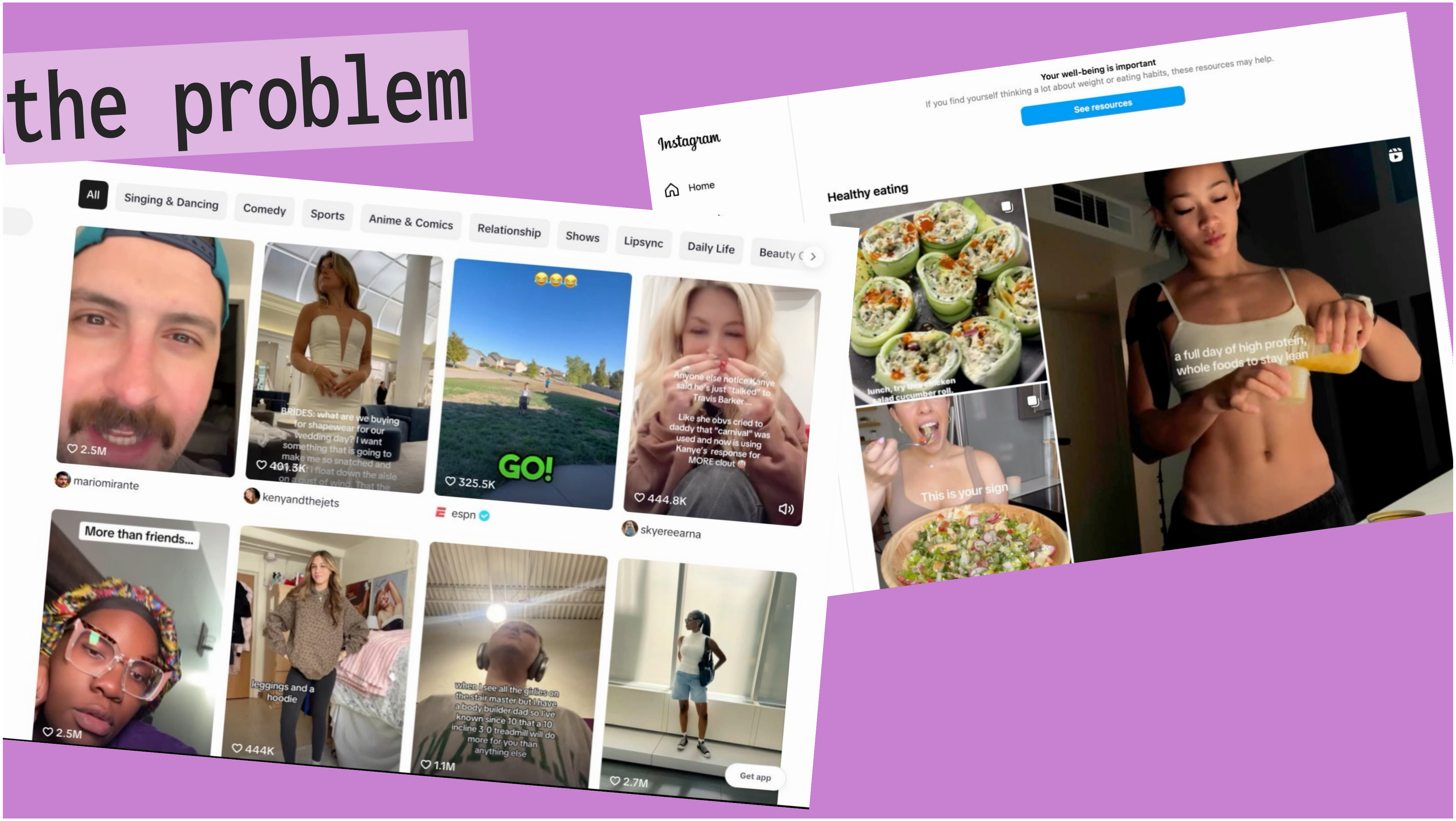
https://bit.ly/nicar-passive

Passive Scraping for social media

(and everything else)

Jonathan Soma
Columbia Journalism School
js4571@columbia.edu
@dangerscarf

the problem



♡ 2.5M mariomirante

BRIDES: what are we buying for shapewear for our wedding day? I want something that is going to make me so snatched and ... t float down the aisle on a gust of wind. That the

♡ 401.3K kenyandthejets

GO!

♡ 325.5K espn

Anyone else notice Kanye said he's just "talked" to Travis Barker ...

Like she obvs cried to daddy that "carnival" was used and now is using Kanye's response for MORE clout 😂

♡ 444.8K skyereearna

More than friends...

♡ 2.5M

leggings and a hoodie

♡ 444K

when I see all the girlies on the stair master but I have a body builder dad so I've known since 10 that a 10 incline 3.0 treadmill will do more for you than anything else

♡ 1.1M

♡ 2.7M

Your well-being is important

If you find yourself thinking a lot about weight or eating habits, these resources may help.

Healthy eating

lunch, try this chicken salad cucumber roll.

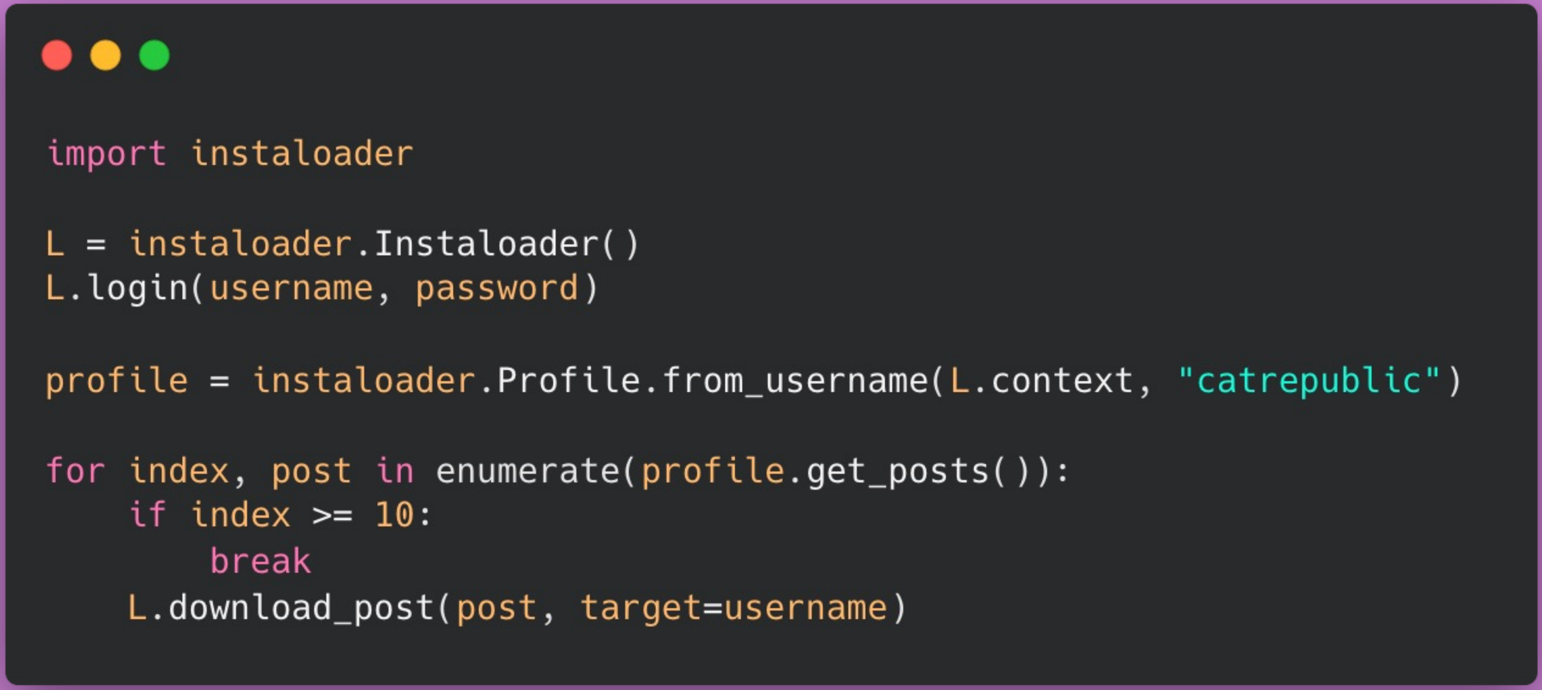
This is your sign

a full day of high protein, whole foods to stay lean

tiers of problem-solving

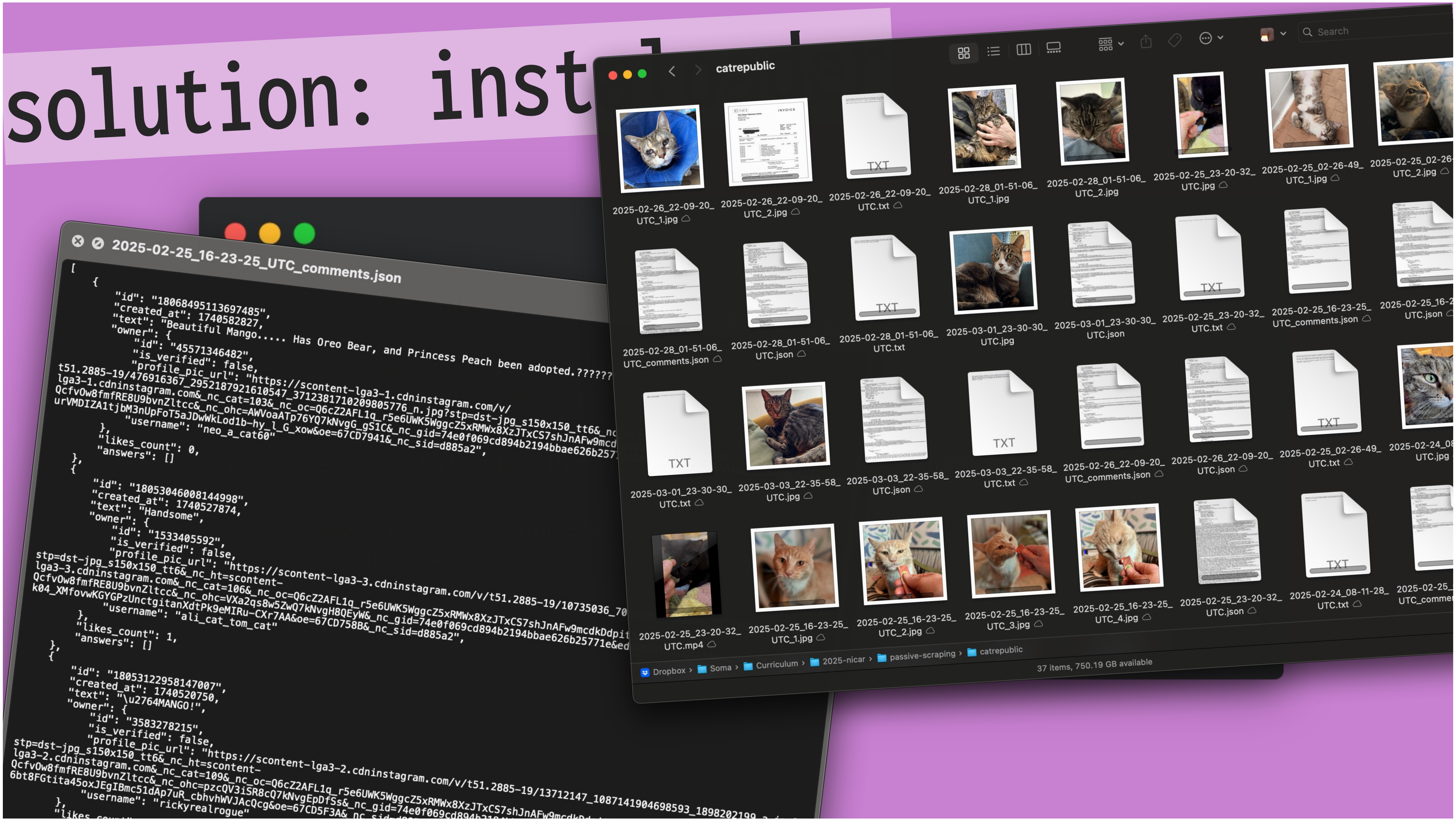
- Using a tool
- Writing a scraper
- Undocumented APIs
- Intercepting browser requests
- Pack-ratting with HAR and WARC/WACZ files

A solution: instaloader



```python
import instaloader

L = instaloader.Instaloader()
L.login(username, password)

profile = instaloader.Profile.from_username(L.context, "catrepublic")

for index, post in enumerate(profile.get_posts()):
    if index >= 10:
        break
    L.download_post(post, target=username)
```

