MY472 - Data for Data Scientists Week 5: HTML, CSS, and Scraping Static Websites

Daniel de Kadt

23 October 2023

Plan for today

- Introduction
- Some key features of the internet
- HTML and CSS
- Fundamentals of web scraping
- Coding

Introduction

Examples

An increasing amount of data is available on the web

- Speeches, biographical information …
- Social media data, articles, press releases …
- Geographic information, conflict data …

These datasets are often provided in an **unstructured format**

Web scraping is the process of extracting this information automatically and transforming it into a **structured dataset**

Why automate?

Copy & pasting is time-consuming, boring, prone to errors, and impractical or infeasible

In contrast, automated web scraping

1. Scales well for large datasets
2. Allows for dynamic data collection
3. Is (mostly) reproducible
4. Involves adaptable techniques
5. Facilitates detecting and fixing errors

When to scrape?

1. Trade-off between your time today and your time in the future. Invest in your future self!
2. Computer time is often cheap; human time more expensive

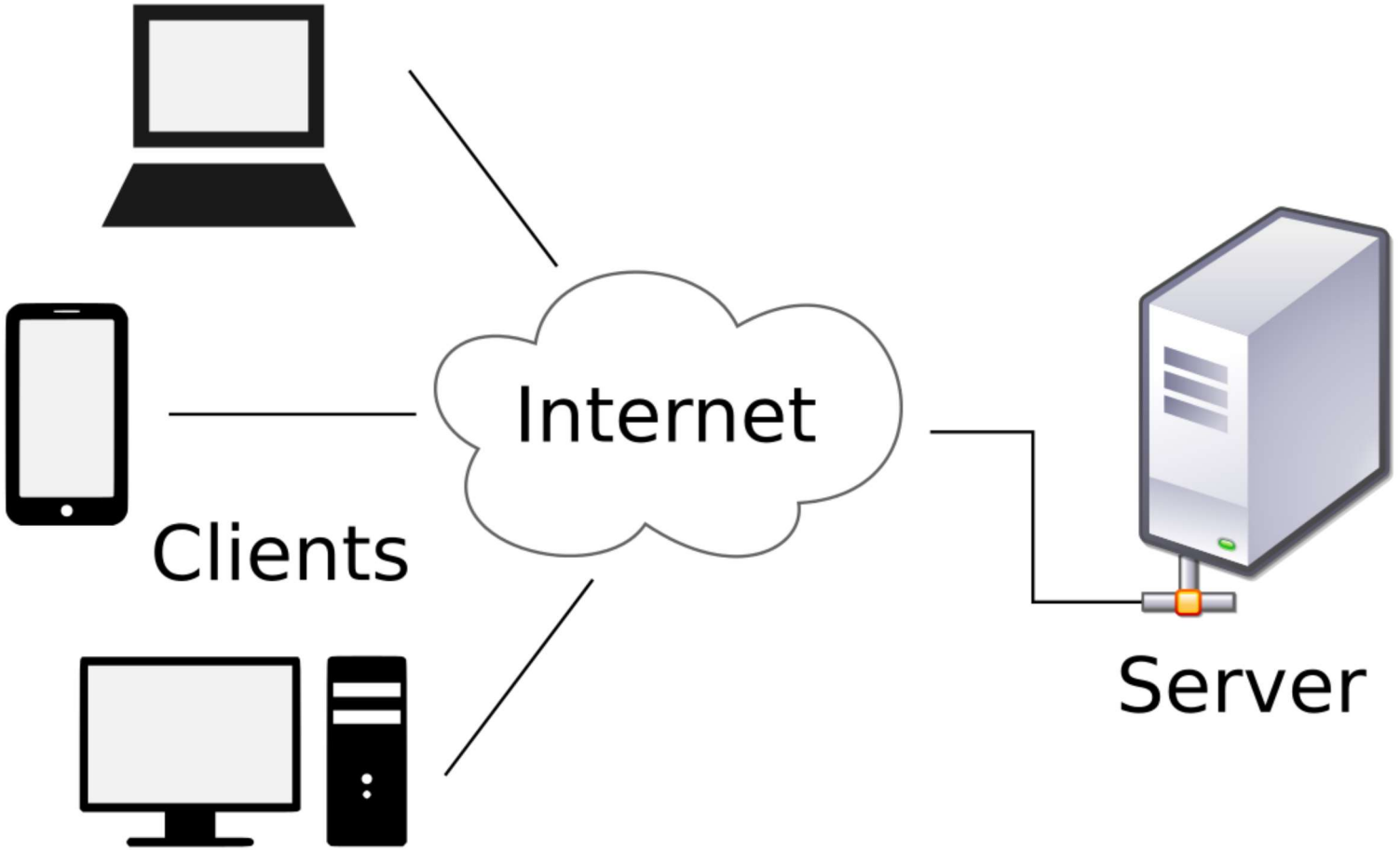
Obtaining data from the web: Two approaches

