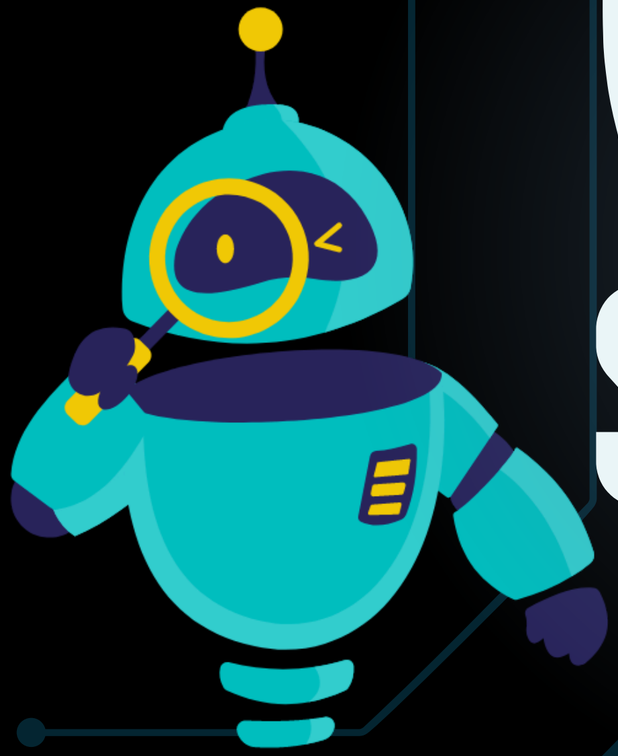


WEB SCRAPING

Jakub Melzacki

AGENDA

TEORIA:

1. Co to?
2. Komu to potrzebne?
3. Legelność
4. Co trzeba znać?
5. Scraper, a Crawler
6. Scrapowanie
7. Biblioteki

SCRAPiNG

Web scraping — to automatyczne pobieranie danych z publicznie dostępnych stron internetowych, w celu ich zapisania i dalszego wykorzystania (np. w analizach, bazach danych czy aplikacjach).


YOU MERELY ADOPTED WEB SCRAPING
I WAS BORN INTO IT, MOLDED BY IT
makeameme.org

The 21 year old Scraper

Scrapes web pages 'to collect data'

Doesn't know what an API is

Keeps CloudFlare and other anti-bot services on their toes

Doesn't properly cache scraped data, often leading to tonnes of requests per second which overloads smaller websites

Forgot English, now only knows XPath

"Is that a webpage? OH GOD I'M GONNA SCRAAAAAAAAPE"

