Web Scraping

Columbia Data Science Society

Carlos Martin and Kevin Lin

Technique of extracting information from websites

Unstructured data → structured data

Why web scraping?

Not every site has an API

Website > API

No rate limits

Today

BeautifulSoup

Requests

Python library

HTTP for humans

Designed as improvement to standard **urllib2**

```
pip install requests
```

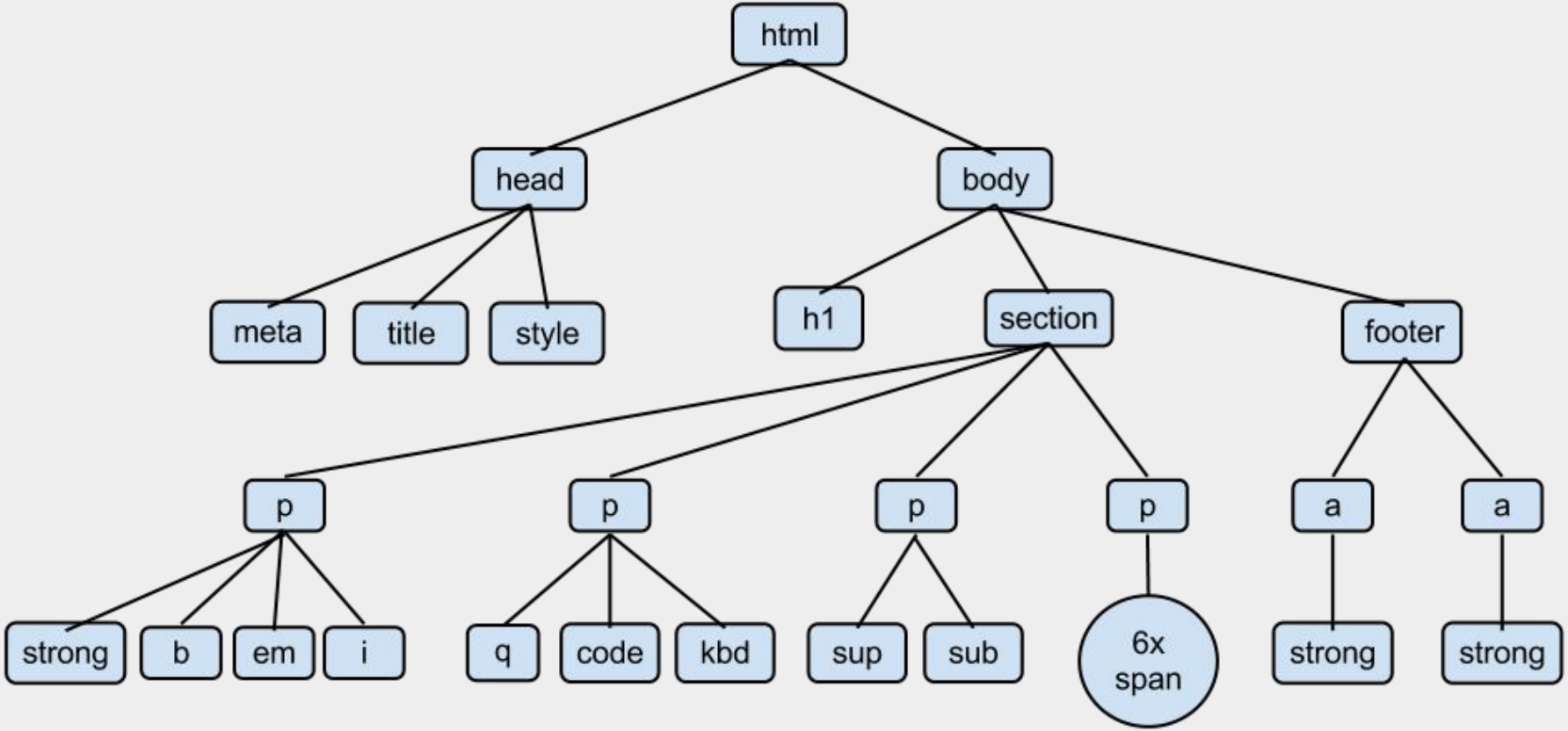
BeautifulSoup

Pulling data out of HTML and XML

Navigate, search, modify the parse tree

```
pip install beautifulsoup4
```

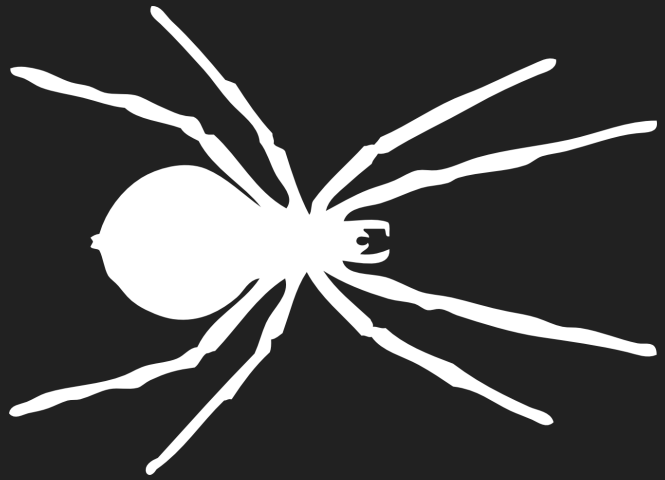
Document Object Model (DOM)

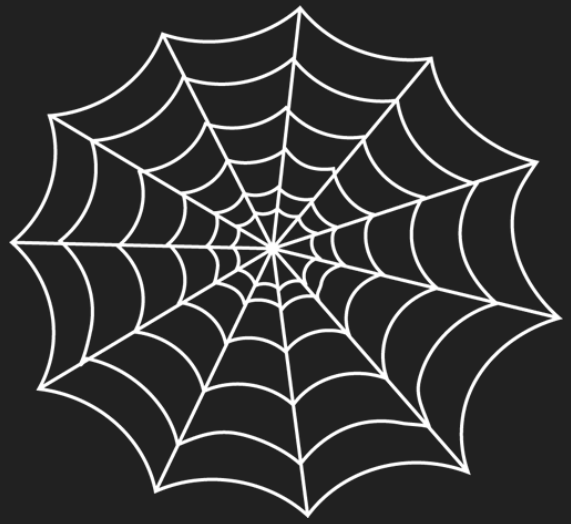


Web crawlers

Web crawlers

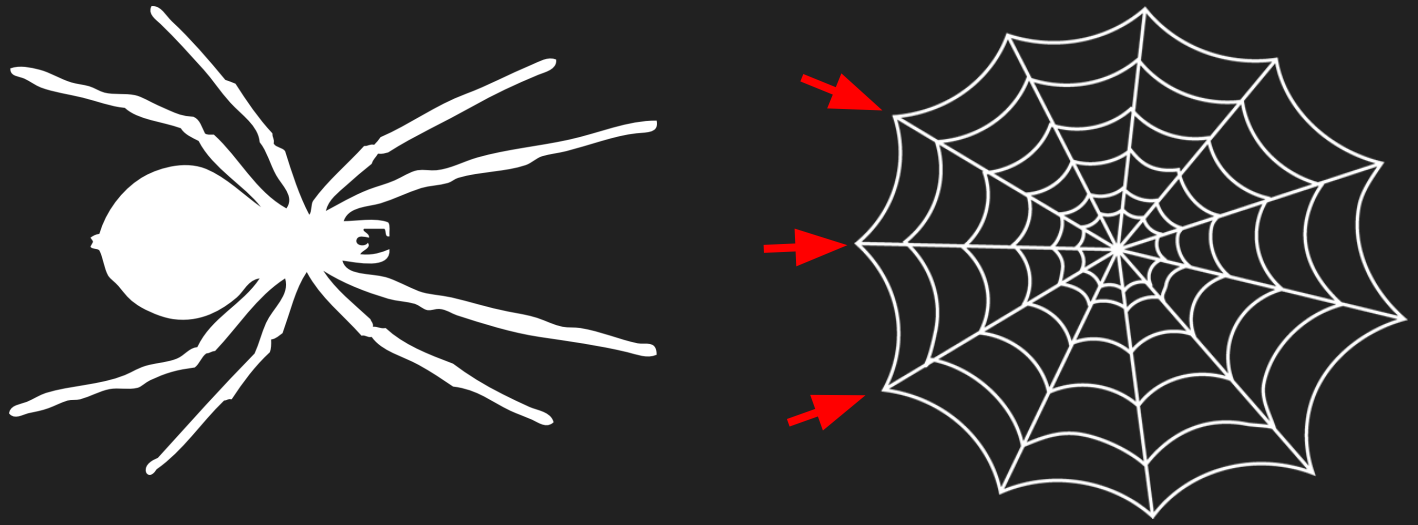
a program that systematically browses the web





