MY472 - Week 4: Basics of HTML and CSS

Pablo Barbera & Akitaka Matsuo

October 23, 2018

Scraping the web

Scraping the web: what?

An increasing amount of data is available on the web:

- Speeches, sentences, biographical information…
- Social media data, newspaper 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.**

Scraping the web: why?

Copy & pasting is time-consuming, boring, prone to errors, and impractical for large datasets

In contrast, automated web scraping:

1. Scales well for large datasets
2. Is reproducible
3. Involved adaptable techniques
4. 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 cheap; human time is expensive

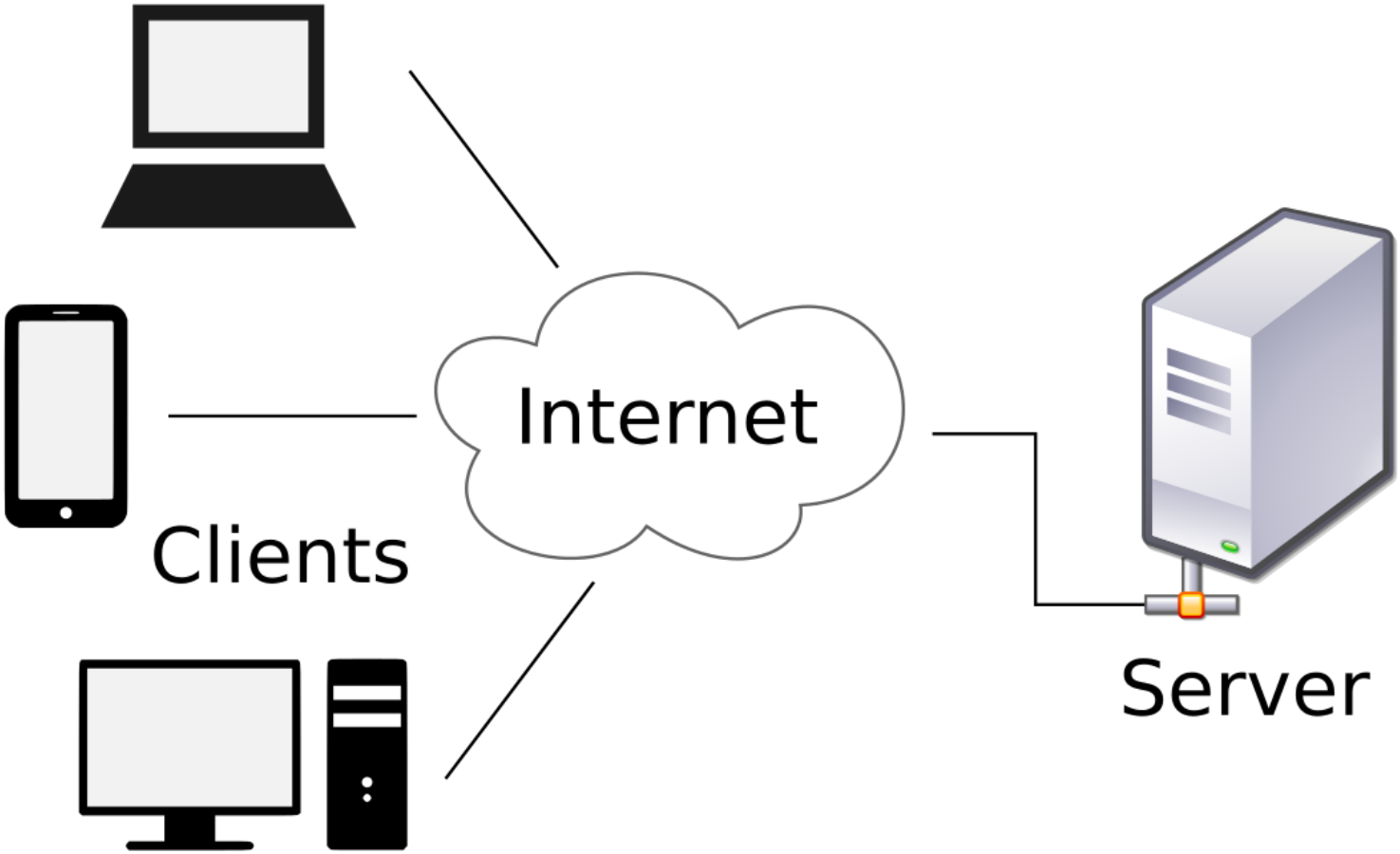
Scraping the web: two approaches

Two different approaches:

1. **Screen scraping**: extract data from source code of website, with html parser and/or regular expressions
   - `rvest` package in R
2. **Web APIs** (application programming interfaces): a set of structured http requests that return JSON or XML data
   - `httr` package to construct API requests
   - Packages specific to each API: weatherData, WDI, Rfacebook,
     - Check CRAN Task View on Web Technologies and Services for examples
   - More on APIs on Week 7

The Internet, how it works

Client-server model



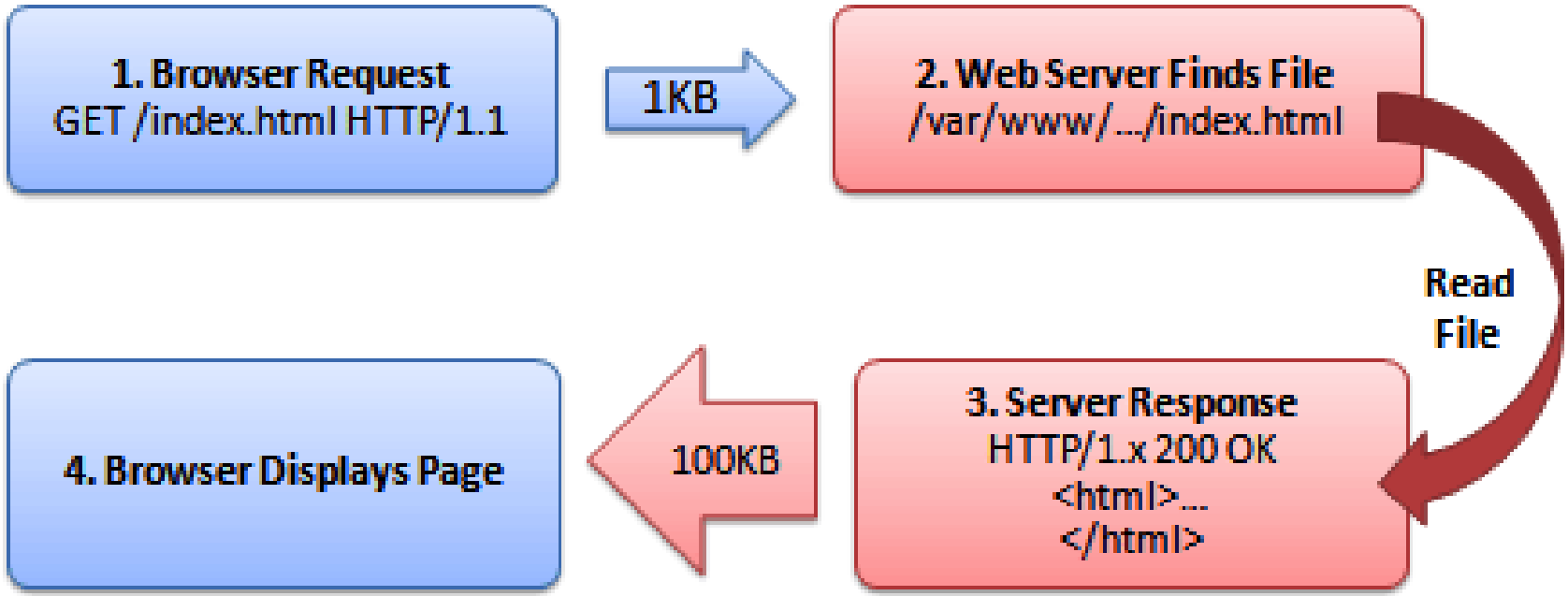
Client-server model

- Client: user computer; tablet; phone; software application; etc.
- Server: Jupyter server on Fabian; mail server; file server; web server; etc.

1. Client make requrests to the server
   - Depending on what you want to get, the request might be
     - HTTP
     - HTTPS
     - SMTP
     - FTP
2. Server returns something

In the case of HTTP request and response

From stackoverflow



Simple example: MY472 website

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

lse-my472.github.io

THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

MY472 Data for Data Scientists

Michaelmas Term 2018

Instructors

- Pablo Barberá, Department of Methodology. *Office hours*: Fridays 12.30-14.30, COL.7.10 (book via LSE for You)
- Akitaka Matsuo, Department of Methodology. *Office hours*: Tuesdays 11.30-12.30, COL.8.02 (book via LSE for You. Only weeks 3, 4, 5, 9)

