Webscraping 101
(with some Python)

Anna Vassilovski

May 25, 2016

Goals

Walk away from this talk knowing:

1. What problems scrapers address
2. How they work
3. How to build one (general steps + Py-example)

Typical Problem

Scenario:

Location	Content	Format
City of Toronto Website	News Releases Since 20xx	Table

Problem: How to get the data?

Scraper

What is a scraper?

Tool to download and extract digital content

aka:

- Machinery behind “Get Data” button
- Provides a custom API

General concept with custom applications

Problem scrapers address

Scenario:

Location	Content	Format
Web	Text	Table
Disk	Images	Non Table
Email	Video	Spreadsheet
		RSS Feed

Problem: How to get the data?

Scraper

Who Uses Scrapers?

- Data aggregators
  - search engines (ex Google), job boards (ex Indeed), event aggregators, real estate related (ex WalkScore)
- Businesses
  - price monitoring, reputation monitoring, market research
- Financial firms
  - signal research, “alternative data”
- Academics
- And many more…

- http://blog.datahut.co/how-real-businesses-use-web-scraping/
- https://www.quora.com/What-are-examples-of-how-real-businesses-use-web-scraping

Problem scrapers address

Scenario:

Location	Content	Format
Web	Text	Table
Disk	Images	Non Table
Email	Video	Spreadsheet
		RSS Feed

Problem: How to get the data?

Scraper

Problem addressed today

Scenario:

Location	Content	Format
Web	Text	Table

Problem: How to get the data?

Scraper

How do Web Scrapers* work?

1. Download webpage from a server
2. Process webpage to output data

Scraper

How do Browsers* work?

1. Download webpage from a server
2. Process webpage to display data

Scraper

How do Web Scrapers* work?

1. Download webpage from a server
2. Process webpage to output data

* customized browser

Scraper

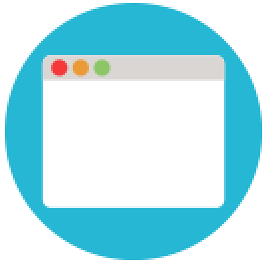
How do Web Scrapers* work?

1. Download webpage from a server – HTTP
2. Process webpage to output data – HTML + CSS + JS

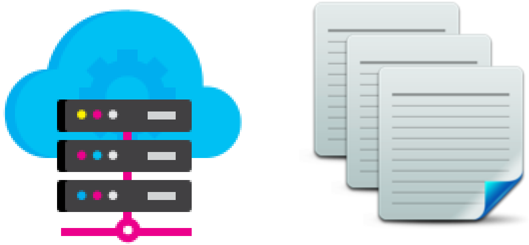
* customized browser

Downloading: How the web works

- Server
  1. Listen for an incoming request
  2. Send out a response
- Browser
  1. Send a request to a server
  2. Receive and process the server response
- HTTP = Language of Request + Response



