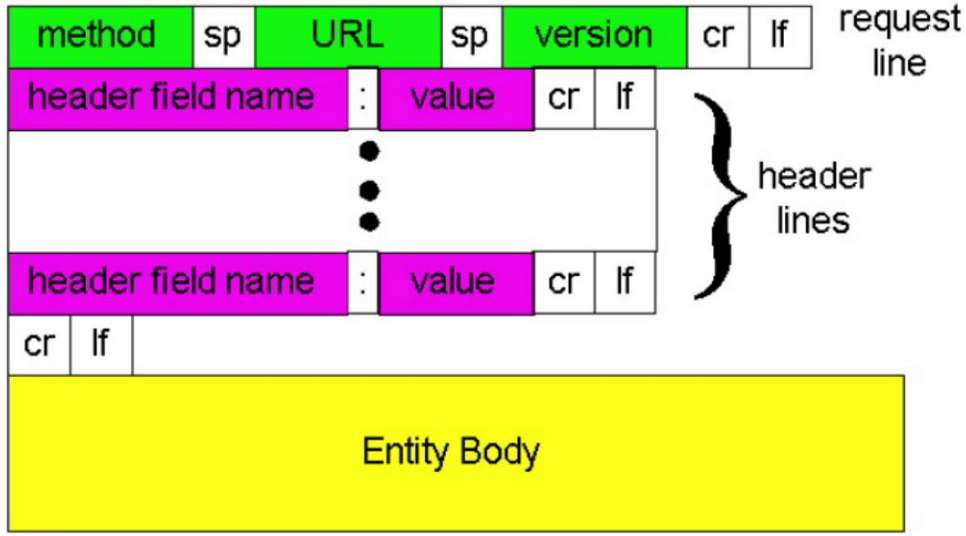


HTTP REQUEST

HTTP Request

SP Method SP URL Http Version CRLF (header-
name:header-value(,header-value)*CRLF)*
CRLF

Message body



- SP: blank space
- CRLF: carriage return
- (): optional • *:

can be repeated • https://
www.w3.org/Protocols/rfc2616/rfc2616-sec5.html

HTTP Request

• **Method** used, usually GET or POST • **URL** of the resource being requested • **Version** of the HTTP protocol being used

• One or more **headers**

– name:value(,value)*

– Blank spaces are part of the value, they are not separators

– Each header on one line

HTTP Request

- The message can be empty (this is the case of the GET method) or not (this is the case of the POST method).

```
GET /en/html/dummy?name=MyName&married=not+single&male=yes HTTP/1.1 Host:
www.explainth.at User-Agent:
Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11) Gecko/20070312 Firefox/1.5.0.11 Accept: text/
xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-gb,en;q=0.5 Accept-
Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 Keep-Alive: 300
Connection: keep-
alive
Referer: http://www.explainth.at/en/misc/httpreq.shtml
```

HTTP Request

- This POST request would be completely equivalent to the previous GET request in terms of sending data from a form to the server. In this case, the content of the message is only the last line.

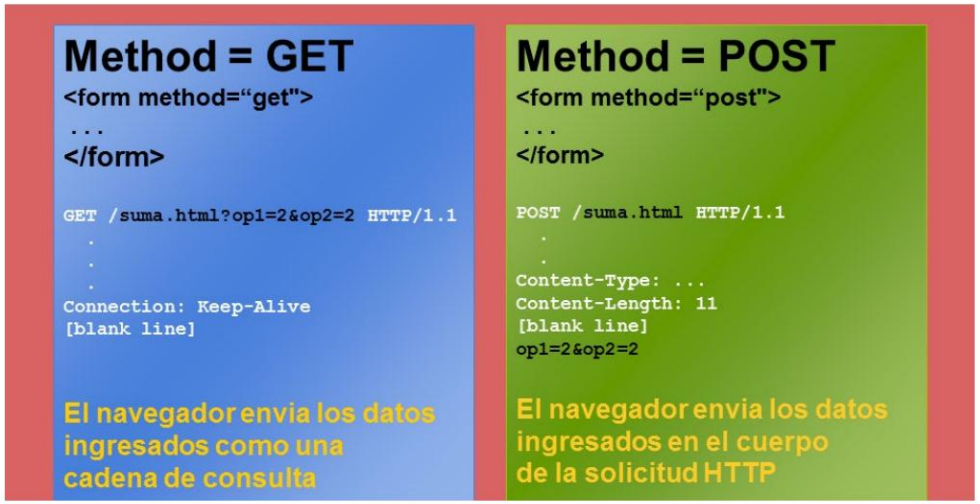
```
POST /en/html/dummy HTTP/1.1
Host: www.explainth.at User-
Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11) Gecko/20070312 Firefox/1.5.0.11 Accept:text/xml ,text/
html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5 Accept-Language: en-gb,en;q=0.5 Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7 Keep-Alive:
300 Connection:
keep-alive
Referer: http://www.explainth.at/en/misc/httpreq.shtml Content-Type:
application/x-www-form-urlencoded
Content-Length: 39

name=MyName&married=not+single&male=yes
```

HTTP Request

- In the form there were three fields, one with the name "name", another with name "married", and the last one with the name "male". In the response, each of the three fields is separated by the "&" symbol. Each field is followed by the value that the user has entered in the form for said field, and separating the name of the field and its value is the "=" sign.