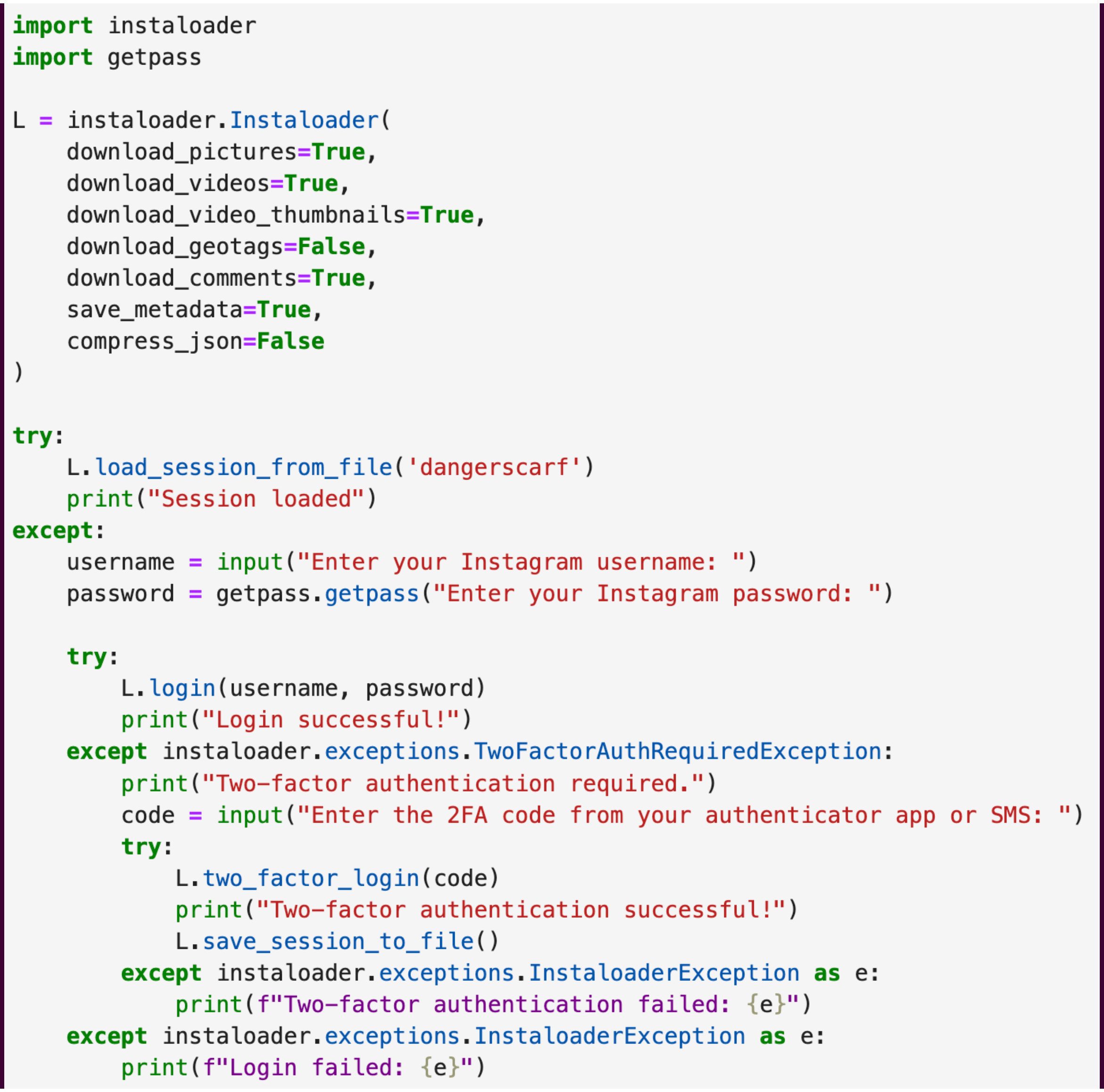
solution: inst



2025-02-25_16-23-25_UTC_comments.json

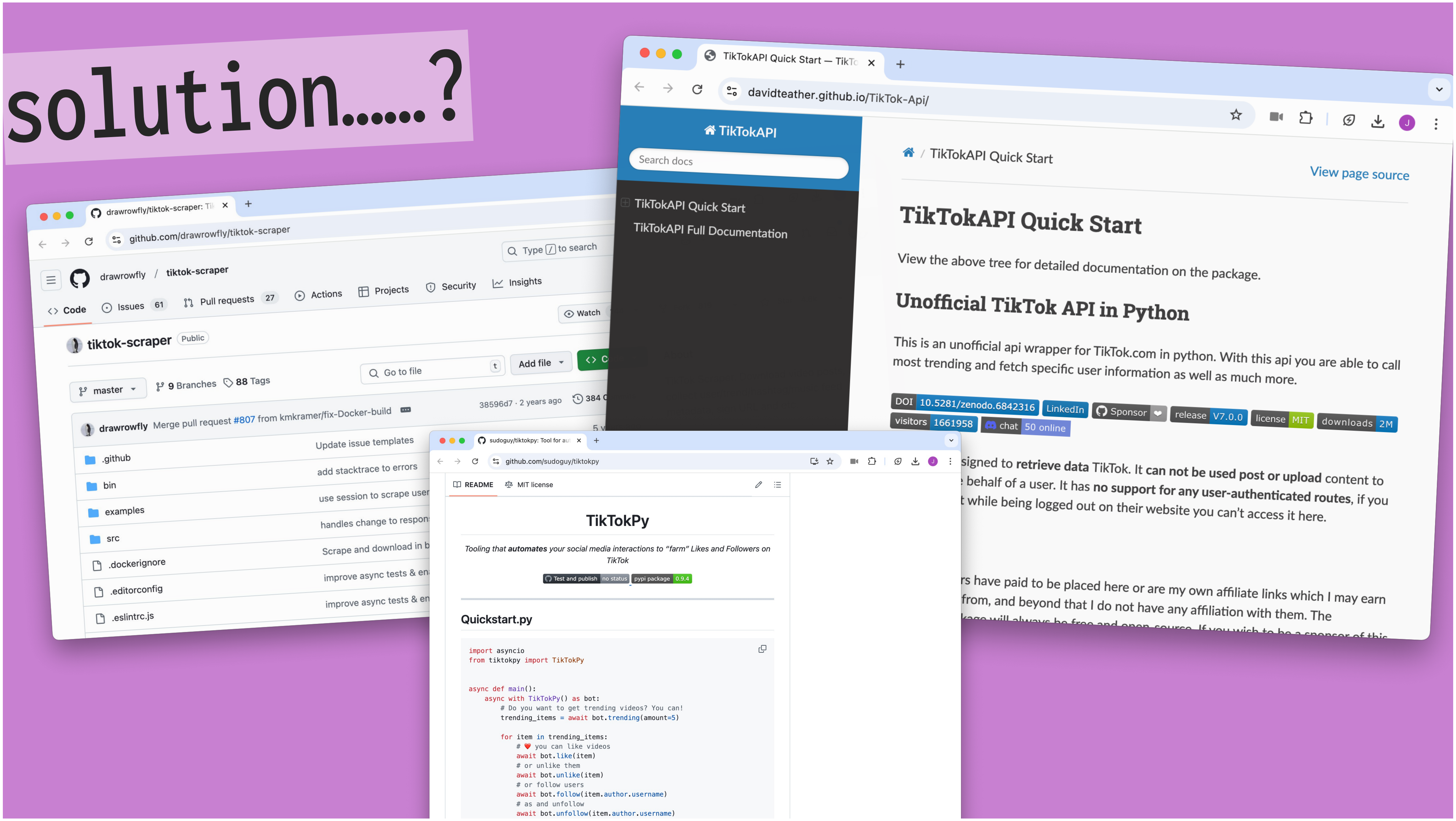
```
[
    {
        "id": "18068495113697485",
        "created_at": 1740582827,
        "text": "Beautiful Mango..... Has Oreo Bear, and Princess Peach been adopted.??????
        "owner": {
            "id": "45571346482",
            "is_verified": false,
            "profile_pic_url": "https://scontent-lga3-1.cdninstagram.com/v/
t51.2885-19/476916367_2952187921610547_3712381710209805776_n.jpg?stp=dst-jpg_s150x150_tt6&_nc
lga3-1.cdninstagram.com&_nc_cat=103&_nc_oc=Q6cZ2AFL1q_r5e6UWK5WggcZ5xRMWx8XzJTxCS7shJnAFw9mcd
QcfvOw8fmfRE8U9bvnZltcc&_nc_ohc=AWVoaATp76YQ7kNvgG_gS1C&_nc_gid=74e0f069cd894b2194bbae626b257
urVMDIZA1tjbM3nUpFoT5aJDwWkLod1b-hy_l_G_xow&oe=67CD7941&_nc_sid=d885a2",
            "username": "neo_a_cat60"
        },
        "likes_count": 0,
        "answers": []
    },
    {
        "id": "18053046008144998",
        "created_at": 1740527874,
        "text": "Handsome",
        "owner": {
            "id": "1533405592",
            "is_verified": false,
            "profile_pic_url": "https://scontent-lga3-3.cdninstagram.com/v/t51.2885-19/10735036_70
stp=dst-jpg_s150x150_tt6&_nc_ht=scontent-
lga3-3.cdninstagram.com&_nc_cat=106&_nc_oc=Q6cZ2AFL1q_r5e6UWK5WggcZ5xRMWx8XzJTxCS7shJnAFw9mcdkDdpit
QcfvOw8fmfRE8U9bvnZltcc&_nc_ohc=VXa2qs8w5ZwQ7kNvgH8QEyW&_nc_gid=74e0f069cd894b2194bbae626b25771e&ed
k04_XMfovwKGYGPzUnctgitanXdtPk9eMIRu-CXr7AA&oe=67CD758B&_nc_sid=d885a2",
            "username": "ali_cat_tom_cat"
        },
        "likes_count": 1,
        "answers": []
    },
    {
        "id": "18053122958147007",
        "created_at": 1740520750,
        "text": "\u2764MANGO!",
        "owner": {
            "id": "3583278215",
            "is_verified": false,
            "profile_pic_url": "https://scontent-lga3-2.cdninstagram.com/v/t51.2885-19/13712147_1087141904698593_1898202199
stp=dst-jpg_s150x150_tt6&_nc_ht=scontent-
lga3-2.cdninstagram.com&_nc_cat=109&_nc_oc=Q6cZ2AFL1q_r5e6UWK5WggcZ5xRMWx8XzJTxCS7shJnAFw9mcdkD
QcfvOw8fmfRE8U9bvnZltcc&_nc_ohc=pzcQV3iSR8cQ7kNvgEpDfSs&_nc_gid=74e0f069cd894b
6bt8FGtita45oxJEgIBmc51dAp7uR_cbhvhWVJAcQcg&oe=67CD5F3A&_nc_sid=d8
            "username": "rickyrealrogue"
        },
        "likes_count"
```

catrepublic
Search
2025-02-26_22-09-20_UTC_1.jpg
2025-02-26_22-09-20_UTC_2.jpg
2025-02-26_22-09-20_UTC.txt
2025-02-28_01-51-06_UTC_1.jpg
2025-02-28_01-51-06_UTC_2.jpg
2025-02-25_23-20-32_UTC.jpg
2025-02-25_02-26-49_UTC_1.jpg
2025-02-25_02-26 UTC_2.jpg
2025-02-28_01-51-06_UTC_comments.json
2025-02-28_01-51-06_UTC.json
2025-02-28_01-51-06_UTC.txt
2025-03-01_23-30-30_UTC.jpg
2025-03-01_23-30-30_UTC.json
2025-02-25_23-20-32_UTC.txt
2025-02-25_16-23-25_UTC_comments.json
2025-02-25_16-2 UTC.json
2025-03-01_23-30-30_UTC.txt
2025-03-03_22-35-58_UTC.jpg
2025-03-03_22-35-58_UTC.json
2025-03-03_22-35-58_UTC.txt
2025-02-26_22-09-20_UTC_comments.json
2025-02-26_22-09-20_UTC.json
2025-02-25_02-26-49_UTC.txt
2025-02-24_08 UTC.jpg
2025-02-25_23-20-32_UTC.mp4
2025-02-25_16-23-25_UTC_1.jpg
2025-02-25_16-23-25_UTC_2.jpg
2025-02-25_16-23-25_UTC_3.jpg
2025-02-25_16-23-25_UTC_4.jpg
2025-02-25_23-20-32_UTC.json
2025-02-24_08-11-28_UTC.txt
2025-02-25_ UTC_comme
Dropbox › Soma › Curriculum › 2025-nicar › passive-scraping › catrepublic
37 items, 750.19 GB available



```python
import instaloader
import getpass

L = instaloader.Instaloader(
    download_pictures=True,
    download_videos=True,
    download_video_thumbnails=True,
    download_geotags=False,
    download_comments=True,
    save_metadata=True,
    compress_json=False
)

try:
    L.load_session_from_file('dangerscarf')
    print("Session loaded")
except:
    username = input("Enter your Instagram username: ")
    password = getpass.getpass("Enter your Instagram password: ")

    try:
        L.login(username, password)
        print("Login successful!")
    except instaloader.exceptions.TwoFactorAuthRequiredException:
        print("Two-factor authentication required.")
        code = input("Enter the 2FA code from your authenticator app or SMS: ")
        try:
            L.two_factor_login(code)
            print("Two-factor authentication successful!")
            L.save_session_to_file()
        except instaloader.exceptions.InstaloaderException as e:
            print(f"Two-factor authentication failed: {e}")
    except instaloader.exceptions.InstaloaderException as e:
        print(f"Login failed: {e}")
```

solution......?



drawrowfly/tiktok-scraper

github.com/drawrowfly/tiktok-scraper

drawrowfly / tiktok-scraper

tiktok-scraper Public

drawrowfly Merge pull request #807 from kmkramer/fix-Docker-build 38596d7 · 2 years ago 384 C

.github Update issue templates

bin add stacktrace to errors

examples use session to scrape user

src handles change to respon

.dockerignore Scrape and download in b

.editorconfig improve async tests & en

.eslintrc.js improve async tests & en

TikTokAPI Quick Start — TikTo

davidteather.github.io/TikTok-Api/

TikTokAPI Quick Start

View the above tree for detailed documentation on the package.

Unofficial TikTok API in Python

This is an unofficial api wrapper for TikTok.com in python. With this api you are able to call most trending and fetch specific user information as well as much more.

signed to **retrieve data** TikTok. It **can not be used post or upload** content to behalf of a user. It has **no support for any user-authenticated routes**, if you t while being logged out on their website you can't access it here.

rs have paid to be placed here or are my own affiliate links which I may earn from, and beyond that I do not have any affiliation with them. The

sudoguy/tiktokpy: Tool for au

github.com/sudoguy/tiktokpy

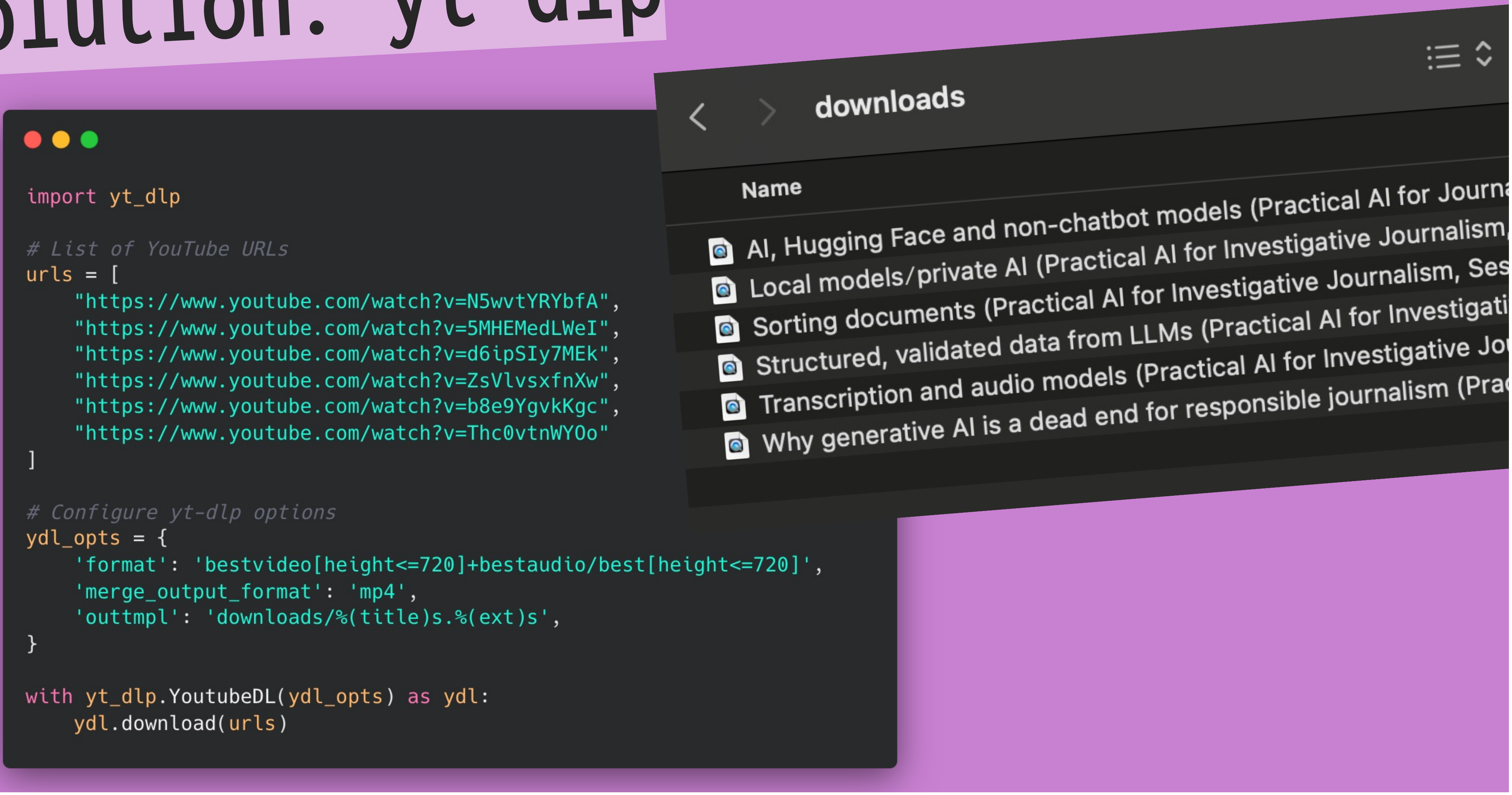
TikTokPy

*Tooling that **automates** your social media interactions to "farm" Likes and Followers on TikTok*