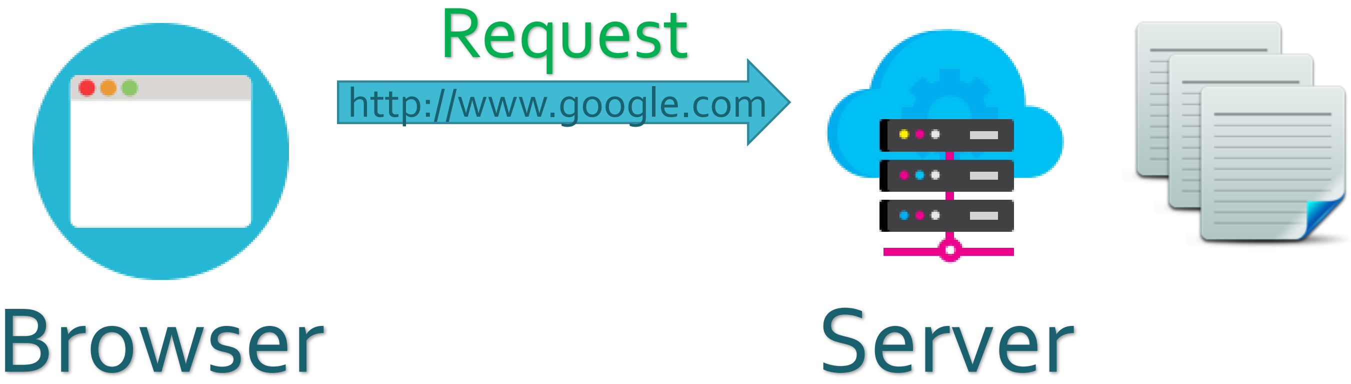
Browser



Server

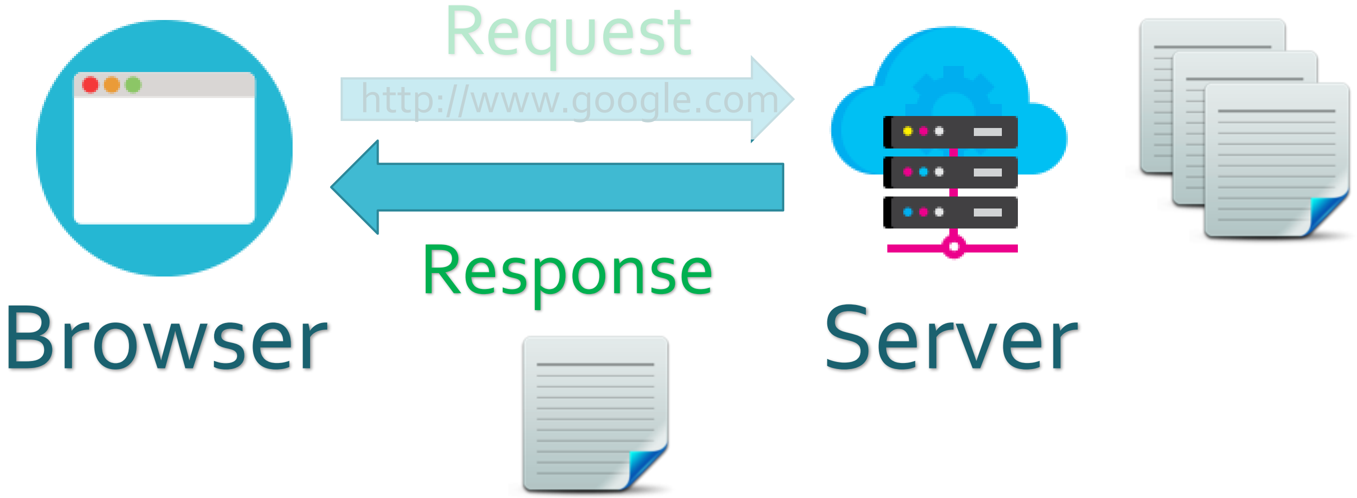
Downloading: How the web works

- Server
  1. Listen for an incoming request
  2. Send out a response
- Browser
  1. Send a request to a server
  2. Receive and process the server response
- HTTP = Language of Request + Response



