Lecture 22: Oct 29, 2018

Web Scraping

- *HTML*
- *Scraping Information*
- *Advanced Scraping*
- *Resources*

James Balamuta
STAT 385 @ UIUC

Announcements

- **hw07** is due **Friday, Nov 2nd, 2018** at **6:00 PM**
- **Group project** member choice phase **has begun!**
  - https://github.com/stat385-fa2018/disc/issues/75
- **EC Opportunity**: Big Data Summit, Nov 8th
  - https://github.com/stat385-fa2018/disc/issues/71
- **Quiz 10** covers Week 9 contents @ CBTF.
  - Window: Oct 30th - Nov 1st
  - Sign up: https://cbtf.engr.illinois.edu/sched
- Want to review your homework or quiz grades?
  Schedule an appointment.

Last Time

- **HTTP(S)**
  - Communicating over the internet.
- **Web + APIs**
  - APIs provide structured interaction with software.
  - Web APIs are more stable than web pages
- **httr with REST**
  - Connect with a REST API using HTTP Verbs
  - Formatting of HTTP Response is: Status, Header, Body
- **JSON**
  - Semi-structured output based on key-value form
  - Common Web API response that is converted to an *R* list

Lecture Objectives

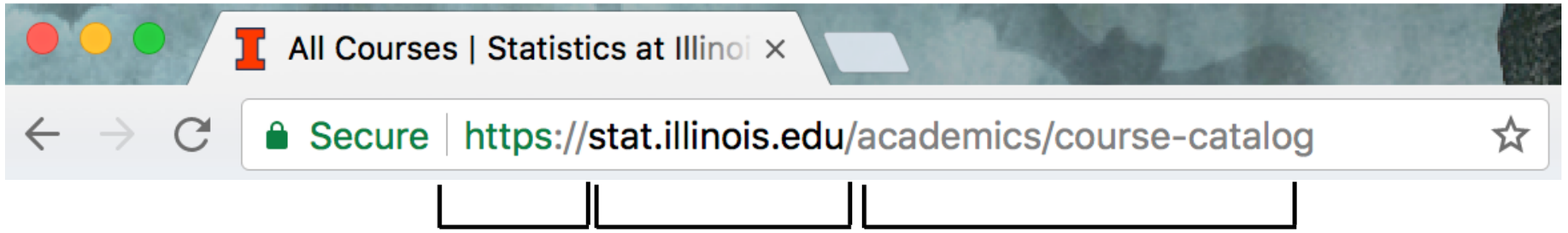
- Understand the **semi-structured form of HTML**
- **Extract** data from a Web Page
- **Describe** the difference between a static and dynamic web page.

HTML

Previously

HTTP(S) Request

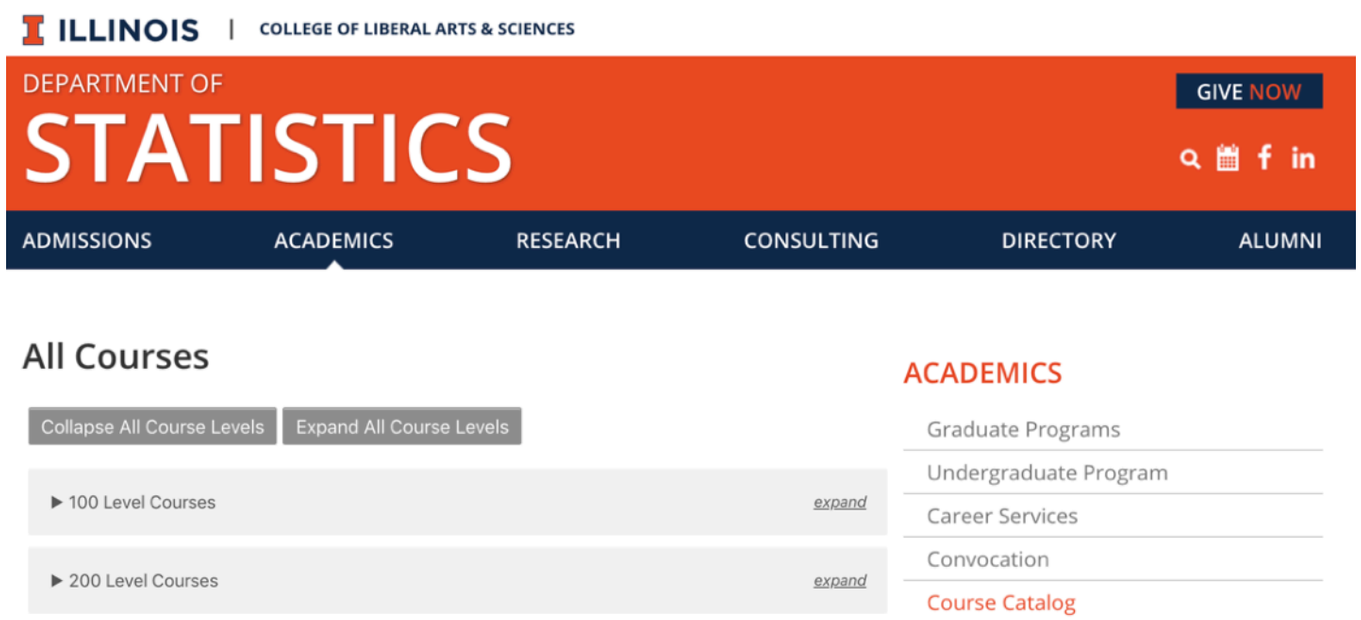
Hyper**t**ext **T**ransfer **P**rotocol (**S**ecure)



Protocol/Scheme
Communication medium for request

Domain
Web Address

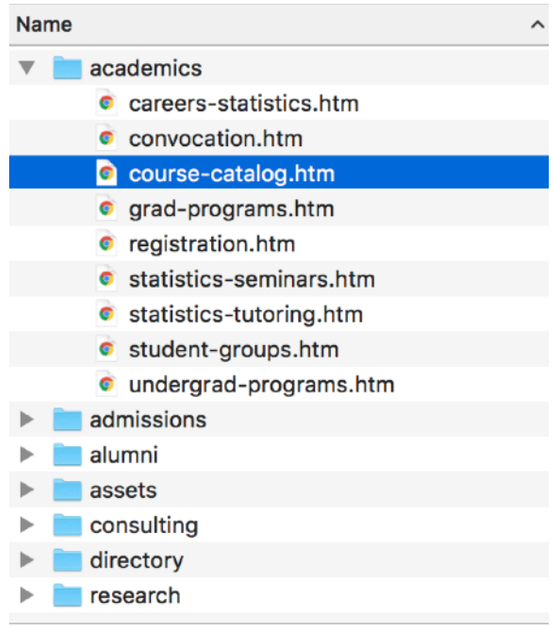
Path
Location of Resource



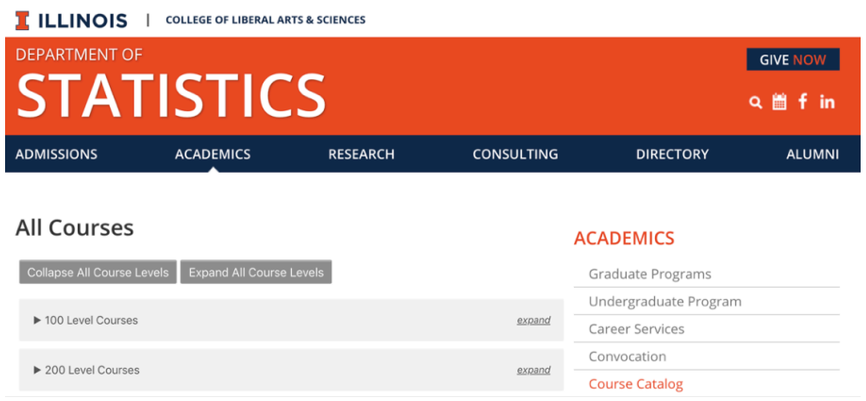
Previously

Hello stat.illinois.edu, I'd like to see the **course catalog**

Source



Hello person's web browser, let me pull that up...



Hello stat.illinois.edu, thank you!

Hello person's web browser, I found it. Here you go!

Previously

Structures of Data

… how data is shaped …

Structured*

Rectangular

~5 - 10%

Semistructured**

key: value

~5 - 10%

```
---
title: "Untitled"
author: "JJB"
date: "1/27/2018"
output: html_document
---
```

Unstructured***

?????????

~80 - 90%

Pinky said,
"Gee, Brain. What are we going to do tonight?"
The Brain replied, "The same thing we do every night, Pinky. Try to take over the world."

---

* Typical form for scientific experiments and company databases
** *RMarkdown* Document Properties (YAML), JavaScript Object Notation (JSON), XML
*** Pure text documents, images, social media posts, and so on. No visible relationship.