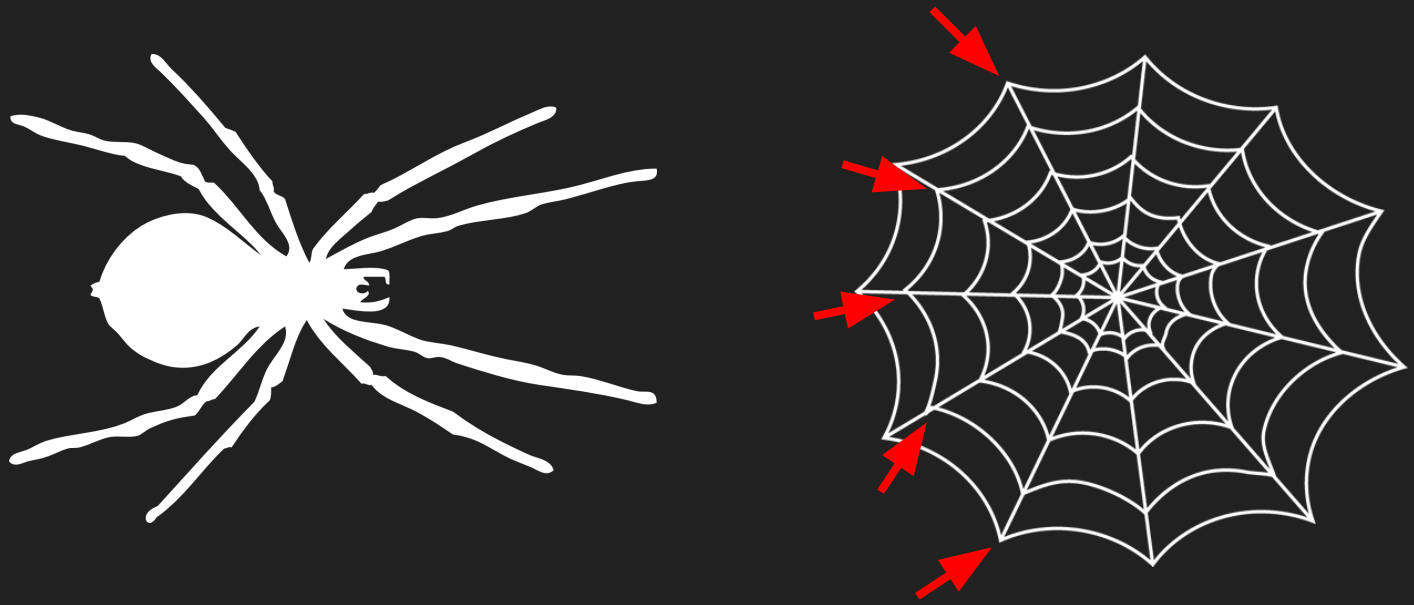
Web crawlers

starts with list of URLs to visit (seeds)