HTTP Request – Methods

- Also called verbs
- Action you want to perform on the resource indicated in the request
  - That resource could be a file that resides on a server, or it could be a program that is running on said server. • GET
  - Get a resource
  - Does not modify anything on the server
  - Mandatory method

HTTP Request – Methods

- HEAD
  - Get the header of a GET request without the content
  - Obtaining the meta-information of the resource
  - To know the size/version…
  - Mandatory method
- POST
  - Sends data to the server to be processed by the resource specified in the petition
  - Data is included in the body of the request
  - Could create a new resource on the server, or update an existing resource

HTTP Request – Methods

- PUT
  - – Send a given resource (a file) to the server
  - – Unlike POST, this method creates a new connection (socket) and uses it to send the resource, which is more efficient than sending it within the body of the message

- DELETE
  - – Delete the specified resource. •

TRACE

  - – Asks the server to send you a response message
  - – It is usually used to diagnose possible connection problems

HTTP Request – Methods

- OPTIONS
  - Asks the server to indicate the HTTP methods it supports for a specific URL
- CONNECT
  - Used to transform an existing connection to an encrypted connection (https).
- PATCH
  - Partially modify an existing resource on the server

HTTP Request – Methods

- Safe methods:
  - – Those that should not change the server state. They only retrieve information –
  - Although the GET method in principle should not change anything, in practice it can be used to send commands to the server, although it is not

recommended • Idempotent methods:

- – Multiple requests should have the same result
- – “Safe methods” should by definition also be “idempotent methods”
- – Again, a bad implementation could skip these practices

HTTP Request – Methods

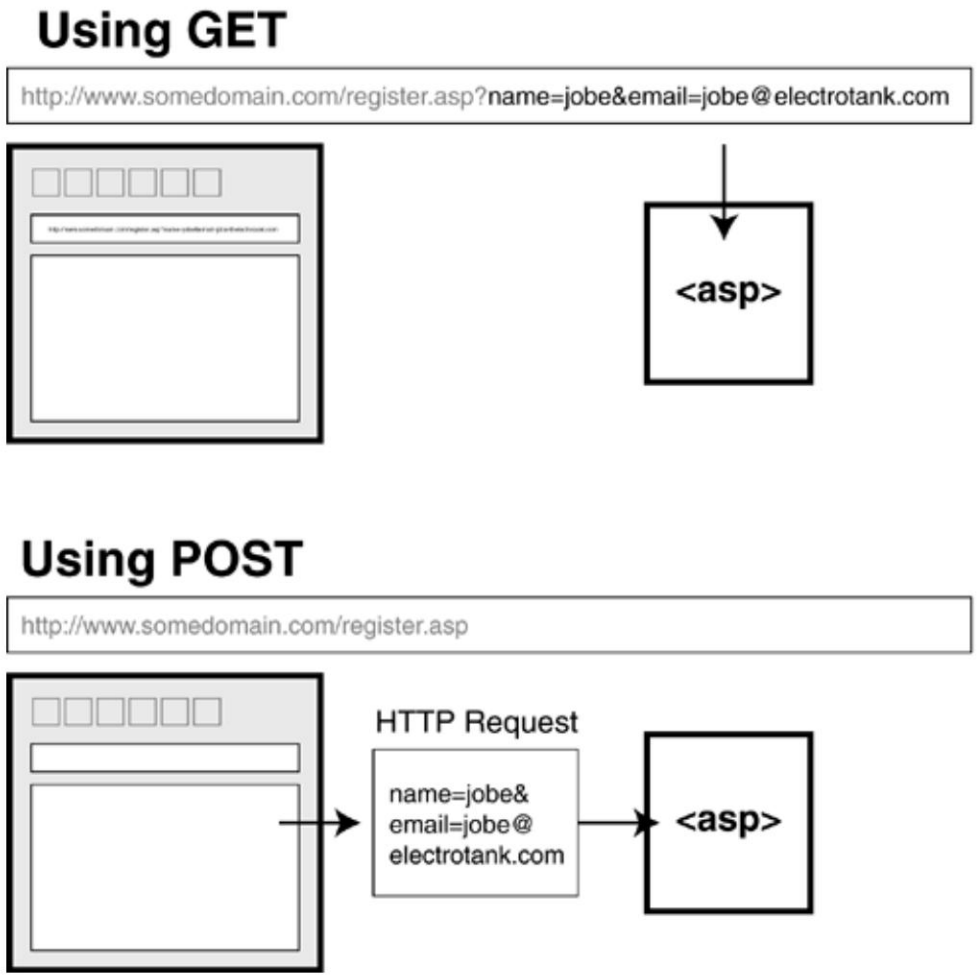
HTTP methods	RFC	Request has Body	Response has Body	Safe	Idempotent	Cacheable
GET	RFC7231	Optional	Forks	Forks	Forks	Forks
HEAD	RFC 7231	Optional	No	Forks	Forks	Forks
POST	RFC 7231	Forks	Forks	No	No	Forks
PUT	RFC7231	Forks	Forks	No	Forks	No
DELETE	RFC 7231	Optional	Forks	No	Forks	No
CONNECT	RFC 7231	Optional	Forks	No	No	No
OPTIONS	RFC 7231	Optional	Forks	Forks	Forks	No
TRACE	RFC 7231	No	Forks	Forks	Forks	No
PATCH	RFC 5789	Forks	Forks	No	No	No

