

メディアアート・プログラミング2

東京藝術大学 芸術情報センター開設科目 後期金曜4限 第10週

2023.12.08 松浦知也 (matsura.tomoya@noc.geidai.ac.jp teach@matsuuratomoya.com)



Webスクレイピング

3 DEGREES OF SEPARATION FROM THE MILITARY-INDUSTRIAL-PRISON- DATA-SURVEILLANCE STATE(2015)

Sam Lavigne

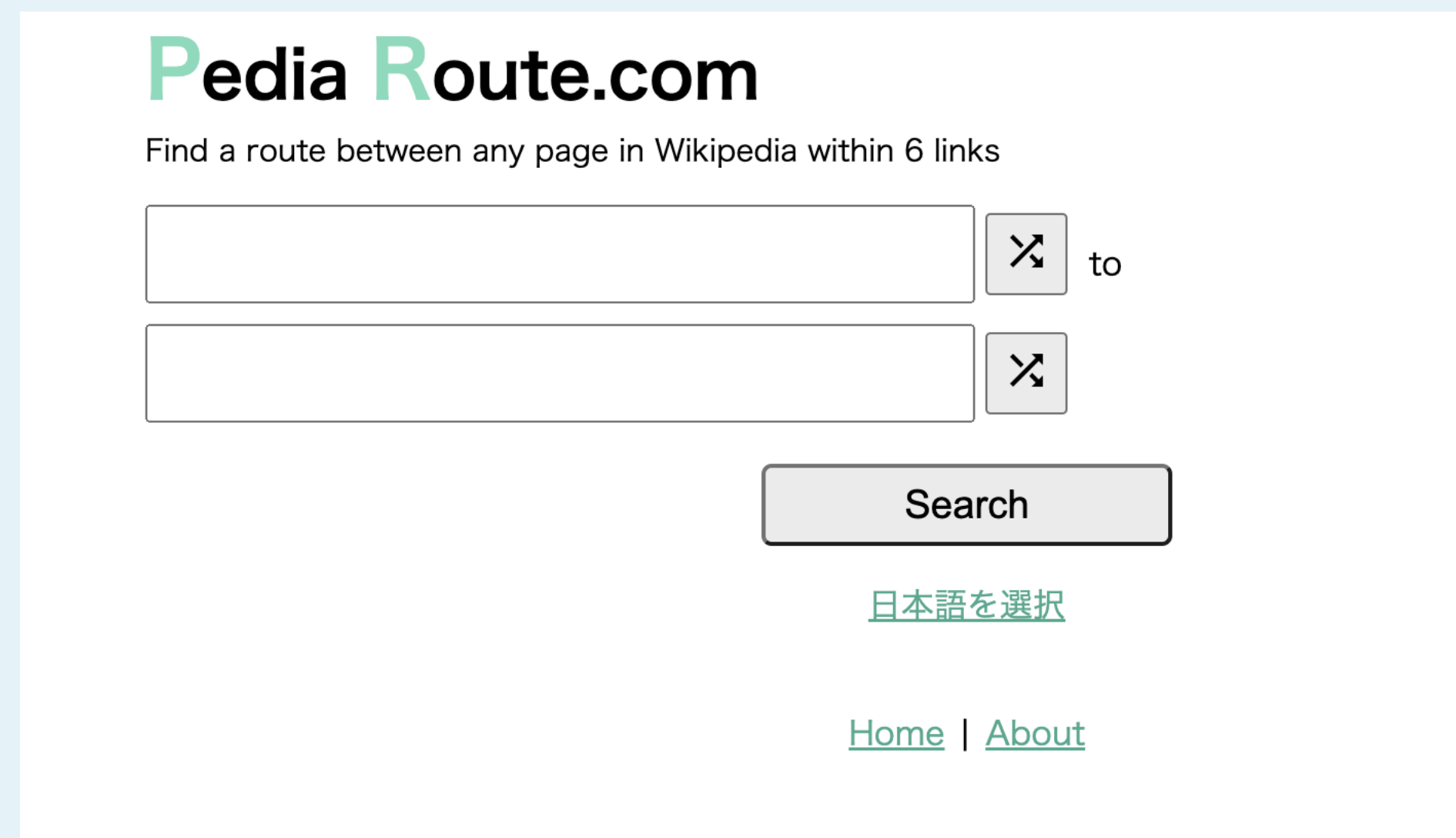


- 自身のLinkedInの「知り合いの知り合い」の中にどれだけ「監視テクノロジー」等に長けた人がいるかを可視化

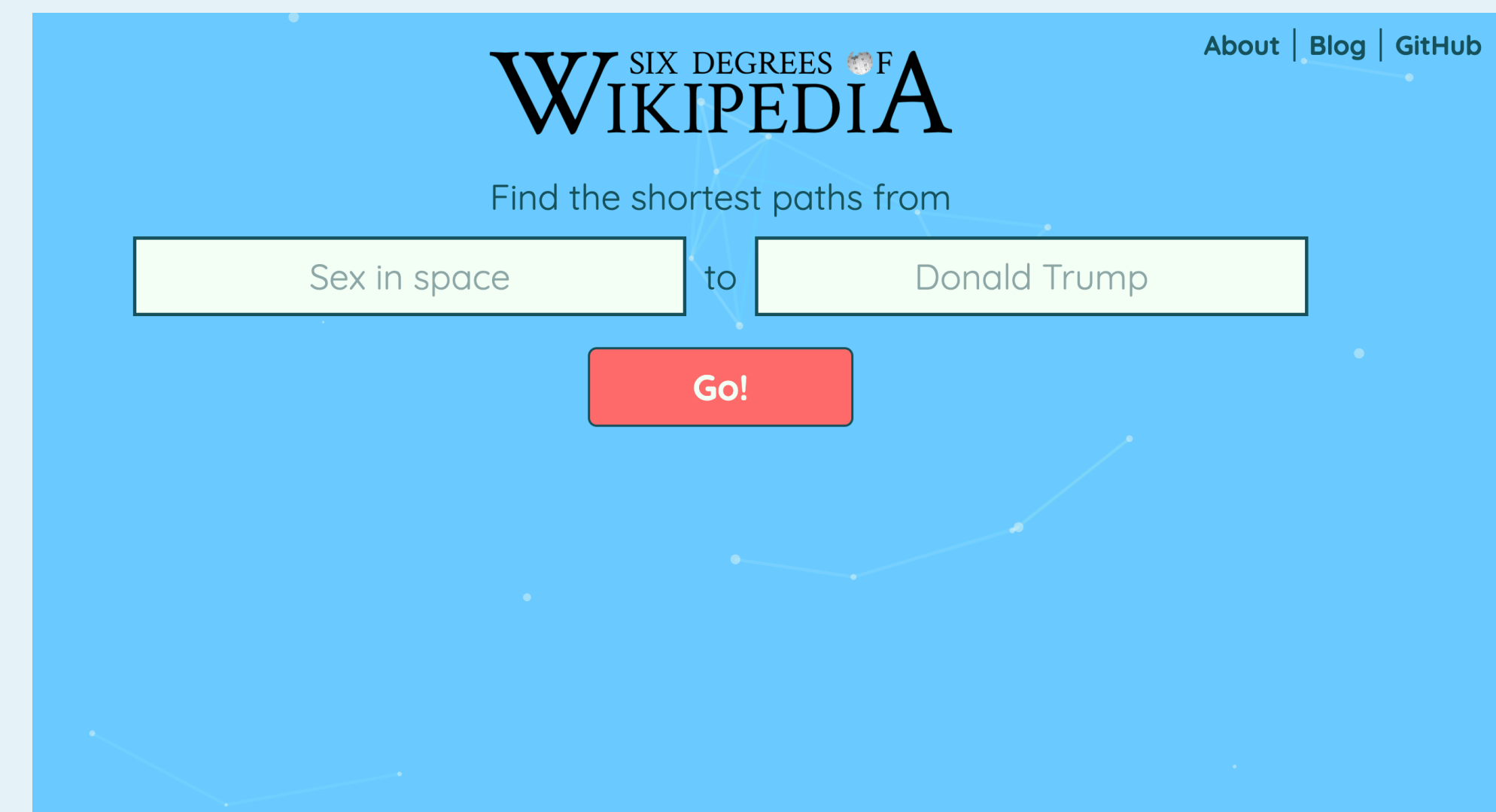
<https://linkedd.s3.amazonaws.com/index.html>