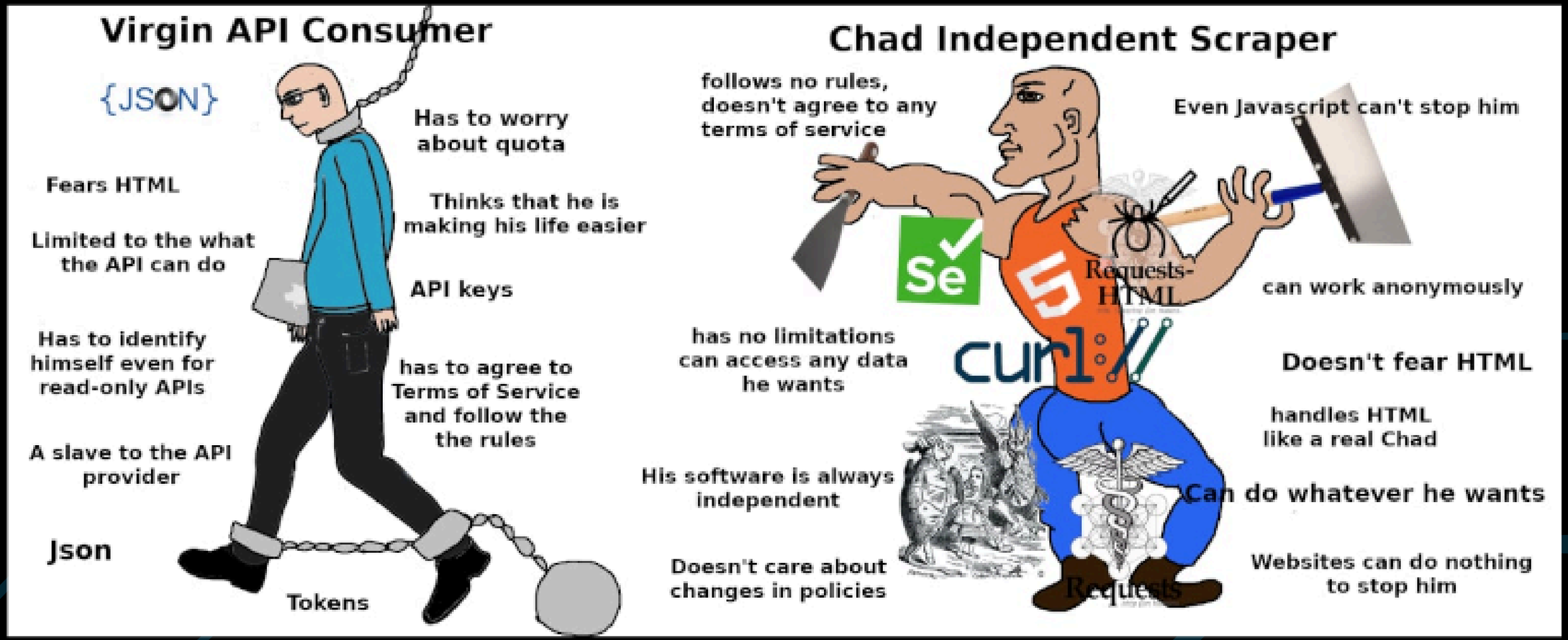
Somehow managed to evade captcha systems - Google is in shock

Retinas cooked to a crisp due to staring at an IDE all day

"Current time? You mean `//*[@id="last-update"]/time/@datetime`?"

Analytics companies fear him

A NA CO TO KOMU?



A NA CO TO KOMU?