source

HTTP Request – Methods

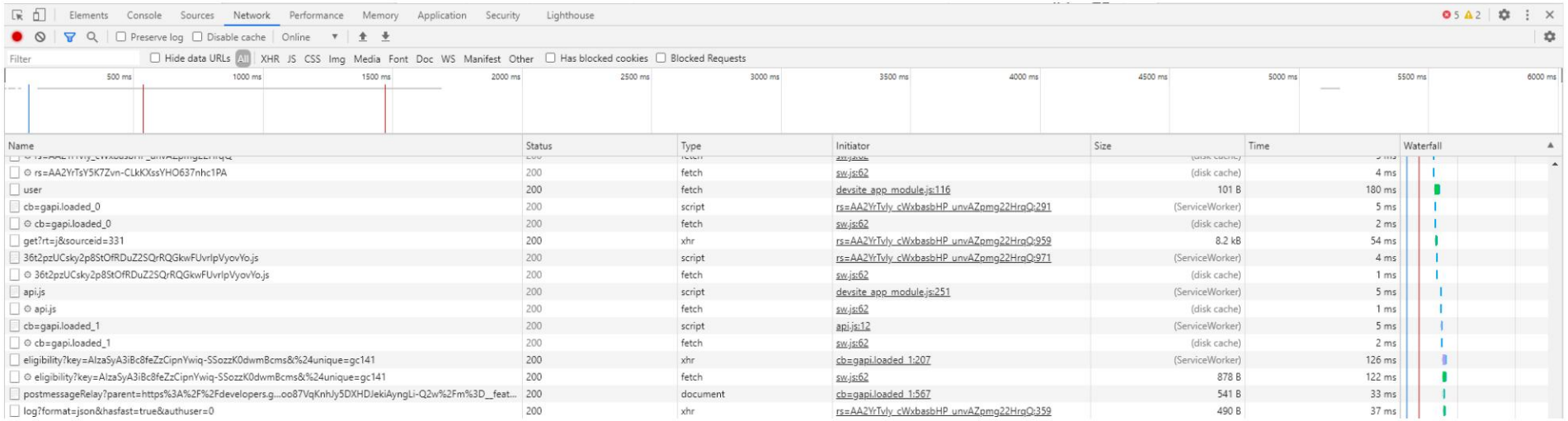
	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data is about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because POST is a little safer than GET because the data sent is part of the URL Never use GET when sending passwords or other server logs parameters are not stored in browser history or in web sensitive information!	
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

HTTP Request – Methods

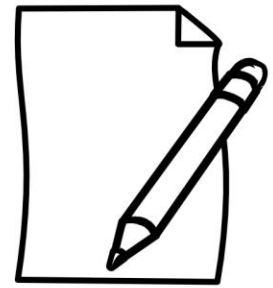


Chrome DevTools

- https://developer.chrome.com/devtools
- Ctrl + Shift + J > Network



HTTP – Exercise 1



- Find a website that has a form
  - Open private browsing
  - Look for a page to log in/register or a contact form • Analyze the

HTTP messages there are

  - Do you identify any?
- Fill out the form with any data and click submit • Analyze

the HTTP message that has been generated with the tools of the browser

- Repeat the steps editing the HTML and changing the method

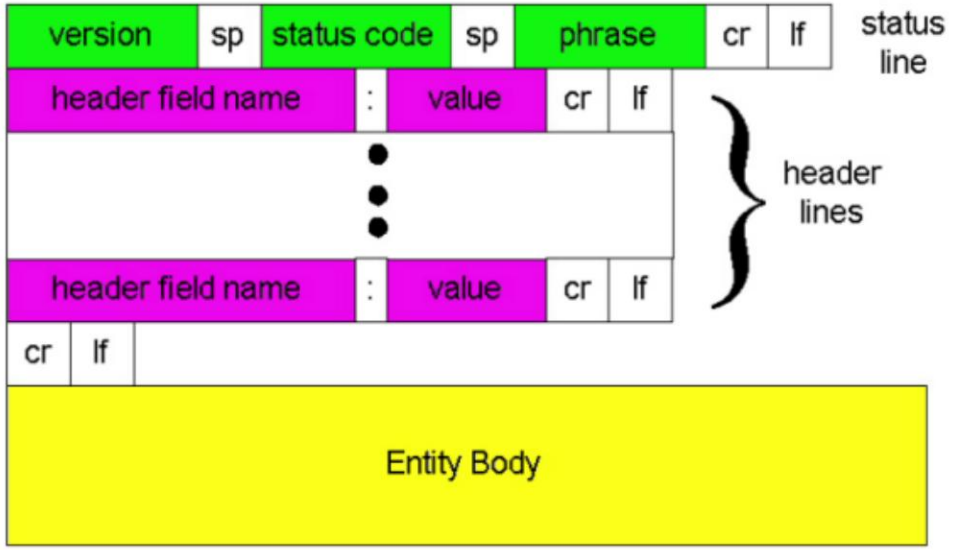
HTTP RESPONSE

HTTP response

A server response in the http protocol follows the following structure:

