Notes at https://github.com/hadley/web-scraping

Scraping websites with R

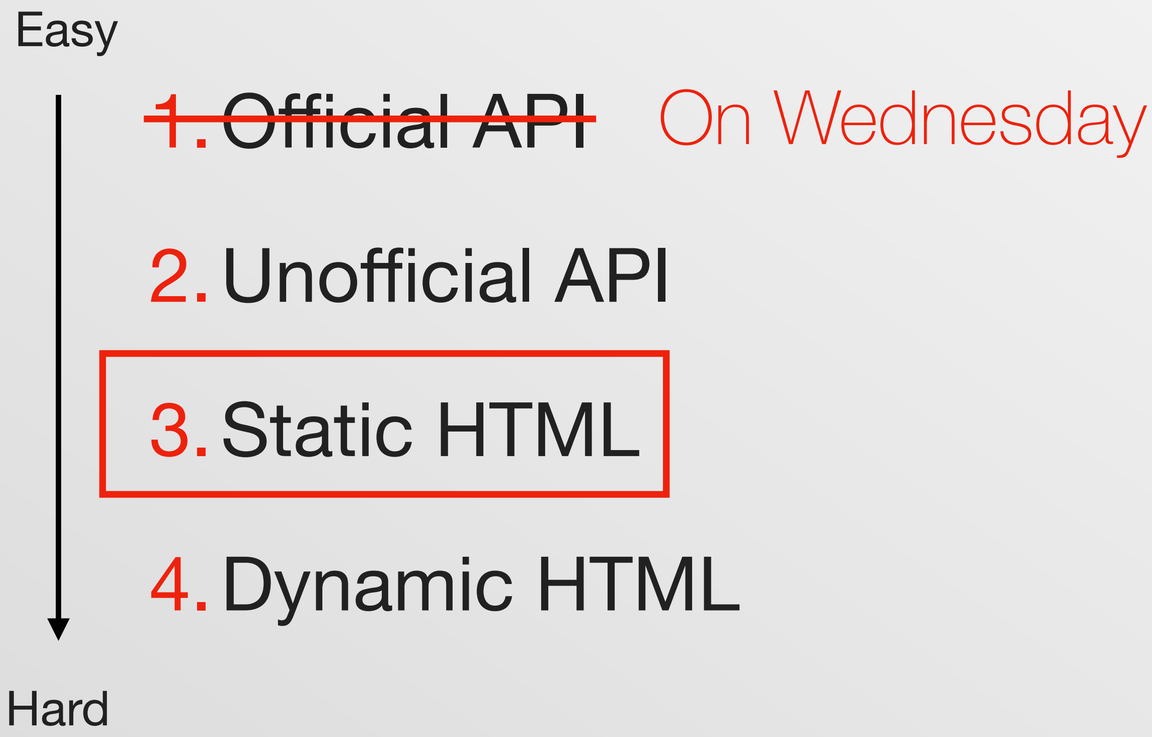
Using rvest and the tidyverse

Hadley Wickham
Chief Scientist, Posit

July 2024

Introductions

Getting data off a website



1. HTML structure
2. rvest basics
3. Extracting data
4. Bonus content

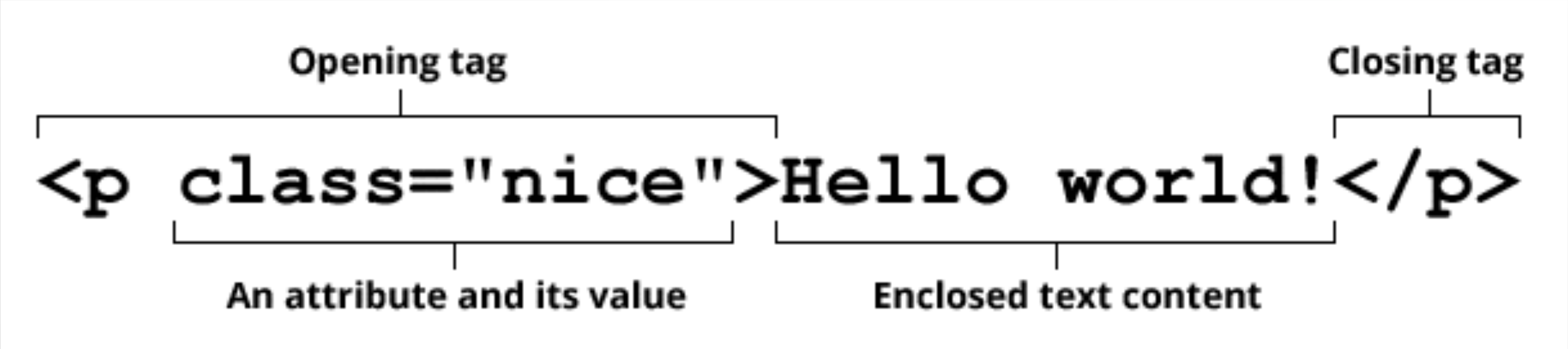
   Pagination, Live HTML, Unofficial APIs, LLMs

