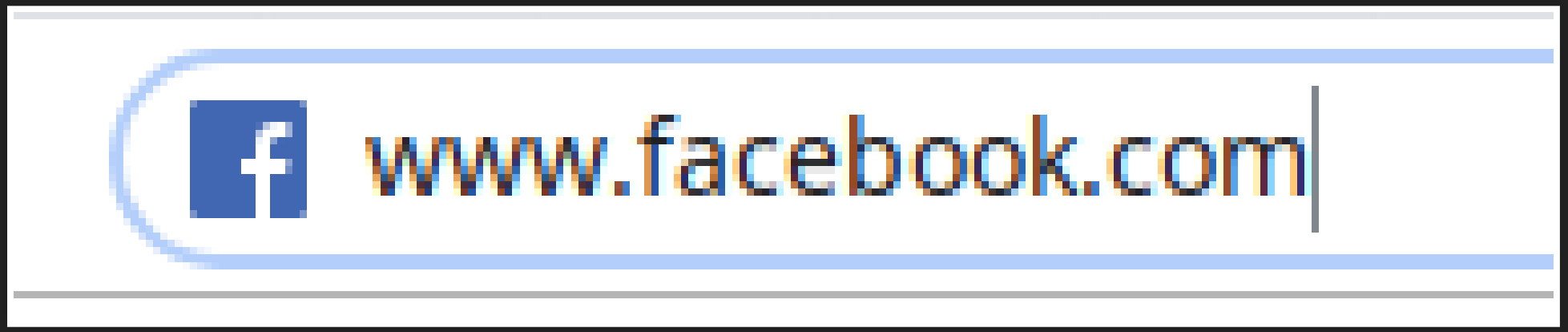
HTTP, REST API, JSON & THE WEB (ALSO REGEXES)

What happens when you press enter?



1. Browser -> DNS: what's the IP of facebook.com?

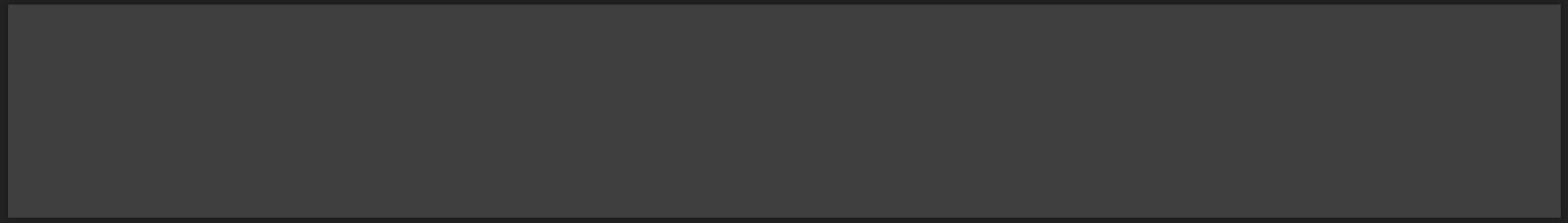
1. Browser -> DNS: what's the IP of facebook.com?
2. DNS -> Browser: 4.4.4.4

1. Browser -> DNS: what's the IP of facebook.com?
2. DNS -> Browser: 4.4.4.4
3. Browser -> 4.4.4.4: send HTTP request

1. Browser -> DNS: what's the IP of facebook.com?
2. DNS -> Browser: 4.4.4.4
3. Browser -> 4.4.4.4: send HTTP request
4. 4.4.4.4 -> Browser: send HTTP response

HTTP request

HTTP request



HTTP request

```
GET
```

HTTP request

```
GET /
```

HTTP request

```
GET / HTTP/1.1
```

HTTP request

```
GET / HTTP/1.1
Host: www.facebook.com
```

HTTP request

```
GET / HTTP/1.1
Host: www.facebook.com
Accept-Language: en

```

HTTP request

```
GET / HTTP/1.1
Host: www.facebook.com
Accept-Language: en
Accept-Encoding: gzip
```