Quickstart.py

```
import asyncio
from tiktokpy import TikTokPy


async def main():
    async with TikTokPy() as bot:
        # Do you want to get trending videos? You can!
        trending_items = await bot.trending(amount=5)

        for item in trending_items:
            # ❤️ you can like videos
            await bot.like(item)
            # or unlike them
            await bot.unlike(item)
            # or follow users
            await bot.follow(item.author.username)
            # as and unfollow
            await bot.unfollow(item.author.username)
```

solution: yt-dlp

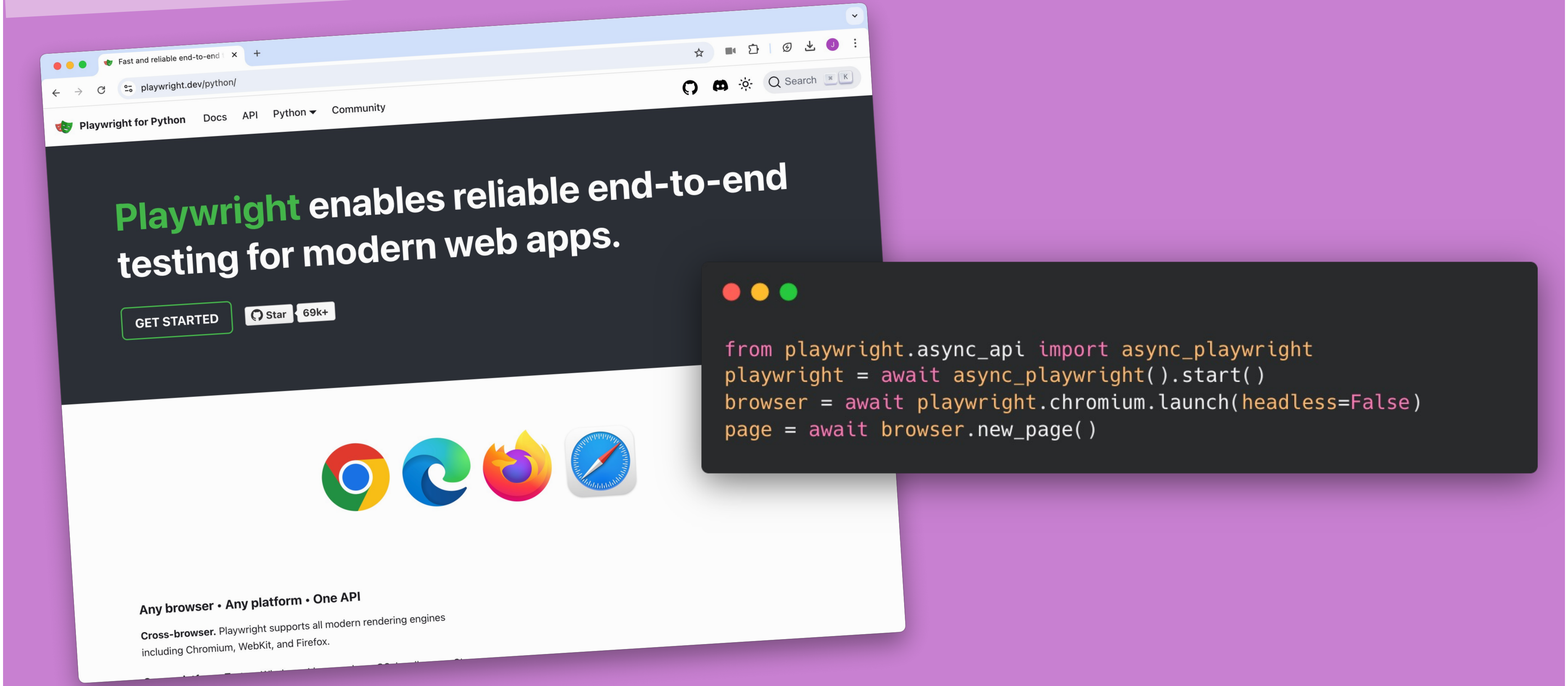


```python
import yt_dlp

# List of YouTube URLs
urls = [
    "https://www.youtube.com/watch?v=N5wvtYRYbfA",
    "https://www.youtube.com/watch?v=5MHEMedLWeI",
    "https://www.youtube.com/watch?v=d6ipSIy7MEk",
    "https://www.youtube.com/watch?v=ZsVlvsxfnXw",
    "https://www.youtube.com/watch?v=b8e9YgvkKgc",
    "https://www.youtube.com/watch?v=Thc0vtnWYOo"
]

# Configure yt-dlp options
ydl_opts = {
    'format': 'bestvideo[height<=720]+bestaudio/best[height<=720]',
    'merge_output_format': 'mp4',
    'outtmpl': 'downloads/%(title)s.%(ext)s',
}

with yt_dlp.YoutubeDL(ydl_opts) as ydl:
    ydl.download(urls)
```

Name

- AI, Hugging Face and non-chatbot models (Practical AI for Journa
- Local models/private AI (Practical AI for Investigative Journalism,
- Sorting documents (Practical AI for Investigative Journalism, Ses
- Structured, validated data from LLMs (Practical AI for Investigati
- Transcription and audio models (Practical AI for Investigative Jou
- Why generative AI is a dead end for responsible journalism (Prac

solution: writing a scraper



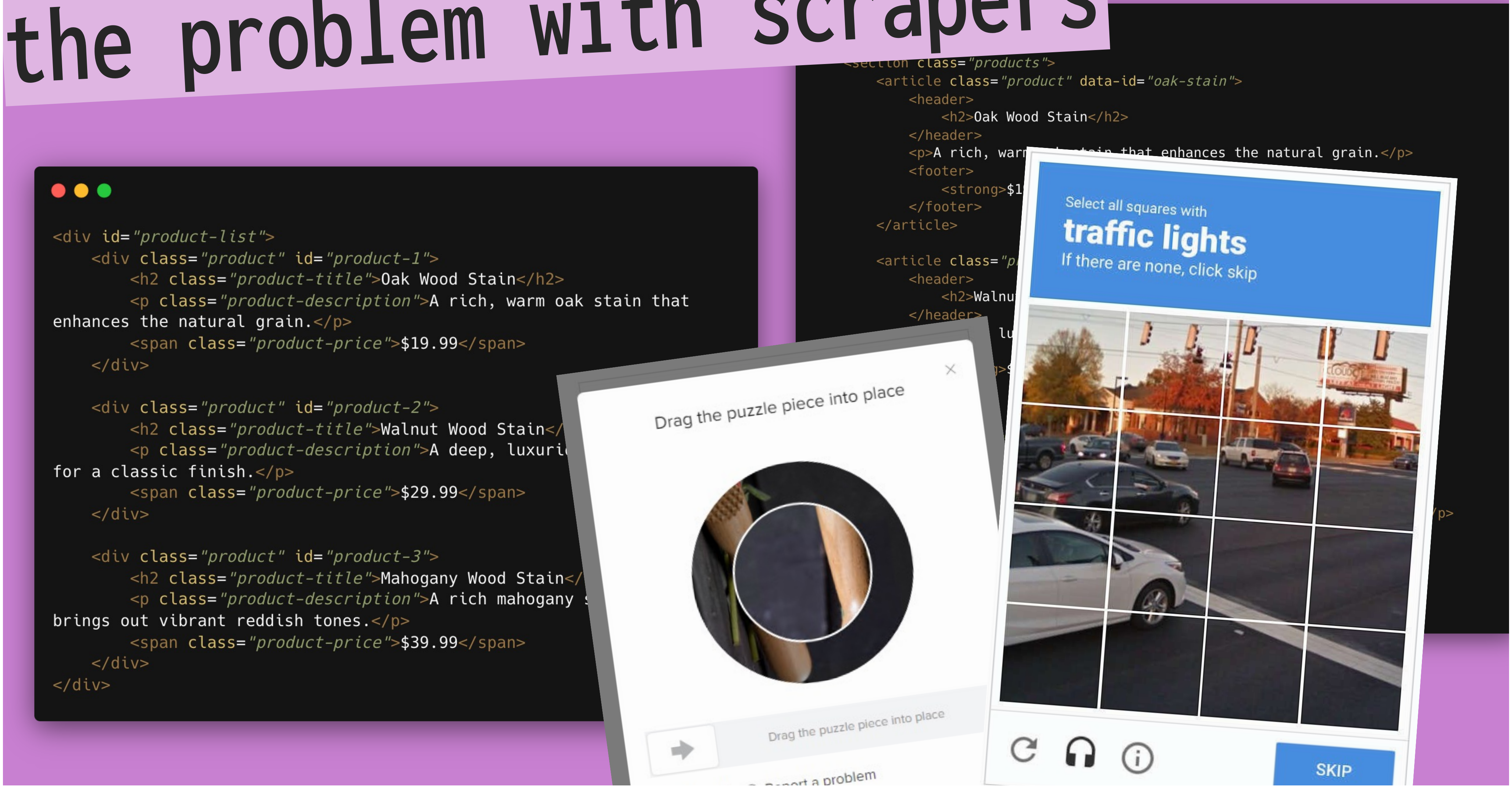
Playwright enables reliable end-to-end testing for modern web apps.

Any browser • Any platform • One API

Cross-browser. Playwright supports all modern rendering engines including Chromium, WebKit, and Firefox.

```
from playwright.async_api import async_playwright
playwright = await async_playwright().start()
browser = await playwright.chromium.launch(headless=False)
page = await browser.new_page()
```

the problem with scrapers



```
<div id="product-list">
    <div class="product" id="product-1">
        <h2 class="product-title">Oak Wood Stain</h2>
        <p class="product-description">A rich, warm oak stain that
enhances the natural grain.</p>
        <span class="product-price">$19.99</span>
    </div>

    <div class="product" id="product-2">
        <h2 class="product-title">Walnut Wood Stain</
        <p class="product-description">A deep, luxuri
for a classic finish.</p>
        <span class="product-price">$29.99</span>
    </div>

    <div class="product" id="product-3">
        <h2 class="product-title">Mahogany Wood Stain</
        <p class="product-description">A rich mahogany s
brings out vibrant reddish tones.</p>
        <span class="product-price">$39.99</span>
    </div>
</div>
```

```
<section class="products">
    <article class="product" data-id="oak-stain">
        <header>
            <h2>Oak Wood Stain</h2>
        </header>
        <p>A rich, warm              that enhances the natural grain.</p>
        <footer>
            <strong>$1
        </footer>
    </article>

    <article class="p
        <header>
            <h2>Walnu
        </header>
                    lu

                 g>$

                                                          p>
```

Drag the puzzle piece into place

Select all squares with

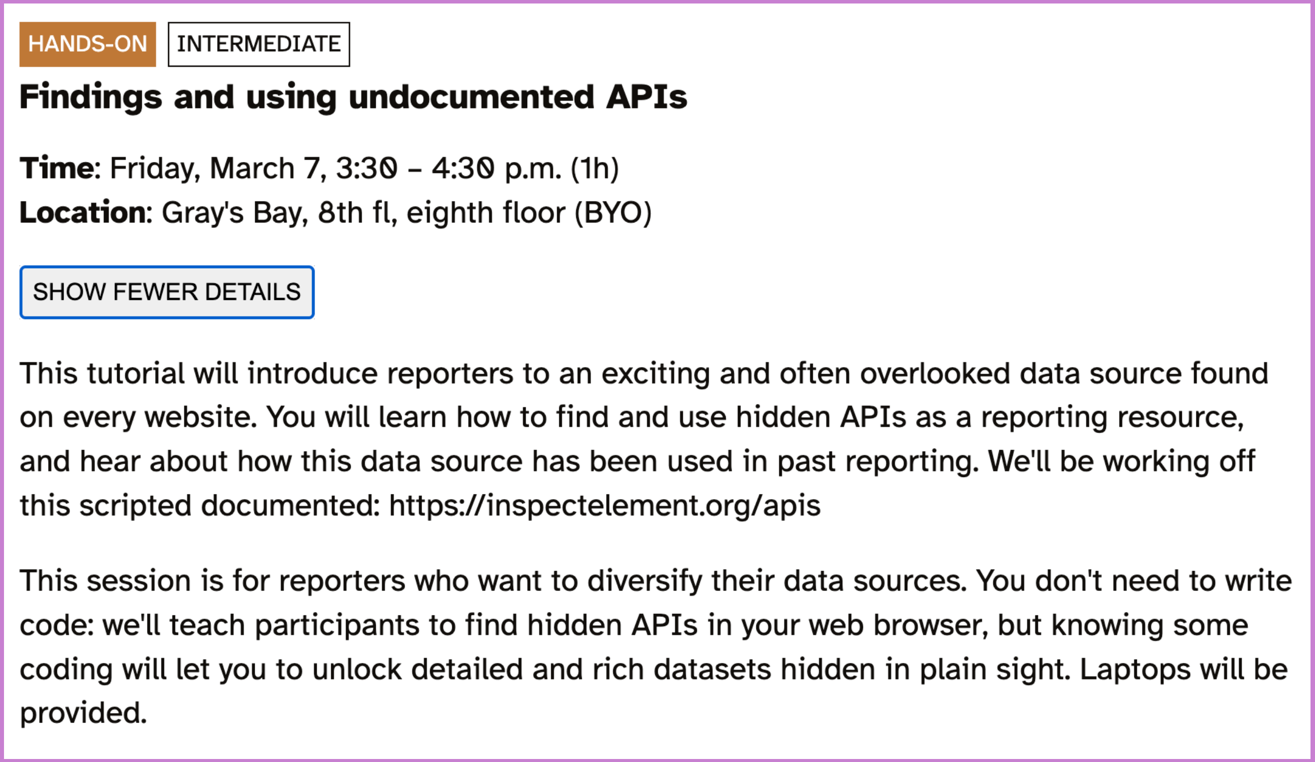
traffic lights

If there are none, click skip

tiers of problem-solving

- ~~Using a tool~~
- ~~Writing a scraper~~
- Undocumented APIs
- Intercepting browser requests
- Pack-ratting with HAR and WARC/WACZ files