Two different approaches

1. **Screen scraping** Extract data from source code of website, with html parser and/or regular expressions
   - `rvest` (this week) and `RSelenium` packages (week 7) in R
2. **Web APIs** (week 8): A set of structured http requests that return JSON or XML data
   - `httr` package to construct API requests
   - Packages specific to each API: For example WDI, Rfacebook,
     - Check CRAN Task View on Web Technologies and Services for examples

Some key features of the internet

Client-server model



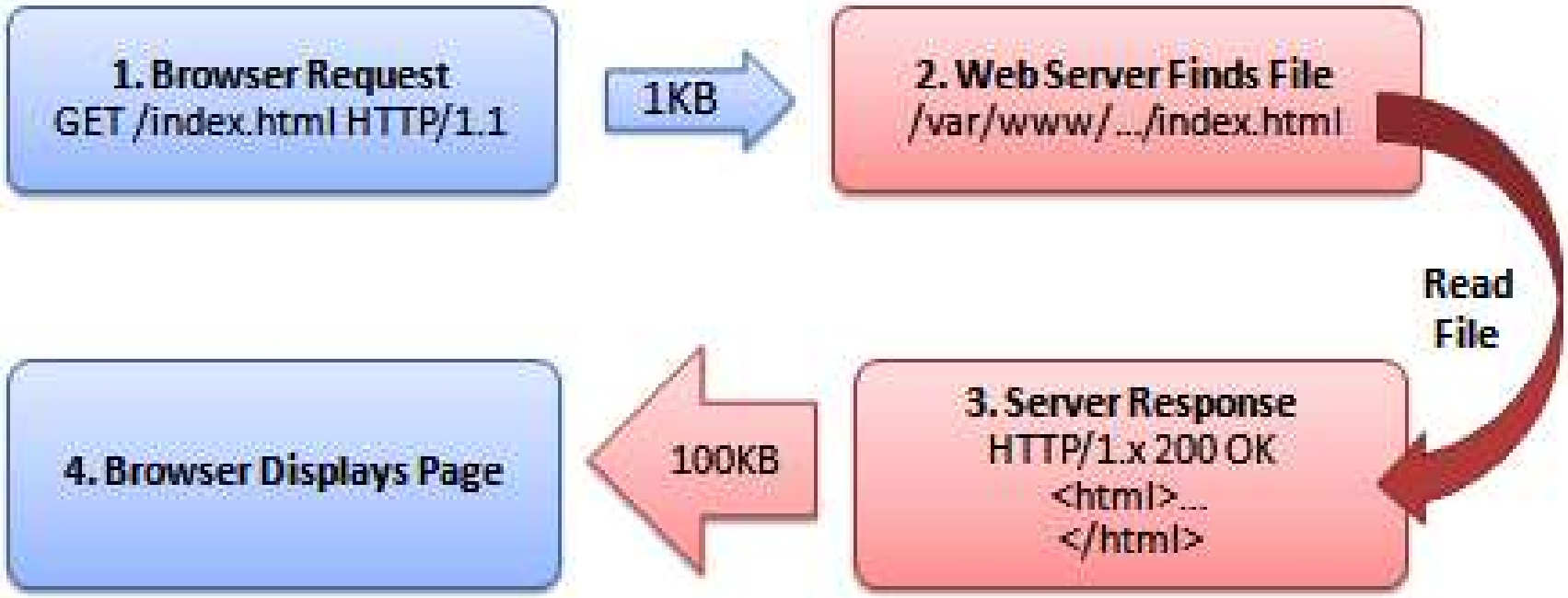
Client-server model

- Client: User computer, tablet, phone, software application, etc.
- Server: Web server, mail server, file server, Jupyter server, etc.