HTTP request

```
GET / HTTP/1.1
Host: www.facebook.com
Accept-Language: en
Accept-Encoding: gzip
```

HTTP response

HTTP request

```
GET / HTTP/1.1
Host: www.facebook.com
Accept-Language: en
Accept-Encoding: gzip
```

HTTP response

```

```

HTTP request

```
GET / HTTP/1.1
Host: www.facebook.com
Accept-Language: en
Accept-Encoding: gzip
```

HTTP response

```
HTTP/1.1 200 OK
```

HTTP request

```
GET / HTTP/1.1
Host: www.facebook.com
Accept-Language: en
Accept-Encoding: gzip
```

HTTP response

```
HTTP/1.1 200 OK
Content-Length: 29769
```

HTTP request

```
GET / HTTP/1.1
Host: www.facebook.com
Accept-Language: en
Accept-Encoding: gzip
```

HTTP response

```
HTTP/1.1 200 OK
Content-Length: 29769
Content-Type: text/html
```

HTTP request

```
GET / HTTP/1.1
Host: www.facebook.com
Accept-Language: en
Accept-Encoding: gzip
```

HTTP response

```
HTTP/1.1 200 OK
Content-Length: 29769
Content-Type: text/html

<html>...</html>
```

HTTP methods

HTTP methods

1. GET - read data

HTTP methods

1. GET - read data
2. POST - update/insert data

HTTP methods

1. GET - read data
2. POST - update/insert data
3. DELETE - delete data

HTTP methods

1. GET - read data
2. POST - update/insert data
3. DELETE - delete data
4. PUT, PATCH, HEAD, OPTIONS, ...

Passing data to server

Passing data to server

Params in URL (usually for GET)

Passing data to server

Params in URL (usually for GET)

```
GET /page?param1=a&param2=b
```

Passing data to server

Params in URL (usually for GET)

```
GET /page?param1=a&param2=b
```

Data in body (usually for POST)

Passing data to server

Params in URL (usually for GET)

```
GET /page?param1=a&param2=b
```

Data in body (usually for POST)

```
POST /page
...

hello
```

HTTP status codes

HTTP status codes

- 2xx - everything is fine

HTTP status codes

- 2xx - everything is fine
- 3xx - redirection

HTTP status codes

- 2xx - everything is fine
- 3xx - redirection
- 4xx - client error

HTTP status codes

- 2xx - everything is fine
- 3xx - redirection
- 4xx - client error
- 5xx - server error

HTTP status codes

- 2xx - everything is fine
- 3xx - redirection
- 4xx - client error
- 5xx - server error
- 403 - forbidden

HTTP status codes

- 2xx - everything is fine
- 3xx - redirection
- 4xx - client error
- 5xx - server error
- 403 - forbidden
- 404 - not found

HTTP status codes

- 2xx - everything is fine
- 3xx - redirection
- 4xx - client error
- 5xx - server error
- 403 - forbidden
- 404 - not found
- 500 - internal server error

HTTP status codes

- 2xx - everything is fine
- 3xx - redirection
- 4xx - client error
- 5xx - server error

- 403 - forbidden
- 404 - not found
- 500 - internal server error
- 418 - I'm a teapot ☕

API (APPLICATION PROGRAMMING INTERFACE)

API (APPLICATION PROGRAMMING INTERFACE)

- interface for using code

API (APPLICATION PROGRAMMING INTERFACE)

- interface for using code
- simplifies usage of complex code

API (APPLICATION PROGRAMMING INTERFACE)

- interface for using code
- simplifies usage of complex code
- hides inner implementation of the code

API (APPLICATION PROGRAMMING INTERFACE)

- interface for using code
- simplifies usage of complex code
- hides inner implementation of the code

REST API

API (APPLICATION PROGRAMMING INTERFACE)

- interface for using code
- simplifies usage of complex code
- hides inner implementation of the code