<https://wikitter.info/>



<https://pediaroute.com/>




<https://www.sixdegreesofwikipedia.com/>

TheirTube(2020)

木原共

The screenshot shows a YouTube channel page for 'Conspiracist'. The channel's profile picture is a robot with a red YouTube play button on its chest and a triangle with an eye on its head. The channel description reads: 'This persona is based on a person who was deeply into conspiracy theories. The videos recommended in this YouTube account tend to suggest global events are in fact conspiracies.' To the right, a 'Watched videos from...' section lists 'Buzzfeed Unsolved Network', 'Uncovered', 'The Truth Factory', and 'shane'. Below this is a browser window showing a grid of video recommendations. The first row includes 'The Maritime Mystery of The Mary Celeste' (1.1M views), 'The Canadian Military Declares War on Canadians' (113K views), 'Switching Lives with Jeffree Star' (35M views), and 'The Search For D. B. Cooper' (5.1M views). The second row includes 'EPISODE 8 THE SUMERIANS FALL OF THE FIRST CITIES' and 'FIREPLACE REALTIME' (4K HDR).

- Youtubeで特定のワードを検索するペルソナを複数作り、その人にレコメンドされる動画を可視化する
- スクレイピングには今日使うpuppeteerを使用&ソースも公開されている


TheirTube Public

👁️ Watch 6


🍴 Fork 14

★ Starred 131

🔗 master ▾
🌿 8 Branches
🏷️ 0 Tags

+ Add file ▾

<> Code ▾


kihapper Merge pull request #5 from guysimard/patch-1

4cb6951 · 3 years ago
🕒 13 Commits

📁 .vscode	Initial Commit	3 years ago
📁 cookies/curator1_cookie	Remove, ignore .DS_Store files	3 years ago
📁 images	Initial Commit	3 years ago
📁 src	Remove, ignore .DS_Store files	3 years ago
📄 .gitignore	Remove, ignore .DS_Store files	3 years ago
📄 README.md	Fixed typo in read me.md	3 years ago
📄 package-lock.json	fixed dependancy issues	3 years ago
📄 package.json	Initial Commit	3 years ago

📖 README
✎ ☰

TheirTube Scraper

Theirtube is a Youtube filter bubble simulator that provides a look into how videos are recommended on other people's YouTube. [🔗their.tube](https://www.their.tube).

TheirTube scraper allows to retrieve your own Youtube recommendation results on your local environment

About

TheirTube scraper allows you to retrieve your own Youtube recommendation results on your local environment

[🔗 www.their.tube](https://www.their.tube)