Simple example: MY472 website

▼ **General**

Request URL: `https://lse-my472.github.io/`

Request Method: `GET`

Status Code: 🟢 `200`

Remote Address: `185.199.110.153:443`

Referrer Policy: `no-referrer-when-downgrade`

Simple example: Request headers

▾ **Request Headers**

:authority: lse-my472.github.io

:method: GET

:path: /

:scheme: https

accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8

accept-encoding: gzip, deflate, br

accept-language: en-US,en;q=0.9,ja;q=0.8,zh-CN;q=0.7,zh-TW;q=0.6,zh;q=0.5

upgrade-insecure-requests: 1

user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.67 Safari/537.36

Simple example: Response headers

▼ **Response Headers**

accept-ranges: bytes
access-control-allow-origin: *
age: 21
cache-control: max-age=600
content-encoding: gzip
content-length: 7753
content-type: text/html; charset=utf-8
date: Fri, 19 Oct 2018 12:51:30 GMT
etag: W/"5bc841de-5085"
expires: Fri, 19 Oct 2018 12:45:38 GMT
last-modified: Thu, 18 Oct 2018 08:18:38 GMT
server: GitHub.com
status: 200
strict-transport-security: max-age=31556952
vary: Accept-Encoding
via: 1.1 varnish
x-cache: HIT
x-cache-hits: 1
x-fastly-request-id: b4184e64b5a061bce2a6b9a85a94b41d80683e90
x-github-request-id: AD84:1E3D:EE3370:1362A72:5BC9CF96
x-served-by: cache-lcy19238-LCY
x-timer: S1539953490.243899,VS0,VE1

Simple example: Reponse contents

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Begin Jekyll SEO tag v2.5.0 -->
<title>lse-my472.github.io | Course handout web page for LSE MY472, Data for Data Scientists (Michaelmas Term 2018).</title>
<meta name="generator" content="Jekyll v3.7.4" />
<meta property="og:title" content="lse-my472.github.io" />
<meta property="og:locale" content="en_US" />
<meta name="description" content="Course handout web page for LSE MY472, Data for Data Scientists (Michaelmas Term 2018)." />
<meta property="og:description" content="Course handout web page for LSE MY472, Data for Data Scientists (Michaelmas Term 2018
<link rel="canonical" href="https://lse-my472.github.io/" />
<meta property="og:url" content="https://lse-my472.github.io/" />
<meta property="og:site_name" content="lse-my472.github.io" />
<script type="application/ld+json">
{"headline":"lse-my472.github.io","@type":"WebSite","url":"https://lse-my472.github.io/","name":"lse-my472.github.io","descrip
<!-- End Jekyll SEO tag -->