REST API

- Idea: represent data as resources (URLs)

API (APPLICATION PROGRAMMING INTERFACE)

- interface for using code
- simplifies usage of complex code
- hides inner implementation of the code

REST API

- Idea: represent data as resources (URLs)
- Operate on resources using HTTP requests

API (APPLICATION PROGRAMMING INTERFACE)

- interface for using code
- simplifies usage of complex code
- hides inner implementation of the code

REST API

- Idea: represent data as resources (URLs)
- Operate on resources using HTTP requests
- Intention of requests is given by HTTP method

Example: provide access to a database of users

Example: provide access to a database of users

Endpoint: /users

Example: provide access to a database of users

Endpoint: /users

- GET /users - download all users

Example: provide access to a database of users

Endpoint: /users

- GET /users - download all users
- GET /users/1 - download user with ID 1

Example: provide access to a database of users

Endpoint: /users

- GET /users - download all users
- GET /users/1 - download user with ID 1
- POST /users - create new user

Example: provide access to a database of users

Endpoint: /users

- GET /users - download all users
- GET /users/1 - download user with ID 1
- POST /users - create new user
- DELETE /users - delete user with ID 1

How is the data transferred?

How is the data transferred?

- Char. encoding: typically UTF-8

How is the data transferred?

- Char. encoding: typically UTF-8
- URL encoding: /hello world -> /hello%20world

How is the data transferred?

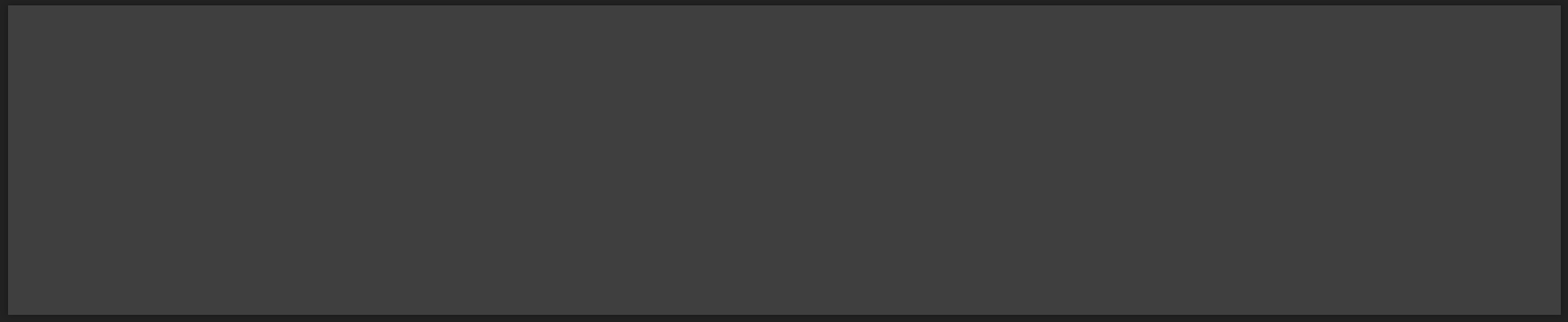
- Char. encoding: typically UTF-8
- URL encoding: /hello world -> /hello%20world
- Data format: historically XML, nowadays JSON

JSON (example)

```
{
  "course": "SPJA",
  "teacher": "Kuba Beránek",
  "year": 2018,
  "attendants": ["Martin Novák", "František Dobrota"]
}
```

JSON (data types)

JSON (data types)



JSON (data types)

```
1
```

JSON (data types)

```
1
"hello"
```

JSON (data types)

```
1
"hello"
true
```

JSON (data types)

```
1
"hello"
true
null
```

JSON (data types)

```
1
"hello"
true
null
[1, 2, ["3", 4, true]]
```

JSON (data types)

```
1
"hello"
true
null
[1, 2, ["3", 4, true]]
{"name": "James", "children": [...]}
```

requests library

```
$ pip install requests
```

Generated request

requests library

```
$ pip install requests
```

```
import requests, json
```

Generated request

```
```

requests library

```
$ pip install requests
```

```
import requests, json
res = requests.post(
```