Web crawlers

identifies links, adds them to list of URLs to visit



Examples

DuckDuckGo

Uses

Scraping

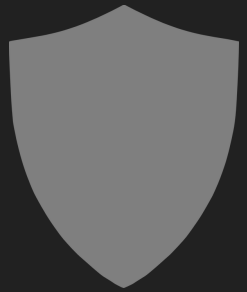
Archiving

Indexing

Validation

Spambots

Crawling policy



Selection policy

Revisit policy

Parallelization policy

Politeness policy

Robots.txt

Robot exclusion standard

Communicates with web crawlers and web robots

Lists areas that should not be processed or scanned

Can include crawl delay (i.e. requests per second) for throttling

See **robotstxt.org/robotstxt.html**

Robots.txt

User-agent: *
Disallow: /cgi-bin/
Disallow: /acis/whatsnew.html
Disallow: /httpd/reports/
Disallow: /itc/ccnmtl/assets/

Sitemap

Robot inclusion standard

XML file that gives URLs available for crawling

Can include last update date, update frequency, page importance, etc.

Allows web crawlers to crawl the site more intelligently

See **sitemaps.org/protocol.html**

Scrapy

web scraping framework for Python

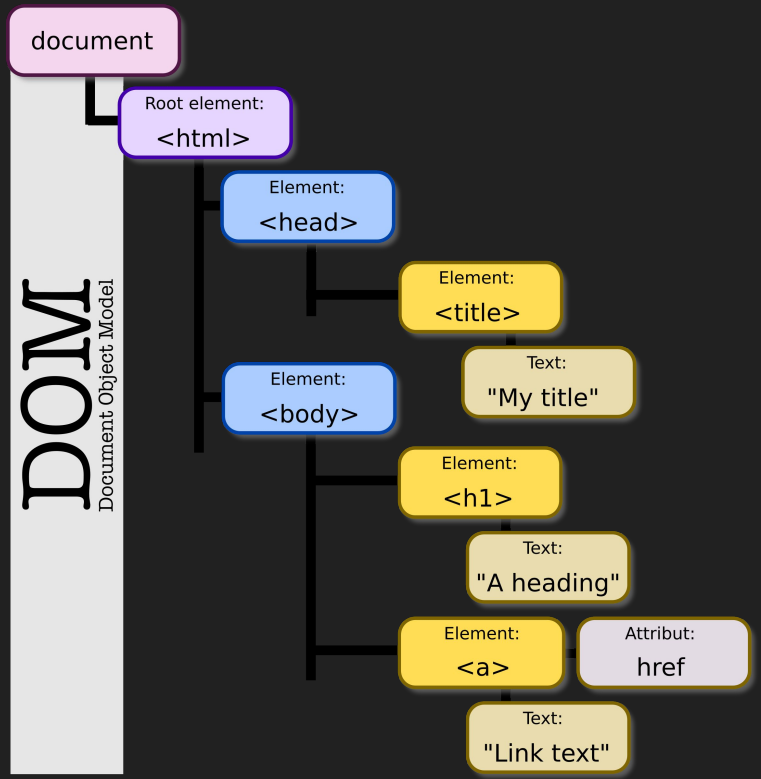
```
pip install scrapy
```

Creating a Scrapy project

```
scrapy startproject myproject

myproject
├── scrapy.cfg
└── myproject
    ├── __init__.py
    ├── items.py
    ├── pipelines.py
    ├── settings.py
    └── spiders
        └── __init__.py
```

Structure of a document



XPath

Language for **selecting nodes** in XML documents

```
/html/head/title

/html/head/title/text()

//td

//div[@class="myclass"]
```

XPath

Language for **selecting nodes** in XML documents

```
/wikimedia/projects/project/editions/*[2]
```

```
<?xml version="1.0" encoding="utf-8"?>
<wikimedia>
 <projects>
  <project name="Wikipedia" launch="2001-01-05">
   <editions>
    <edition language="English">en.wikipedia.org</edition>
    <edition language="German">de.wikipedia.org</edition>
    <edition language="French">fr.wikipedia.org</edition>
    <edition language="Polish">pl.wikipedia.org</edition>
   </editions>
  </project>
  <project name="Wiktionary" launch="2002-12-12">
   <editions>
    <edition language="English">en.wiktionary.org</edition>
    <edition language="French">fr.wiktionary.org</edition>
    <edition language="Vietnamese">vi.wiktionary.org</edition>
    <edition language="Trukish">tr.wiktionary.org</edition>
   </editions>
  </project>
 </projects>
</wikimedia>
```