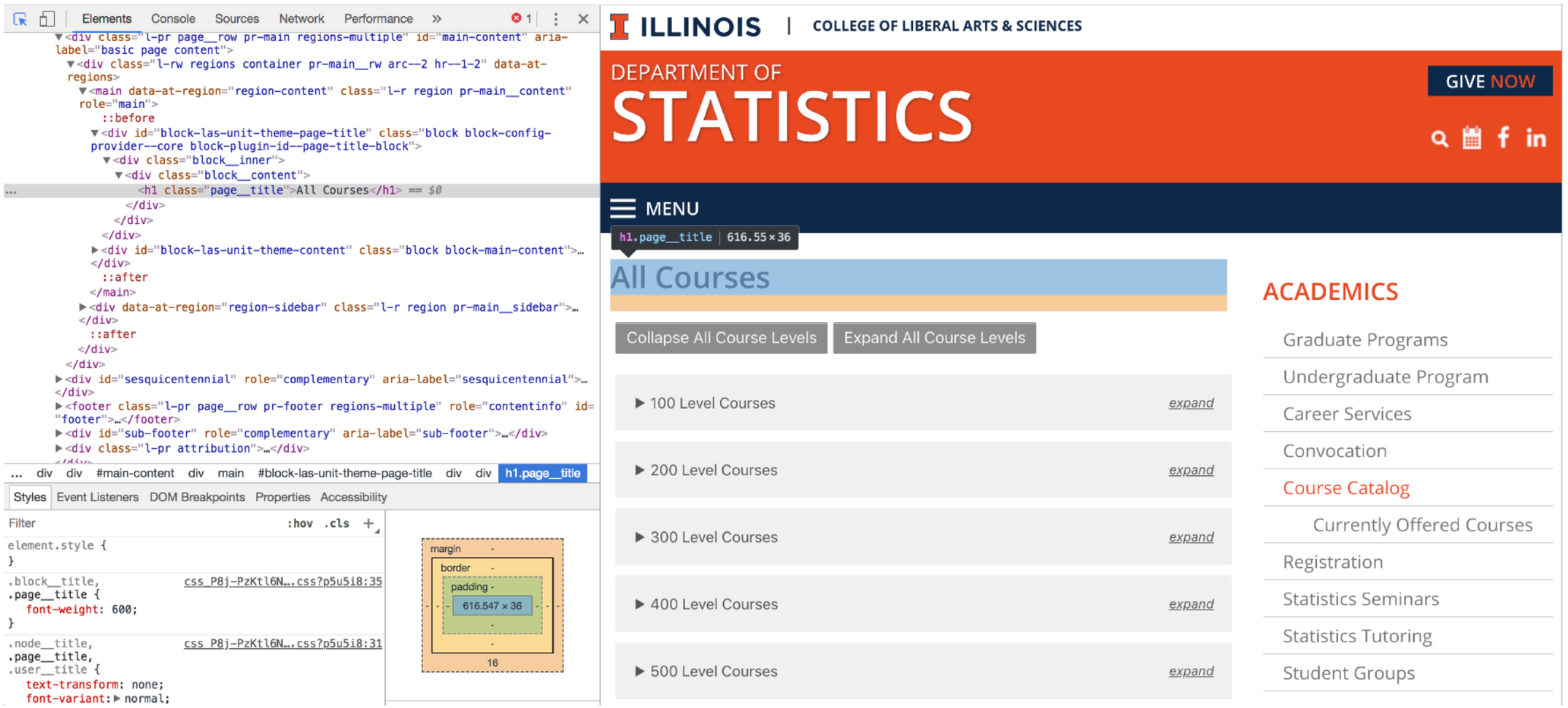
HTML

Hyper **T**ext **M**arkup **L**anguage

the language of the web

HTML Source

Browser View



https://stat.illinois.edu/academics/course-catalog

HTML Elements

… breakdown of keys and value location …

Generic Element

Element Name
Individual HTML element found in the web page

Attribute Name
Mapping to set specific behaviors

Attribute Value
Value assigned to the attribute

Content
Displayed on the website

```
<element attribute = "property"> content </element>
```

Example Element for URIs

Element Name
Individual HTML element found in the web page

Attribute Name
Mapping to set specific behaviors

Attribute Value
Value assigned to the attribute

Content
Displayed on the website

```
<a href = "http://google.com"> Link to Google </a>
```

Semi-structure of HTML

… **key**:**value** pairing …

HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Title of Page</title>
</head>
<body>
<h1 align = "center"> First order heading (large) </h1>
<p> Paragraph for text with a
  <a href = "http://www.stat.illinois.edu">link!</a>
  <h2> Top Beverages </h2>
  <ol>
    <li> Tea </li>
    <li> Coffee </li>
    <li> Milkshakes </li>
  </ol>
</p>
<table border = “1”>
  <tr>
    <th> Name </th>
    <th> Salary </th>
  </tr>
  <tr>
    <td> Joshua Tree </td>
    <td> 66,666 </td>
  </tr>
  <tr>
    <td> Aaron Thomas </td>
    <td> 78,921.40 </td>
  </tr>
</table>
<!-- Comment -->
</body>
</html>
```

Key

Value

Rendered in Web Browser

First order heading (large)

Paragraph for text with a link!

Top Beverages

1. Tea
2. Coffee
3. Milkshakes

Name	Salary
Joshua Tree	66,666
Aaron Thomas	78,921.40

HTML **DOM**

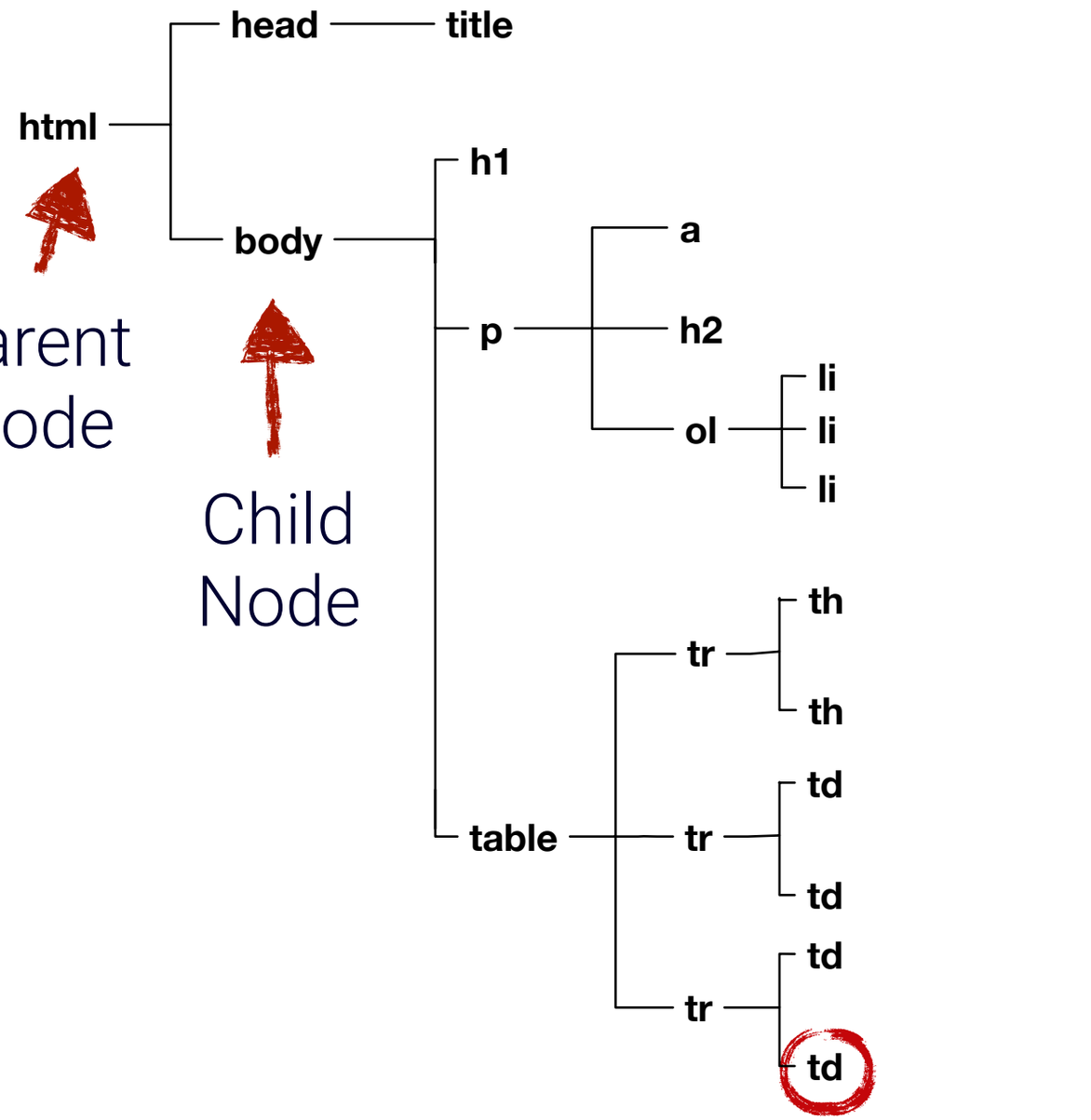
… **D**ocument **O**bject **M**odel Trees …

```
<!DOCTYPE html>
<html>
<head>
<title>Title of Page</title>
</head>
<body>
<h1 align = "center"> First order heading (large) </h1>
<p> Paragraph for text with a
  <a href = "http://www.stat.illinois.edu">link!</a>
  <h2> Top Beverages </h2>
  <ol>
    <li> Tea </li>
    <li> Coffee </li>
    <li> Milkshakes </li>
  </ol>
</p>
<table border = “1”>
  <tr>
    <th> Name </th>
    <th> Salary </th>
  </tr>
  <tr>
    <td> Joshua Tree </td>
    <td> 66,666 </td>
  </tr>
  <tr>
    <td> Aaron Thomas </td>
    <td> 78,921.40 </td>
  </tr>
</table>
<!-- Comment -->
</body>
</html>
```

Key

Value

DOM Tree of Web Page



Parent Node

Child Node

Common HTML Tags

… breakdown of keys and value location …

HTML

Rendered

 R.A. Fisher

Bold *Italics*

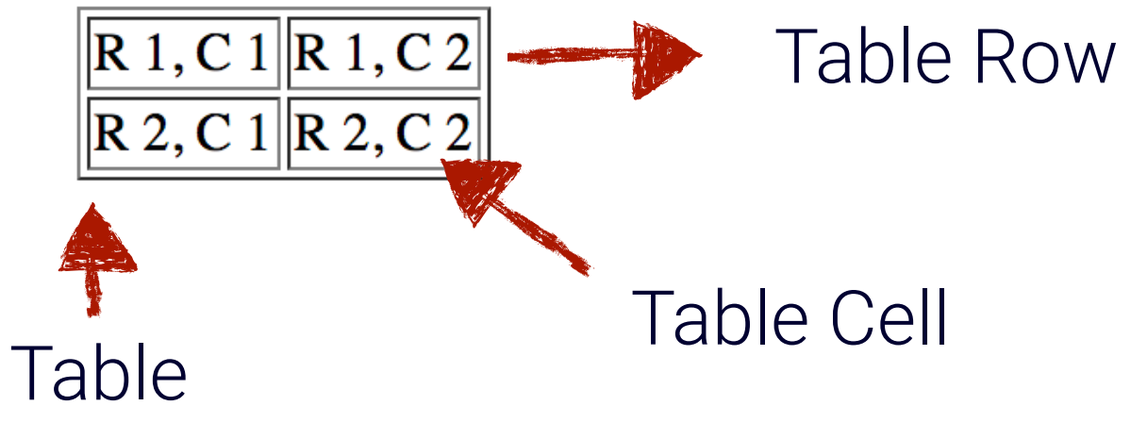
R.A. Fisher

Table

Table Row

Table Cells