http-version SP status-code SP explanation-phrase CRLF (header-name:
header-value (","header-value)* CRLF)*
CRLF

Message body



HTTP response

- **Status code:** indicates whether the request was successful or if there was an error with her
- **Phrase:** explanation of the code
- **Headers:** same structure as in requests • **Body:** the

server's response

HTTP response

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8 Content-

Length: length

<html>

<head> <title> Title of our first page </title> </head> <body> Hello world! </body> </

html>

HTTP Response – Status Codes

• Three-digit numbers •

They are part of HTTP responses • They

explain what happened when trying to carry out a request • Various

categories: – 1xx:

Messages –

2xx: Operation completed
successfully – 3xx: Redirection

– 4xx: Error on the client's side

– 5xx: Server error

• List

HTTP Response – Status Codes

Codes 1xx: Information •

100 Continue: Connection accepted, request continues •

101 Switching Protocols: Changing protocols • 102

Processing: Still processing the request

HTTP Response – Status Codes

Codes 2xx: Operation completed successfully

• 200 OK: The request has been completed

successfully • 201 Created: Request OK and a new resource

has been created • 202 Accepted: Request accepted but the process has not finished y

• 204 No Content: No Content

• 205 Reset Content: Tells the user agent to reload the document • 206

Partial Content: Partial content

HTTP Response – Status Codes

Codes 3xx: Redirect • 301

Moved Permanently: contains the new URL • 302

Found: The change is temporary •

304 Not modified: Not modified, the cached version can be used • 307

Temporary Redirect: Very similar to 302 • 308

Permanent Redirect: Very similar to 301

HTTP Response – Status Codes

Codes 4xx: Error on the part of the client • 400 Bad Request: Malformed or invalid request • 401 Unauthorized: The client must register

• 403 Forbidden: The client does not have access rights

• 404 Not Found: The indicated resource does not exist

• 405 Method Not Allowed: That HTTP method is not allowed • 409 Conflict: For example, multiple simultaneous edits • 410 Gone: The resource has been permanently deleted • 418 I'm a teapot: April's fool's joke

HTTP Response – Status Codes

Codes 5xx: Server Error • 500

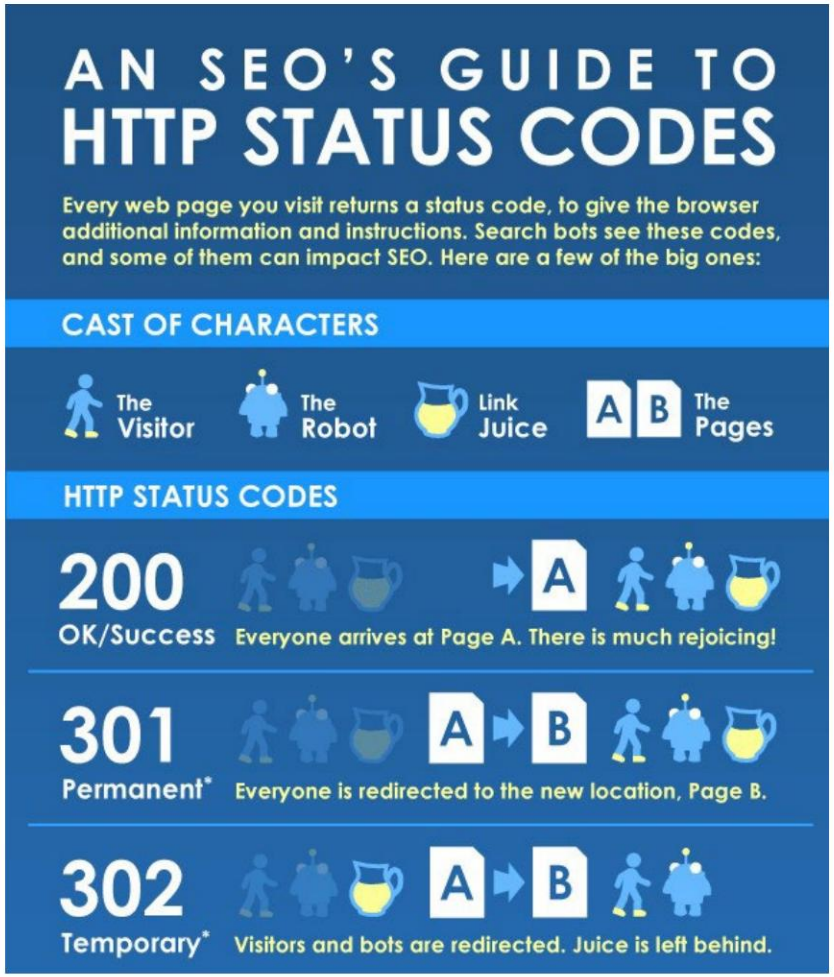
Internal Server Error: the server does not know how to handle the situation • 502 Bad