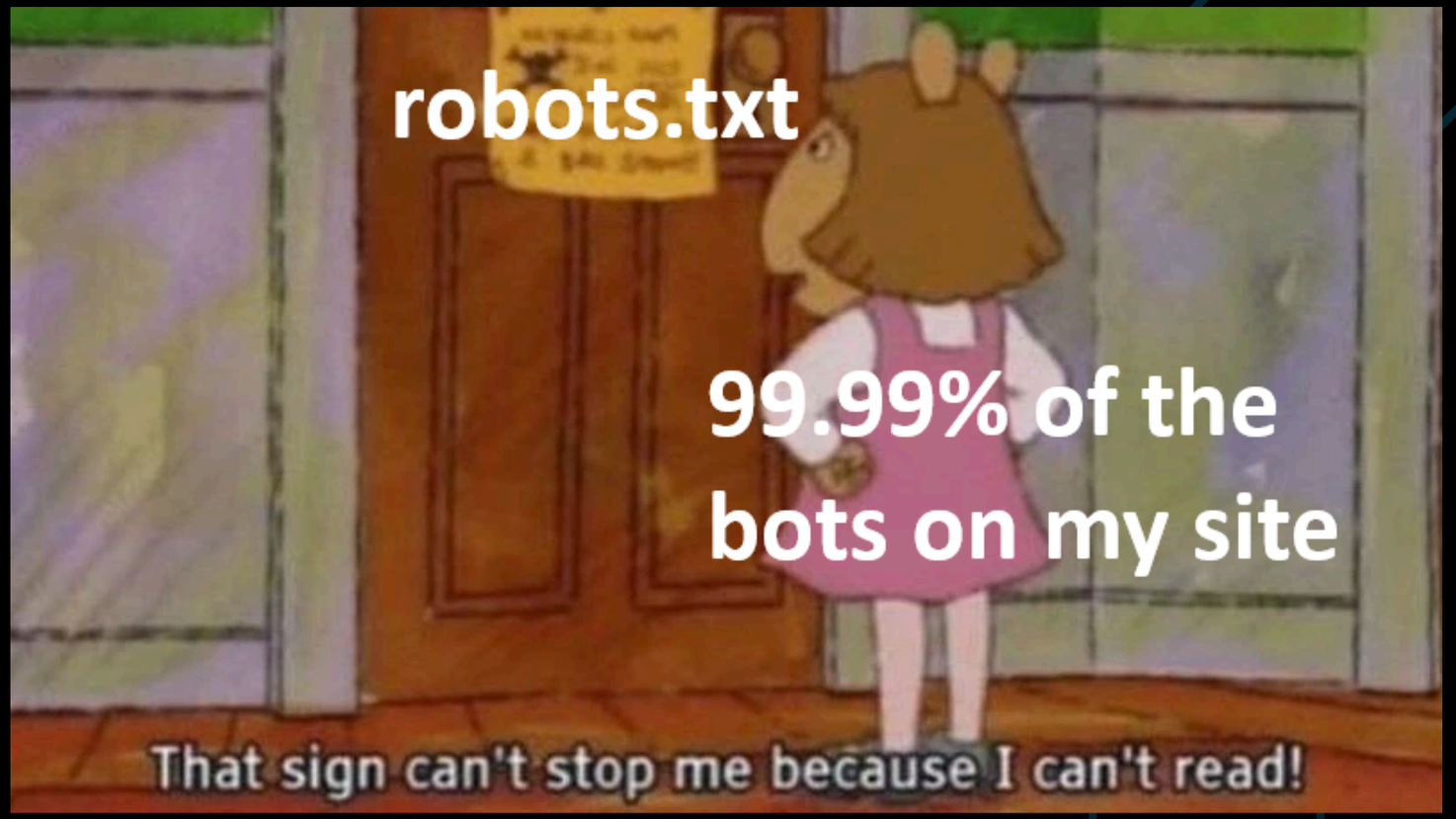
BEZ DANYCH NIE MA DANYCH

PRACA DYPLOMOWA

LIMITY I OGRANICZENIA NIE INTERESUJĄ MNIE!