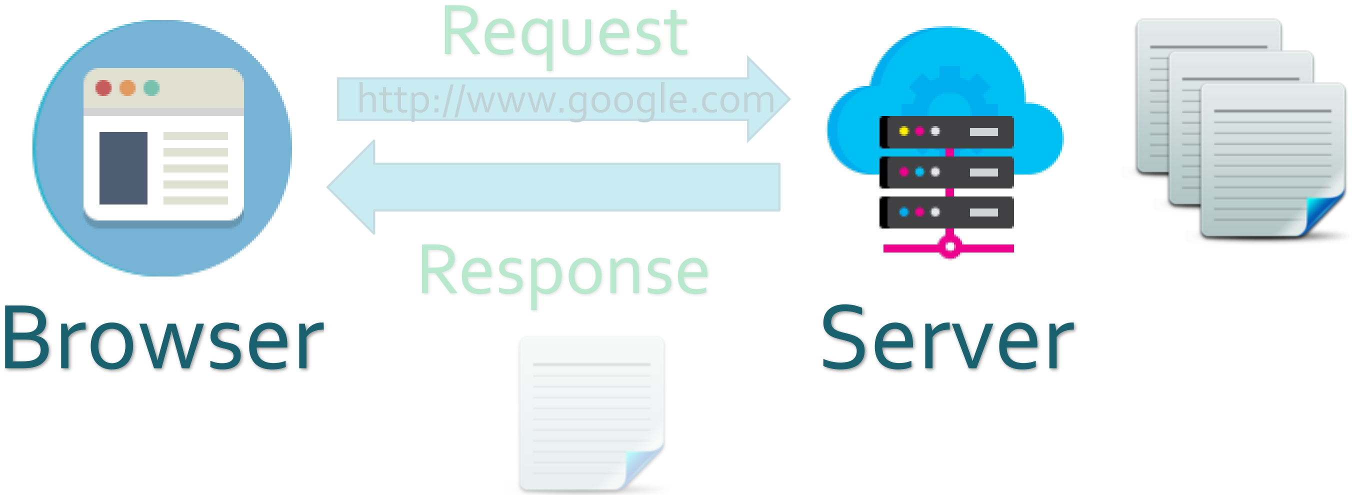
Downloading: How the web works

- Server
  1. Listen for an incoming request
  2. Send out a response
- Browser
  1. Send a request to a server
  2. Receive and process the server response
- HTTP = Language of Request + Response



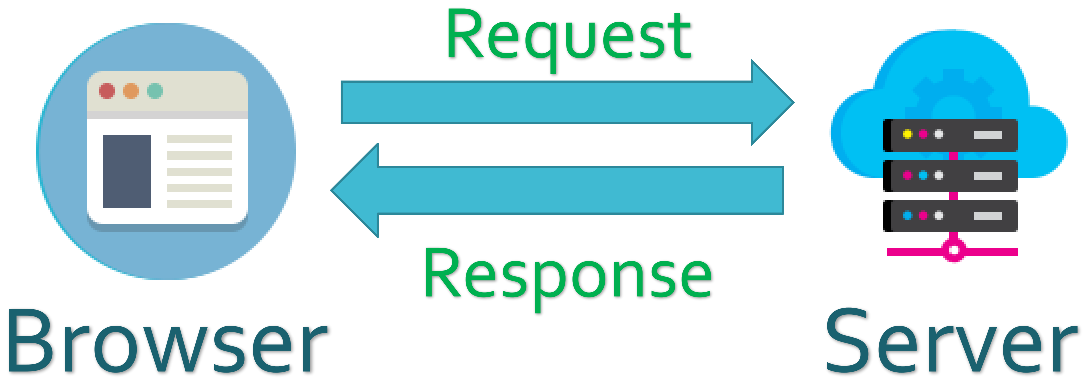
Downloading: How the web works

- Server
    1. Listen for an incoming request
    2. Send out a response
- Browser
    1. Send a request to a server
    2. Receive and process the server response
- HTTP = Language of Request + Response