nodejs
youtube
puppeteer

- 📖 Readme
- 📈 Activity
- ★ 131 stars
- 👁️ 6 watching
- 🍴 14 forks

Report repository


Releases

No releases published


Packages

No packages published

Contributors 3



kihapper kihapper



gnepick John Karahalie

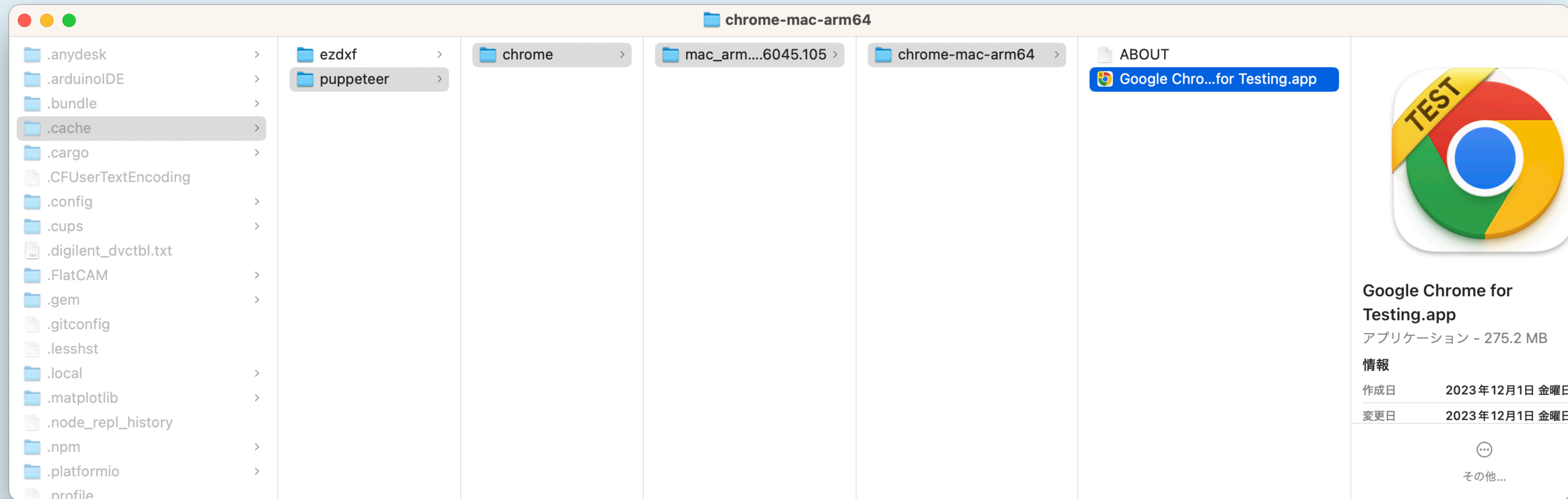
<https://github.com/kihapper/TheirTube>

puppeteerを使ってみよう

インストール

- 空のフォルダを新規作成、VSCodeにドラッグ&ドロップ
- ターミナルを開いてnpm init
- npm install -S puppeteer でインストール

```
{ } package.json > { } dependencies > puppeteer
1   {
2     "name": "20231201-scraping_test",
3     "version": "1.0.0",
4     "description": "",
5     "main": "index.js",
6     "scripts": {
7       "test": "echo \"Error: no test specified\" && exit 1"
8     },
9     "author": "",
10    "license": "ISC",
11    "dependencies": {
12      "puppeteer": "^21.5.2"
13    }
14  }
15
```



- puppeteerは自動でテスト用Google Chromeをインストールするが、サイズが大きいためnode_modulesではなく~/.cacheに入る
- 演習室のMacで作業する場合は毎回npm install すること

とりあえず開く&スクリーンショット撮影

1_launch.js

```
const puppeteer = require('puppeteer');

const main = async () => {
  let browser = await puppeteer.launch({ headless: false });
  let page = await browser.newPage();
  await page.goto('https://www.youtube.com/');
  await page.screenshot({ path: './ss.png' });
  await page.close();
  await browser.close();
};

main();
```

とりあえず開く&スクリーンショット撮影

1_launch.js

```
const puppeteer = require('puppeteer');

const main = async () => {
  let browser = await puppeteer.launch({ headless: false });
  let page = await browser.newPage();
  await page.goto('https://www.youtube.com/');
  await page.screenshot({ path: './ss.png' });
  await page.close();
  await browser.close();
};

main();
```