solution: undocumented APIs

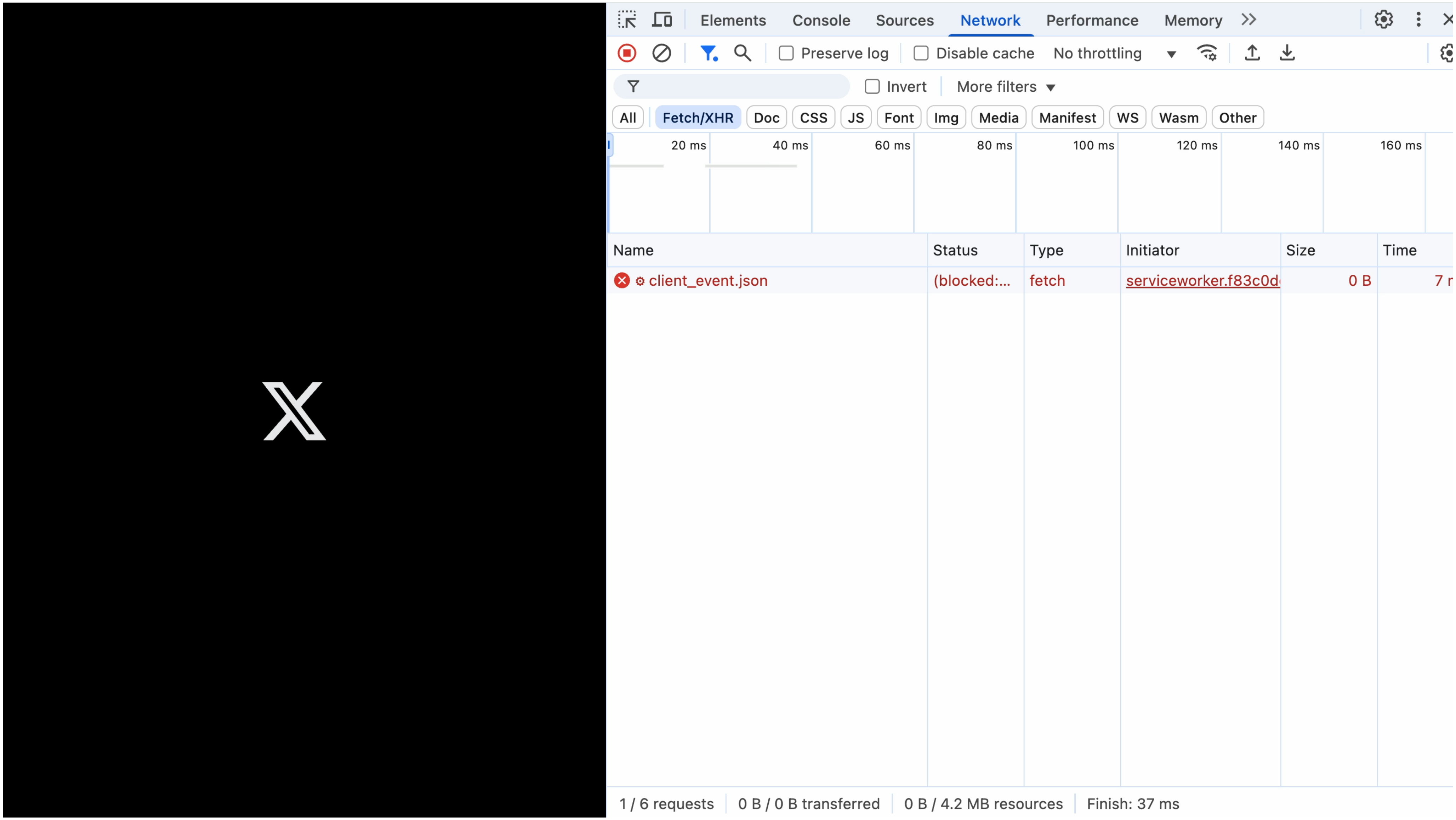


Findings and using undocumented APIs

Time: Friday, March 7, 3:30 – 4:30 p.m. (1h)
Location: Gray's Bay, 8th fl, eighth floor (BYO)

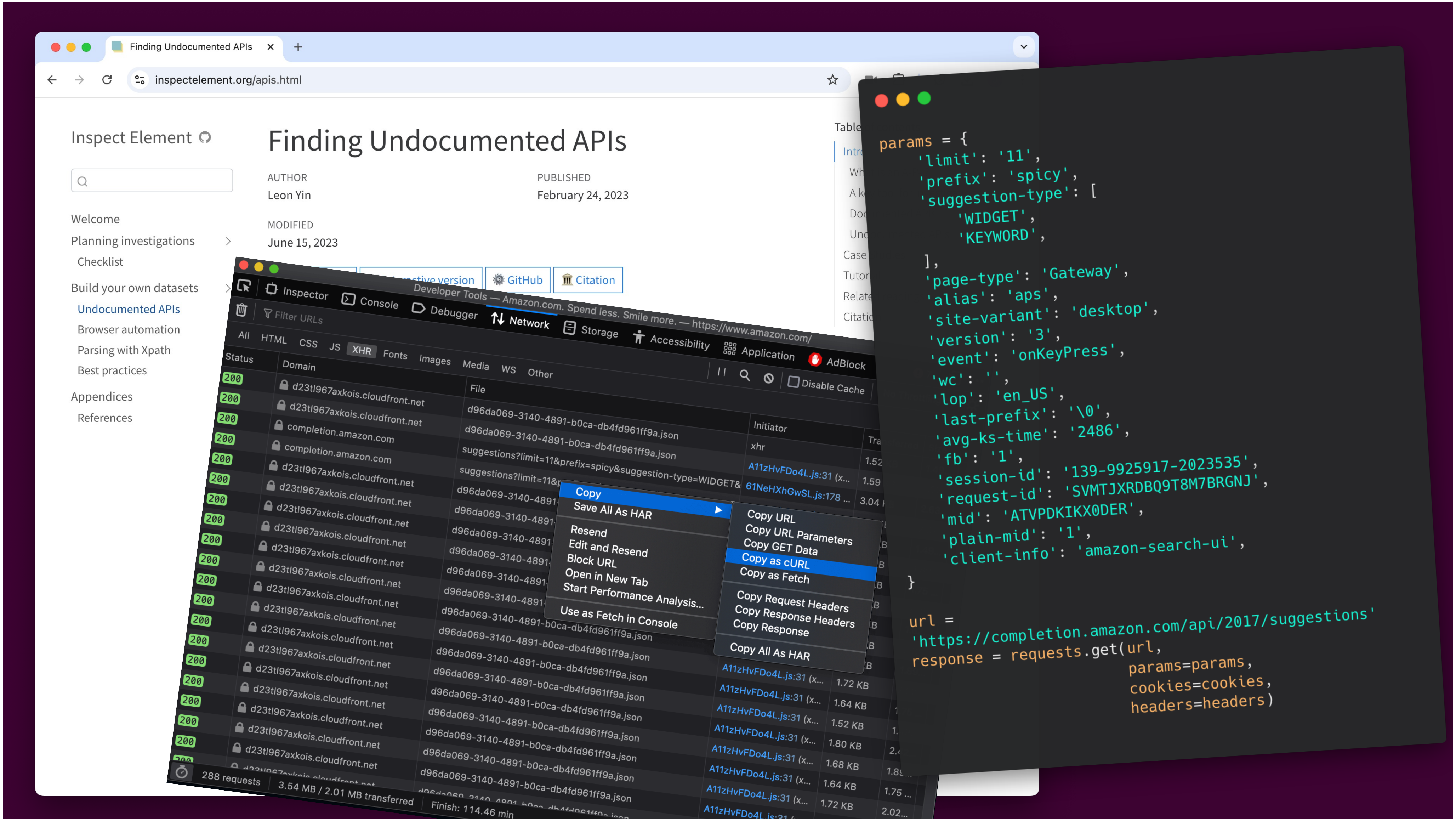
This tutorial will introduce reporters to an exciting and often overlooked data source found on every website. You will learn how to find and use hidden APIs as a reporting resource, and hear about how this data source has been used in past reporting. We'll be working off this scripted documented: https://inspectelement.org/apis

This session is for reporters who want to diversify their data sources. You don't need to write code: we'll teach participants to find hidden APIs in your web browser, but knowing some coding will let you to unlock detailed and rich datasets hidden in plain sight. Laptops will be provided.



Name	Status	Type	Initiator	Size	Time
client_event.json	(blocked:...	fetch	serviceworker.f83c0d	0 B	7 r

1 / 6 requests | 0 B / 0 B transferred | 0 B / 4.2 MB resources | Finish: 37 ms



Inspect Element

Finding Undocumented APIs

AUTHOR
Leon Yin

PUBLISHED
February 24, 2023

MODIFIED
June 15, 2023

Developer Tools — Amazon.com. Spend less. Smile more. — https://www.amazon.com/

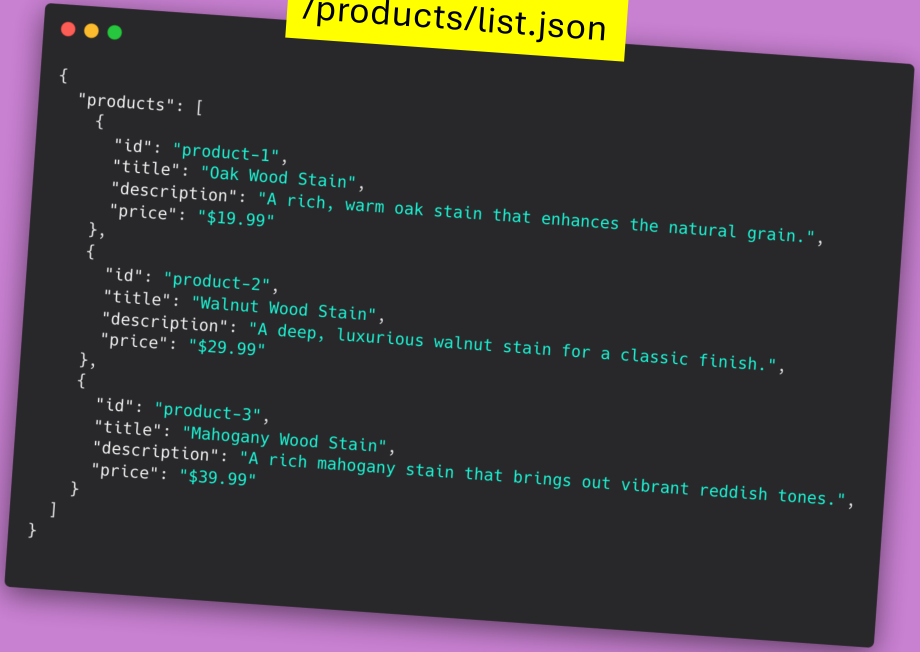
Status	Domain	File	Initiator	Tra
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json		
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json	xhr	1.52
200	completion.amazon.com	suggestions?limit=11&prefix=spicy&suggestion-type=WIDGET&	A11zHvFDo4L.js:31 (x...	1.59
200	completion.amazon.com	suggestions?limit=11&p	61NeHXhGwSL.js:178 ...	3.04
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891		
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891		
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891		
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891		
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891		
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891		
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json		
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json		
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json	A11zHvFDo4L.js:31 (x...	1.72 KB
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json	A11zHvFDo4L.js:31 (x...	1.64 KB
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json	A11zHvFDo4L.js:31 (x...	1.52 KB
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json	A11zHvFDo4L.js:31 (x...	1.80 KB
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json	A11zHvFDo4L.js:31 (x...	1.68 KB
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json	A11zHvFDo4L.js:31 (x...	1.64 KB
200	d23tl967axkois.cloudfront.net	d96da069-3140-4891-b0ca-db4fd961ff9a.json	A11zHvFDo4L.js:31 (x...	1.72 KB

288 requests | 3.54 MB / 2.01 MB transferred | Finish: 114.46 min

```
params = {
    'limit': '11',
    'prefix': 'spicy',
    'suggestion-type': [
        'WIDGET',
        'KEYWORD',
    ],
    'page-type': 'Gateway',
    'alias': 'aps',
    'site-variant': 'desktop',
    'version': '3',
    'event': 'onKeyPress',
    'wc': '',
    'lop': 'en_US',
    'last-prefix': '\0',
    'avg-ks-time': '2486',
    'fb': '1',
    'session-id': '139-9925917-2023535',
    'request-id': 'SVMTJXRDBQ9T8M7BRGNJ',
    'mid': 'ATVPDKIKX0DER',
    'plain-mid': '1',
    'client-info': 'amazon-search-ui',
}

url =
'https://completion.amazon.com/api/2017/suggestions'
response = requests.get(url,
                        params=params,
                        cookies=cookies,
                        headers=headers)
```

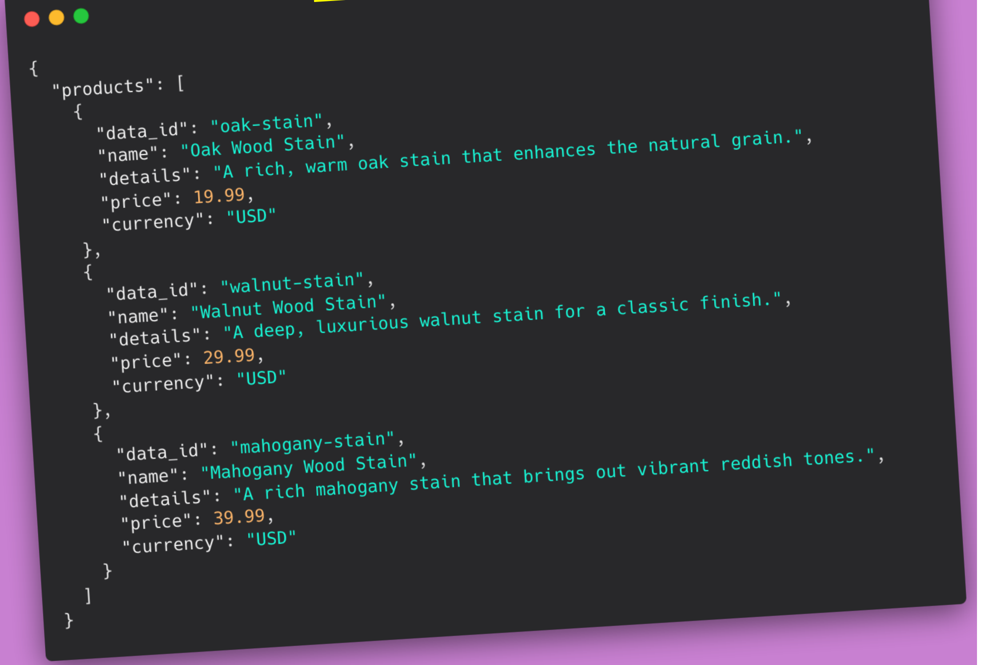
the problem: they change (and we're lazy!)