CSS Selectors

Selects nodes based on their CSS style

Each stylesheet rule has a **selector pattern** that matches a set of HTML elements

```
p{
 color:#0000FF;
}
```

```
<h1 class="class1">
  Heading
</h1>
<p id="id1">
  Paragraph
</p>
<p class="class1">
  Paragraph
</p>
```

```
.class1{
 color:#0000FF;
}
```

```
<h1 class="class1">
  Heading
</h1>
<p id="id1">
  Paragraph
</p>
<p class="class1">
  Paragraph
</p>
```

```
#id1{
 color:#0000FF;
}
```

```
<h1 class="class1">
  Heading
</h1>
<p id="id1">
  Paragraph
</p>
<p class="class1">
  Paragraph
</p>
```

```
p.class1{
 color:#0000FF;
}
```

```
<h1 class="class1">
  Heading
</h1>
<p id="id1">
  Paragraph
</p>
<p class="class1">
  Paragraph
</p>
```

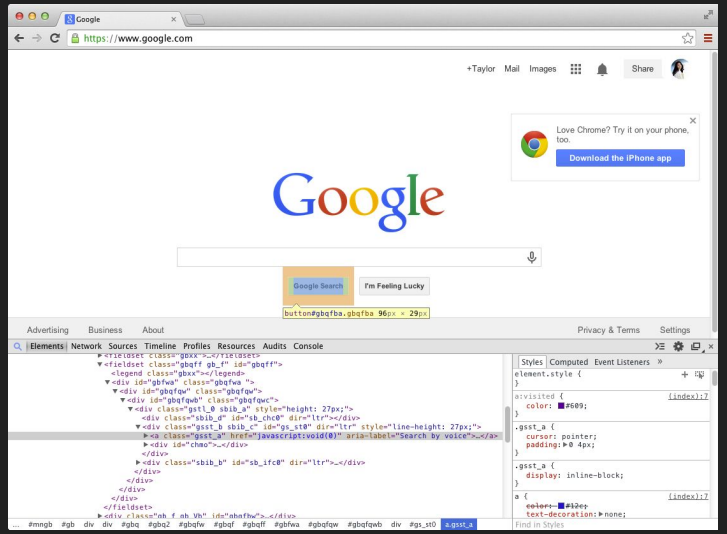
Obtaining the XPath

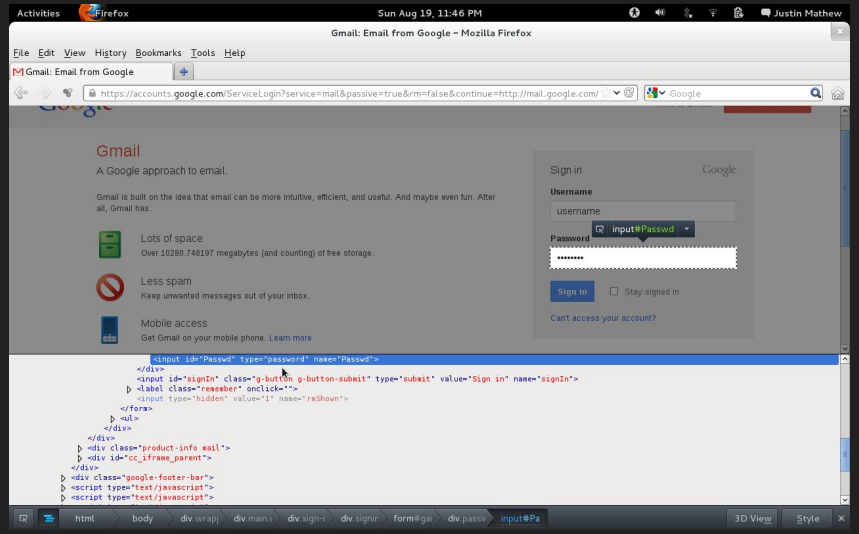
Chrome Developer Tools

Inspect
Copy > Copy XPath

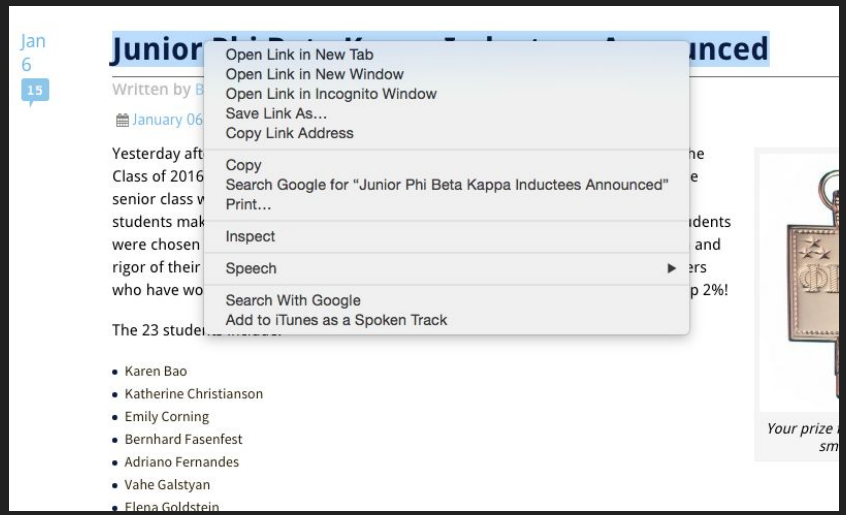
Firebug HTML Panel

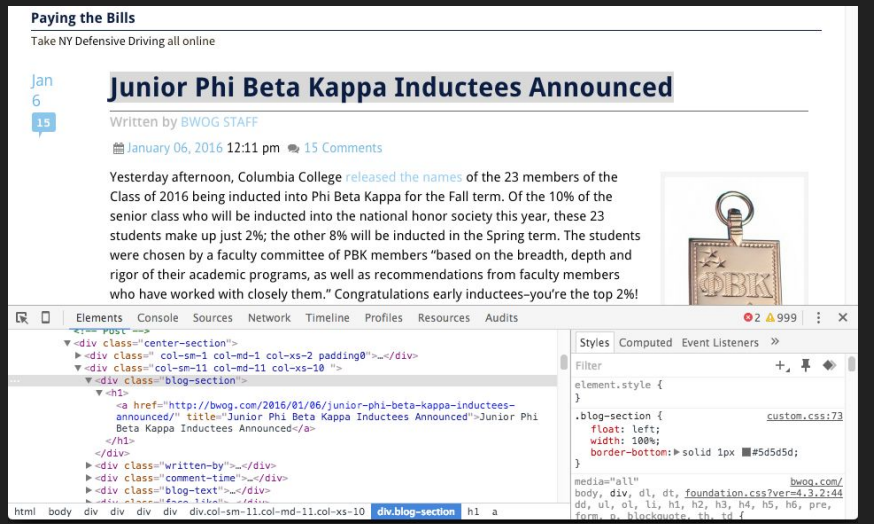
Inspect Element with Firebug
Copy XPath





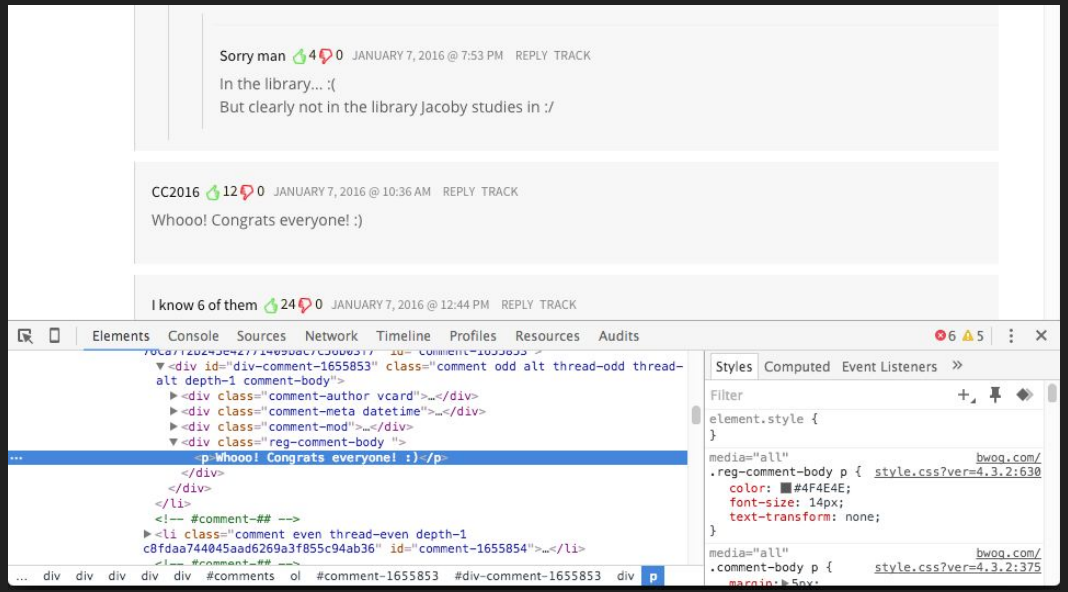
Using XPath selectors