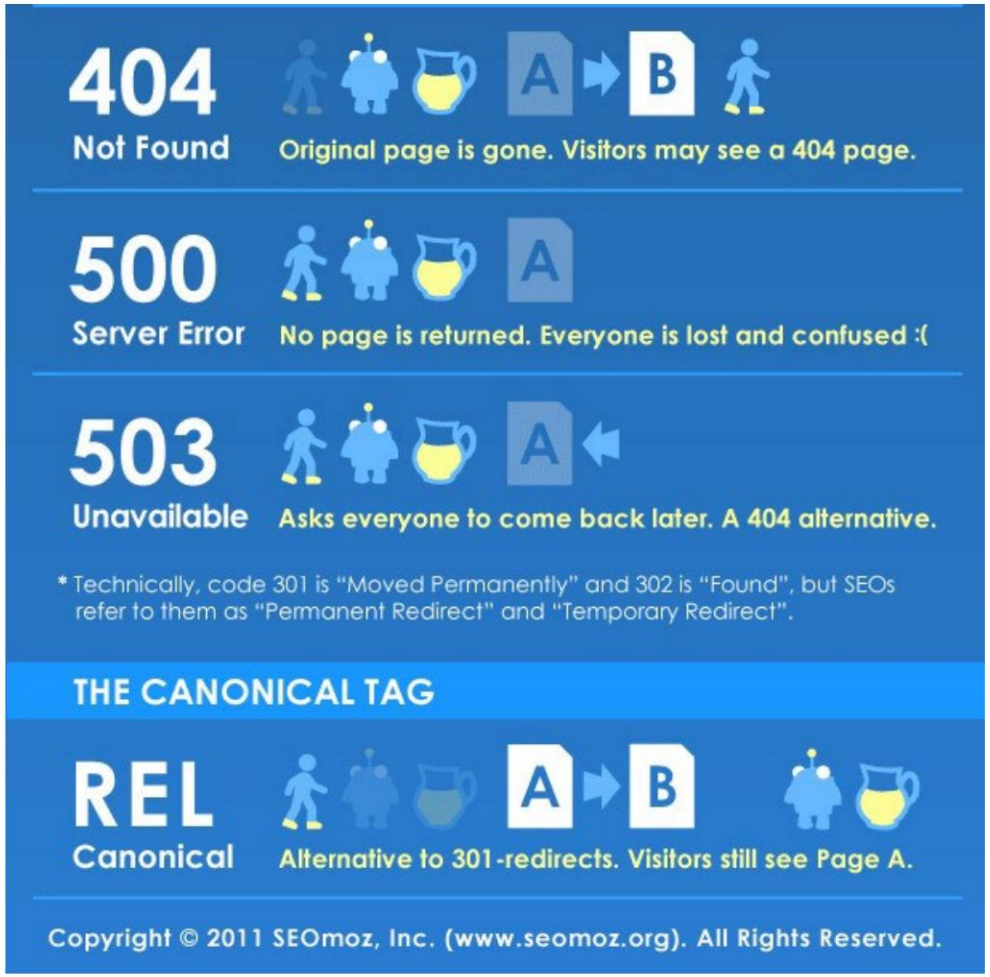
Gateway • 503 Service Unavailable: e.g. overloaded or under maintenance • 504 Gateway Timeout

