NodeJS

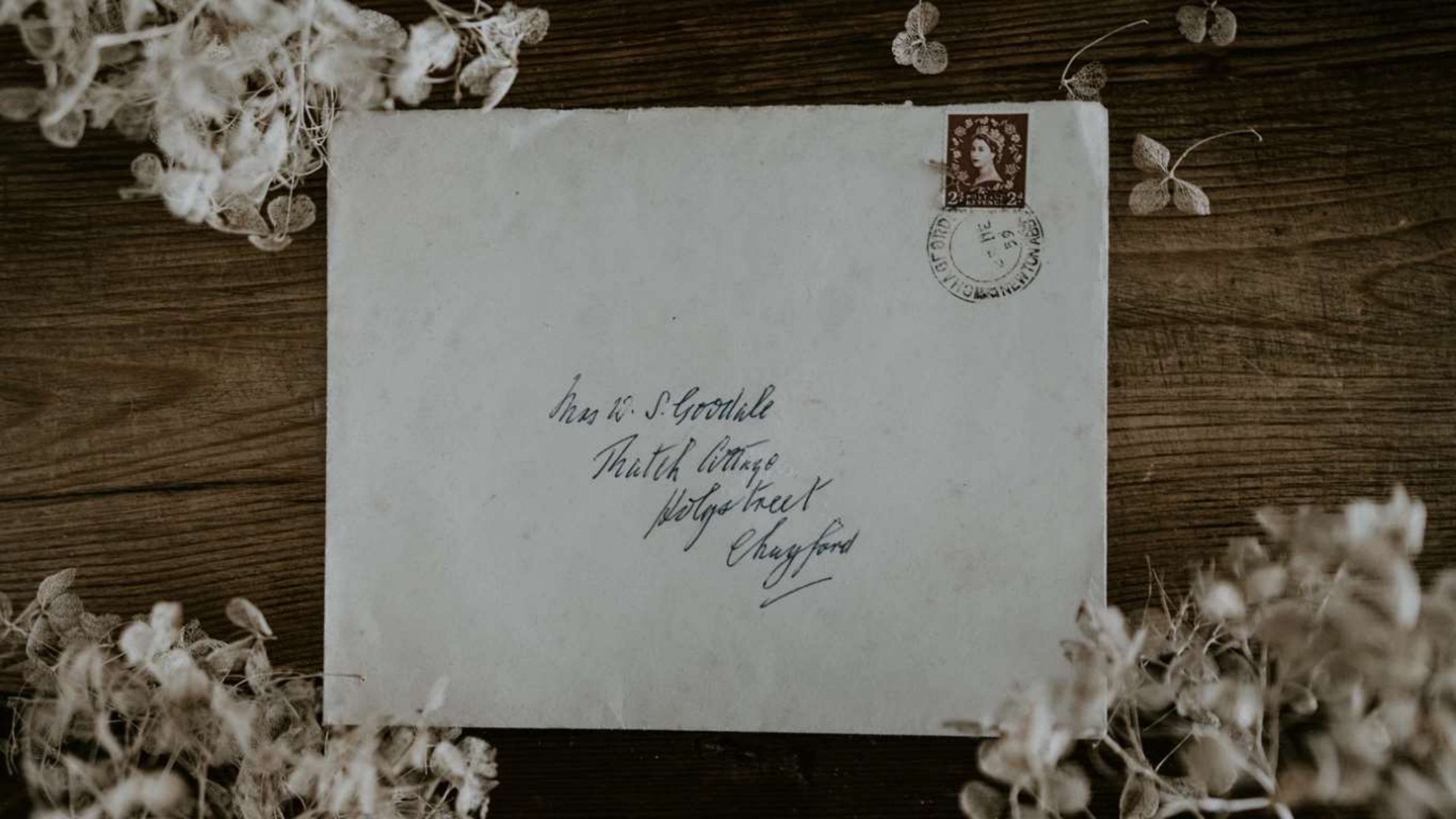
środowisko i technologia ServerSide

PAWEŁ ŁUKASZUK

Rok akademicki: 2020/2021, Semestr: 2, Wykład: 1

Other protocols

- User Datagram Protocol (UDP): substitute communication protocol to TCP implemented primarily for creating loss-tolerating and low-latency linking between different applications.
- Post office Protocol (POP): POP3 is designed for receiving incoming emails.
- Simple mail transport Protocol (SMTP): designed to send and distribute outgoing E-Mail.
- File Transfer Protocol (FTP): allows users to transfer files from one machine to another. Types of files may include program files, multimedia files, text files, and documents, etc.
- Hyper Text Transfer Protocol (HTTP): designed for transferring a hypertext among two or more systems.
- Hyper Text Transfer Protocol Secure (HTTPS): standard protocol to secure the communication among two computers one using the browser and other fetching data from web server.