AUTOMATYZACJA

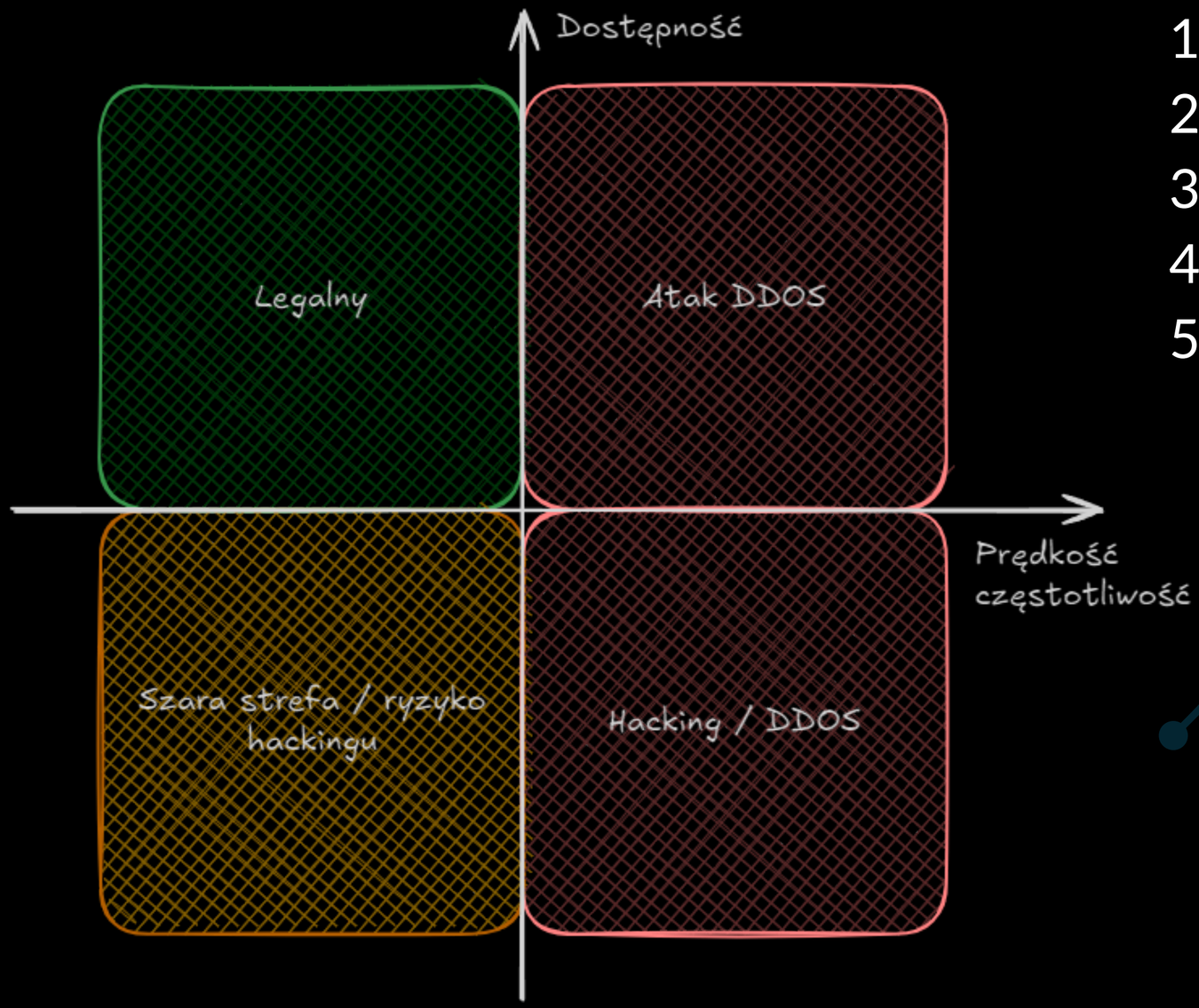
A TO WGL LEGALNE?



A TO WGL LEGALNE?

TAK, ALE...

TECHNICZNIE



PRAWNIE

1. Ochrona danych osobowych (RODO)
2. Własność intelektualna
3. Prawo konkurencji
4. Bezpieczeństwo systemów IT
5. Zakłócanie działania usług

CO TRZEBA ZNAĆ?

```html
<!DOCTYPE html>
<html lang="pl">
<body>
    <h1 id="head">Example Header</h1>
    <p>This is an example paragraph.</p>
    <div class="data-list">
        <div class="data-item">Item 1</div>
        <div class="data-item">Item 2</div>
        <div class="data-item">Item 3</div>
    </div>
    <div>
        <a href="next_page.html">Next page</a>
    </div>
</body>
```

1.HTML

1. Tagi
2. Struktura
3. Atrybuty

2.NAWIGACJA

```html
<!DOCTYPE html>
<html lang="pl">
<body>
    <h1 id="head">Example Header</h1>
    <p>This is an example paragraph.</p>
    <div class="data-list">
        <div class="data-item">Item 1</div>
        <div class="data-item">Item 2</div>
        <div class="data-item">Item 3</div>
    </div>
    <div>
        <a href="next_page.html">Next page</a>
    </div>
</body>
```

CSS

```css
#head
.data-item
.data-list > .data-item
.data-list > .data-item:first-child
.data-list > .data-item:nth-child(3)
div > a
a[href="next_page.html"]
#head + p
```