And many more codes

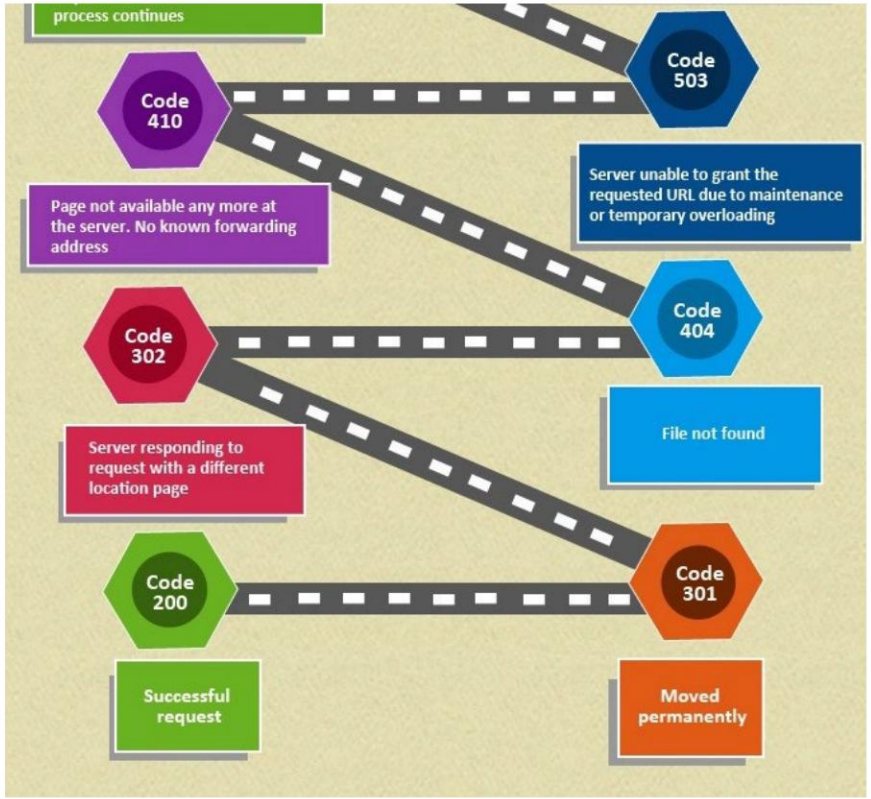
• There are also unofficial codes

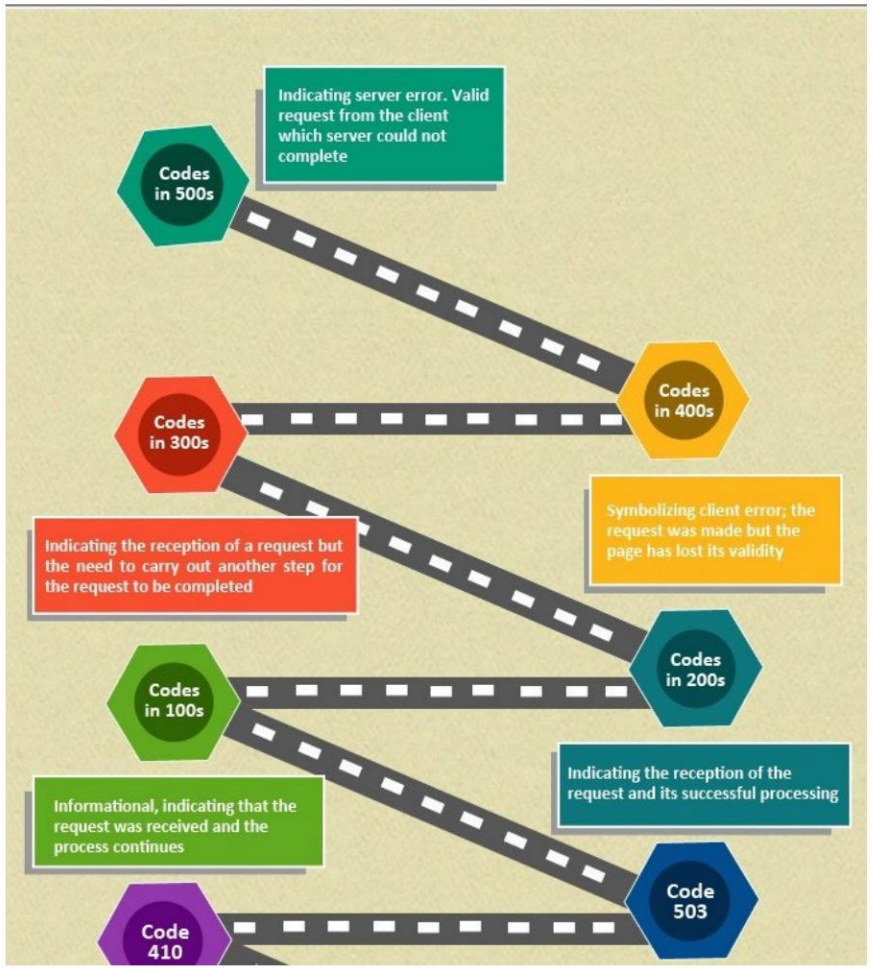
HTTP Response – Status Codes





HTTP Response – Status Codes