Mrs D. S. Goodlake
Thatch Cottage
Hollys Street
Chagford

HTTP

- HyperText Transfer Protocol - a protocol for transferring hypertext documents
- provides a standardized way for client and server to communicate and exchange data
- determines the form of client requests for data and the form of the server's response to the request
- is used to transfer various types of data, such as images, videos, documents, etc.
- is based on TCP (Transmission Control Protocol)
- by default uses port 80

HTTP

- HTTP is connectionless
- HTTP is stateless
- HTTP delivers all types of data

- HTTP is insecure (issue resolved by HTTPS)

HTTP versions

- 1991 - 0.9
- 1996 - 1.0
- **1997 - 1.1**
- **2015 - 2.0**
- draft - 3.0

Protocol	Desktop	Mobile	Both
	5.60%	0.57%	2.97%
HTTP/0.9	0.00%	0.00%	0.00%
HTTP/1.0	0.08%	0.05%	0.06%
HTTP/1.1	40.36%	45.01%	42.79%
HTTP/2	53.96%	54.37%	54.18%

Figure 20.3. HTTP version usage by request.

https://almanac.httparchive.org/en/2019/http2

HTTP

- RFC-2616 (HTTP/1.1)
- RFC-7540 (HTTP/2)

From the developer point of view:

- resources
- methods
- headers
- response codes
- authentication
- redirections

HTTP resources

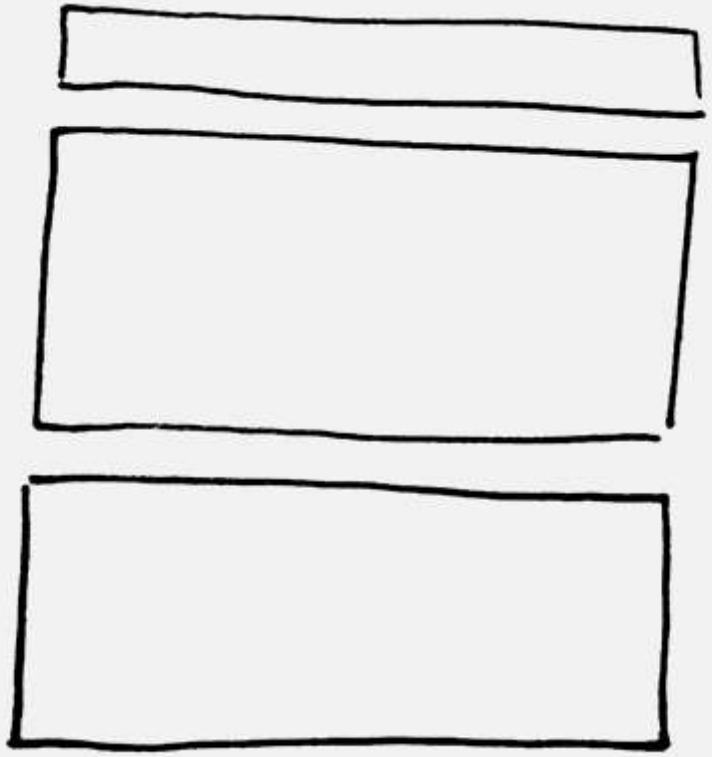
The target of an HTTP request is called a "resource", whose nature isn't defined further.

It can be a document, a photo, or anything else. Each resource is identified by a Uniform Resource Identifier (URI) used throughout HTTP for identifying resources.