Generated request

```
POST
```

requests library

```
$ pip install requests
```

```
import requests, json
res = requests.post("github.com/users",
```

Generated request

```
POST /users HTTP/1.1
Host: github.com
```

requests library

```
$ pip install requests
```

```
import requests, json
res = requests.post("github.com/users",
  headers={"Content-Type": "application/json"},
```

Generated request

```
POST /users HTTP/1.1
Host: github.com
Content-Type: application/json
```

requests library

```
$ pip install requests
```

```
import requests, json
res = requests.post("github.com/users",
  headers={"Content-Type": "application/json"},
  data=json.dumps({"param": 1})
)
```

Generated request

```
POST /users HTTP/1.1
Host: github.com
Content-Type: application/json

{"param": 1}
```

requests library

```
$ pip install requests
```

```
import requests, json
res = requests.post("github.com/users",
  headers={"Content-Type": "application/json"},
  data=json.dumps({"param": 1})
)
res.content  # raw binary data
```

Generated request

```
POST /users HTTP/1.1
Host: github.com
Content-Type: application/json

{"param": 1}
```

requests library

```
$ pip install requests
```

```
import requests, json
res = requests.post("github.com/users",
  headers={"Content-Type": "application/json"},
  data=json.dumps({"param": 1})
)
res.content  # raw binary data
res.json     # parsed JSON data
```

Generated request

```
POST /users HTTP/1.1
Host: github.com
Content-Type: application/json

{"param": 1}
```

requests library

```
$ pip install requests
```

```
import requests, json
res = requests.post("github.com/users",
  headers={"Content-Type": "application/json"},
  data=json.dumps({"param": 1})
)
res.content  # raw binary data
res.json     # parsed JSON data
res.text     # text data
```

Generated request

```
POST /users HTTP/1.1
Host: github.com
Content-Type: application/json

{"param": 1}
```

OAuth client

```
$ pip install oauth2
```

```
client = create_client()
res, data = client.request("github.com/api",
  method="POST", headers=...)
assert res['status'] == 200
```

REGULAR EXPRESSIONS

REGULAR EXPRESSIONS

standardized language for searching and manipulating text

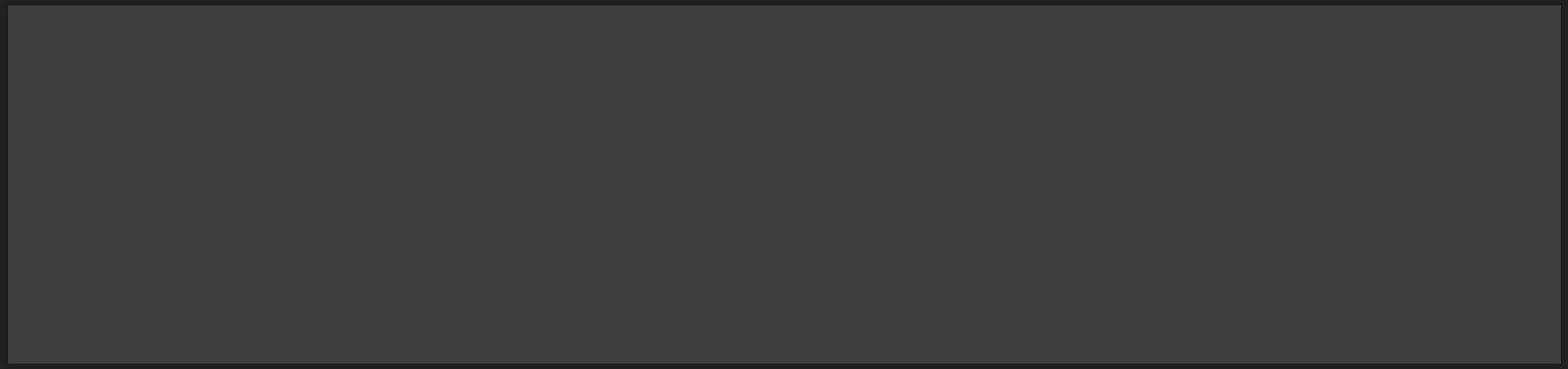
REGULAR EXPRESSIONS

standardized language for searching and manipulating text

```
(?:\w|-)*@(?:\w|-)\.\w{2-4}
```