XPATH

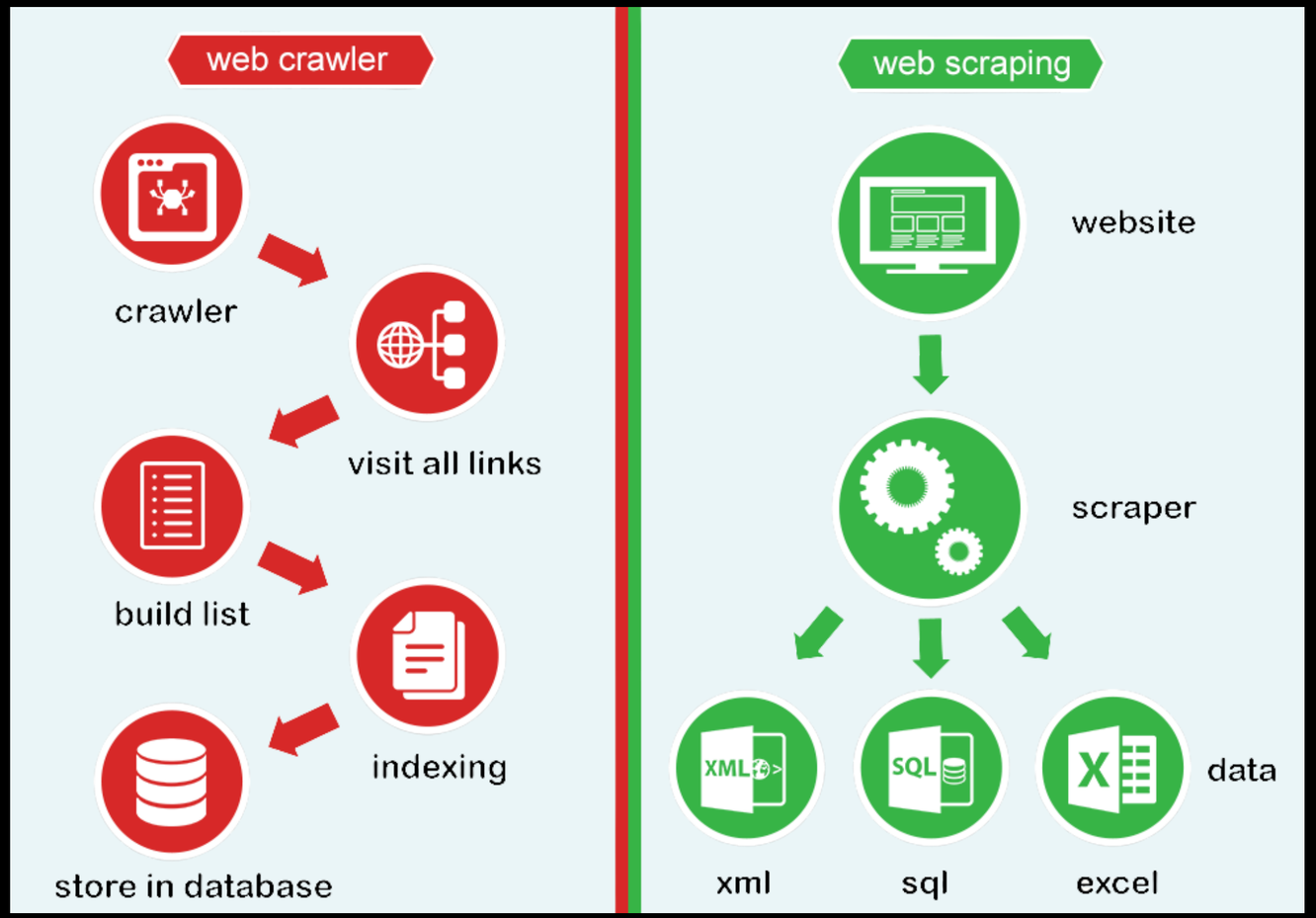
```xpath
//h1[@id="head"]
//p
//*[@class="data-item"]
//div[@class="data-list"]/div[@class="data-item"]
(//div[@class="data-list"]/div[@class="data-item"])[1]
(//div[@class="data-list"]/div[@class="data-item"])[3]
//body/div[last()]/a
//a[@href="next_page.html"]
//h1[@id="head"]/following-sibling::p[1]
```