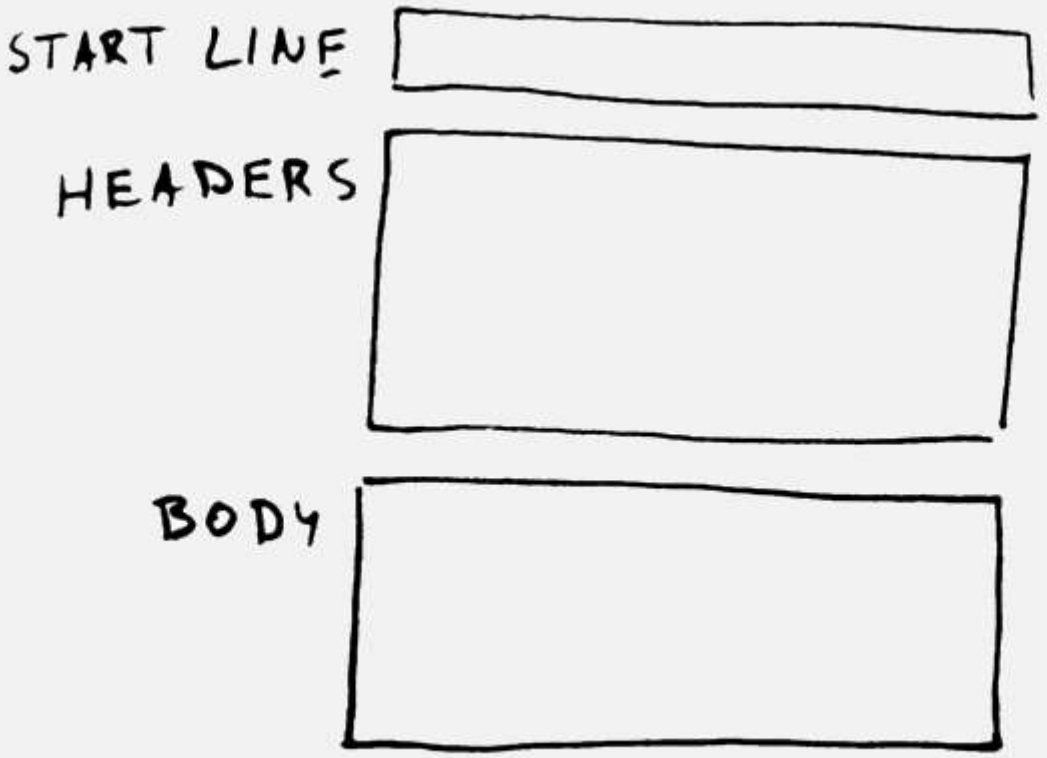
HTTP messages

- Request HTTP message
- Response HTTP message

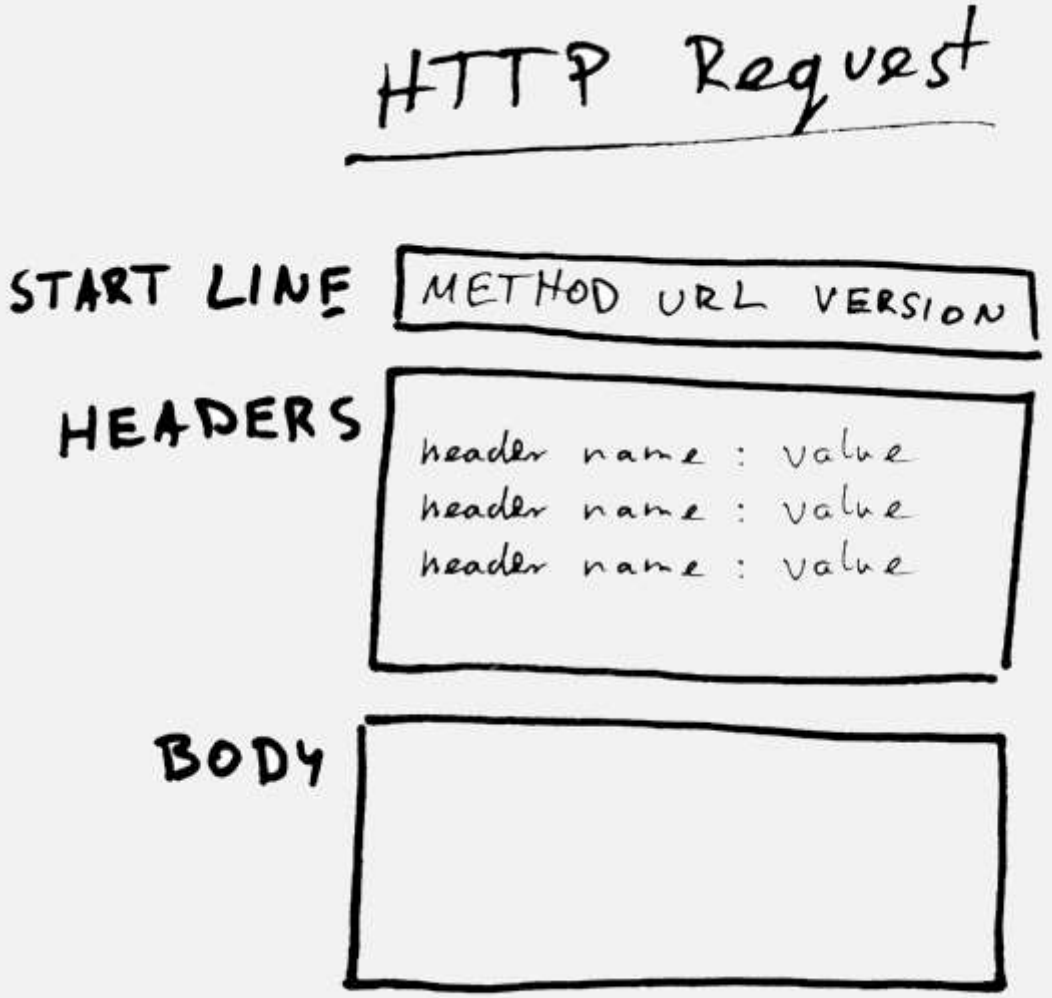
HTTP message



HTTP message



HTTP request message



HTTP request message

POST /test HTTP/1.1

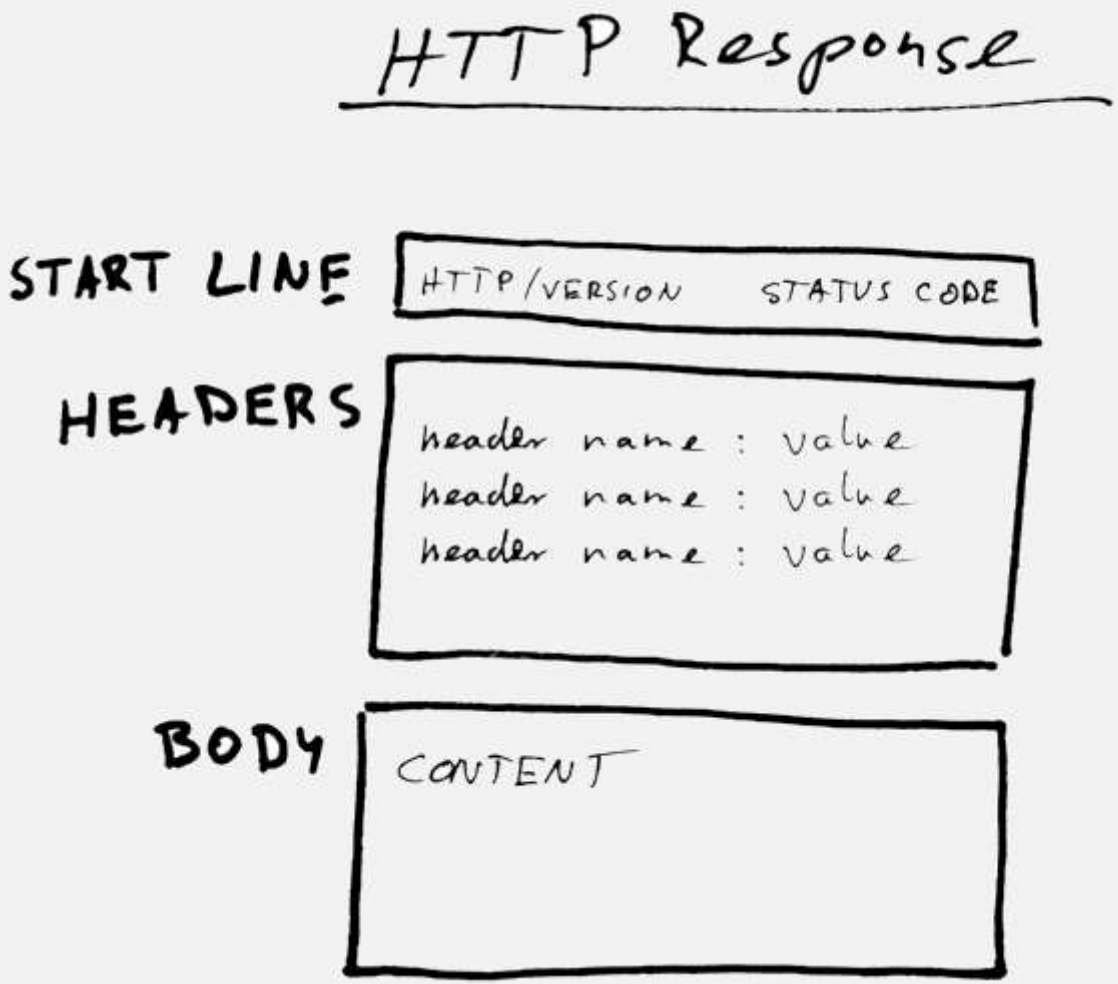
Host: foo.example

Content-Type: application/x-www-form-urlencoded

Content-Length: 27

field1=value1&field2=value2

HTTP response message



HTTP reponse message

HTTP/1.1 200 OK

Cache-Control: no-cache

Content-Type: application/json; charset=utf-8

Content-Length: 292

Date: Sat, 12 Jan 2019 07:11:48 GMT

{ ... user data }

HTTP methods

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource.

Each of them implements a different semantic.

Some common features are shared by a group of them: e.g. a request method can be safe, idempotent, or cacheable.

GET

The GET method is used to request data from the specified resource.

Requests using the GET method should only retrieve the data.

Examples:

GET /users