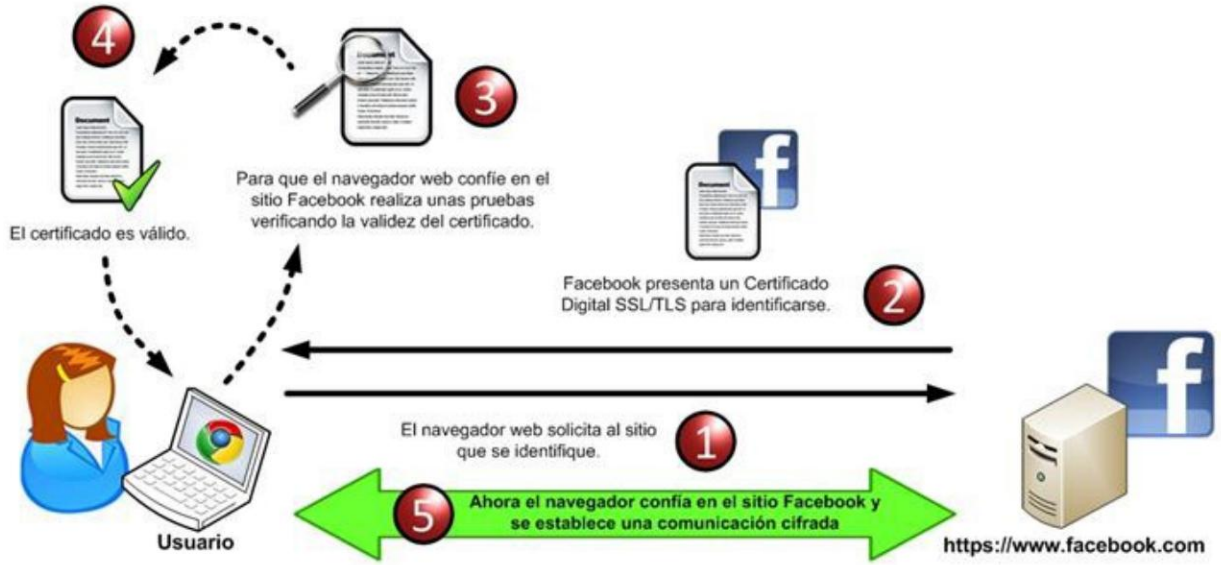


HTTPS

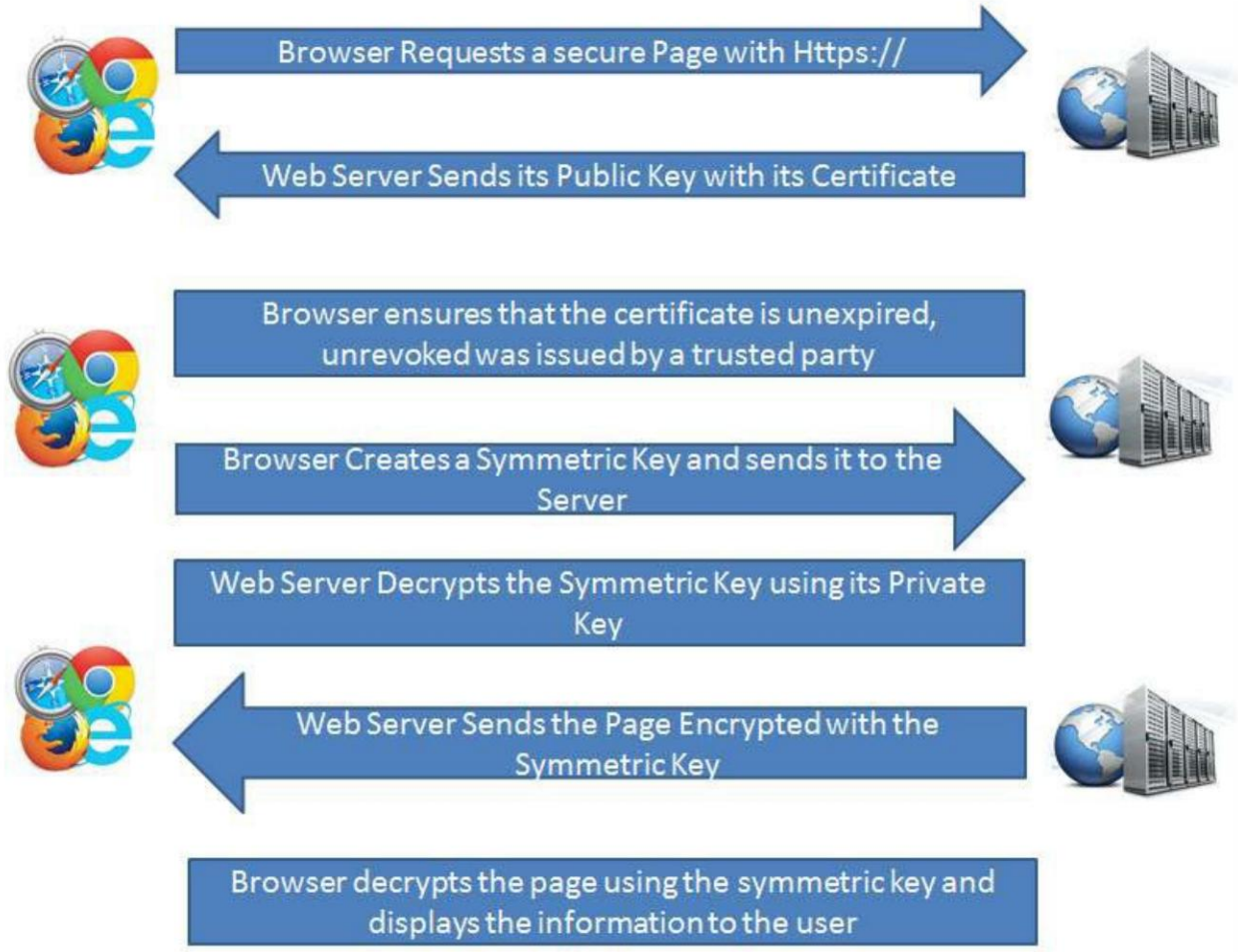
- Hypertext Transfer Protocol Secure
- Application protocol based on the HTTP protocol •