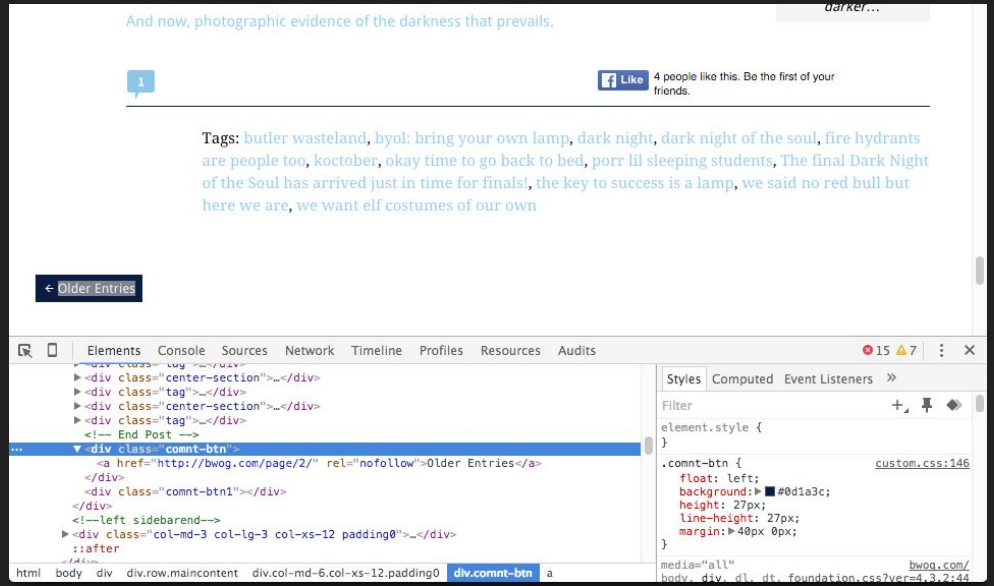

```
//div[@class="blog-section"]
```

Extracting comments




```
//div[@class="reg-comment-body "]
```

Following links




```
//div[@class="comnt-btn"]//@href
```

Wait for it...

```
scrapy crawl bwog -o comments.json
```

...and on it goes!

Next steps

Scrapy Shell (interactive shell console)

Feed Exports (JSON, CSV, XML)

Item Pipelines (writing to MongoDB, duplicates filter)

doc.scrapy.org

for more information

Code from this tutorial

github.com/carlosgmartin/web-scraping