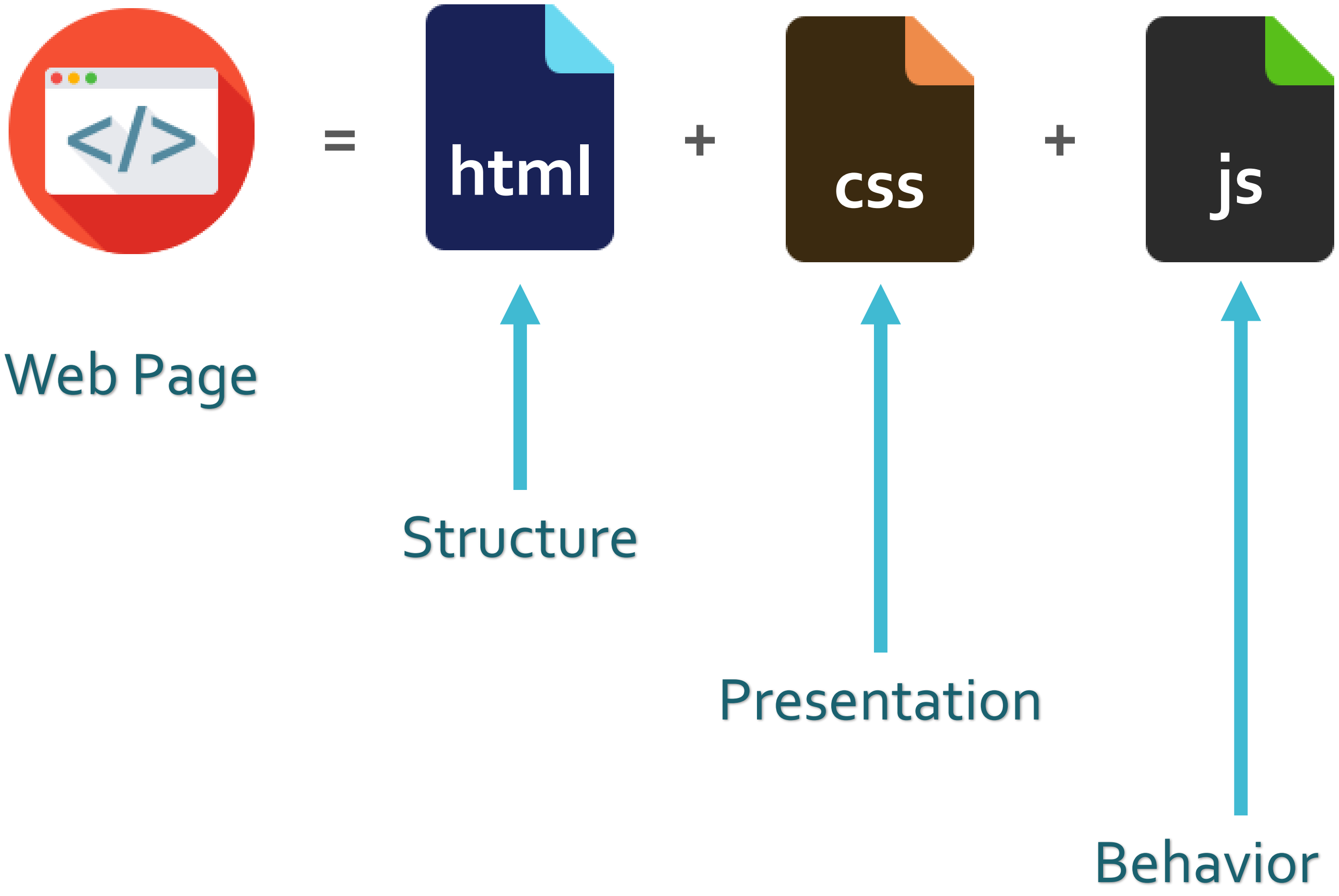


Downloading: How the web works

- Server
  1. Listen for an incoming request
  2. Send out a response
- Browser
  1. Send a request to a server
  2. Receive and process the server response
- HTTP = Language of Request + Response



Processing: How web content gets displayed



Web Page

Structure

Presentation

Behavior

How do Web Scrapers* work?

1. Download webpage from a server – HTTP
2. Process webpage to output data – HTML + CSS + JS

How do Web Scrapers* work?