Regex intro (https://regex101.com/)

```
abc          match literal characters 'abc'
.            match anything
\s           match whitespace
\d           match digit (1, 2, etc.)
[abc]        match any character from the brackets (a, b or c)
\d*          match digit 0 or more times
\d+          match digit 1 or more times
\d{1,3}      match digit between 1-3 times
(x.*)        capture text inside the parantheses
a|b          match a or b
(A|B)*       match repeated As and Bs
```



```
1) Vepřo knedlo zelo 115,-
```

```
(\d)\)\s*(.*?)\s*(\d+),-
```

```

```

```
1) Vepřo knedlo zelo 115,-
```

```
(\d)\)\s*(.*?)\s*(\d+),-
```

```
(\d)   zaznamenej číslici (ID jídla)
```

```
1) Vepřo knedlo zelo 115,-
```

```
(\d)\)\s*(.*?)\s*(\d+),-
```

```
(\d)   zaznamenej číslici (ID jídla)
\)     následovanou znakem konce závorky ')'
```

```
1) Vepřo knedlo zelo 115,-
```

```
(\d)\)\s*(.*?)\s*(\d+),-
```

```
(\d)   zaznamenej číslici (ID jídla)
\)     následovanou znakem konce závorky ')'
\s*    ignoruj mezery
```

```
1) Vepřo knedlo zelo 115,-
```

```
(\d)\)\s*(.*?)\s*(\d+),-
```

```
(\d)   zaznamenej číslici (ID jídla)
\)     následovanou znakem konce závorky ')'
\s*    ignoruj mezery
(.*?)  zaznamenej cokoliv, co najdeš (název jídla)
```

```
1) Vepřo knedlo zelo_115,-
```

```
(\d)\)\s*(.*?)\s*(\d+),-
```

```
(\d)   zaznamenej číslici (ID jídla)
\)     následovanou znakem konce závorky ')'
\s*    ignoruj mezery
(.*?)  zaznamenej cokoliv, co najdeš (název jídla)
\s*    ignoruj mezery
```

```
1) Vepřo knedlo zelo 115,-
```

```
(\d)\)\s*(.*?)\s*(\d+),-
```

```
(\d)   zaznamenej číslici (ID jídla)
\)     následovanou znakem konce závorky ')'
\s*    ignoruj mezery
(.*?)  zaznamenej cokoliv, co najdeš (název jídla)
\s*    ignoruj mezery
(\d+)  zaznamenej sekvenci číslic (cena)
```

```
1) Vepřo knedlo zelo 115,-
```

```
(\d)\)\s*(.*?)\s*(\d+),-
```

```
(\d)   zaznamenej číslici (ID jídla)
\)     následovanou znakem konce závorky ')'
\s*    ignoruj mezery
(.*?)  zaznamenej cokoliv, co najdeš (název jídla)
\s*    ignoruj mezery
(\d+)  zaznamenej sekvenci číslic (cena)
,-     následovanou znaky ',-'
```

Regex in Python

Regex in Python

```
import re
```

Regex in Python

```
import re
regex = re.compile('param="([\w/.]+)"')
```

Regex in Python

```
import re
regex = re.compile('param="([\w/.]+)"')
matches = regex.findall('param="images/data.jpg"')
```

Regex in Python

```
import re
regex = re.compile('param="([\w/.]+)"')
matches = regex.findall('param="images/data.jpg"')
# ['images/data.jpg']
```