```
<table border = "1">
    <tr>
        <td> R1, C1 </td>
        <td> R1, C2 </td>
    </tr>
    <tr>
        <td> R2, C1 </td>
        <td> R2, C2 </td>
    </tr>
</table>
```



Your Turn

1. Identify all elements in the web page source.
2. What attributes are present?
3. What are the attribute properties?
4. What information can be extracted?

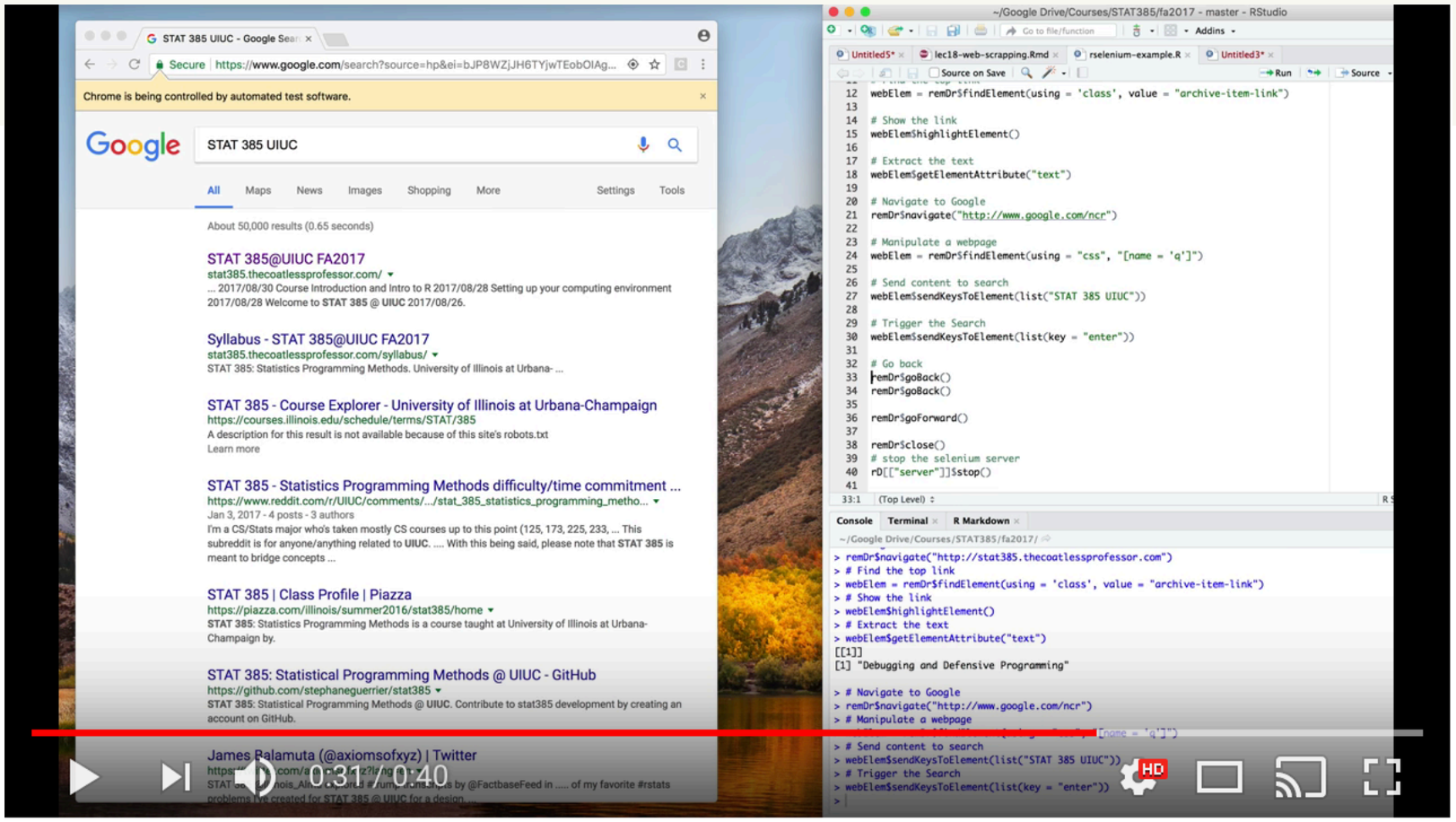
```
<!DOCTYPE html>
<html>
<head>
<title>House of Statistics</title>
</head>
<body bgcolor = “blue”>
<h3 align = “left”> My Personal Website </h3>
<hr />
 See my work on <a href = "http://www.github.com">GitHub!</a>
<!-- We should probably put a side bar in here... -->
<p> Check out the hypnotoad! </p>
<img src = “https://i.imgur.com/oV8yyJa.gif” />
<h2 align = “right”> Important Dates </h2>
<p>
   <i> December 27th </i> - An Unlucky Person’s Birthday
</p>
<h4> Grocery List </h4>
<ul>
  <li> Yogurt </li>
  <li> Greengage Plums </li>
  <li> Saltine Crackers </li>
</ul>
</body>
</html>
```

Scraping Information

Previously

Definition:

Web scraping (*web harvesting* or *web data extraction*) is a computer software technique of extracting information from websites. Source



https://www.youtube.com/watch?v=Vt6f8A35-1w

Breakdown of Web Scraping

… flow of the web scraping process …

1. **Obtain the identifier for the data to be extracted …**
   … either the html element information or css selector …

2. **Download and read into R the webpage …**
   … use rvest's **read_html()** to read the web page into *R* …

3. **Extract the nodes …**
   … picks out the elements on the page with the identifier …

4. **Retrieve the content from nodes …**
   … obtain text, table, or attribute information via ***html_*()*** …

… extracting content from *static* web pages …

```
install.packages("rvest")
library("rvest")
```

Function	Description
read_html(x)	Read in HTML Files
html_nodes(x, css, xpath)	Extract Nodes Matching Tag, e.g. **<tag></tag>**
html_node(x, css, xpath)	Extract only one node matching the tag
html_table(x, header, trim, fill, dec)	Convert HTML tables to data.frame, e.g. **<table></table>**
html_text(x, trim)	Extract content between tags
html_name(x)	Extract the name of HTML tag, e.g. **<tag></tag>** gives **"tag"**
html_attrs(x)	Extract all attributes on a tag, e.g. attribute = "value"
html_attr(x, name, default)	Extract values associate with specific attribute.

Web Page as a String

… direct embedding …

First order heading (large)

Paragraph for text with a link!

Top Beverages

1. Tea
2. Coffee
3. Milkshakes

Name	Salary
Joshua Tree	66,666
Aaron Thomas	78,921.40

```
#install.packages("rvest")
library("rvest")

sample_webpage = '<!DOCTYPE html>
<html>
<head>
<title>Title of Page</title>
</head>
<body>
<h1 align = "center"> First order heading (large) </h1>
<p> Paragraph for text with a
  <a href = "http://www.stat.illinois.edu">link!</a>
  <h2> Top Beverages </h2>
  <ol>
    <li> Tea </li>
    <li> Coffee </li>
    <li> Milkshakes </li>
  </ol>
</p>
<table border =  “1”>
 <tr>
   <th> Name </th>
   <th> Salary </th>
 </tr>
 <tr>
   <td> Joshua Tree </td>
   <td> 66,666 </td>
 </tr>
 <tr>
   <td> Aaron Thomas </td>
   <td> 78,921.40 </td>
 </tr>
</table>
<!-- Comment -->
</body>
</html>'
```

Read HTML

... 3 ways to read into *R* a website ...

```
# Option 1: Read Web Page as a String from R
my_webpage = read_html(sample_webpage)

# Option 2: Read an active Web Page via its URI
my_webpage = read_html("http://domain.com/path/to/sample_webpage.html")

# Option 3: Read a Web Page Saved on the Local File System
my_webpage = read_html("~/Documents/sample_webpage.html")

my_webpage
# {xml_document}
# <html>
# [1] <head>\n<meta http-equiv="Content-Type" content="text/#html; charset=UTF-8">\n<title>Title  ...
# [2] <body>\n<h1 align="center"> First order heading (large) </h1>\n<p> Paragraph for text with ...
```

Previously

Definition:

Piping is the act of taking one value and immediately placing it into another function to form a flow of results.

Left Function
Transmitting function result
`rnorm(10)`

Pipe Operator
Facilitate moving left result to the function on right

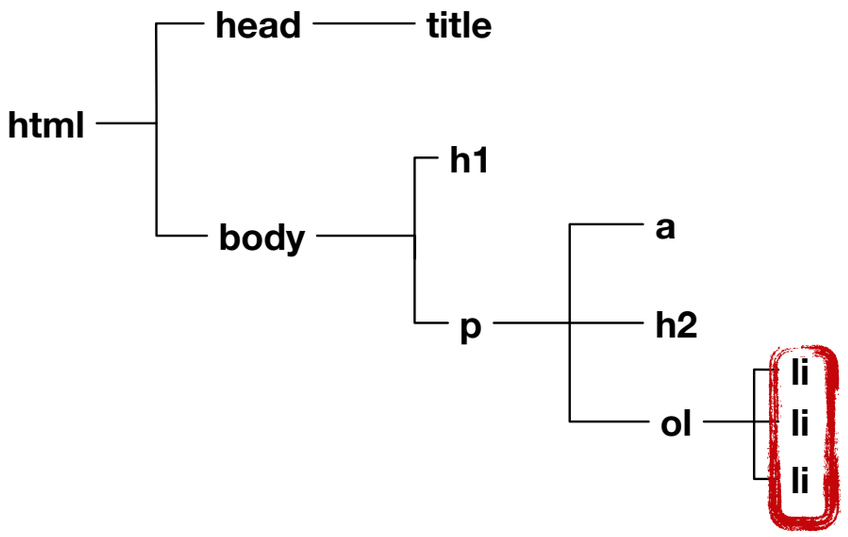
Right Function
Receiving function result in first parameter
`abs(rnorm(10))`

```
rnorm(10) %>% abs()
```

%>% is read as "and, then"

Extract Nodes

… selecting HTML elements / keys …



HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Title of Page</title>
</head>
<body>
<h1 align = "center"> First order heading (large) </h1>
<p> Paragraph for text with a
  <a href = "http://www.stat.illinois.edu">link!</a>
  <h2> Top Beverages </h2>
  <ol>
    <li> Tea </li>
    <li> Coffee </li>
    <li> Milkshakes </li>
  </ol>
</p>
<!-- Comment -->
</body>
</html>
```

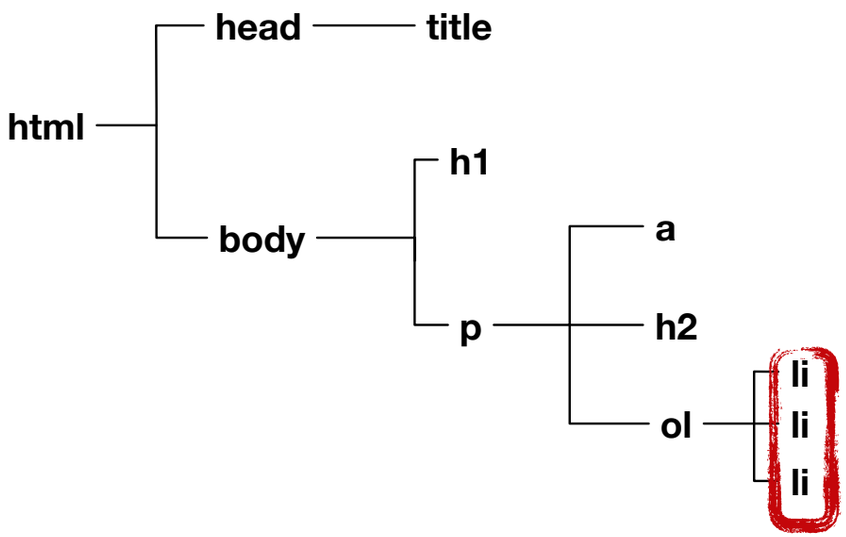
```
# Retrieve only the first
# instances of the li element
# on the page.
my_webpage %>%
   html_node("li")
# {xml_nodeset (1)}
# [1] <li> Tea </li>


# Retrieve all instances
# of the li element on page.
my_webpage %>%
   html_nodes("li")
# {xml_nodeset (3)}
# [1] <li> Tea </li>
# [2] <li> Coffee </li>
# [3] <li> Milkshakes </li>


# Which should you prefer?
```