GET /users/1

HEAD

The HEAD method asks for a response identical to that of a GET request, but without the response body. Used to check the availability of the resource.

Examples:

HEAD /users

HEAD /users/1

POST

The POST method sends a request for the server to accept the entity contained in the request as a new object identified by the sent URL.

Example:

POST /users

```
{
  "firstname" : "Jan",
  "lastname" : "Kowalski"
}
```

PUT

The PUT method sends a request for the object to be stored under the provided URL identifier.

If the URL identifier refers to an already existing resource it is modified, if the URL identifier does not point to an existing resource then the server can create a resource with that URL identifier.

Example:

```
PUT /users/10
{
  "firstname" : "Jan",
  "lastname" : "Kowalski"
}
```

PATCH

The PATCH method is used for partial modifications of the resource

Example:

PATCH /users/10

```
{
  "lastname" : "Nowak"
}
```

DELETE

The DELETE method deletes a specific resource

Example:

DELETE /users/1

OPTIONS

The OPTIONS method returns the HTTP methods supported by the server for the specified URL.

Examples:

OPTIONS /users

OPTIONS /users/1

TRACE

The TRACE method performs a message loop-back test along the path to the target resource, providing a useful debugging mechanism.

Examples:

TRACE /users

TRACE /users/1

CONNECT

The CONNECT method establishes a tunnel to the server identified by the target resource.

Idempotency

Multiple method calls under the same initial conditions will always have the same result.

METHOD	IDEMPOTENT
GET	YES
HEAD	YES
POST	NO
PUT	YES
DELETE	YES
CONNECT	NO
OPTIONS	YES
TRACE	YES
PATCH	NO

Safety [not security]

{Brak modyfikacji danych oraz skutków ubocznych}

METHOD	IS SAFE
GET	YES
HEAD	YES
POST	NO
PUT	NO
DELETE	NO
CONNECT	NO
OPTIONS	YES
TRACE	YES
PATCH	NO

Cacheability

A cacheable response is an HTTP response that can be cached, that is stored to be retrieved and used later, saving a new request to the server.