headlessをfalseにすると実際にブラウザが立ち上がってページ遷移の様子がチェックできる

とりあえず開く&スクリーンショット撮影

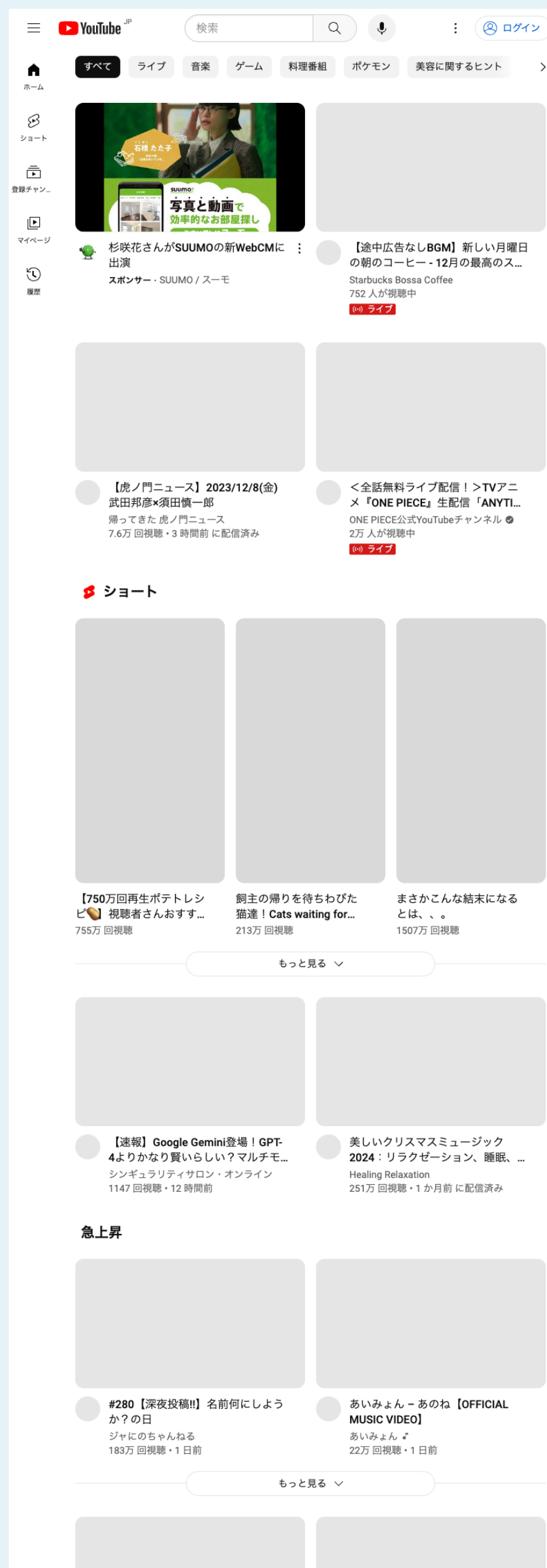
1_launch.js

```
const puppeteer = require('puppeteer');

const main = async () => {
  let browser = await puppeteer.launch({ headless: false });
  let page = await browser.newPage();
  await page.goto('https://www.youtube.com/');
  await page.screenshot({ path: './ss.png', fullPage: true });
  await page.close();
  await browser.close();
};

main();
```

fullPage:trueを追加するとページ全体のスクリーンショットも撮れる



ただし、Webページがスクリプトで動的に画像を読み込んだりしている場合、スクロール操作を自動化したり、ある程度の秒数待つ、のような操作をする必要も

その他のオプション

```
await puppeteer.launch({ headless: false ,args: ["--window-size=1920,1080"],timeout:0})
```

- ・ ウィンドウサイズによって画面のレイアウトや、出てくるHTMLタグが変わることもある
- ・ 標準だと30秒経って読み込みが終わらなかったり、要素が取得できないとエラーで終了
- ・ timeoutに0を指定すると無限に待ち続ける

asyncとawait

- これまでのコードは実行箇所がソースコードのどこか一つにあり、上から下に実行されていた
- Web系など、処理に時間がかかるものは順番に実行していると遅くなりがち
- 「処理Aの応答が帰ってきたらBを行う」という約束だけを決めて、処理を先に進めるのがasync
- 例えばファイル読み書きのためのfsモジュールもrequire("fs").promisesとするとasyncバージョンが使える

awaitを使わずにasync関数を繋げる

2_async_intro.js

```
const puppeteer = require('puppeteer');

const main = async () =>
  puppeteer.launch({ headless: false })
    .then(browser => browser.newPage())
    .then(page => page.goto('https://www.youtube.com/'))
    .then(_response => page.screenshot({ path: './ss.png' }))
    .then(_ => page.close())
    .then(_ => browser.close()));

main();
```