Retrieving Content

… obtaining values as text …



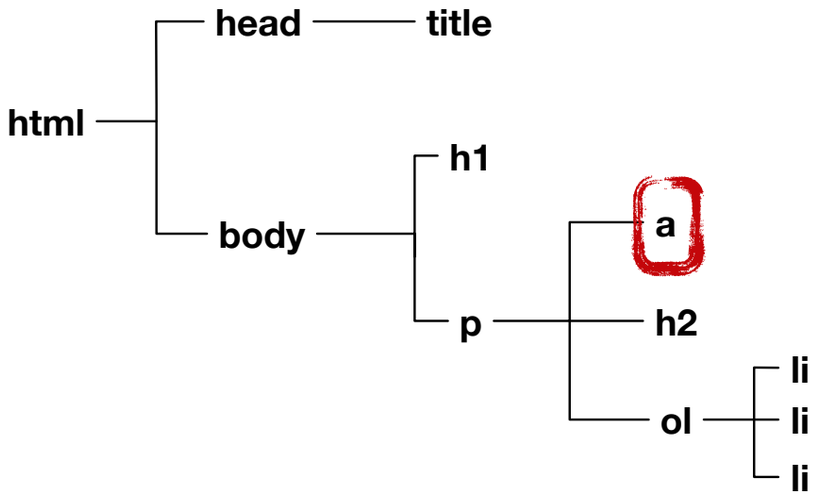
HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Title of Page</title>
</head>
<body>
<h1 align = "center"> First order heading (large) </h1>
<p> Paragraph for text with a
  <a href = "http://www.stat.illinois.edu">link!</a>
  <h2> Top Beverages </h2>
  <ol>
    <li> Tea </li>
    <li> Coffee </li>
    <li> Milkshakes </li>
  </ol>
</p>
<!-- Comment -->
</body>
</html>
```

```
# Retrieve the text content
# in between all instances
# of the li element.
my_webpage %>%
  html_nodes("li") %>%
  html_text()
# [1] "Tea"
# [2] "Coffee"
# [3] "Milkshakes"
```

Retrieving Attribute Content

… obtaining the text values …



HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Title of Page</title>
</head>
<body>
<h1 align = "center"> First order heading (large) </h1>
<p> Paragraph for text with a
  <a href = "http://www.stat.illinois.edu">link!</a>
  <h2> Top Beverages </h2>
  <ol>
    <li> Tea </li>
    <li> Coffee </li>
    <li> Milkshakes </li>
  </ol>
</p>
<!-- Comment -->
</body>
</html>
```

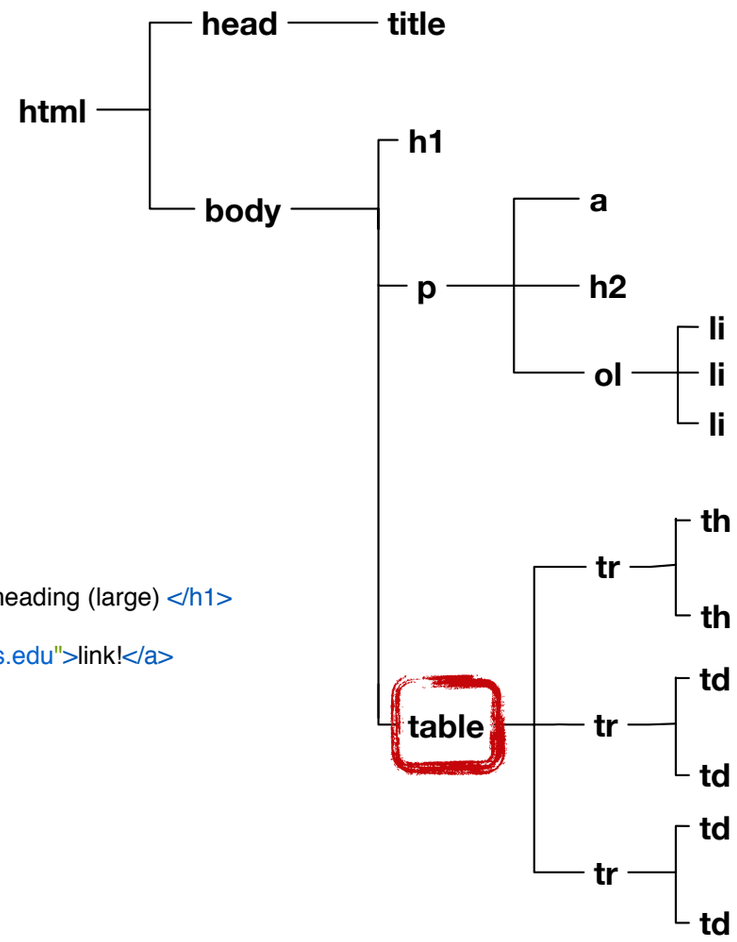
```
# Extract all a elements and, then…

# Retrieve only the attribute
# value for href.
my_webpage %>%
   html_nodes("a") %>%
   html_attr("href")
# [1]
# "http://www.stat.illinois.edu"
```

```
# Retrieve all the attributes
# on all a elements.
my_webpage %>%
   html_nodes("a") %>%
   html_attrs()
# [[1]]
#                        href
# "http://www.stat.illinois.edu"
```

Retrieving Table Content

… obtaining the cell information …




```
<!DOCTYPE html>
<html>
<head>
<title>Title of Page</title>
</head>
<body>
<h1 align = "center"> First order heading (large) </h1>
<p> Paragraph for text with a
  <a href = "http://www.stat.illinois.edu">link!</a>
  <h2> Top Beverages </h2>
  <ol>
    <li> Tea </li>
    <li> Coffee </li>
    <li> Milkshakes </li>
  </ol>
</p>
<table border = “1”>
  <tr>
    <th> Name </th>
    <th> Salary </th>
  </tr>
  <tr>
    <td> Joshua Tree </td>
    <td> 66,666 </td>
  </tr>
  <tr>
    <td> Aaron Thomas </td>
    <td> 78,921.40 </td>
  </tr>
</table>
<!-- Comment -->
</body>
</html>
```

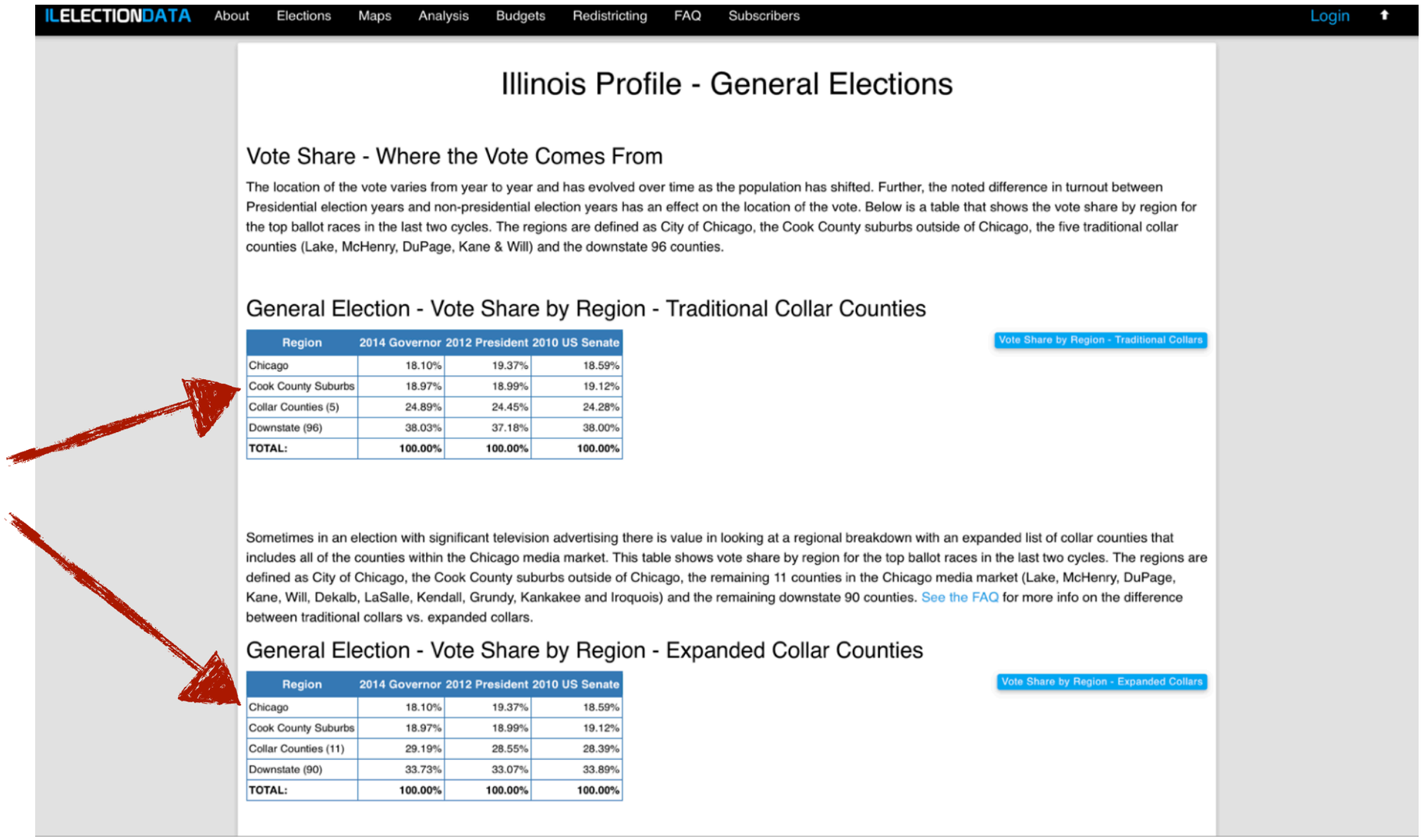
```
# Extract the table element.
# Retrieve the contents as
# a data.frame.
my_webpage %>%
    html_node("table") %>%
    html_table()

# Make sure the node is
# a table before trying to
# use html_table()!
```

Live Web Page

… web scraping illinoiselectiondata.com …

We want to obtain all tables



ILELECTIONDATA About Elections Maps Analysis Budgets Redistricting FAQ Subscribers Login

Illinois Profile - General Elections

Vote Share - Where the Vote Comes From

The location of the vote varies from year to year and has evolved over time as the population has shifted. Further, the noted difference in turnout between Presidential election years and non-presidential election years has an effect on the location of the vote. Below is a table that shows the vote share by region for the top ballot races in the last two cycles. The regions are defined as City of Chicago, the Cook County suburbs outside of Chicago, the five traditional collar counties (Lake, McHenry, DuPage, Kane & Will) and the downstate 96 counties.