/products/list.json



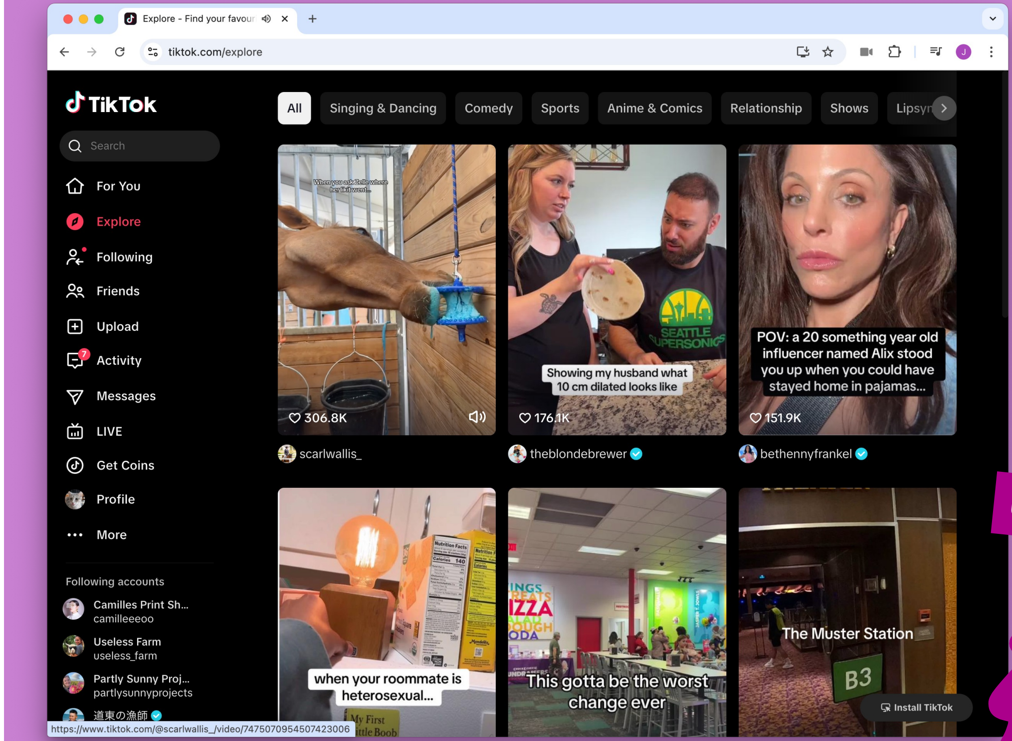
```
{
  "products": [
    {
      "id": "product-1",
      "title": "Oak Wood Stain",
      "description": "A rich, warm oak stain that enhances the natural grain.",
      "price": "$19.99"
    },
    {
      "id": "product-2",
      "title": "Walnut Wood Stain",
      "description": "A deep, luxurious walnut stain for a classic finish.",
      "price": "$29.99"
    },
    {
      "id": "product-3",
      "title": "Mahogany Wood Stain",
      "description": "A rich mahogany stain that brings out vibrant reddish tones.",
      "price": "$39.99"
    }
  ]
}
```

/api/v2/products



```
{
  "products": [
    {
      "data_id": "oak-stain",
      "name": "Oak Wood Stain",
      "details": "A rich, warm oak stain that enhances the natural grain.",
      "price": 19.99,
      "currency": "USD"
    },
    {
      "data_id": "walnut-stain",
      "name": "Walnut Wood Stain",
      "details": "A deep, luxurious walnut stain for a classic finish.",
      "price": 29.99,
      "currency": "USD"
    },
    {
      "data_id": "mahogany-stain",
      "name": "Mahogany Wood Stain",
      "details": "A rich mahogany stain that brings out vibrant reddish tones.",
      "price": 39.99,
      "currency": "USD"
    }
  ]
}
```

solution: intercepting API calls



Showing my husband what 10 cm dilated looks like

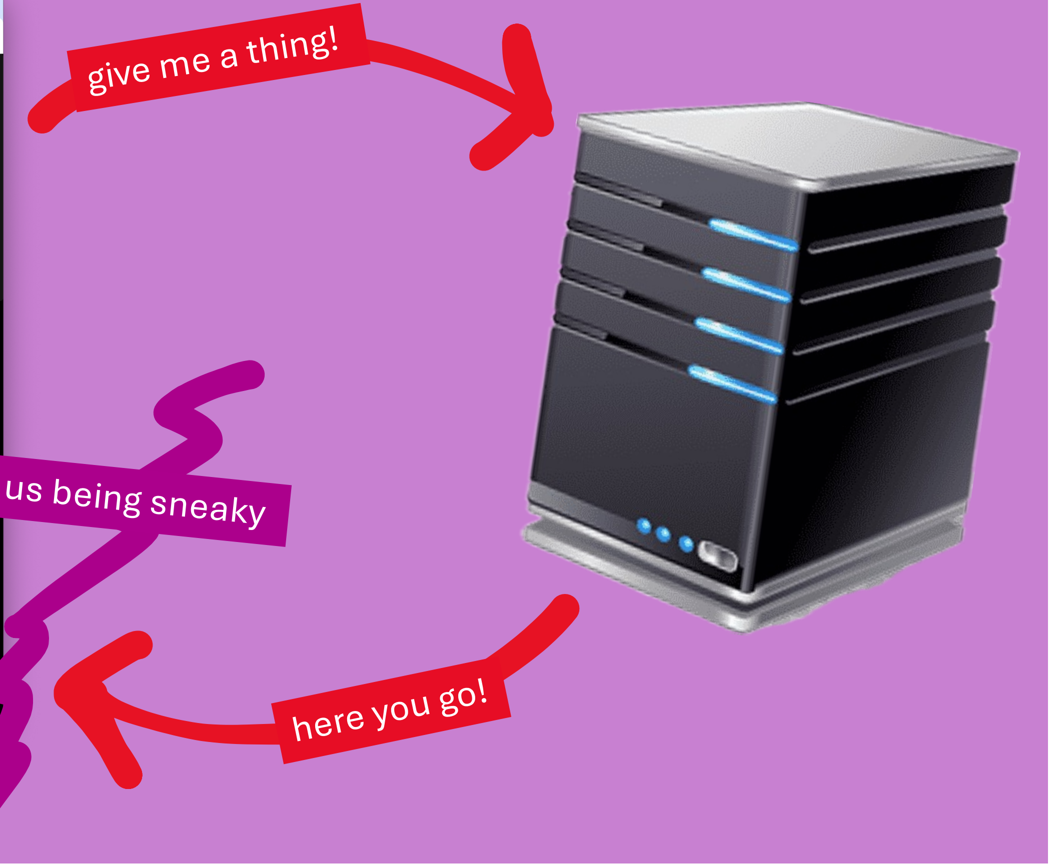
POV: a 20 something year old influencer named Alix stood you up when you could have stayed home in pajamas...

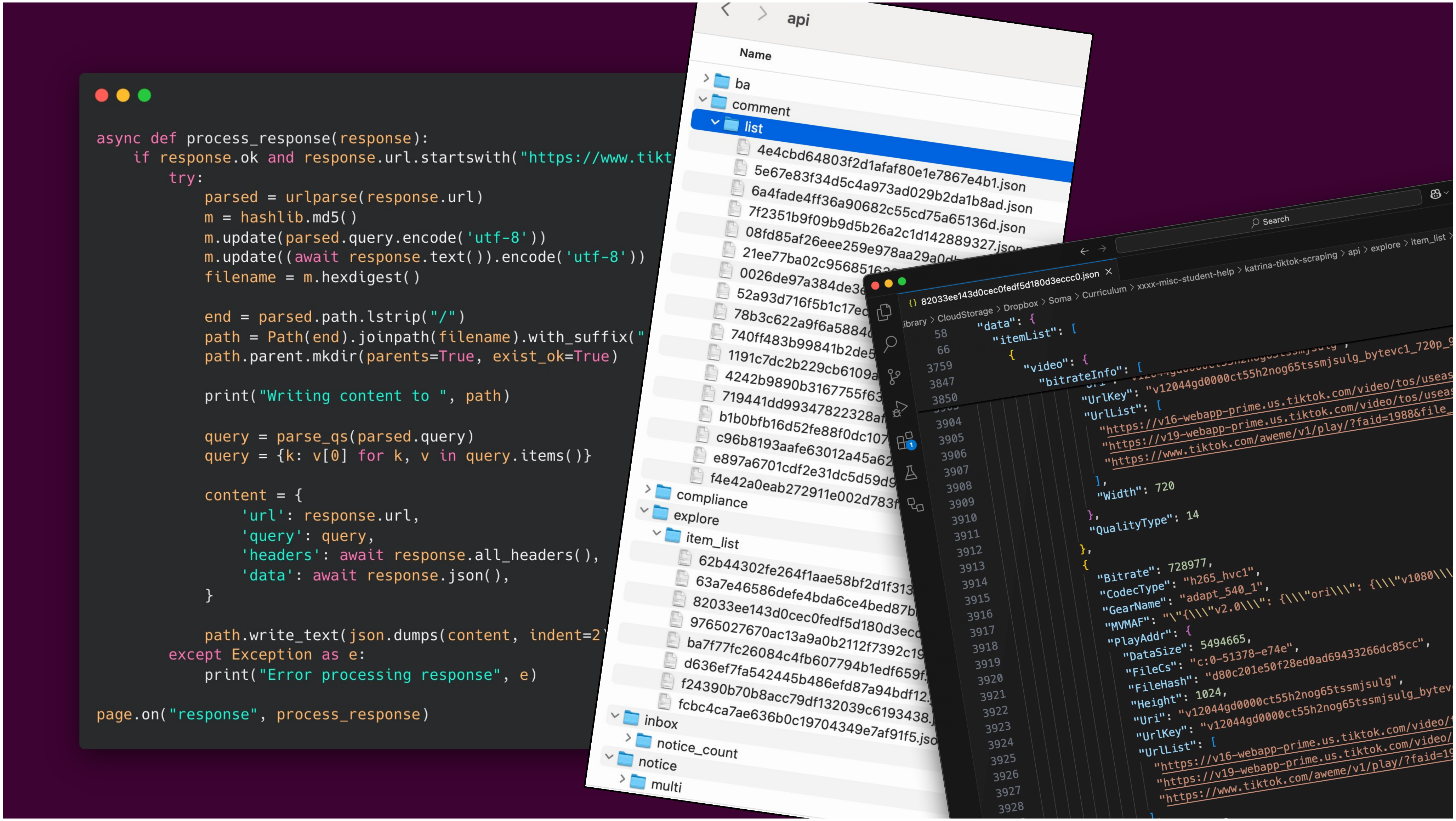
when your roommate is heterosexual...

This gotta be the worst change ever

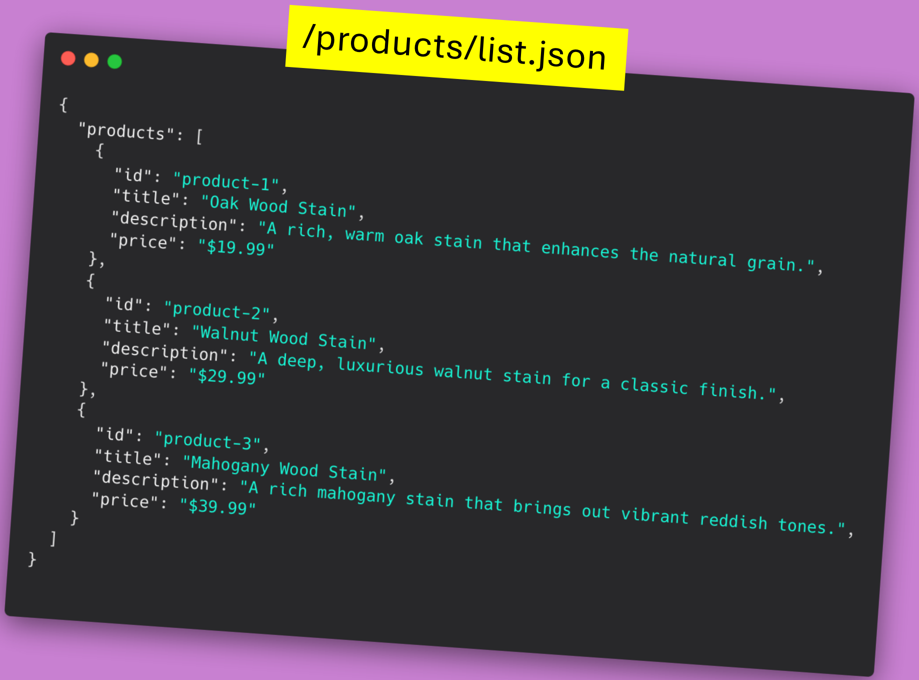
The Muster Station

https://www.tiktok.com/@scarlwallis_/video/7475070954507423006



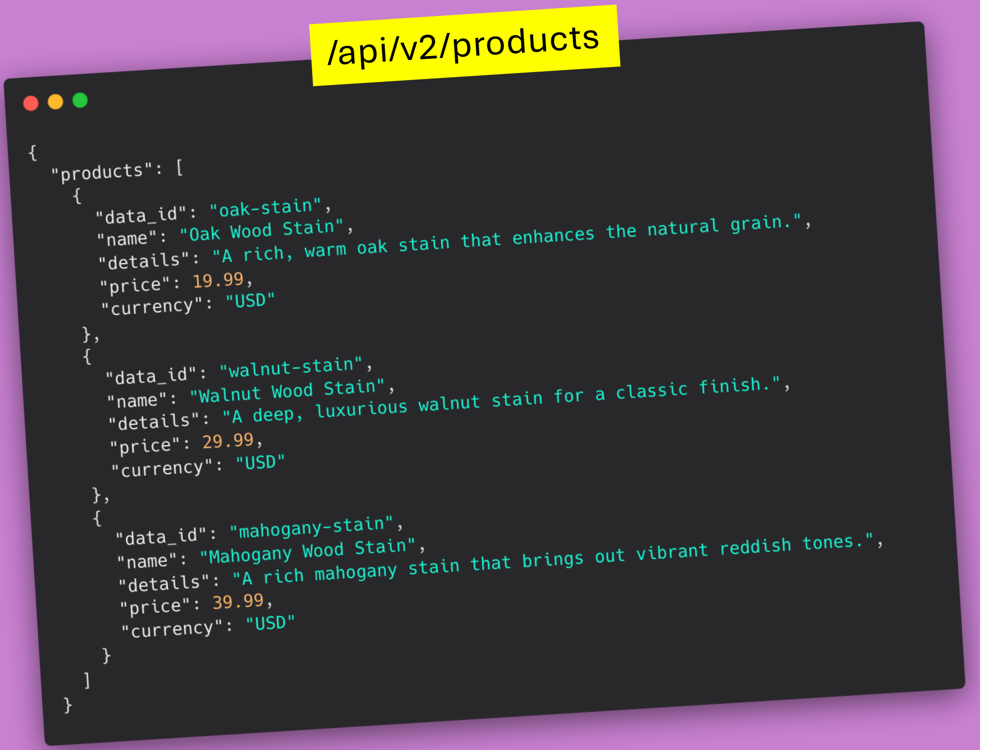


the problem: they change (and we're lazy!)



