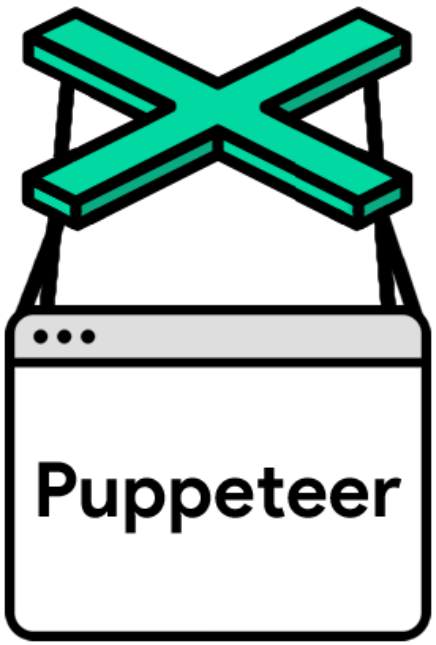
MASTER OF PUPPETS

https://przemyslawjanpietrzak.github.io/przemyslawjanpietrzak.github.io/puppeteer/dist

Puppeteer



OVERVIEW

* Driver on Chromium
* Written in Node.js
* async/await syntax

Syntax

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

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();
  await page.goto('https://news.ycombinator.com', {waitUntil: 'networkidle2'});
  await page.pdf({path: 'hn.pdf', format: 'A4'});

  await browser.close();
})();
```

CHAPTER I

HTML => PDF

CHAPTER II

Web scraping

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

let bookingUrl = 'https://booking.com';
(async () => {
  const browser = await puppeteer.launch({ headless: false });
  const page = await browser.newPage();
  await page.setViewport({ width: 1920, height: 926 });
  await page.goto(bookingUrl);

  const list = await page.evaluate(() => {
    const list = document.querySelector('#list-selector').children;
    console.log(list);
    return Array
      .from(list)
      .map(i =>  i.querySelector(".bh-carousel--new__price").textContent);
  });
  console.log(list);
})();
```

CHAPTER III
Lighthouse

```
chrome(async protocol => {
  const { Page, Runtime } = protocol;

  await Promise.all([ Page.enable(), Runtime.enable() ]);
  Page.navigate({ url });
  Page.loadEventFired(async () => {
    const timing = await Runtime.evaluate({
      expression: 'JSON.stringify(window.performance.timing)'
    });
    const paint = await Runtime.evaluate({
      expression: 'JSON.stringify(performance.getEntriesByType("paint"))'
    });
    const mark = await Runtime.evaluate({
      expression: 'JSON.stringify(performance.getEntriesByType("mark"))'
    });
    protocol.close();
    launcher.kill();
    resolve({
      timing: JSON.parse(timing.result.value),
      paint: JSON.parse(paint.result.value),
      mark: JSON.parse(mark.result.value)
    });
  });
})
```

CHAPTER IV
Gremlins

```
async () => {
  browser = await puppeteer.launch({
    headless,
  });
      page = await browser.newPage();
      page.on('pageerror', (error) => {
          errors.push(error);
        });

        await page.goto(host);
  await page.setViewport({height, width });
});
```

EPILOGUE

WHEN NOT?

* E2E testing
* Visual regression testing
* Web Scraping static pages
* Weak error handling

INSTEAD OF?

*** Cypress.io or Selenium for E2E testing**

*** Backstop.js for visual regression testing**

*** BeautifulSoup for web scraping static pages**

Thank you :*