1. Client makes request to the server
   - Depending on what you want to get, the request might be
     - HTTP: Hypertext Transfer Protocol
     - HTTPS: Hypertext Transfer Protocol Secure
     - SMTP: Simple Mail Transfer Protocol
     - FTP: File Transfer Protocol
2. Server returns response

Request and response in the case of HTTP

From StackOverflow



Simple example: MY472 website

Let's see a very simple example of https://lse-my472.github.io

MY472 Data for Data Scientists

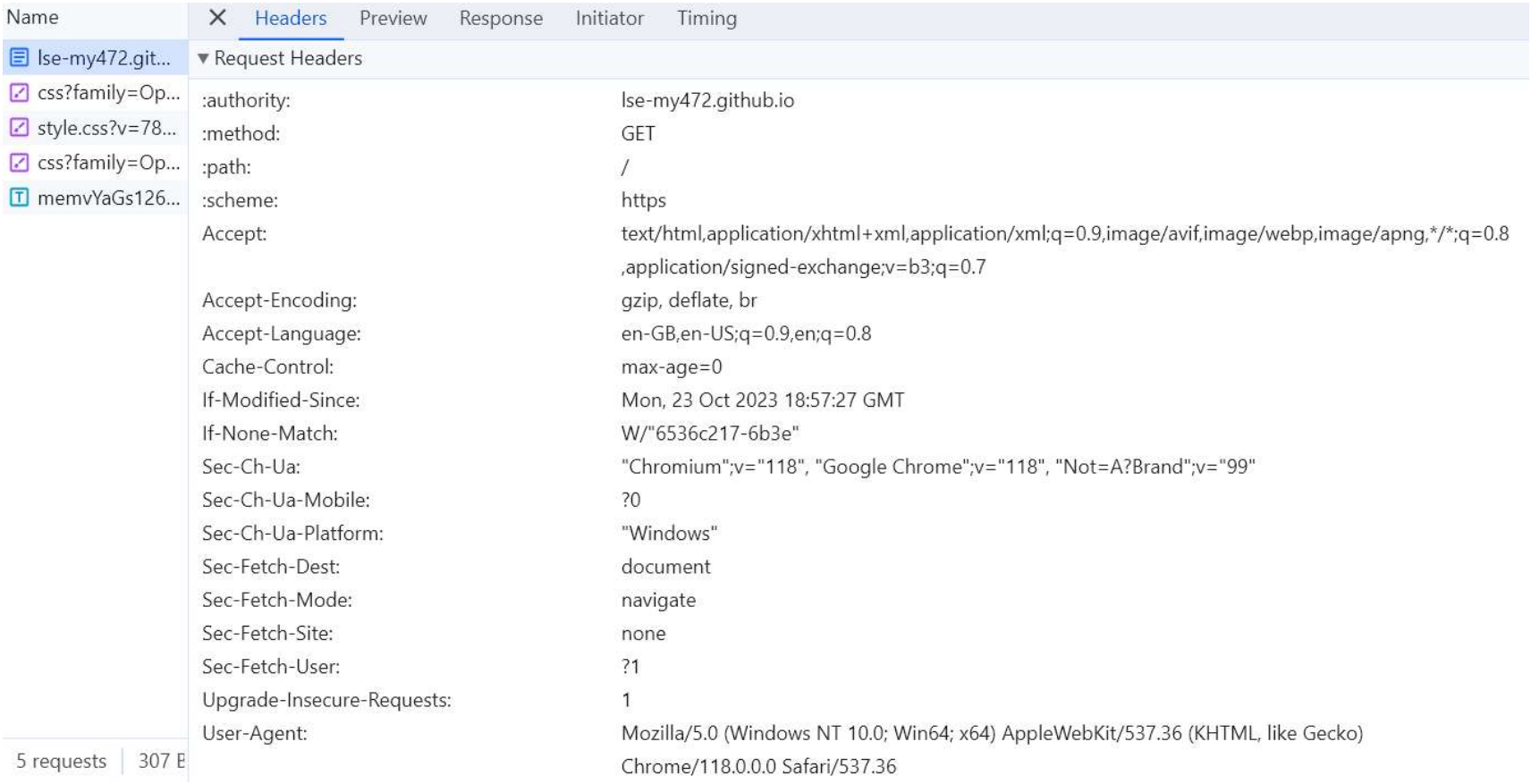
Michaelmas Term 2023

Main course repo

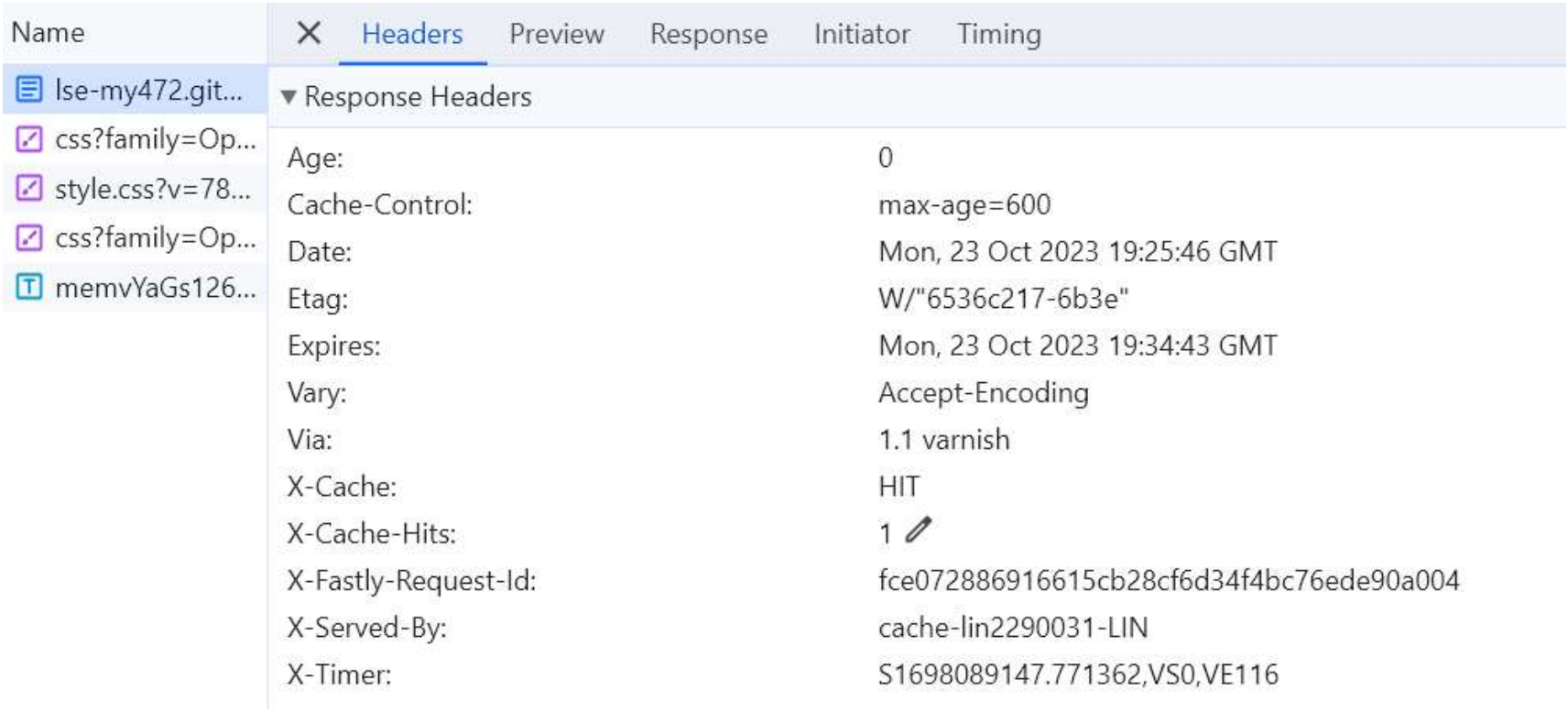
Moodle page

Simple example: MY472 website