General Election - Vote Share by Region - Traditional Collar Counties

Vote Share by Region - Traditional Collars

Region	2014 Governor	2012 President	2010 US Senate
Chicago	18.10%	19.37%	18.59%
Cook County Suburbs	18.97%	18.99%	19.12%
Collar Counties (5)	24.89%	24.45%	24.28%
Downstate (96)	38.03%	37.18%	38.00%
TOTAL:	**100.00%**	**100.00%**	**100.00%**

Sometimes in an election with significant television advertising there is value in looking at a regional breakdown with an expanded list of collar counties that includes all of the counties within the Chicago media market. This table shows vote share by region for the top ballot races in the last two cycles. The regions are defined as City of Chicago, the Cook County suburbs outside of Chicago, the remaining 11 counties in the Chicago media market (Lake, McHenry, DuPage, Kane, Will, Dekalb, LaSalle, Kendall, Grundy, Kankakee and Iroquois) and the remaining downstate 90 counties. See the FAQ for more info on the difference between traditional collars vs. expanded collars.

General Election - Vote Share by Region - Expanded Collar Counties

Vote Share by Region - Expanded Collars

Region	2014 Governor	2012 President	2010 US Senate
Chicago	18.10%	19.37%	18.59%
Cook County Suburbs	18.97%	18.99%	19.12%
Collar Counties (11)	29.19%	28.55%	28.39%
Downstate (90)	33.73%	33.07%	33.89%
TOTAL:	**100.00%**	**100.00%**	**100.00%**

How can we **retrieve elements uniquely?**

SelectorGadget

… obtaining unique identifiers for page elements …

SelectorGadget Screencast from Andrew Cantino on Vimeo.

http://selectorgadget.com/

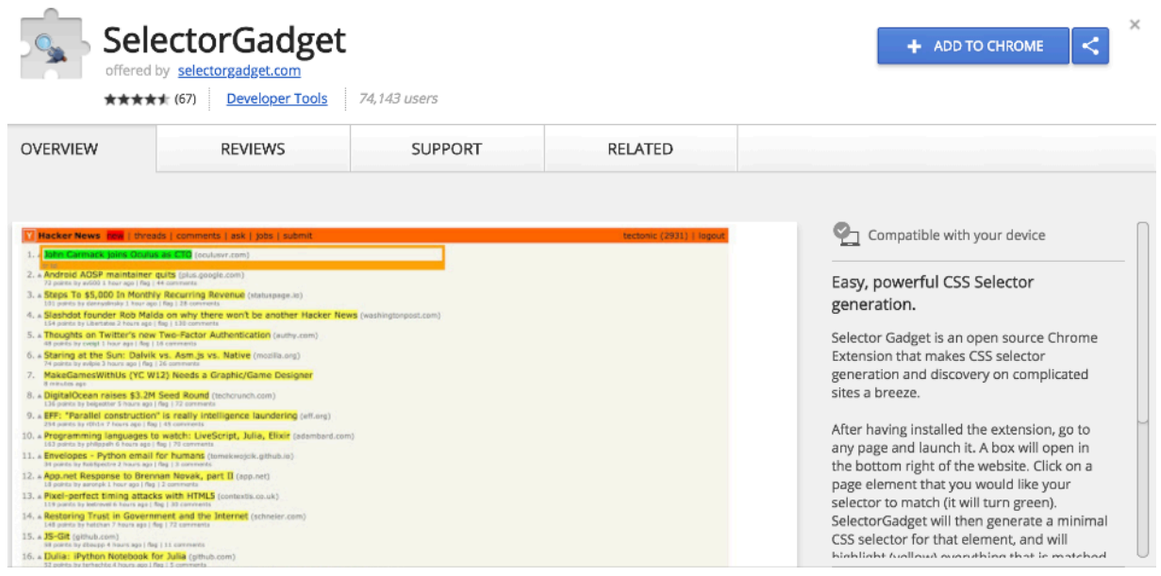
rvest: SelectorGadget Vignette

Install SelectorGadget

… Bookmark vs. Browser …

Hold the click and drag

[Bookmark link on page](#)
(contains JavaScript)



[Chrome Extension](#)

Example Selectors

... possible ways to select HTML ...

Selector	Example	Description
.class	.greentext	Find all elements with class="greentext"
#id	#newsblock	Find elements with id="newsblock"
tag	p	Find all <p> elements
tag tag	p span	Find all and <p> elements
tag > tag	p > span	Find all elements with <p> as a parent
[attribute]	[href]	Find all elements that have a href attribute
[attribute=value]	[href="http://google.com/"]	Find all elements with href="https://google.com/"

For more, please see: https://www.w3schools.com/cssref/css_selectors.asp

SelectorGadget in Action *

… web scraping illinoiselectiondata.com …

ILELECTIONDATA About Elections Maps Analysis Budgets Redistricting FAQ Subscribers Login

Illinois Profile - General Elections

Vote Share - Where the Vote Comes From

The location of the vote varies from year to year and has evolved over time as the population has shifted. Further, the noted difference in turnout between Presidential election years and non-presidential election years has an effect on the location of the vote. Below is a table that shows the vote share by region for the top ballot races in the last two cycles. The regions are defined as City of Chicago, the Cook County suburbs outside of Chicago, the five traditional collar counties (Lake, McHenry, DuPage, Kane & Will) and the downstate 96 counties.

General Election - Vote Share by Region - Traditional Collar Counties

Vote Share by Region - Traditional Collars

Region	2014 Governor	2012 President	2010 US Senate
Chicago	18.10%	19.37%	18.59%
Cook County Suburbs	18.97%	18.99%	19.12%
Collar Counties (5)	24.89%	24.45%	24.28%
Downstate (96)	38.03%	37.18%	38.00%
TOTAL:	**100.00%**	**100.00%**	**100.00%**

div table

Selected

Remove element selection

Sometimes in an election with significant television advertising there is value in looking at a regional breakdown with an expanded list of collar counties that includes all of the counties within the Chicago media market. This table shows vote share by region for the top ballot races in the last two cycles. The regions are defined as City of Chicago, the Cook County suburbs outside of Chicago, the remaining 11 counties in the Chicago media market (Lake, McHenry, DuPage, Kane, Will, Dekalb, LaSalle, Kendall, Grundy, Kankakee and Iroquois) and the remaining downstate 90 counties. See the FAQ for more info on the difference between traditional collars vs. expanded collars.

General Election - Vote Share by Region - Expanded Collar Counties

Vote Share by Region - Expanded Collars

Region	2014 Governor	2012 President	2010 US Senate
Chicago	18.10%	19.37%	18.59%
Cook County Suburbs	18.97%	18.99%	19.12%
Collar Counties (11)	29.19%	28.55%	28.39%
Downstate (90)	33.73%	33.07%	33.89%
TOTAL:	**100.00%**	**100.00%**	**100.00%**

Also included, click to deselect

.myTable-blue Clear (10) Toggle Position XPath Help X

Generated Element Selector

Using the Selector

… Retrieving Tables of Voting Information…

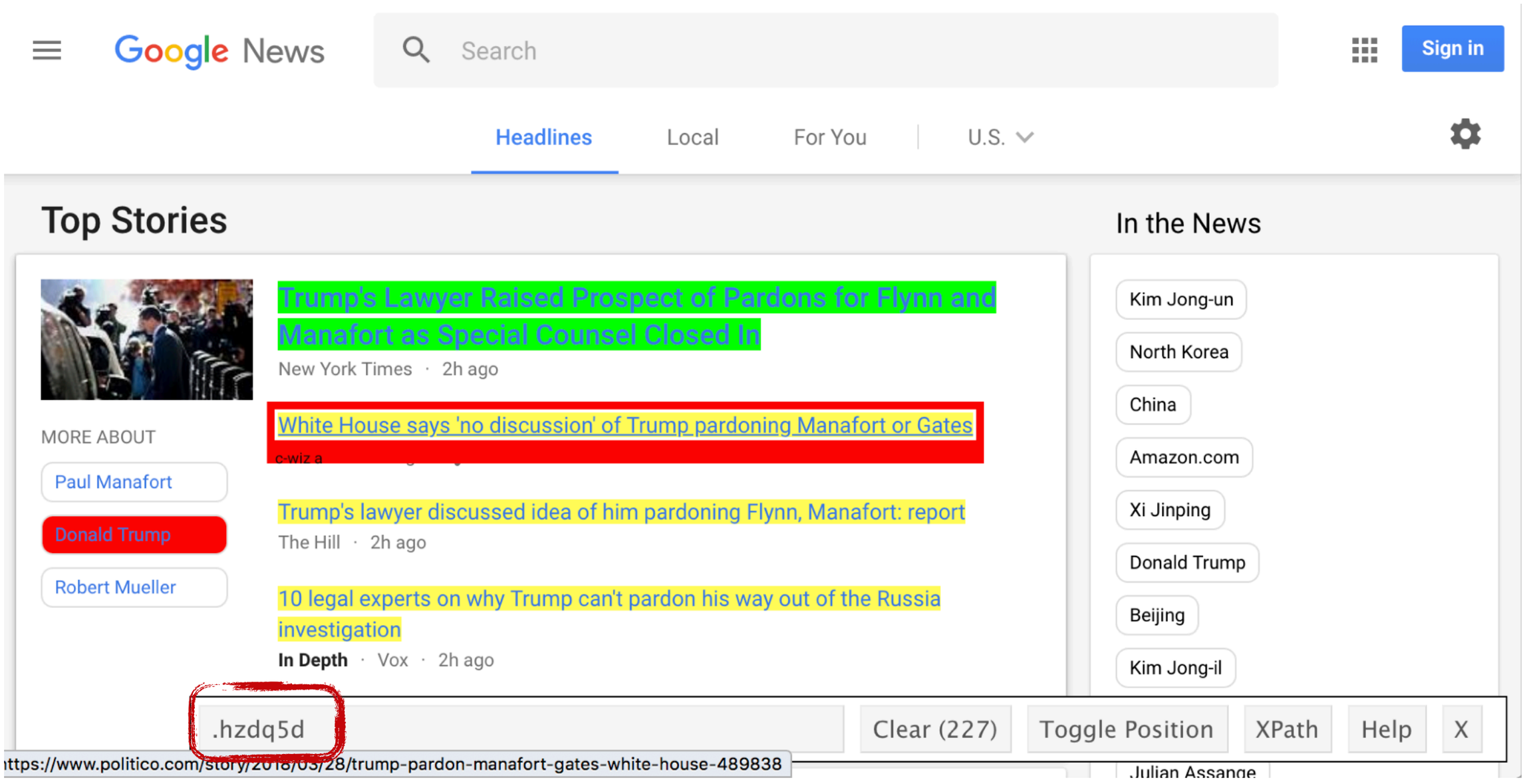
```
# Read in the Web Page to R
# without saving into the server.
il_profile = read_html("https://illinoiselectiondata.com/elections/ILprofile.php")

# Select the elements that are tables.
# Extract the contents as a data.frame.
il_profile %>%
  html_nodes(".myTable-blue") %>%
  html_table()
```

New Selections

... web scraping news.google.com ...