/products/list.json

```json
{
  "products": [
    {
      "id": "product-1",
      "title": "Oak Wood Stain",
      "description": "A rich, warm oak stain that enhances the natural grain.",
      "price": "$19.99"
    },
    {
      "id": "product-2",
      "title": "Walnut Wood Stain",
      "description": "A deep, luxurious walnut stain for a classic finish.",
      "price": "$29.99"
    },
    {
      "id": "product-3",
      "title": "Mahogany Wood Stain",
      "description": "A rich mahogany stain that brings out vibrant reddish tones.",
      "price": "$39.99"
    }
  ]
}
```



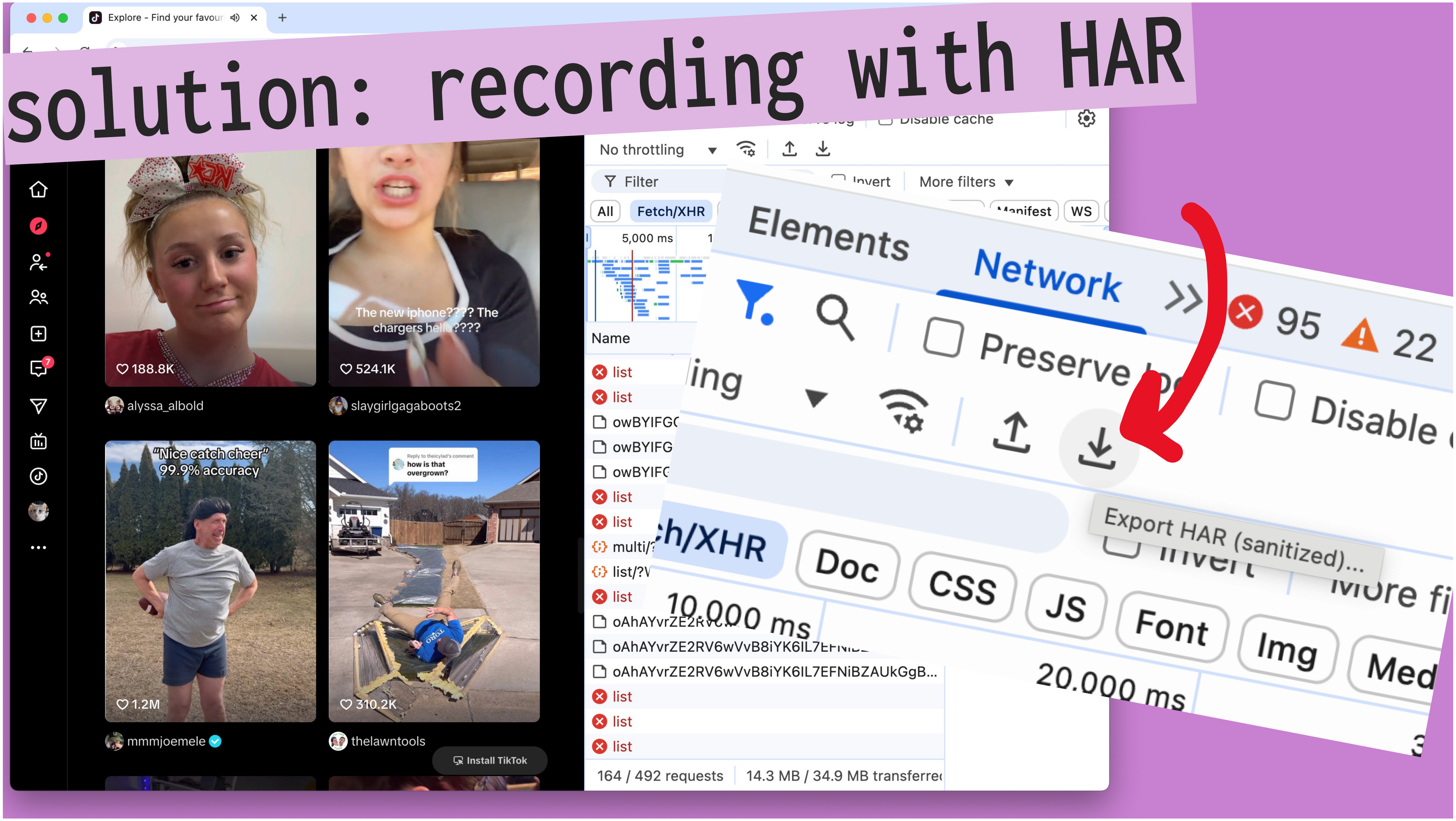
/api/v2/products

```json
{
  "products": [
    {
      "data_id": "oak-stain",
      "name": "Oak Wood Stain",
      "details": "A rich, warm oak stain that enhances the natural grain.",
      "price": 19.99,
      "currency": "USD"
    },
    {
      "data_id": "walnut-stain",
      "name": "Walnut Wood Stain",
      "details": "A deep, luxurious walnut stain for a classic finish.",
      "price": 29.99,
      "currency": "USD"
    },
    {
      "data_id": "mahogany-stain",
      "name": "Mahogany Wood Stain",
      "details": "A rich mahogany stain that brings out vibrant reddish tones.",
      "price": 39.99,
      "currency": "USD"
    }
  ]
}
```

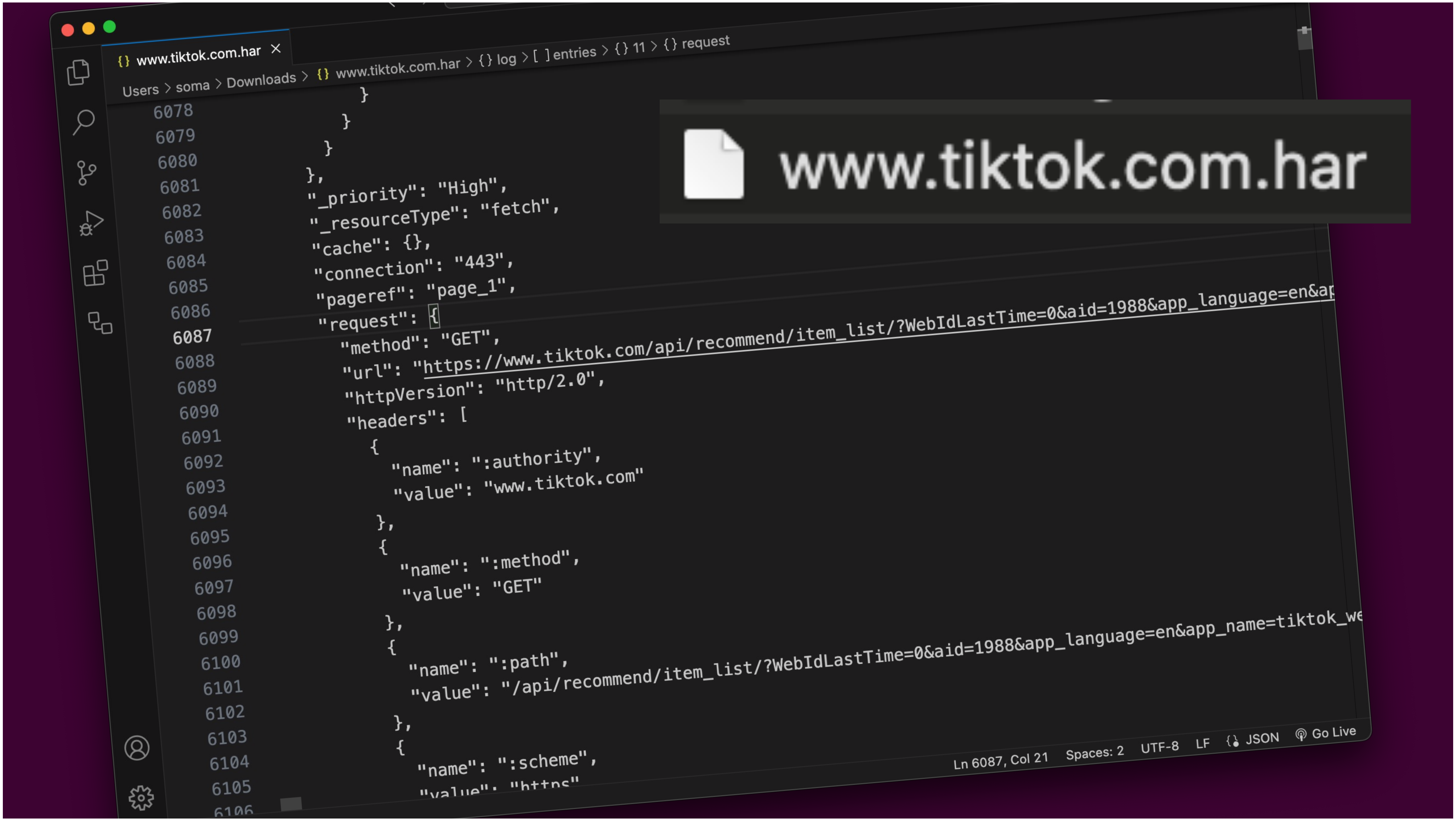
tiers of problem-solving

- ~~Using a tool~~
- ~~Writing a scraper~~
- ~~Undocumented APIs~~
- ~~Intercepting browser requests~~
- Pack-ratting with HAR and WARC/WACZ files

solution: recording with HAR



www.tiktok.com.har



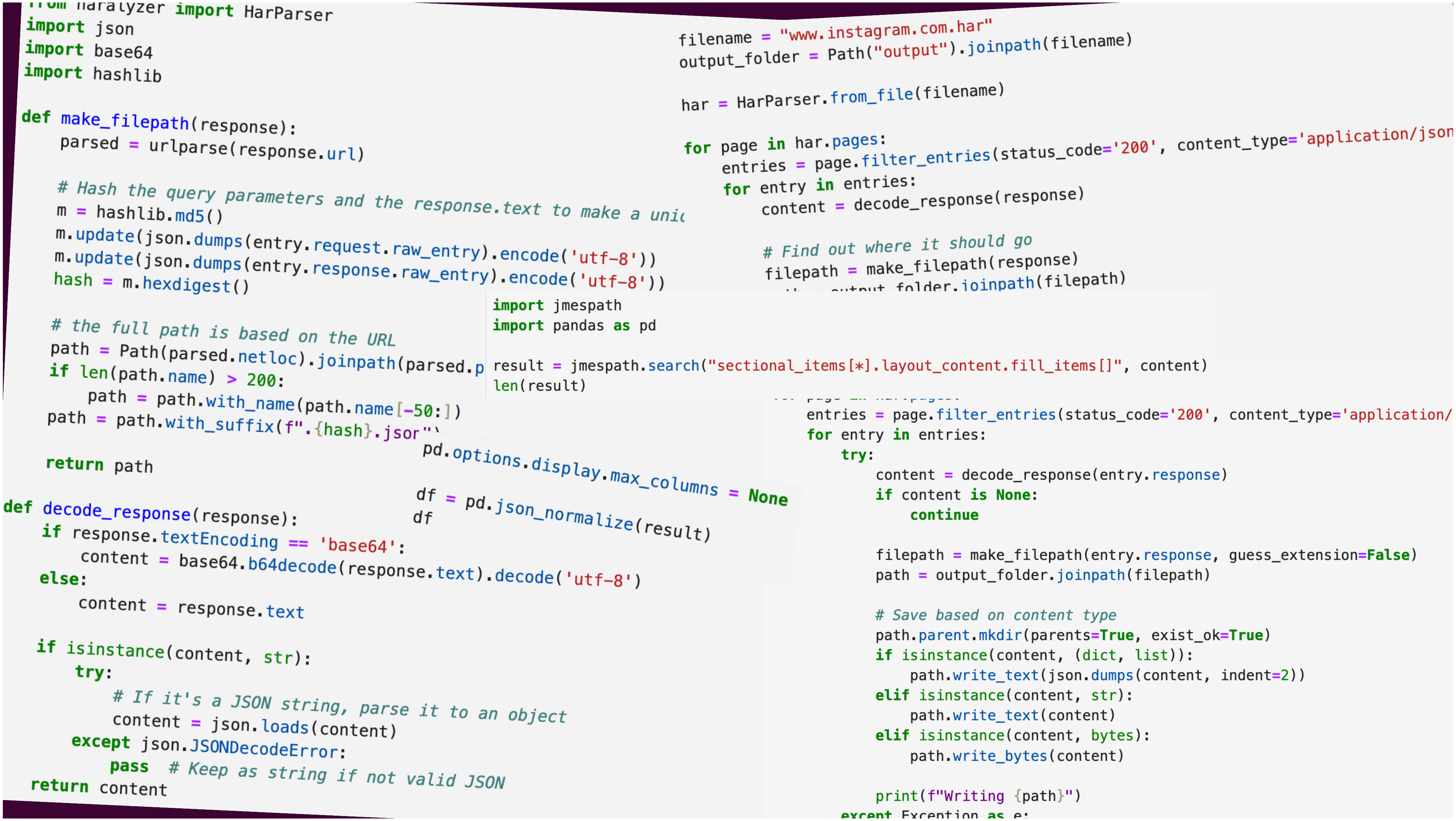
www.tiktok.com.har

{} www.tiktok.com.har ×

Users > soma > Downloads > {} www.tiktok.com.har > {} log > [ ] entries > {} 11 > {} request

```
            }
          }
        }
      },
      "_priority": "High",
      "_resourceType": "fetch",
      "cache": {},
      "connection": "443",
      "pageref": "page_1",
      "request": {
        "method": "GET",
        "url": "https://www.tiktok.com/api/recommend/item_list/?WebIdLastTime=0&aid=1988&app_language=en&ap
        "httpVersion": "http/2.0",
        "headers": [
          {
            "name": ":authority",
            "value": "www.tiktok.com"
          },
          {
            "name": ":method",
            "value": "GET"
          },
          {
            "name": ":path",
            "value": "/api/recommend/item_list/?WebIdLastTime=0&aid=1988&app_language=en&app_name=tiktok_we
          },
          {
            "name": ":scheme",
            "value": "https"
```