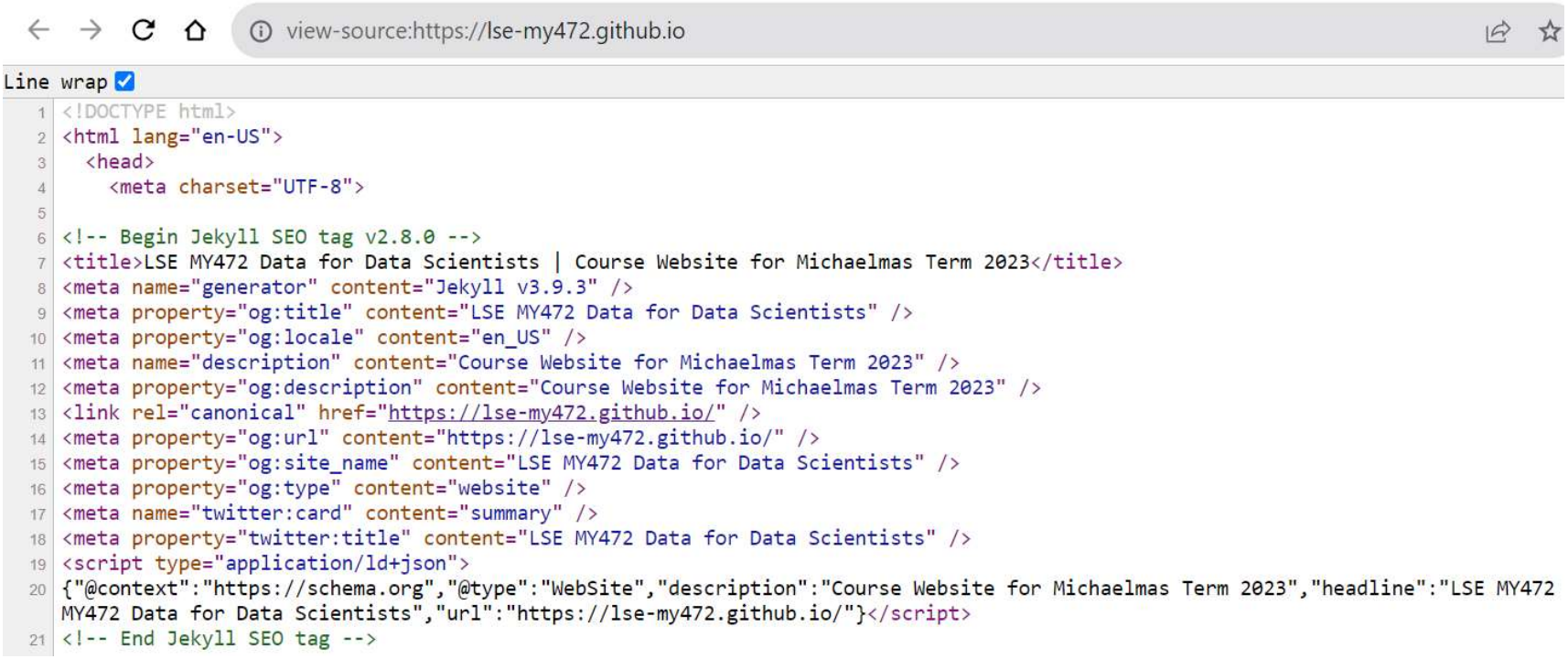
Simple example: Request headers



Simple example: Response headers



Simple example: Reponse content



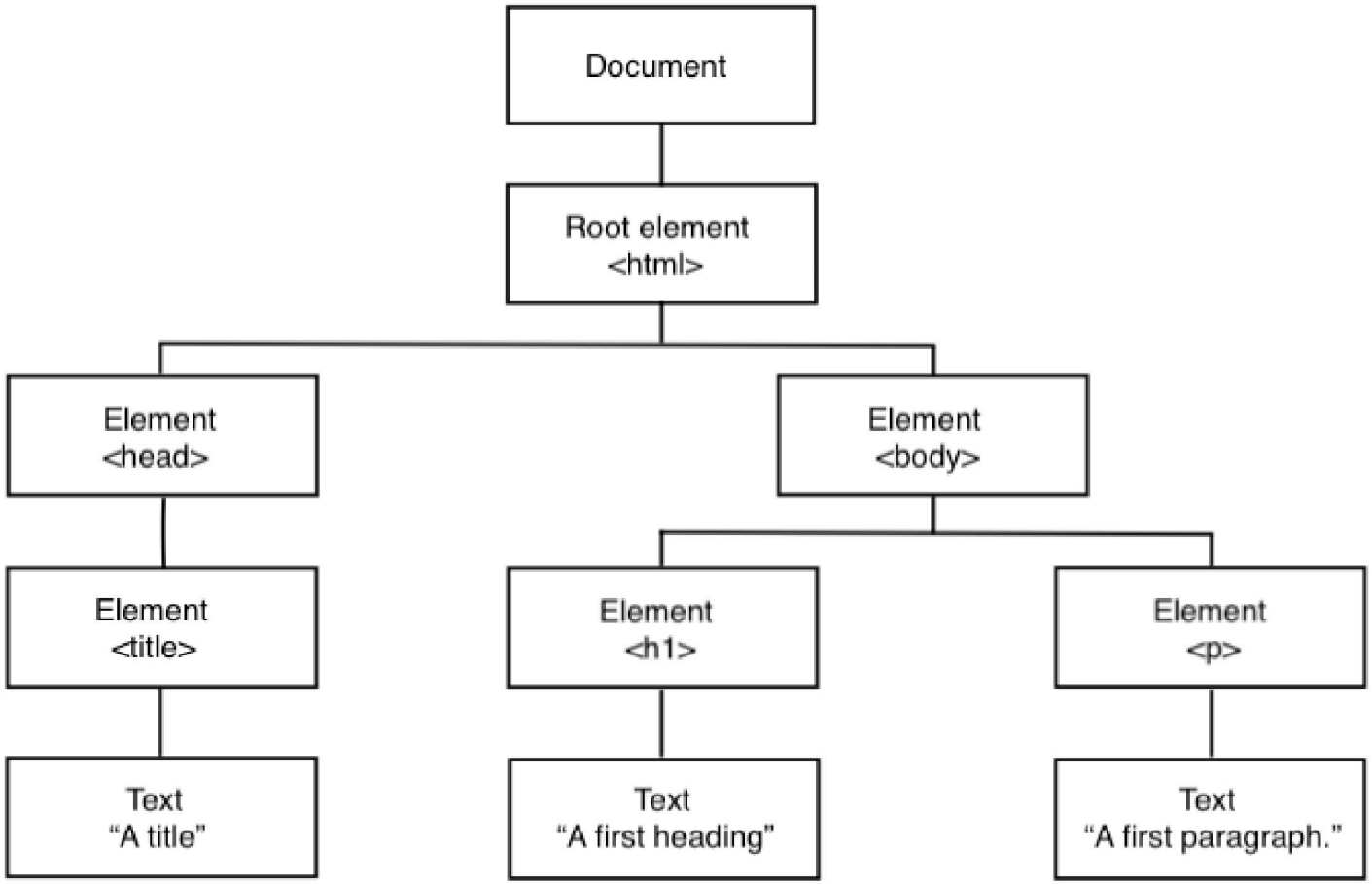
HTML and CSS

HTML

HTML: Hypertext Markup Language

- HTML displays mostly **static** content
- Many contents of dynamic webpages cannot be found in HTML
  - Example: Google Maps
- Understanding what is static and dynamic in a webpage is a crucial first step for web scraping

HTML tree structure



A very simple HTML file

```
<!DOCTYPE html>
<html>
    <head>
        <title>A title</title>
    </head>
    <body>
        <h1>A first heading</h1>
        <p>A first paragraph.</p>
    </body>
</html>
```

From: https://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro

Slightly more features

```
<!DOCTYPE html>
<html>
    <head>
        <title>A title</title>
    </head>
    <body>
        <h1>A first heading</h1>
        <p>A first paragraph.</p>
        <p>A second paragraph with some <b>formatted</b> text.</p>
        <p>A third paragraph with a <a href="http://www.lse.ac.uk">hyperlink</a>.</p>
    </body>
</html>
```

With some content divisions

```
<!DOCTYPE html>
<html>
    <head>
        <title>A title</title>
    </head>
    <body>
        <div>
            <h1>Heading of the first division</h1>
            <p>A first paragraph.</p>
            <p>A second paragraph with some <b>formatted</b> text.</p>
            <p>A third paragraph with a <a href="http://www.lse.ac.uk">hyperlink</a>.</p>
        </div>
        <div>
            <h1>Heading of the second division</h1>
            <p>Another paragraph with some text.</p>
        </div>
    </body>
</html>
```

Beyond plain HTML

1. **Cascading Style Sheets (CSS)** Style sheet language which describes formatting of HTML components, useful for us because of selectors

2. **Javascript**: Adds functionalities to the websites, e.g. change content/structure after website has been loaded