3.API

4.REGEX

```
\d+
[A-Z]{3}
\w+@\w+\.\w+
\d{4}-\d{2}-\d{2}
items/(\d+)/view
```

CRAWLER, A SCRAPER



SCRAPOWANIE

SCRAPOWANIE

DANE DOSTARCZANE PRZEZ API

DANE UMIESZCZONE W JS

DANE UMIESZCZONE W HTML

DANE Z API

NIEZABEZPIECZONE API → WYKORZYSTUJEMY API BEZPOŚRENIO

LEKKO ZABEZPIECZONE API → WYKORZYSTUJEMY API BEZPOŚREDNIO

AUTH ZABEZPIECZONE API → WYŁAPUJEMY REQUEST Z PRZEGLĄDARKI

MOCNO ZABEZPIECZONE API → RENDERUJEMY I SCRAPUJEMY HTML

DANE Z JS

DANE SĄ W POSTACI JSON → WYSZUKUJEMY JSON i PARSUJEMY

DANE SĄ "ZASZYFROWANE" → DEKODUJEMY I PARSUJEMY

BRAK KLUCZA → RENDERUJEMY I SCRAPUJEMY HTML

DANE Z HTML

BADAMY STRUKTURĘ STRONY

SZUKAMY OTOCZENIA NASZYCH DANYCH

SZUKAMY PUNKTÓW ORIENTACYJNYCH (UNIKALNE KLASY/ID CSS)

SZUKAMY SCIEŻKI DO DANYCH OD PUNKTÓW ORIENTACYNYCH

IMPLEMENTUJEMY ŚCIĘŻKĘ W BIBLIOTECE

CIESZYMY SIĘ Z NASZYCH DANYCH!

BIBLIOTEKI

REQUESTS

http for humans

```python
import requests

# przykład: pobieramy stronę główną PJATK
url = "https://pja.edu.pl"

response = requests.get(url)
response.raise_for_status()
# fragment HTML
print(response.text[:500])
```

PARSEL

```
from parsel import Selector
sel = Selector(text=site_html)

# --- CSS SELECTORS ---
header = sel.css("h1::text").get()
print("Nagłówek:", header)

items = sel.css(".data-item::text").getall()
print("Elementy:", items)

link = sel.css("a::attr(href)").get()
print("Link:", link)

# --- XPATH ---
header_x = sel.xpath("//h1/text()").get()
print("Nagłówek (XPath):", header_x)

items_x = sel.xpath("//div[@class='data-item']/text()").getall()
print("Elementy (XPath):", items_x)

link_x = sel.xpath("//a/@href").get()
print("Link (XPath):", link_x)
```

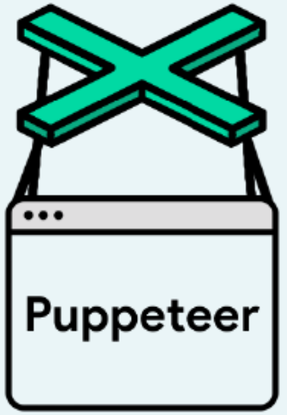
PLAYWRIGHT

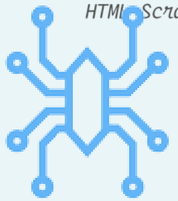
```
playwright install --with-deps chromium
```

```python
from playwright.sync_api import sync_playwright
import time

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    page = browser.new_page()
    page.goto("https://example.com")
    time.sleep(5)
    browser.close()
```

INNE BIBLIOTEKI





TWORZENIE ŚRODOWISKA WARSZTATÓW

```
python -m venv .venv
.\.venv\Scripts\activate # Windows
source .venv/Scripts/activate # Bash
pip install requests parsel playwright
playwright install --with-deps chromium
```