Ln 6087, Col 21 Spaces: 2 UTF-8 LF {} JSON Go Live



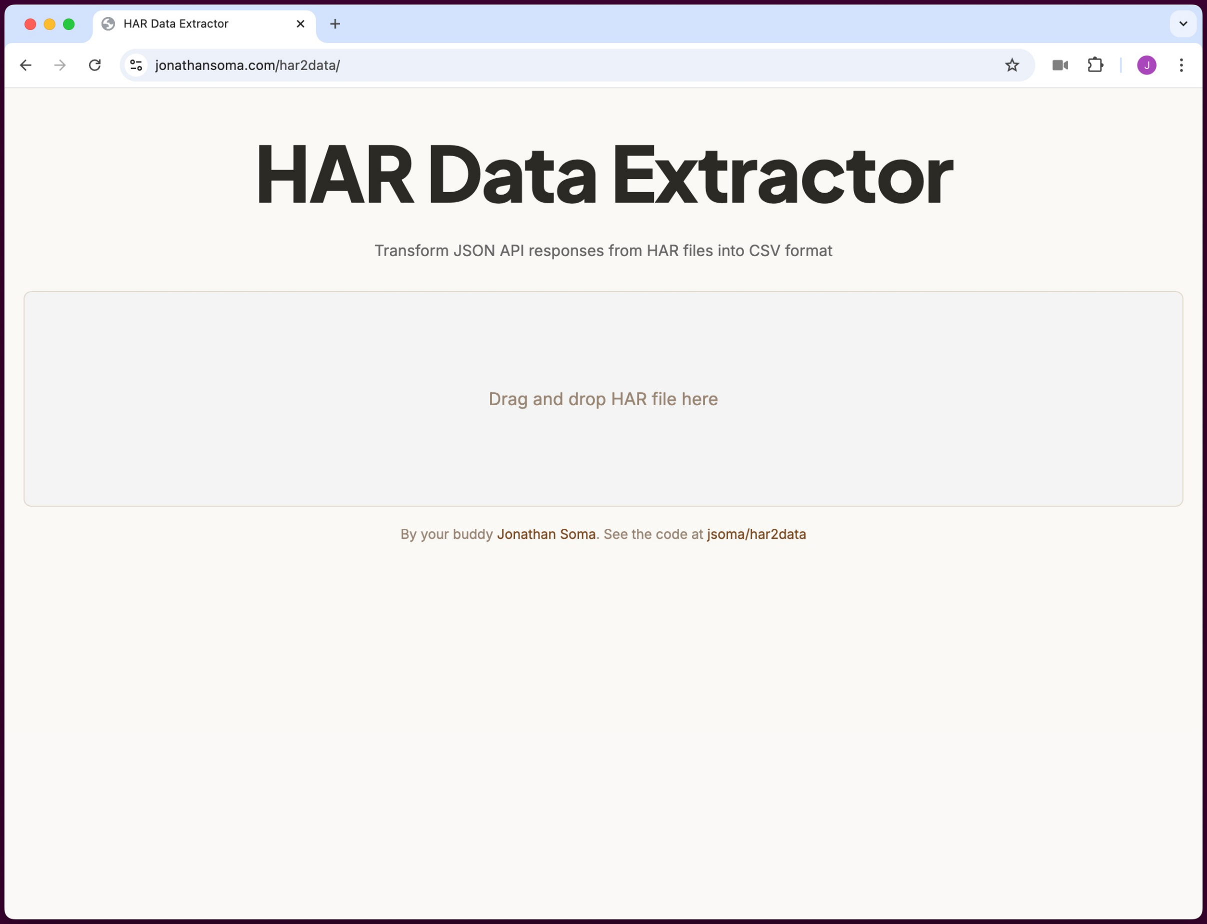
```python
from haralyzer import HarParser
import json
import base64
import hashlib

def make_filepath(response):
    parsed = urlparse(response.url)

    # Hash the query parameters and the response.text to make a unic
    m = hashlib.md5()
    m.update(json.dumps(entry.request.raw_entry).encode('utf-8'))
    m.update(json.dumps(entry.response.raw_entry).encode('utf-8'))
    hash = m.hexdigest()

    # the full path is based on the URL
    path = Path(parsed.netloc).joinpath(parsed.p
    if len(path.name) > 200:
        path = path.with_name(path.name[-50:])
    path = path.with_suffix(f".{hash}.jsor")

    return path

def decode_response(response):
    if response.textEncoding == 'base64':
        content = base64.b64decode(response.text).decode('utf-8')
    else:
        content = response.text

    if isinstance(content, str):
        try:
            # If it's a JSON string, parse it to an object
            content = json.loads(content)
        except json.JSONDecodeError:
            pass  # Keep as string if not valid JSON
    return content
```

```python
filename = "www.instagram.com.har"
output_folder = Path("output").joinpath(filename)

har = HarParser.from_file(filename)

for page in har.pages:
    entries = page.filter_entries(status_code='200', content_type='application/json
    for entry in entries:
        content = decode_response(response)

        # Find out where it should go
        filepath = make_filepath(response)
        path = output_folder.joinpath(filepath)
```

```python
import jmespath
import pandas as pd

result = jmespath.search("sectional_items[*].layout_content.fill_items[]", content)
len(result)
```

```python
pd.options.display.max_columns = None

df = pd.json_normalize(result)
df
```

```python
for page in har.pages:
    entries = page.filter_entries(status_code='200', content_type='application/
    for entry in entries:
        try:
            content = decode_response(entry.response)
            if content is None:
                continue

            filepath = make_filepath(entry.response, guess_extension=False)
            path = output_folder.joinpath(filepath)

            # Save based on content type
            path.parent.mkdir(parents=True, exist_ok=True)
            if isinstance(content, (dict, list)):
                path.write_text(json.dumps(content, indent=2))
            elif isinstance(content, str):
                path.write_text(content)
            elif isinstance(content, bytes):
                path.write_bytes(content)

            print(f"Writing {path}")
        except Exception as e:
```



HAR Data Extractor

Transform JSON API responses from HAR files into CSV format

Drag and drop HAR file here

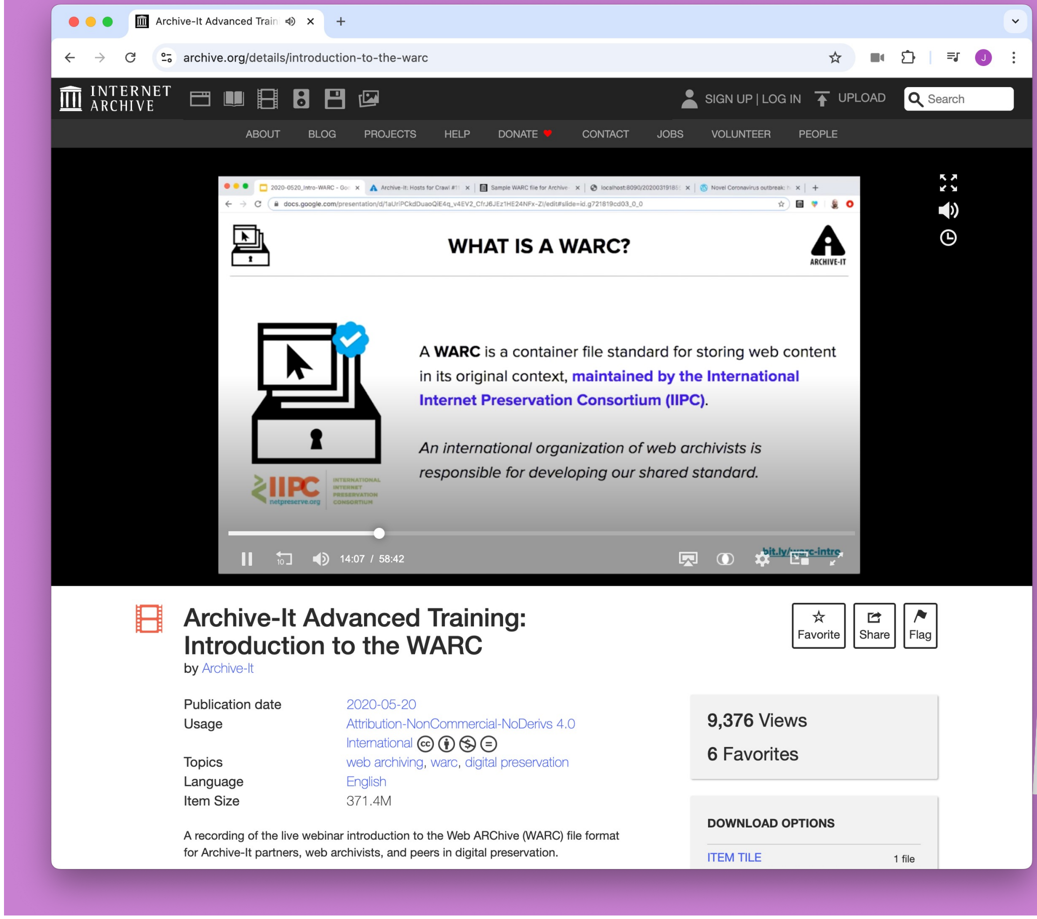
By your buddy Jonathan Soma. See the code at jsoma/har2data

the problem

- Giant files!!!
- Chunking!!!
- Base64 encoding!!!
- Other things I probably don’t even know about yet!!!
- (But they do also sometimes work, too, it just depends)

solution: saving WARC files

ty jeremy merrill


Archive-It Advanced Train
archive.org/details/introduction-to-the-warc
INTERNET ARCHIVE
SIGN UP | LOG IN
UPLOAD
Search
ABOUT
BLOG
PROJECTS
HELP
DONATE
CONTACT
JOBS
VOLUNTEER
PEOPLE
2020-0520_Intro-WARC - Goo
Archive-It: Hosts for Crawl #1
Sample WARC file for Archive-
localhost:8090/20200319185
Novel Coronavirus outbreak: h
docs.google.com/presentation/d/1aUrIPCkdDuaoQiE4q_v4EV2_CfrJ6JEz1HE24NFx-Zl/edit#slide=id.g721819cd03_0_0
ARCHIVE-IT
IIPC netpreserve.org INTERNATIONAL INTERNET PRESERVATION CONSORTIUM
14:07 / 58:42
bit.ly/warc-intro
Favorite
Share
Flag

WHAT IS A WARC?

A **WARC** is a container file standard for storing web content in its original context, **maintained by the International Internet Preservation Consortium (IIPC)**.

An international organization of web archivists is responsible for developing our shared standard.

Archive-It Advanced Training: Introduction to the WARC

by Archive-It

Publication date	2020-05-20
Usage	Attribution-NonCommercial-NoDerivs 4.0 International 🅭
Topics	web archiving, warc, digital preservation
Language	English
Item Size	371.4M

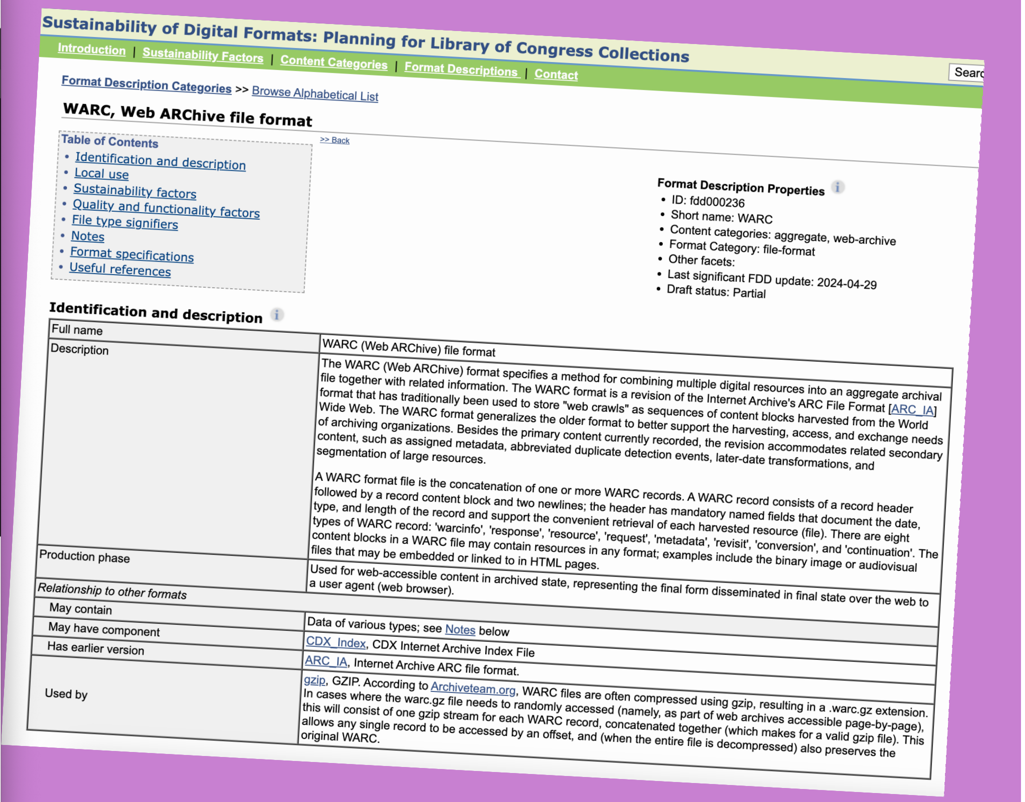
A recording of the live webinar introduction to the Web ARChive (WARC) file format for Archive-It partners, web archivists, and peers in digital preservation.

9,376 Views

6 Favorites

DOWNLOAD OPTIONS

ITEM TILE 1 file


Sustainability of Digital Formats: Planning for Library of Congress Collections
Introduction | Sustainability Factors | Content Categories | Format Descriptions | Contact
Searc
Format Description Categories >> Browse Alphabetical List
>> Back

WARC, Web ARChive file format

Table of Contents

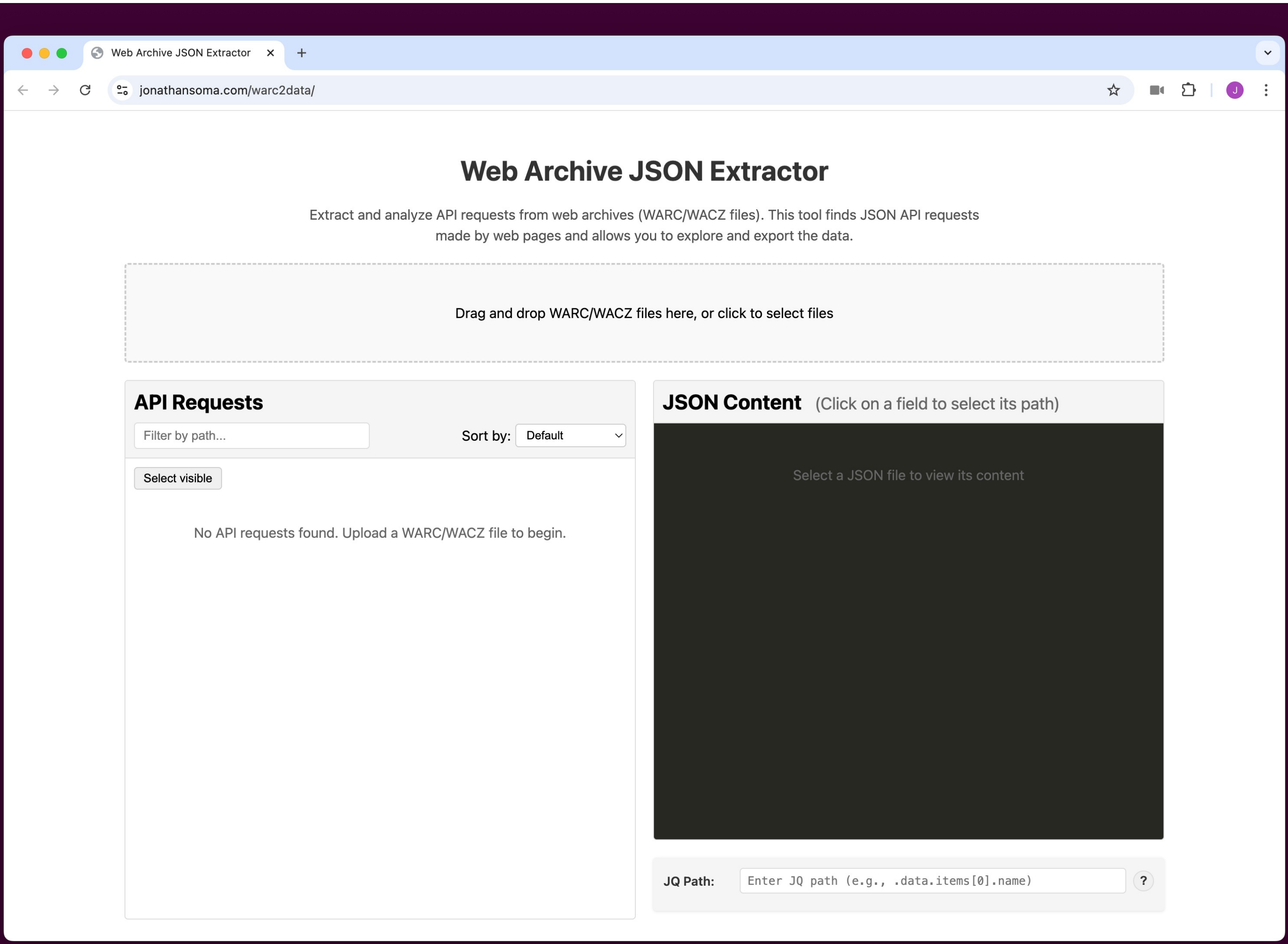
- Identification and description
- Local use
- Sustainability factors
- Quality and functionality factors
- File type signifiers
- Notes
- Format specifications
- Useful references