Adding some simple CSS (1/2)

```
<!DOCTYPE html>
<html>
    <head>

        <!-- CSS start -->
        <style>
        p {
        color: green;
        }
        </style>
        <!-- CSS end -->

        <title>A title</title>
    </head>
    <body>
        <div>
            <h1>Heading of the first division</h1>
            <p>A first paragraph.</p>
            <p>A second paragraph with some <b>formatted</b> text.</p>
            <p>A third paragraph with a <a href="http://www.lse.ac.uk">hyperlink</a>.</p>
        </div>
        <div>
            <h1>Heading of the second division</h1>
            <p>Another paragraph with some text.</p>
        </div>
    </body>
</html>
```

Adding some simple CSS (2/2)

```
<!DOCTYPE html>
<html>
    <head>

        <!-- CSS start -->
        <style>
        .text-about-web-scraping {
          color: orange;
        }
        .division-two h1 {
        color: green;
          }
        </style>
        <!-- CSS end -->

        <title>A title</title>
    </head>
    <body>
        <div>
            <h1>Heading of the first division</h1>
            <p>A first paragraph.</p>
            <p>A second paragraph with some <b>formatted</b> text.</p>
            <p class="text-about-web-scraping">A third paragraph now containing some text about web scraping ...</p>
        </div>
        <div class="division-two">
            <h1>Heading of the second division</h1>
            <p>Another paragraph with some text.</p>
            <p class="text-about-web-scraping">A last paragraph discussing some web scraping ...</p>
        </div>
    </body>
```