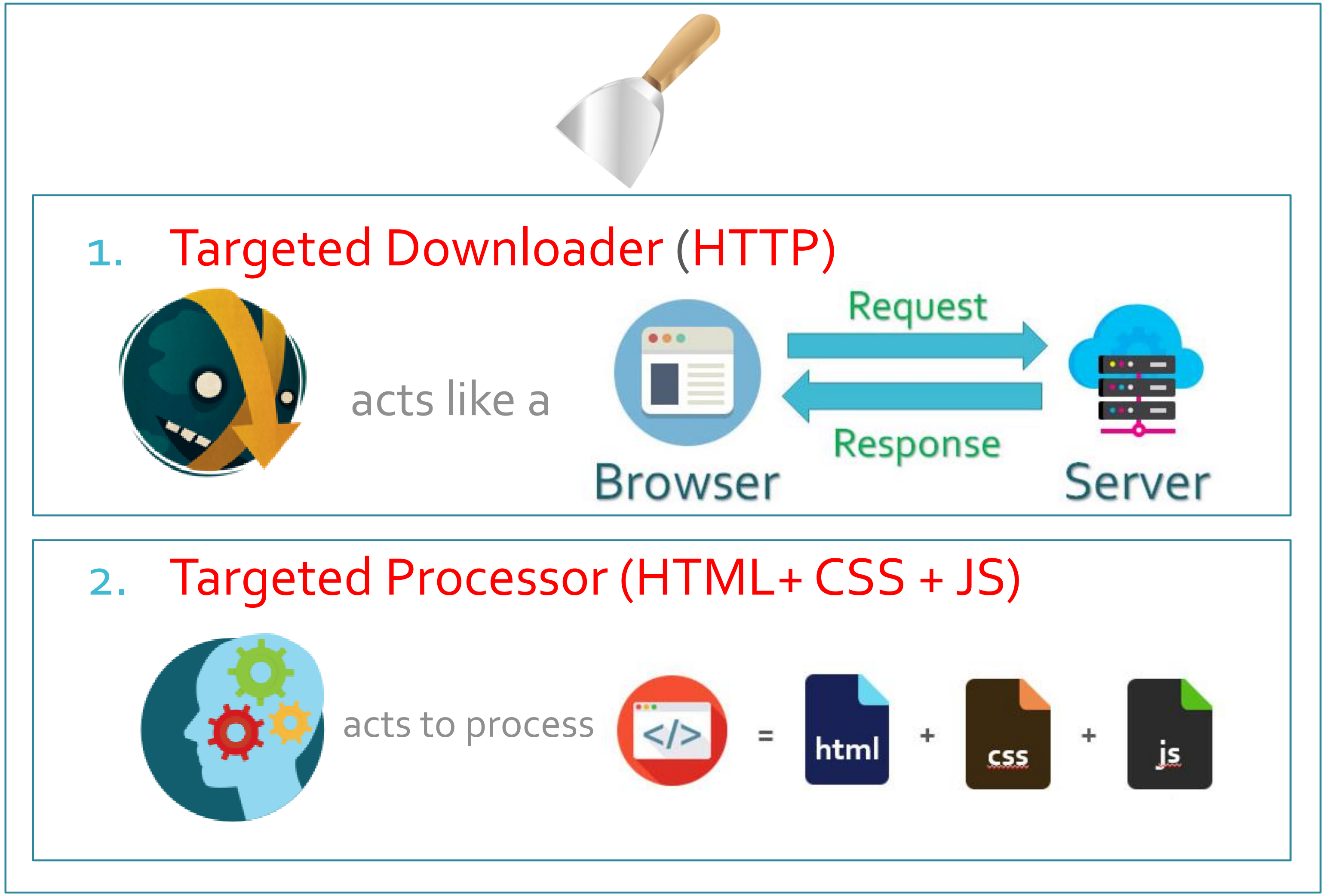
Components

1. Download webpage from a server – HTTP
2. Process webpage to output data – HTML + CSS + JS

Scraper = Downloader + Processor

How do Web Scrapers* work?

Components



1. Targeted Downloader (HTTP)

acts like a

2. Targeted Processor (HTML+ CSS + JS)

acts to process

How do I build a scraper?