Format Description Properties

- ID: fdd000236
- Short name: WARC
- Content categories: aggregate, web-archive
- Format Category: file-format
- Other facets:
- Last significant FDD update: 2024-04-29
- Draft status: Partial

Identification and description

Full name	WARC (Web ARChive) file format
Description	The WARC (Web ARChive) format specifies a method for combining multiple digital resources into an aggregate archival file together with related information. The WARC format is a revision of the Internet Archive's ARC File Format [ARC_IA] format that has traditionally been used to store "web crawls" as sequences of content blocks harvested from the World Wide Web. The WARC format generalizes the older format to better support the harvesting, access, and exchange needs of archiving organizations. Besides the primary content currently recorded, the revision accommodates related secondary content, such as assigned metadata, abbreviated duplicate detection events, later-date transformations, and segmentation of large resources. A WARC format file is the concatenation of one or more WARC records. A WARC record consists of a record header followed by a record content block and two newlines; the header has mandatory named fields that document the date, type, and length of the record and support the convenient retrieval of each harvested resource (file). There are eight types of WARC record: 'warcinfo', 'response', 'resource', 'request', 'metadata', 'revisit', 'conversion', and 'continuation'. The content blocks in a WARC file may contain resources in any format; examples include the binary image or audiovisual files that may be embedded or linked to in HTML pages.
Production phase	Used for web-accessible content in archived state, representing the final form disseminated in final state over the web to a user agent (web browser).
Relationship to other formats	
May contain	Data of various types; see Notes below
May have component	CDX_Index, CDX Internet Archive Index File
Has earlier version	ARC_IA, Internet Archive ARC file format.
Used by	gzip, GZIP. According to Archiveteam.org, WARC files are often compressed using gzip, resulting in a .warc.gz extension. In cases where the warc.gz file needs to randomly accessed (namely, as part of web archives accessible page-by-page), this will consist of one gzip stream for each WARC record, concatenated together (which makes for a valid gzip file). This allows any single record to be accessed by an offset, and (when the entire file is decompressed) also preserves the original WARC.



Web Archive JSON Extractor

Extract and analyze API requests from web archives (WARC/WACZ files). This tool finds JSON API requests made by web pages and allows you to explore and export the data.

Drag and drop WARC/WACZ files here, or click to select files

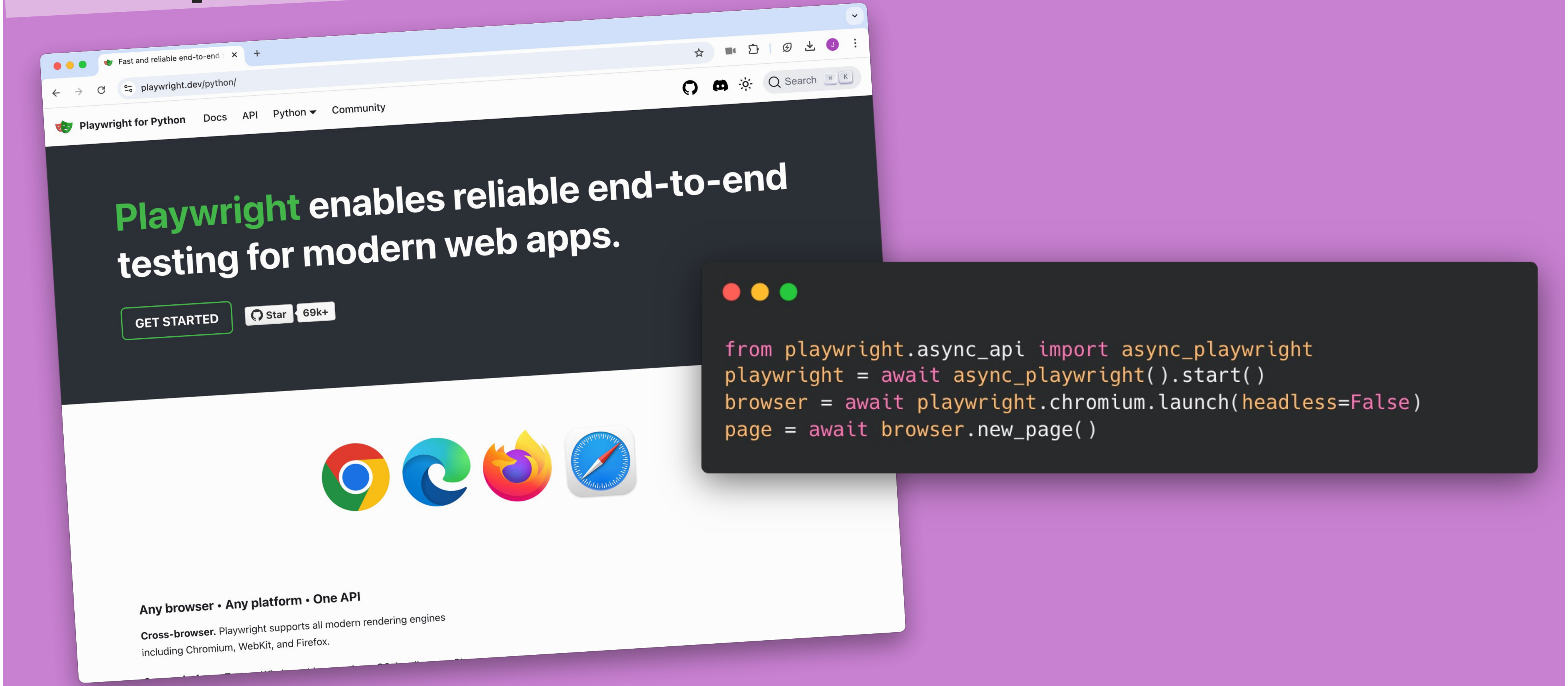
API Requests

No API requests found. Upload a WARC/WACZ file to begin.

JSON Content (Click on a field to select its path)

Select a JSON file to view its content

the problem: automation..?



Playwright enables reliable end-to-end testing for modern web apps.

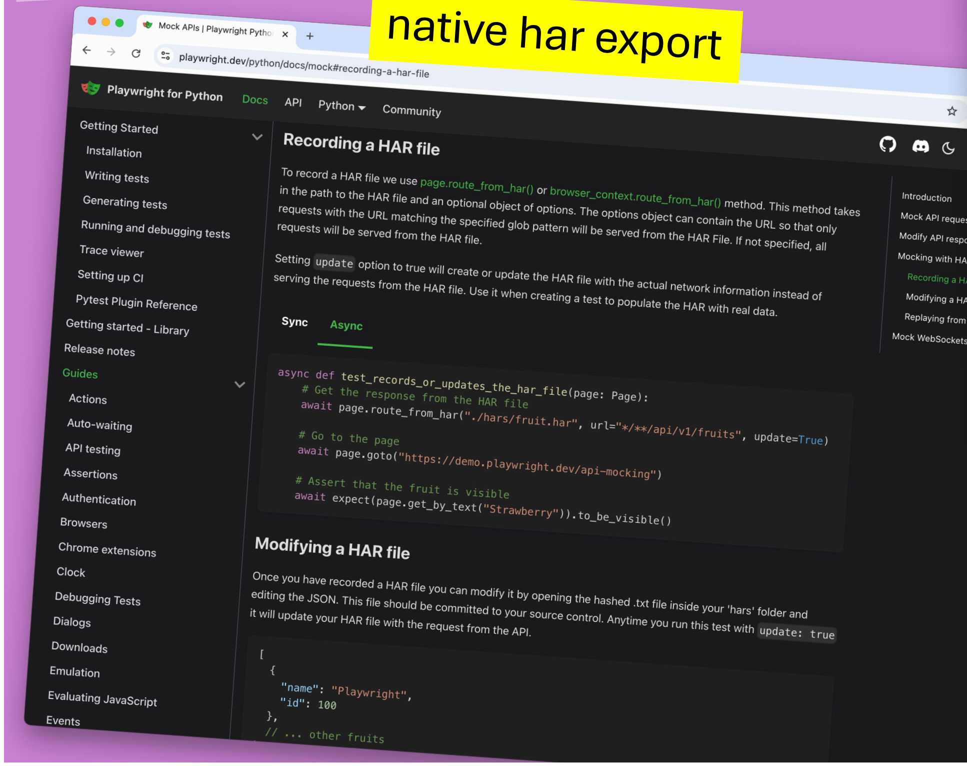
Any browser • Any platform • One API

Cross-browser. Playwright supports all modern rendering engines including Chromium, WebKit, and Firefox.

```python
from playwright.async_api import async_playwright
playwright = await async_playwright().start()
browser = await playwright.chromium.launch(headless=False)
page = await browser.new_page()
```

just kidding! Automate with Playwright!

native har export



Recording a HAR file

To record a HAR file we use page.route_from_har() or browser_context.route_from_har() method. This method takes in the path to the HAR file and an optional object of options. The options object can contain the URL so that only requests with the URL matching the specified glob pattern will be served from the HAR File. If not specified, all requests will be served from the HAR file.

Setting `update` option to true will create or update the HAR file with the actual network information instead of serving the requests from the HAR file. Use it when creating a test to populate the HAR with real data.

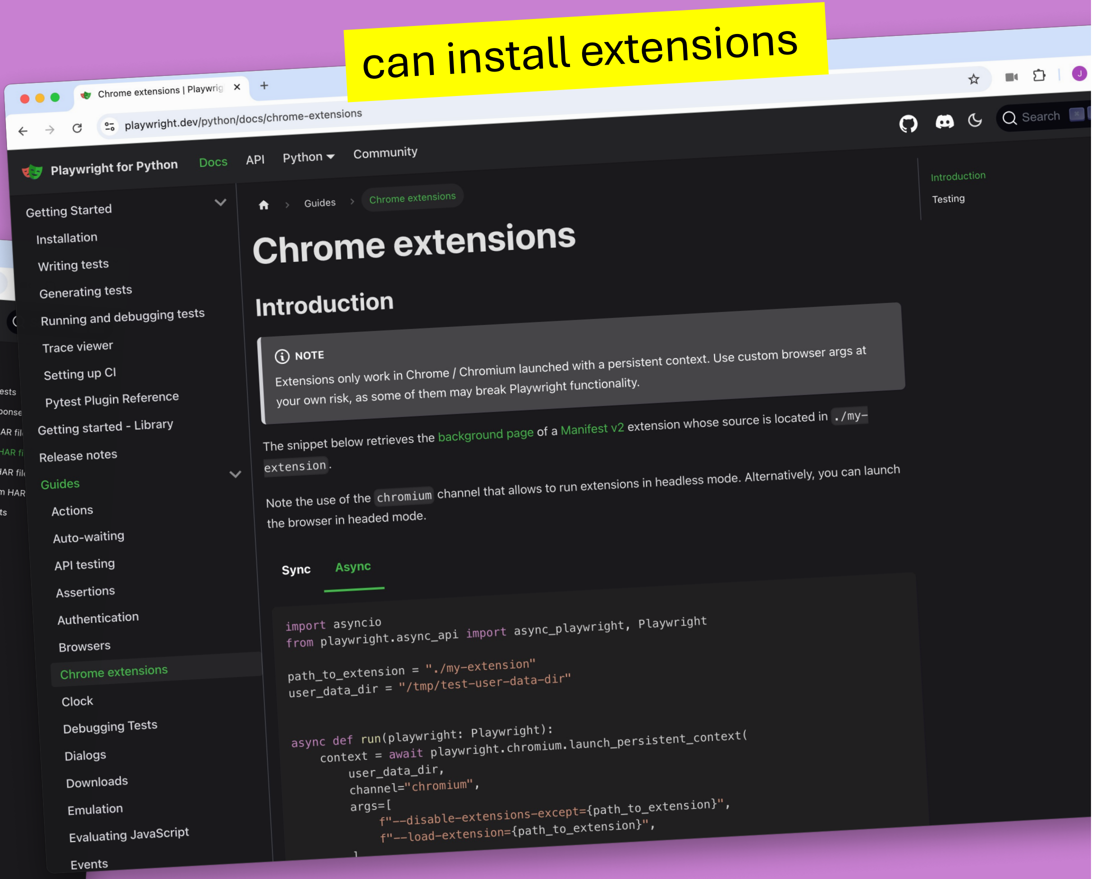
```
async def test_records_or_updates_the_har_file(page: Page):
    # Get the response from the HAR file
    await page.route_from_har("./hars/fruit.har", url="*/**/api/v1/fruits", update=True)

    # Go to the page
    await page.goto("https://demo.playwright.dev/api-mocking")

    # Assert that the fruit is visible
    await expect(page.get_by_text("Strawberry")).to_be_visible()
```

Modifying a HAR file

Once you have recorded a HAR file you can modify it by opening the hashed .txt file inside your 'hars' folder and editing the JSON. This file should be committed to your source control. Anytime you run this test with `update: true` it will update your HAR file with the request from the API.

```
[
  {
    "name": "Playwright",
    "id": 100
  },
  // ... other fruits
```

can install extensions



Chrome extensions

Introduction

> ⓘ NOTE
>
> Extensions only work in Chrome / Chromium launched with a persistent context. Use custom browser args at your own risk, as some of them may break Playwright functionality.

The snippet below retrieves the background page of a Manifest v2 extension whose source is located in `./my-extension`.

Note the use of the `chromium` channel that allows to run extensions in headless mode. Alternatively, you can launch the browser in headed mode.

```
import asyncio
from playwright.async_api import async_playwright, Playwright

path_to_extension = "./my-extension"
user_data_dir = "/tmp/test-user-data-dir"


async def run(playwright: Playwright):
    context = await playwright.chromium.launch_persistent_context(
        user_data_dir,
        channel="chromium",
        args=[
            f"--disable-extensions-except={path_to_extension}",
            f"--load-extension={path_to_extension}",
        ]
```

https://bit.ly/nicar-passive

Passive Scraping for social media

(and everything else)

Jonathan Soma
Columbia Journalism School
js4571@columbia.edu
@dangerscarf