METHOD	IS CACHEABLE
GET	YES
HEAD	YES
POST	YES *
PUT	NO
DELETE	NO
CONNECT	NO
OPTIONS	NO
TRACE	NO
PATCH	NO

Body

Not every method can use body

METHOD	REQUEST HAS BODY	RESPONSE HAS BODY
GET	OPTIONAL	YES
HEAD	OPTIONAL	NO
POST	YES	YES
PUT	YES	YES
DELETE	OPTIONAL	YES
CONNECT	OPTIONAL	YES
OPTIONS	OPTIONAL	YES
TRACE	NO	YES
PATCH	YES	YES

HTTP headers

- contain meta-data for the request
- may be present in any method
- have form key:value
- there are headers with defined meanings that should result in specific behavior of the receiving side (client or server)
- developers can define their own headers and give them meanings
- custom ones should contain x- prefix
- there are no restrictions in the standard, but most servers only allow a certain number of headers of a predefined size

* letter size of the headers is ignored.

HTTP headers request examples

- Accept - specifies the client's acceptable response format. Accept: application/json
- Accept-Language - customer-accepted response language. Accept-Language: pl-PL
- Host – server domain hame. Host:wp.pl
- User-Agent – client identifier. Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.101 Safari/537.36
- Cookie – cookie related to request

HTTP headers response examples

- Server - the name of the server. Server:Apache/2.4.10 (Unix) OpenSSL/1.0.1e-fips mod_bwlimited/1.4
- Date – date and time when response was sent. Date:Sat, 21 Oct 2017 10:44:53 GMT
- Content-type – type of returned data. Content-type: text/html; charset=UTF-8
- Set-cookie - command for the browser to set a value in a cookie.
- Content-length – response size in bytes
- Location - address server wants the browser to redirect to Location: https://www.wp..pl/

HTTP content type

Content Type (MIME) – Multipurpose Internet Mail Extensions

- Application / JSON
- Application / XML
- Image / JPG (GIF, PNG)
- Text / Plain
- … and many more

HTTP status code

- 1XX - information codes
- 2XX - success codes
- 3XX - redirect codes
- 4XX - client application error codes
- 5XX - HTTP server error codes

HTTP status code 1xx

- 100 Continue
- 101 Switching Protocols
- 110 Connection Timed Out
- 111 Connection refused

HTTP status code 2xx

- 200 OK
- 201 Created
- 202 Accepted
- 204 No Content

HTTP status code 3xx

- 301 Moved Permanently
- 302 Moved Temporarily
- 303 See Other
- 304 Not Modified

HTTP status code 4xx

- 400 Bad Reqest
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable
- 415 Unsupported Media Type

HTTP status code 5xx

- 500 Internal Server Error
- 503 Service Unavailable
- 504 Gateway Timeout

HTTP status code ☺

- 402 - Payment required
- 418 - I’m a teapot
- 451 - Unavailable For Legal Reason

HTTPS

S in HTTP stands for „secure” ☺

HTTP is not a secure protocol - it sends information in plain text in an unencrypted way

This is sufficient to display simple pages or even complex services as long as they do not display or send sensitive data to them

The solution is to use the HTTPS extension (port 443)

HTTPS uses cryptographic mechanisms to ensure the confidentiality of transmitted information, both symmetric and asymmetric algorithms, as well as certificates and PKI


250MM
LUX-TOOLS

{your-favourite-browser-name} Dev Tools

It is used to preview requests, the DOM tree, JS code, security settings or performance testing

Allows you to simulate a poor connection, as well as the operation of the application completely offline

You can check how the application will behave on a particular screen resolution or on a cell phone

POSTMAN

Convenient yet powerful graphical tool for testing http communication

HTTP client as a desktop application.

Allows you to easily build HTTP requests using header methods, bodies.

Saves executed requests so it is easy to replay them.

https://www.postman.com

Fiddler

A tool that allows you to monitor, intercept, debug and modify HTTP/HTTPS traffic between your computer and web applications.

It is able to make the life of network administrators much easier, all connections are described in detail and the interface gives you the possibility to set up breakpoints or use scripts, among other things.

only for Windows ☹

https://www.telerik.com/fiddler

Library

- https://developer.mozilla.org/en-US/docs/Web/HTTP
- https://tools.ietf.org/html/rfc2616
- Michał Zalewski – „Splątana sieć” / „The Tangled Web”

Sample APIs:

- https://github.com/public-apis/public-apis
- https://public-apis.xyz