*



---

* The class selector found for the title links are time sensitive. If you repeat this procedure, you will receive a different class. The only way to change this is to look into the link attributes.

Time-limited Selectors

… retrieving titles for Google News …

```
# Read in the Web Page to R without saving into the server.
gnews = read_html("https://news.google.com")

# Select story titles and retrieve the text.
gnews %>%
  html_nodes(".hzdq5d") %>%
  html_text()
#  [1] "Trump's Lawyer Raised Prospect of Pardons for Flynn and Manafort as Special
Counsel Closed In"
#  [2] "Trump's lawyer allegedly raised possibility of pardons for Manafort, Flynn last
summer"
#  [3] "White House says 'no discussion' of Trump pardoning Manafort or Gates"
```

Your Turn

Find the top listed stars of The Thomas Crown Affair

Extract the financial information from the Open Secrets campaign initiative.

https://www.opensecrets.org/races/election?id=IL

Hint: You will need to use html_table(x, fill = TRUE)

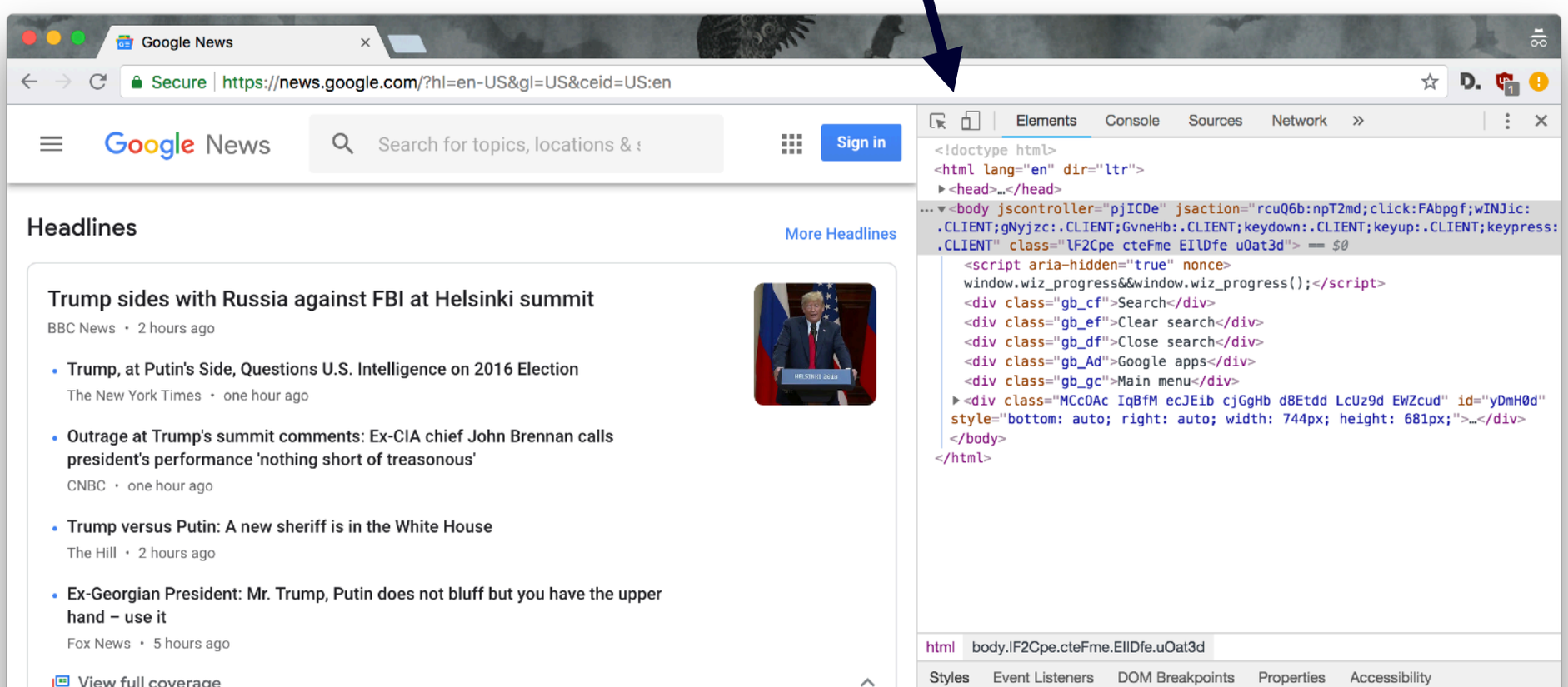
Advanced Scraping

Web Developer Tools

… exploring a website's source code …

Open Chrome's Web Developer Tools

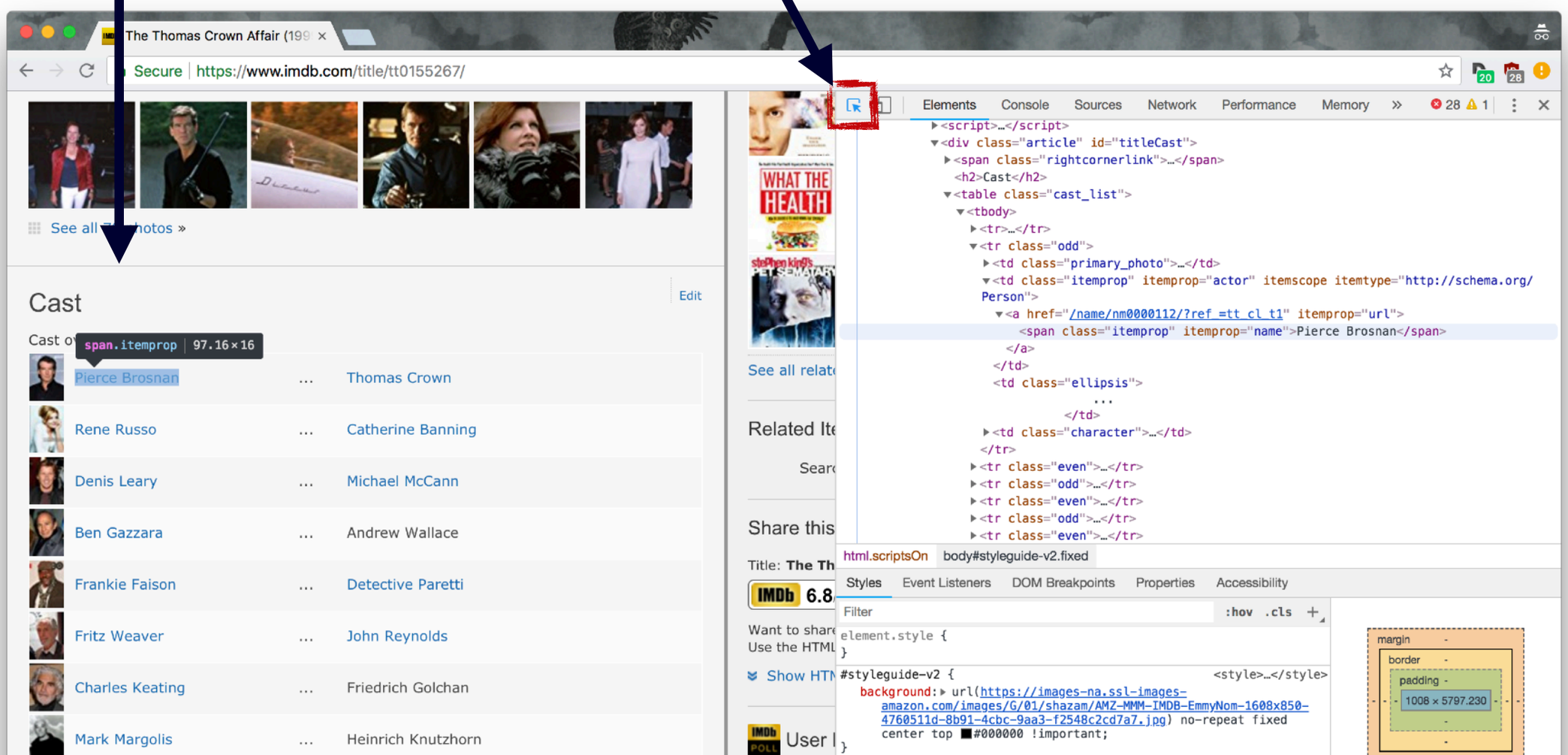
- Windows: **Ctrl** + **Shift** + **I**
- macOS: **Command** + **Option** + **I**