    <link rel="stylesheet" href="/assets/css/style.css?v=183b95c9358bbbd7c16f509a11ff112c9f74c481">
  </head>
  <body>
    <div class="container-lg px-3 my-5 markdown-body">
```

HTML and CSS

HTML

HTML: Hyper-Text Markup Language

- HTML displays mostly **static** contents.
- Many contents of dynamic webpages cannot be found anywhere in html
  - Example: google maps
- Understanding what's static and what's dynamic in a webpage is a crucial first step for web scraping

A simplest html file

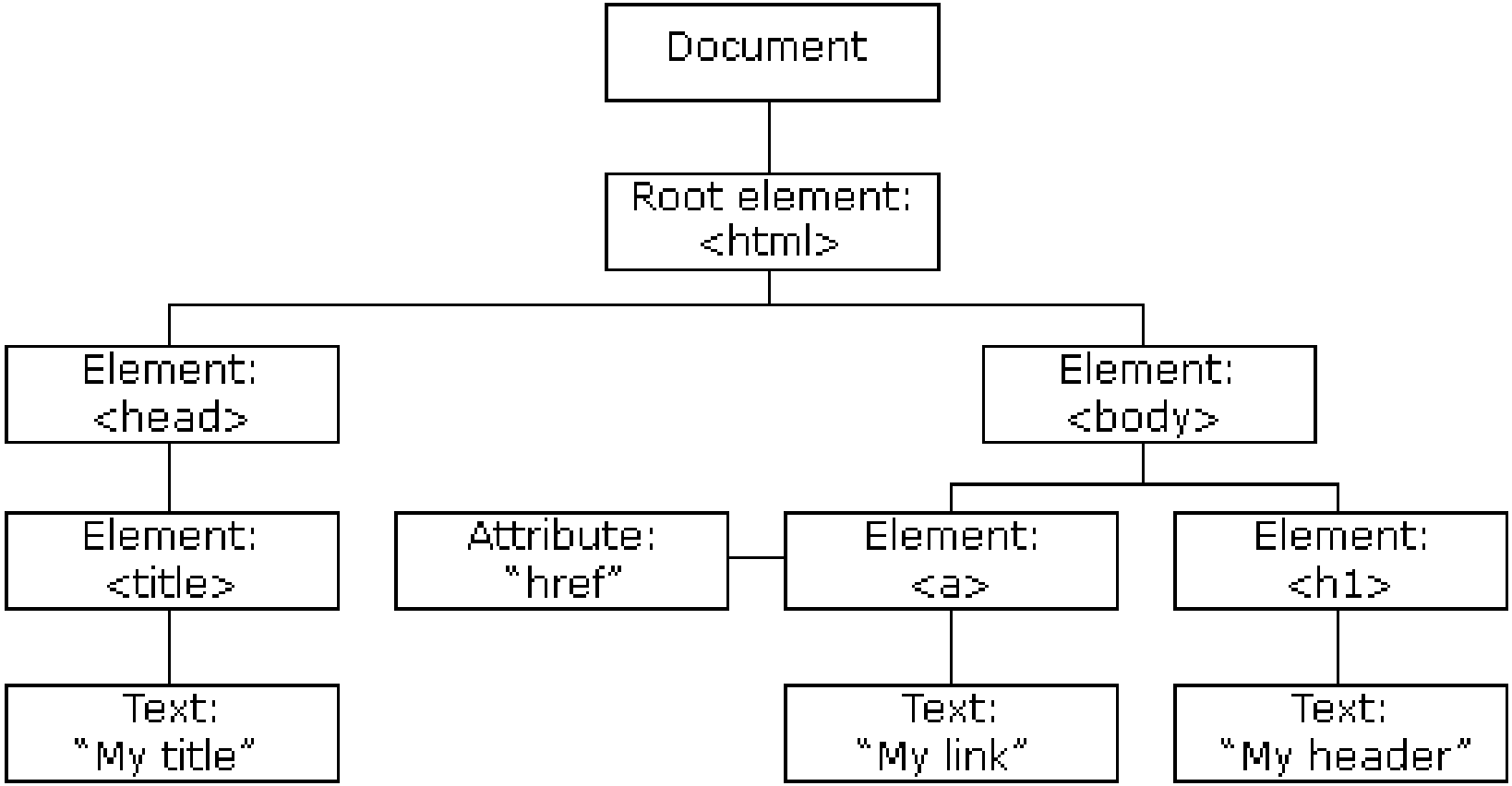
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro

```
<!DOCTYPE html>
<html>
<head>
    <title>Page Title</title>
</head>
<body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
</body>
</html>
```

Another simple html file

```html
<!DOCTYPE html>
<html>
<head>
    <title>My Title</title>
</head>
<body>
    <h1>My Header</h1>
    <a href="http://kenbenoit.net">My link</a>
</body>
</html>
```

HTML Structure



Beyond HTML

1. **Cascading Style Sheets (CSS)** Describes formatting of HTML components, useful for us!

2. **Javascript**: adds functionalities to the website (e.g. change content/structure after website has been loaded)

Webscraping, three main scenarios

Scinario 1: Data in table format

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search

WIKIPEDIA
The Free Encyclopedia

Main page

International court

From Wikipedia, the free encyclopedia

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

Scinario 2: Data in unstructured format

Scinario 3: hidden behind web forms

Three main scenarios

1. Data in **table** format
   - Automatic extraction with `rvest`
2. Data in **unstructured** format
   - Element identification
     - `selectorGadget`
     - **Inspect** in browser
   - Identify the target with **CSS** or **xpath** selector
   - Automatic extraction with `rvest`
3. Data hidden `behind web forms`
   - Automation of web browser behavior with `selenium`

CSS Selector

We use it to select particular element from a webpage:

- 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 div='maintitle'>my main title</div>`
  - selector: `#maintitle`

CSS Selector

- selecting by tag structure
  - example html code: (link inside `div` tag)
    `<div><a href='https://www.google.com'>Google Link</a></div>`
    - selector: `div a`

Rerefernece: https://www.w3schools.com/cssref/css_selectors.asp

A similar but more powerful selector is **xpath**

The rules of the game

1. Respect the hosting site's wishes:
   - Check if an API exists or if data are available for download
   - Keep in mind where data comes from and give credit (and respect copyright if you want to republish the data!)
   - Some websites *disallow* scrapers on `robots.txt` file
2. Limit your bandwidth use:
   - Wait one or two seconds 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?