ZADANIA

API/JS

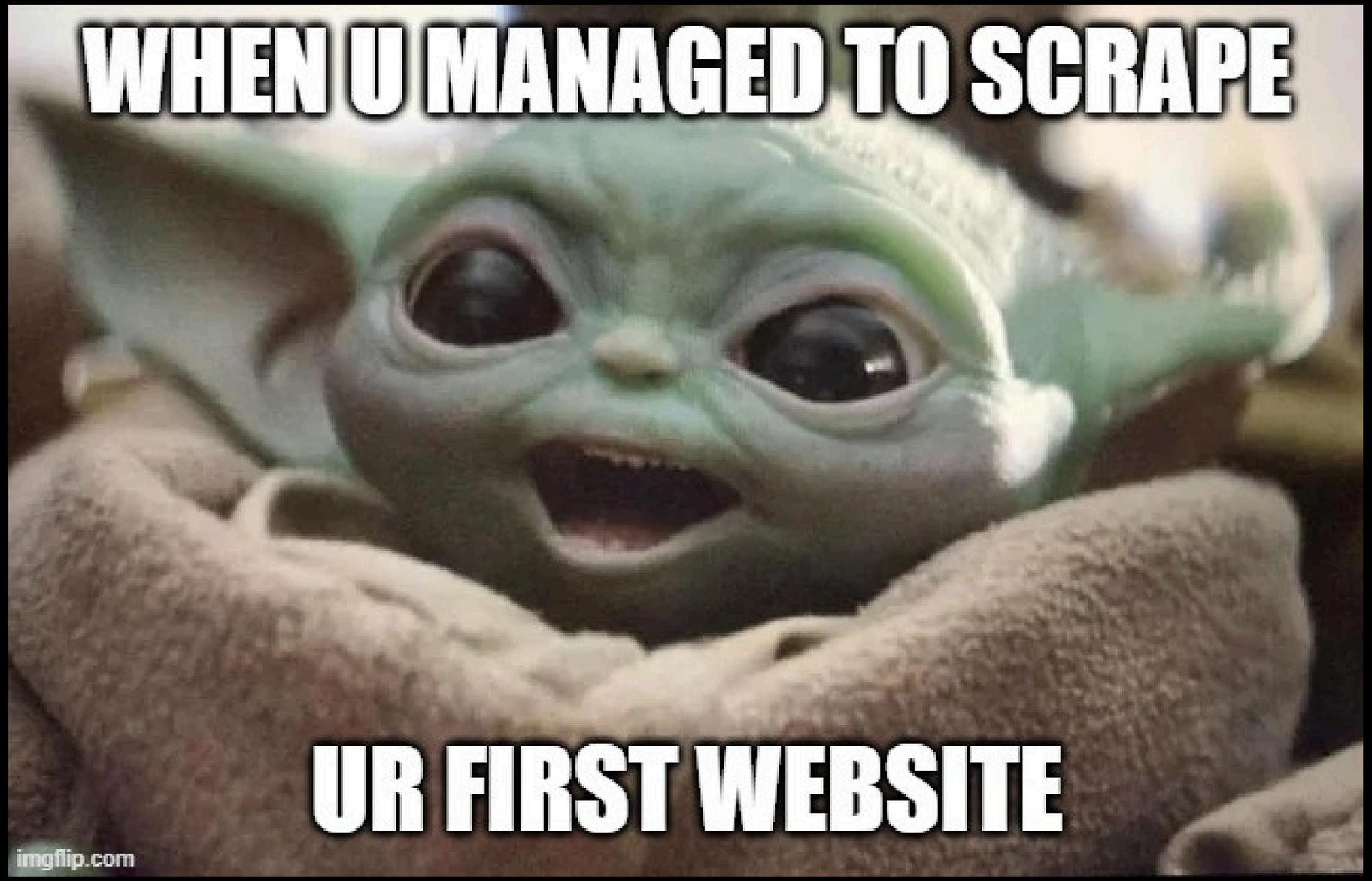
1. filmweb.pl/search#/film
2. imdb.com/event/ev0000003
3. idealwine.com
4. swiatksiazki.pl/ksiazki/ksiazki-3799.html
5. premierleague.com/en/matches

HTML

1. betclic.pl
2. bricklink.com/catalogList.asp
3. myanimelist.net/topanime.php
4. lubimyczytac.pl/katalog/ksiazki
5. gol.gg/tournament/list/

SPECJALNE STRONY

1. scrapethissite.com/pages/
2. toscrape.com


WHEN U MANAGED TO SCRAPE
UR FIRST WEBSITE
imgflip.com