Example: IMDb

… selecting story headlines more robustly …

1. Click on the selector
2. Mouse over element on webpage
3. Click to Select

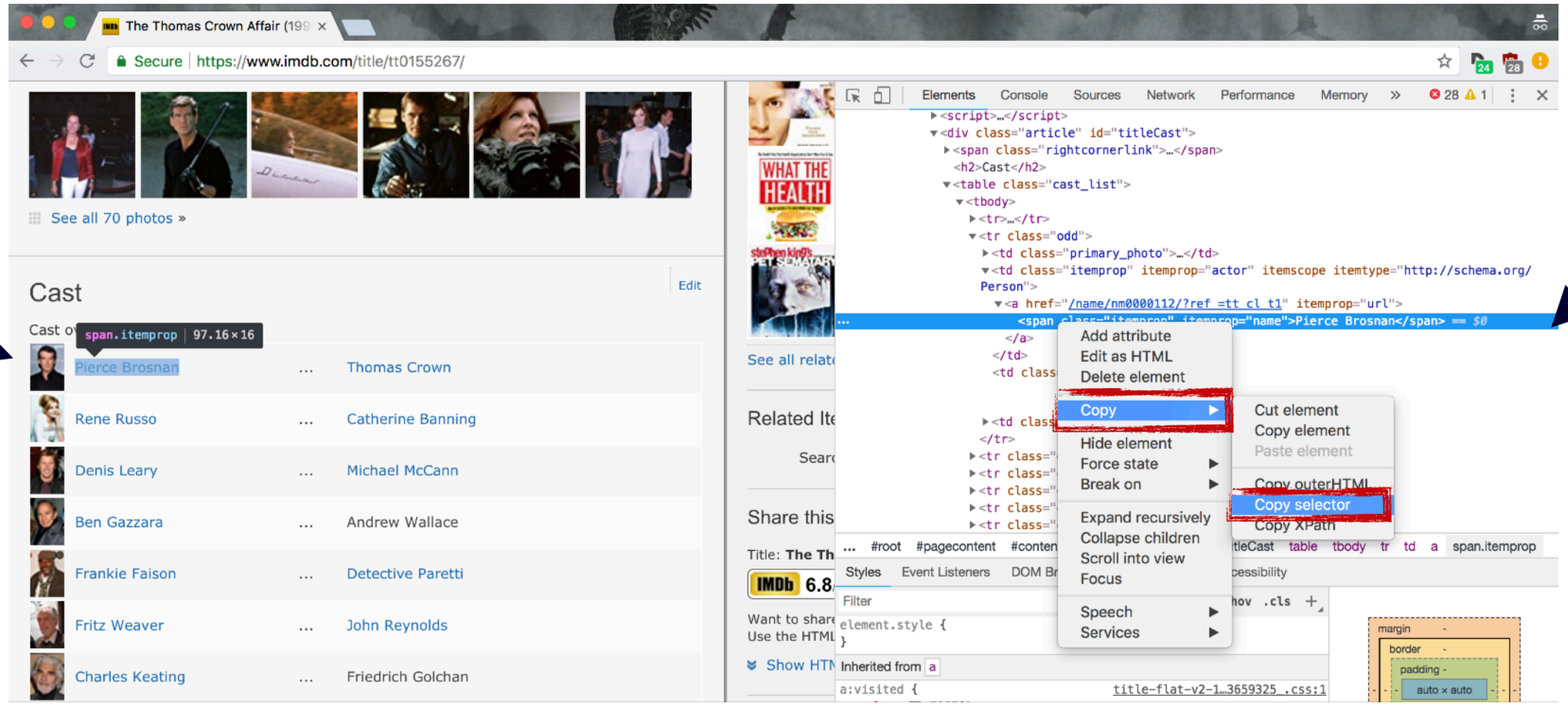


Retrieve Selector

… note this is still a unique selector …

Selected

Right Click



#titleCast > table > tbody > tr:nth-child(2) > td.itemprop > a > span

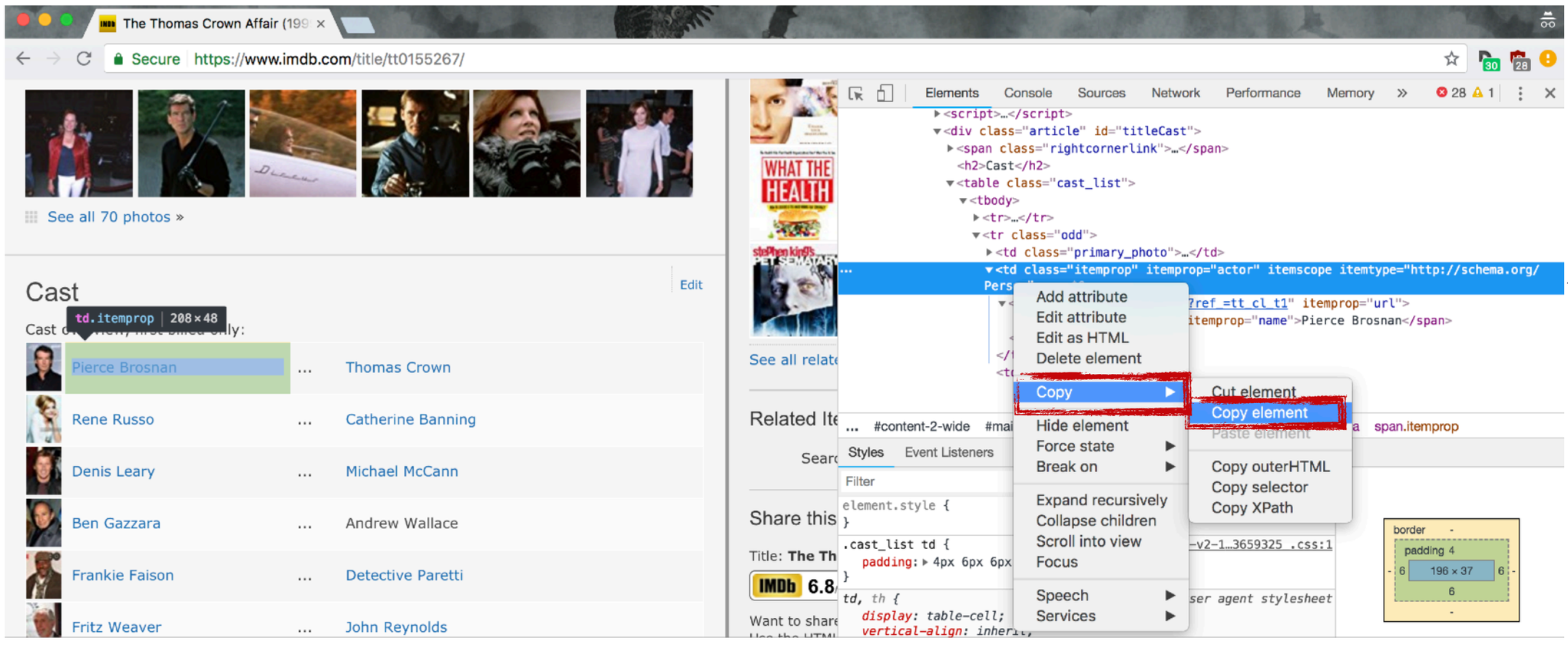
How can we **generalize to obtain all actors?**

Finding Commonalities in Semistructure

```
<td class="itemprop" itemprop="actor" itemscope=""
itemtype="http://schema.org/Person">
<a href="/name/nm0000112/?ref_=tt_cl_t1" itemprop="url">
<span class="itemprop" itemprop="name">Pierce Brosnan</
span>
</a> </td>
```

Selected

Right Click



```
# Read in the Movie
imdb_movie = read_html("https://www.imdb.com/title/tt0155267/")

# Create a CSS selector based on two or more element attributes.
imdb_movie %>%
  html_nodes("td[itemprop=\"actor\"] span[itemprop=\"name\"]") %>%
  html_text()
#  [1] "Pierce Brosnan"  "Rene Russo"      "Denis Leary"
#  [4] "Ben Gazzara"     "Frankie Faison"  "Fritz Weaver"
#  [7] "Charles Keating" "Mark Margolis"   "Faye Dunaway"
# [10] "Michael Lombard" "Bill Ambrozy"    "Michael Bahr"
# [13] "Robert D. Novak" "Joe H. Lamb"     "James Saito"
```

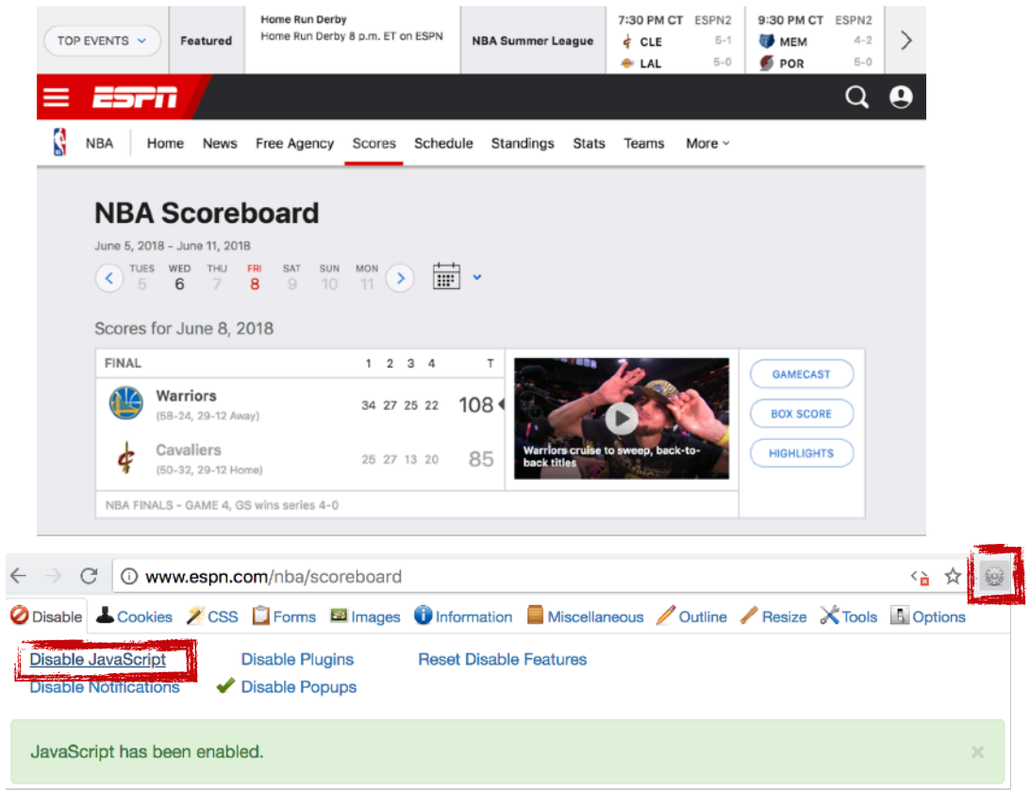
Creating a CSS Selector

… ensuring the right values are extracted …

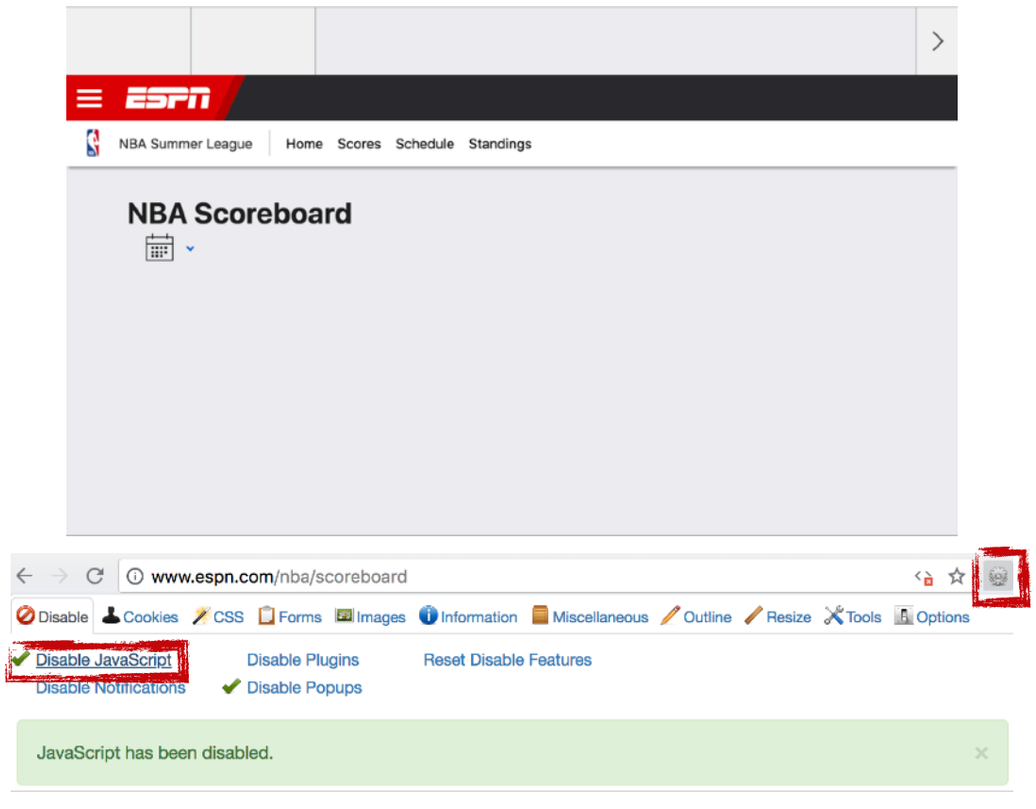
---

* The use of \" is to escape the quotation character as it is already used.
** We opt to not specify a > as there is another element nested.