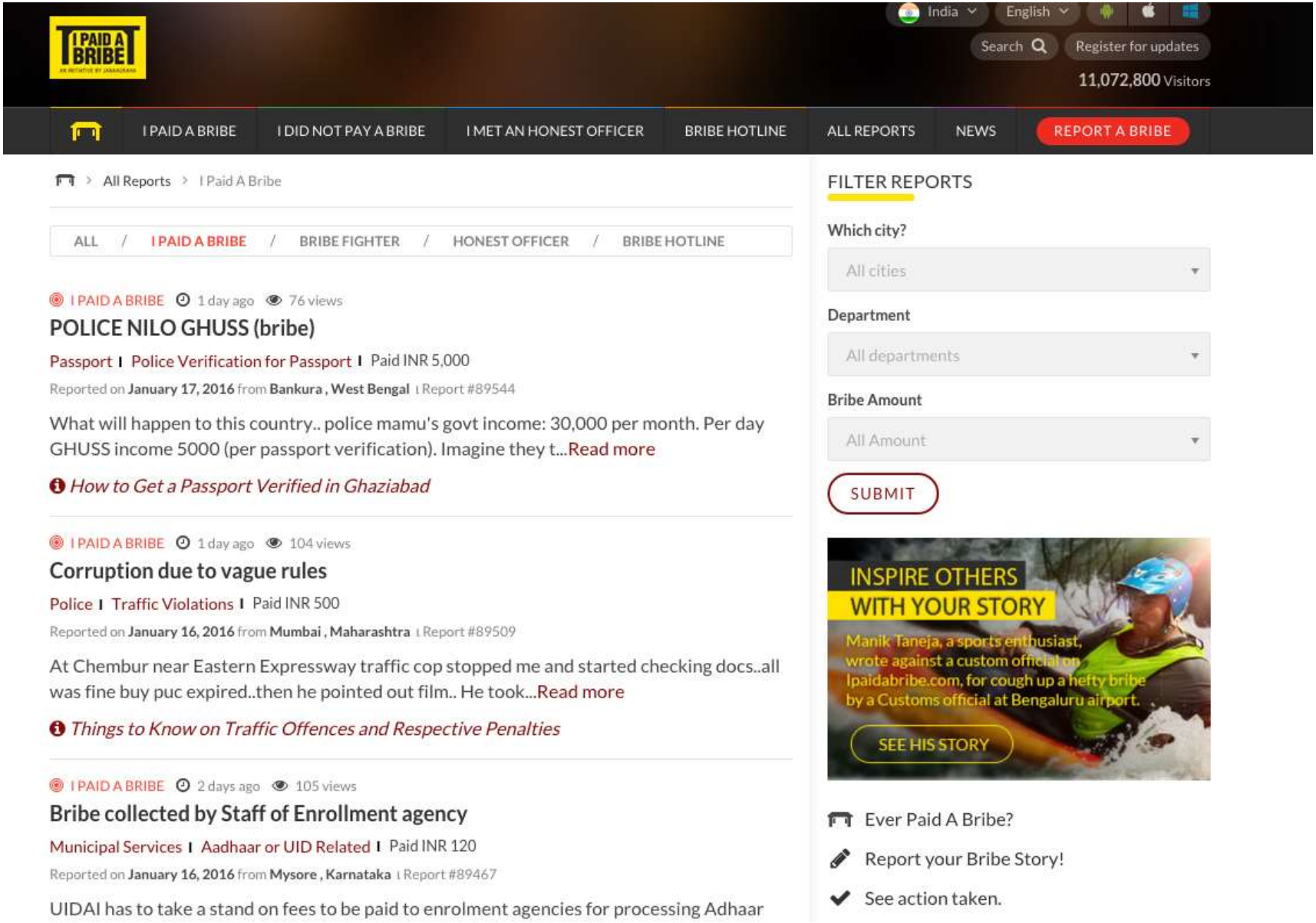
Fundamentals of web scraping

Scenario 1: Data in table format

List of international courts [edit]

Name	Scope	Years active	Subject matter
International Court of Justice	Global	1945–present	General disputes
International Criminal Court	Global	2002–present	Criminal prosecutions
Permanent Court of International Justice	Global	1922–1946	General disputes
Appellate Body	Global	1995–present	Trade disputes within the WTO
International Tribunal for the Law of the Sea	Global	1994–present	Maritime disputes
African Court of Justice	Africa	2009–present	Interpretation of AU treaties
African Court on Human and Peoples' Rights	Africa	2006–present	Human rights
COMESA Court of Justice	Africa	1998–present	Trade disputes within COMESA
ECOWAS Community Court of Justice	Africa	1996–present	Interpretation of ECOWAS treaties
East African Court of Justice	Africa	2001–present	Interpretation of EAC treaties
SADC Tribunal	Africa	2005–2012	Interpretation of SADC treaties

Scenario 2: Data in unstructured format



www.ipaidabribe.com/reports/paid

Scenario 3: Hidden behind web forms

Three main scenarios

1. Data in *table* format
   - Automatic extraction with **rvest** or select specific table with *inspect element* in browser
2. Data in *unstructured* format
   - Element identification key in this case
     - *Inspect element* in browser
   - Identify the target e.g. with *CSS* (this week) or *XPath* selector (week 7)
   - Automatic extraction with **rvest**
3. Data hidden *behind web forms* (week 7)
   - Element identification to e.g. find text boxes, buttons, and results
   - Automation of web browser with **RSelenium**

Identifying elements via CSS selector (1/2)

- Selecting by tag-name
  - Example html code: `<h3>This is the main item</h3>`
  - Selector: `h3`
- Selecting by class
  - Example html code: `<div class  = 'itemdisplay'>This is the main item</div>`
  - Selector: `.itemdisplay`
- Selecting by id
  - Example html code: `<div id = 'maintitle'>my main title</div>`
  - Selector: `#maintitle`

Identifying elements via CSS selector (2/2)

- Selecting by tag structure
  - Example html code (hyperlink tag `a` inside `div` tag): `<div><a href = 'https://www.google.com'>Google Link</a></div>`
  - Selector: `div a`
- Selecting by nth child of a parent element
  - Example html code: `<body><p>First paragraph</p><p>Second paragraph.</p></body>`
  - Selector of second paragraph: `body > p:nth-child(2)`

You don't have to figure these out yourself: inspect!

Reference and further examples:
https://www.w3schools.com/cssref/css_selectors.asp

The rules of the game

1. Respect the hosting site's wishes
   - Check if an API exists or if data are available for download
   - Respect copyright and ethics; what are you allowed to do?
   - Keep in mind where data comes from and give credit
   - Some websites disallow scrapers via `robots.txt` file
2. Limit your bandwidth use
   - Wait some time after each hit
   - Scrape only what you need, and just once
3. When using APIs, read documentation
   - Is there a batch download option?
   - Are there any rate limits?
   - Can you share the data?

Coding

Markdown files this week

- 01-selecting-elements.Rmd
- 02-scraping-tables.Rmd