1. Identify interesting question
2. Identify target website with data to answer question
3. Investigate website structure
4. Write scraper (downloader + processor)
5. Test scraper
6. Deploy scraper – get data
7. Optional: repeat / refine

How do I write a scraper?

Downloader:

1. Identify URL to webpage with data
2. Request webpage with URL
3. Receive response

Processor:

1. Read response
2. Extract relevant data from response
3. Output data (to screen, file, db etc.)

Demo

Demo Examples

1. Mississauga:
    1. Pure HTML
    2. Table format
    3. Single page
2. Burlington:
    1. Pure HTML
    2. Div format (not much different from tables)
    3. Multiple pages
3. Toronto:
    1. Shell HTML + JavaScript data injection
    2. JSON format (after some text wrangling)
    3. Multiple pages

Problem scrapers address

Scenario:

Location

- Web
- Disk
- Email

Content

- Text
- Images
- Video

Format

- Table
- Non Table
- Spreadsheet
- RSS Feed

Problem: How to get the data?

Scaper

Reconnaissance Toolkit

1. Chrome DevTools (Examine Content / HTTP)
2. Postman (HTTP Requests)

Python Implementation Toolkit

Downloading:

requests for HTTP calls

Processing:

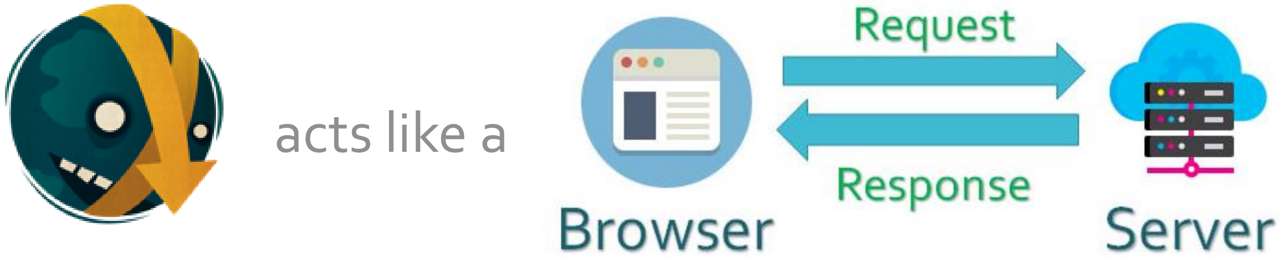
BeautifulSoup
json
(limited to Parsing in this case)

Scraper considerations

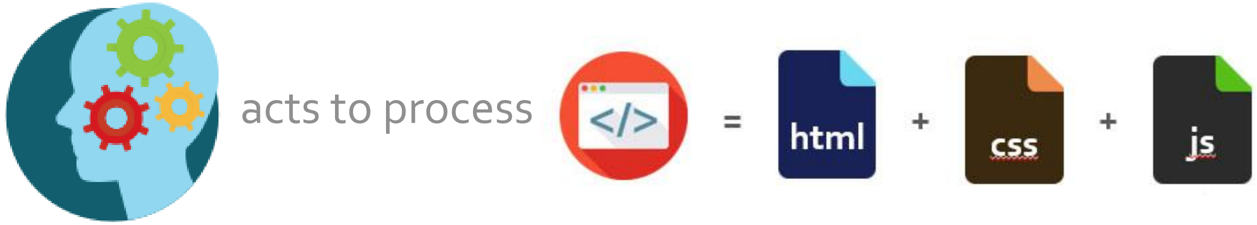
1. Timing of requests
2. Structuring your downloading / processing code
3. What content to extract
4. But different cases add levels of complexity on top of this…

How do Web Scrapers* work?

1. Targeted Downloader (HTTP)



2. Targeted Processor (HTML+ CSS + JS)



Goals

Walk away from this talk knowing:

1. What problems scrapers address
2. How they work
3. How to build one (general steps + Py-example)

Q&A