Enabled JavaScript (default)



Disabled JavaScript



Website: http://www.espn.com/nba/scoreboard

Checking for JavaScript

… hidden content error …

Install Web Developer Extension for Chrome.

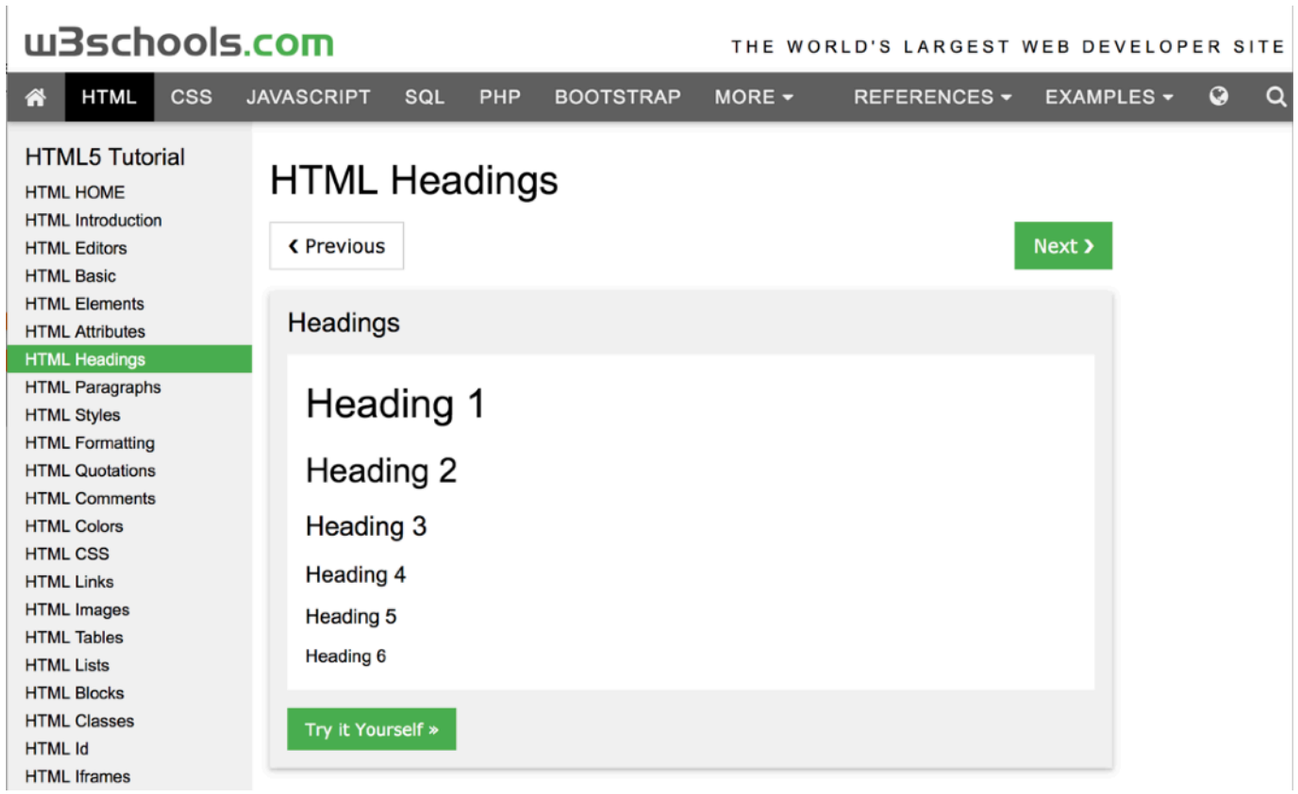
Disable JavaScript, reload the page, and watch content disappear.

- One solution is to use phantomjs to render the page locally and then scrape it.
- Another solution is to use RSelenium to replicate a web browser and scrape data. This isn't very stable though.

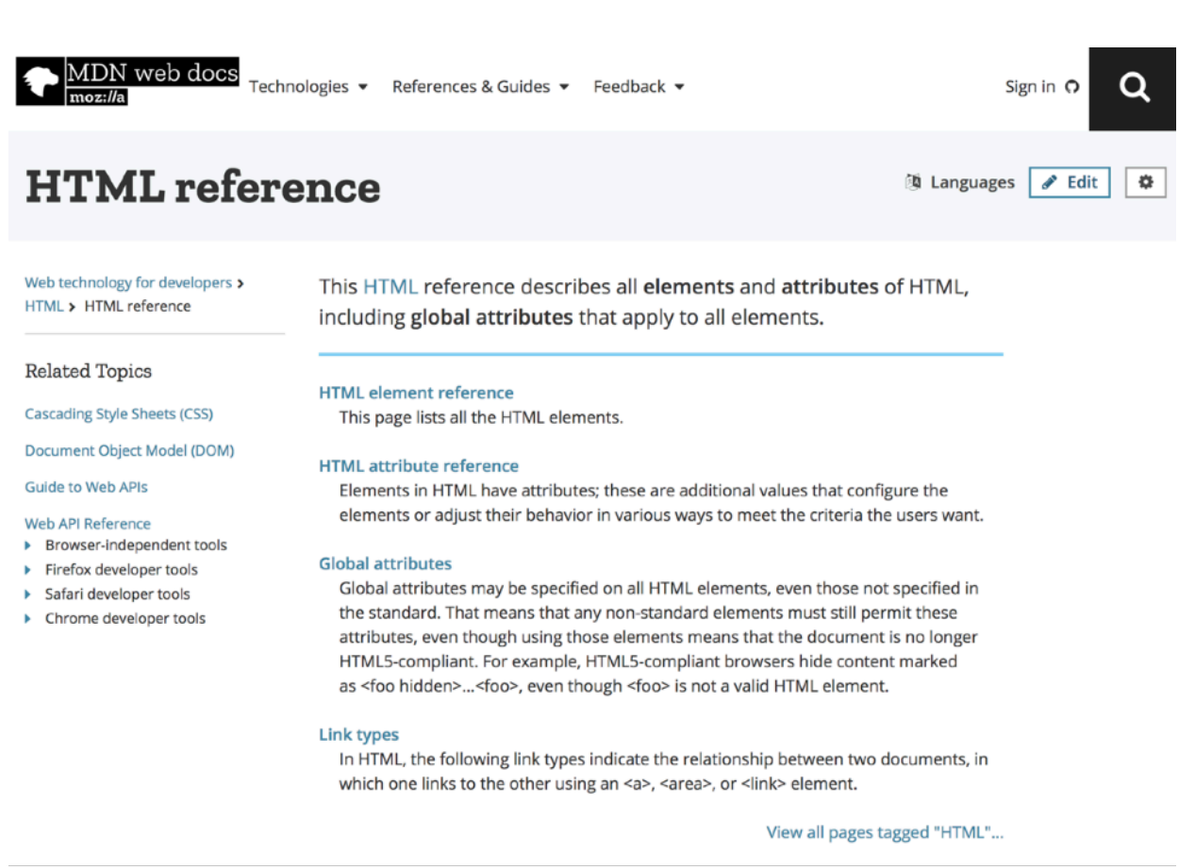
Resources

HTML Reference

… additional HTML help …



[W3Schools HTML Reference]

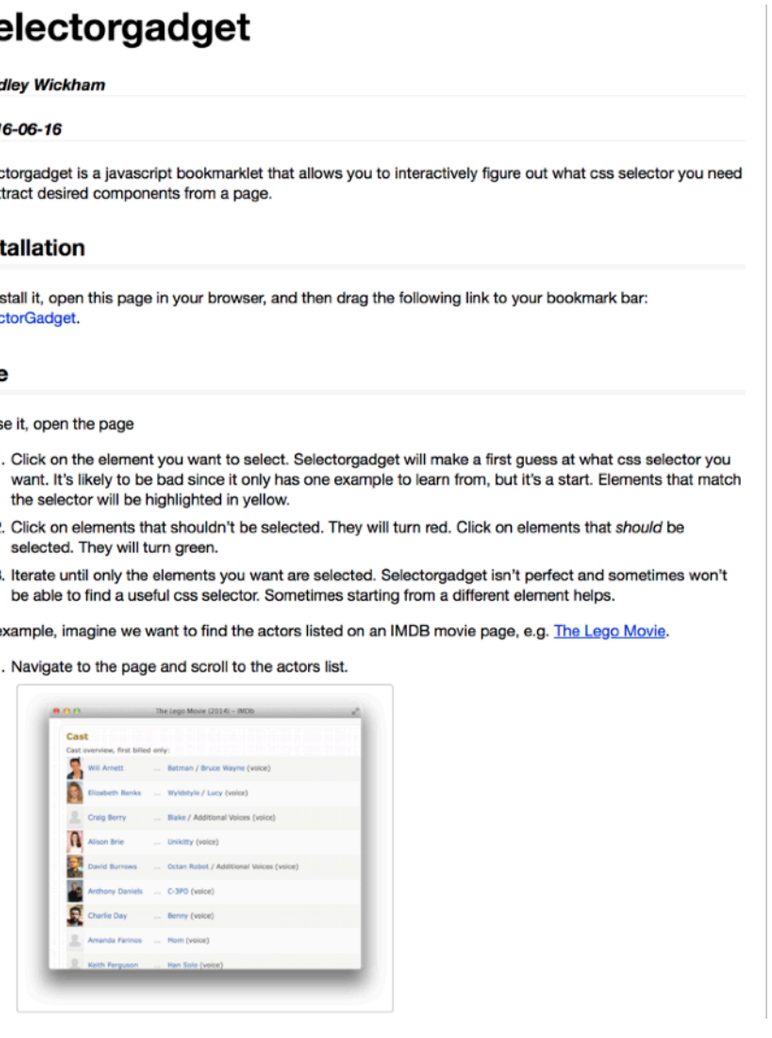


[Mozilla's HTML Reference]

SelectorGadget

… identify elements …

http://selectorgadget.com/



rvest: SelectorGadget Vignette

Recap

- **HTTP(S)**
  - Way to retrieve websites.
- **HTML**
  - Semi-structured Data
  - Language of the web
- **Pipe Operator**
  - Facilitate the flow of multi-step problems
- **Scraping Information**
  - Extracts values from HTML Tags