Intended for secure data transfer. • Uses SSL/

TLS-based encryption (port 443)



HTTPS

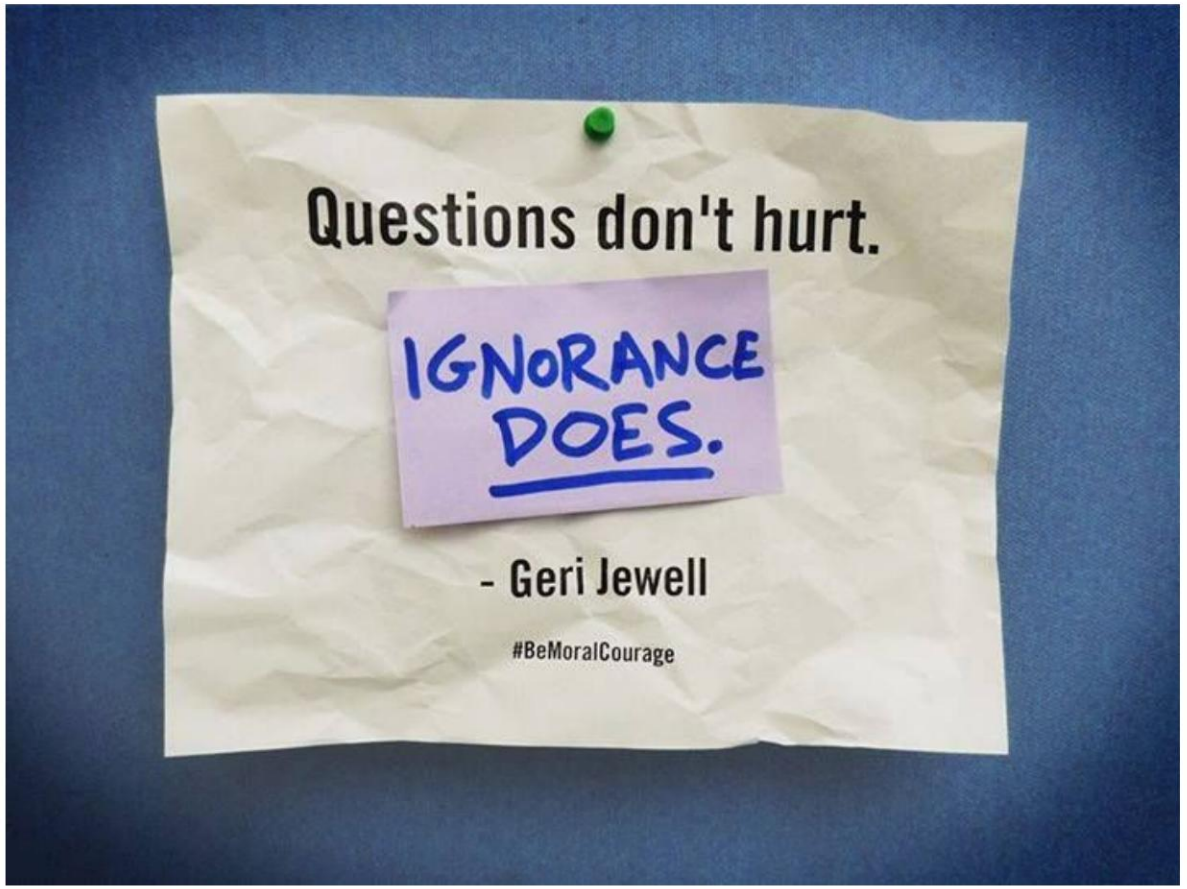


Postman

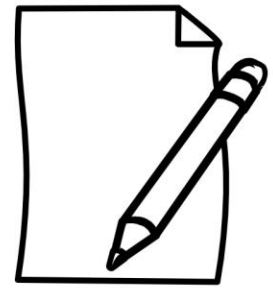
- API development platform
  - Send HTTP requests
  - Testing
  - Simulate endpoints
- Before it was a browser extension • Now it is a standalone app • It can also be used from the browser:

  https://go.postman.co/build
- Web: https://www.postman.com/

Doubts



HTTP – Exercise 2



- Install Postman or use it from the browser
- Try using each of the 9 HTTP methods
  - Analyze the requests and responses
- Try some of the status codes at: https://httpstat.us/

References

- HTTP - Hypertext Transfer Protocol
  - https://www.w3.org/Protocols/
  - https://www.w3.org/Protocols/History.html
  - https://www.w3.org/Protocols/Classic.html
  - https://developer.mozilla.org/en-US/docs/Web/HTTP
- Articles about HTTP
  - https://code-maze.com/http-series/
- E-Book
  - Introduction to HTTP (https://launchschool.com/books/http)
- Gourley, D., Totty, B., Sayer, M., Aggarwal, A., & Reddy, S. (2002). HTTP: the definitive guide. "O'Reilly Media, Inc."