さっき実際に実行されていたのはこういうコード。

「Aしたら次にその結果を使ってBする」という処理の合成をしてからまとめて実行
(括弧がだんだんネストされてしまう)

await

```
const puppeteer = require('puppeteer');

const main = async () => {
  let browser = await puppeteer.launch({ headless: false });
  let page = await browser.newPage();
  await page.goto('https://www');
  await page.screenshot({ path: 'screenshot.png' });
  await page.close();
  await browser.close();
};

main();
```

asyncな関数を実行すると、Promise<A>という値が返ってくる。これに対して.thenが実行できる

awaitを使うと、その処理が終わるまで待機してPromiseの中身を取り出してくれる = 普通の関数のように扱える

mapとPromise.allを使って並列処理

3_map_async.js

```
const puppeteer = require('puppeteer');

const take_screen_shot = async (browser, url, index) => {
  let page = await browser.newPage();
  let path = `${index}.png`;
  await page.goto(url);
  await page.screenshot({ path: path });
  await page.close();
}

const main = async () => {
  let browser = await puppeteer.launch({ headless: false });
  const urllist = ["https://www.youtube.com",
    "https://www.google.com",
    "https://www.geidai.ac.jp",
    "https://yahoo.co.jp"]
  await Promise.all(urllist.map((url, index) => take_screen_shot(browser, url, index)));
  await browser.close();
}
```

mapとPromise.allを使って並列処理

3_map_async.js

```
const puppeteer = require('puppeteer');

const take_screen_shot = async (browser, url, index) => {
  let page = await browser.newPage();
  let path = `${index}.png`;
  await page.goto(url);
  await page.screenshot({ path: path });
  await page.close();
}

const main = async () => {
  let browser = await puppeteer.launch({ headless: false });
  const urllist = ["https://www.youtube.com",
    "https://www.google.com",
    "https://www.geidai.ac.jp",
    "https://yahoo.co.jp"]
  await Promise.all(urllist.map((url, index) => take_screen_shot(browser, url, index)));
  await browser.close(),
}
```

配列要素に対してasyncな関数をmapで全部に適用

Promise.all()はPromise<A>の配列を受け取って全部の処理が完了するまでのPromiseを返す = 自動で並列実行！

自動でスクロールダウンする

```
async function autoScroll(page, maxScrolls) {
  await page.evaluate(async maxScrolls => {
    await new Promise((then) => {
      let distance = 1000;
      let scrolls = 0;
      let timer = setInterval(() => {
        window.scrollBy(0, distance);
        scrolls += 1;
        if (scrolls >= maxScrolls) {
          clearInterval(timer);
          then();
        }
      }, 100);
    });
  }, maxScrolls);
}

exports.autoScroll = autoScroll;
```

autoscroll.jsとして保存

別ファイルの関数を呼び出す

4_auto_scroll.js

```
const puppeteer = require('puppeteer');
const autoScroll = require("./autoscroll.js").autoScroll;

const main = async () => {
  let browser = await puppeteer.launch({ headless: false });
  let page = await browser.newPage();
  await page.goto('https://www.youtube.com/');
  await autoScroll(page, 10);
  await page.screenshot({ path: "ss_full.png", fullPage: true });
  await page.close();
  await browser.close();
}
main();
```

リンクを辿ってみる

5_youtube_hopper.js

```
const puppeteer = require('puppeteer');
const autoScroll = require('./autoscroll.js').autoScroll;

const main = async () => {
  let browser = await puppeteer.launch({ headless: false });
  let page = await browser.newPage();
  await page.goto('https://www.youtube.com/');
  await autoScroll(page, 1);
  for (let i = 0; i < 10; i++) {
    const urlList = await page.$$eval("a#thumbnail", elems => elems.map(elem => elem.href));
    const url = urlList[0];
    console.log(url)
    await page.goto(url);
    await autoScroll(page, 10);
    await page.screenshot({ path: `${i}.png`, fullPage: true });
  }
  await page.close();
  await browser.close();
}
main();
```