HTML structure

HTML is a tree

```
<!doctype html>
<html lang="en-US">
  <head>
    <title>Page title</title>
    <meta  ... >
    <script  ... ></script>
     ...
  </head>
  <body>
     ...
  </body>
</html>
```

The tree is made up of elements

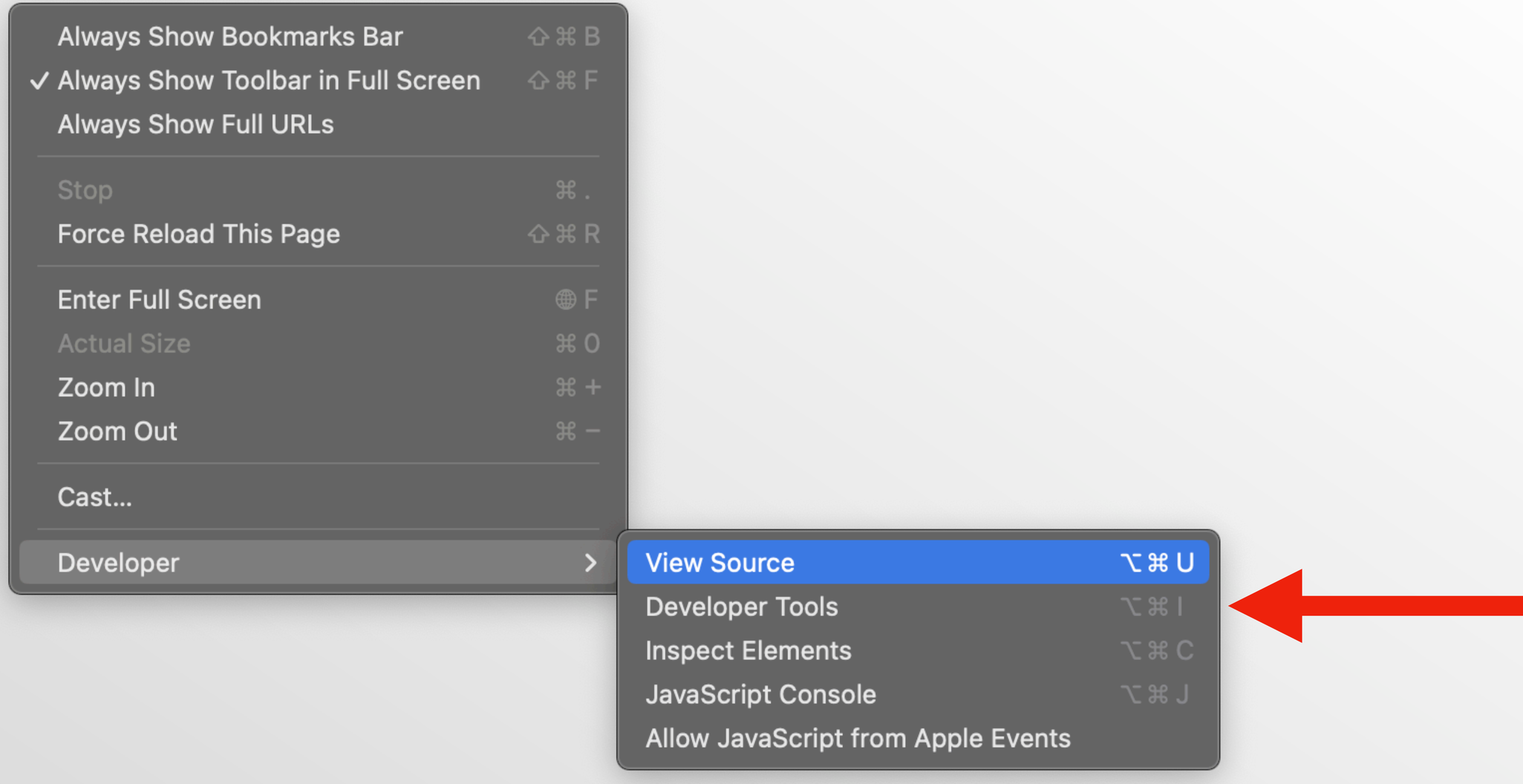


https://developer.mozilla.org/en-US/docs/Glossary/Element

The stuff you see on a page comes from the body

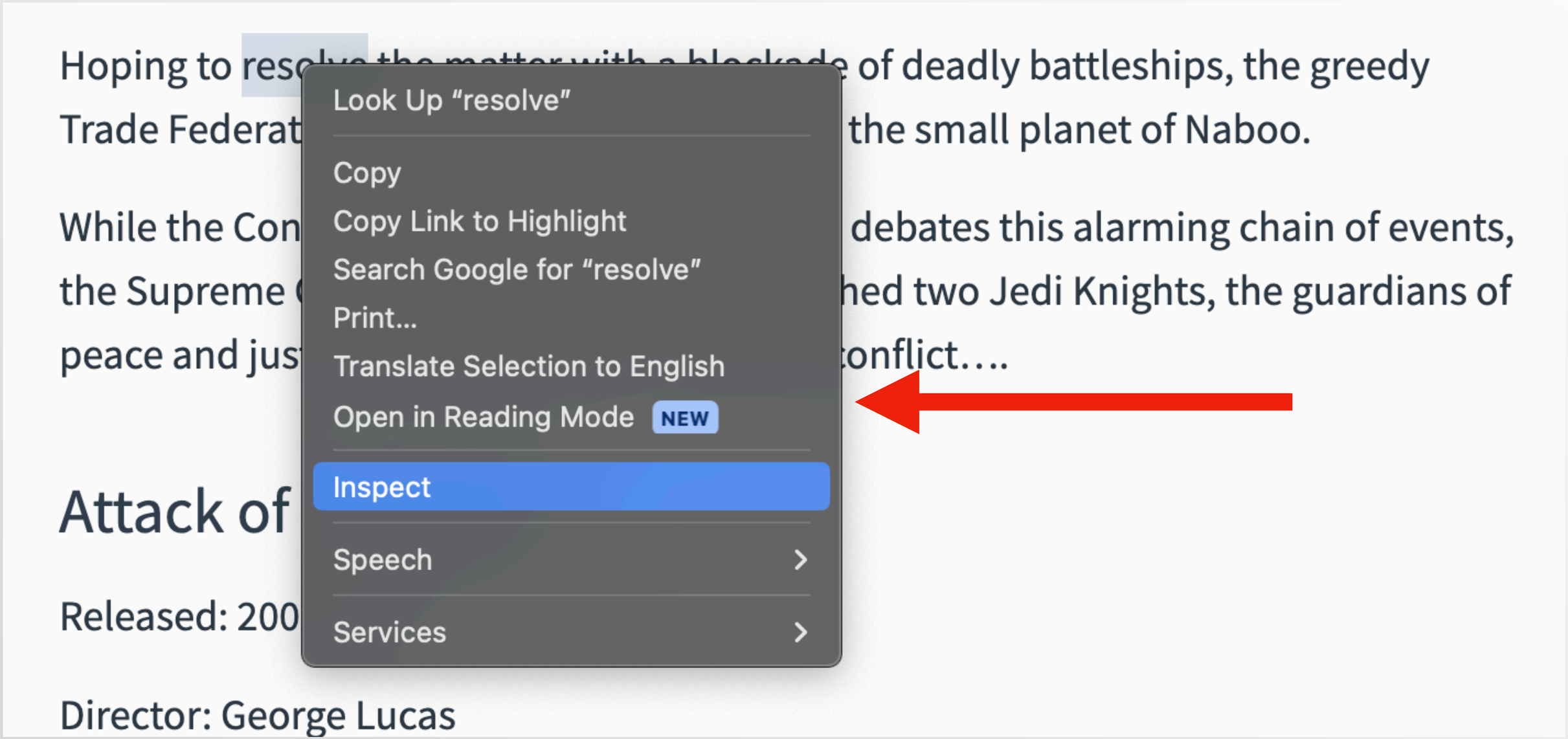
```
<body>
  <h1>Top level heading</h1>
  <p>A paragraph containing text that is <b>bold</b> and
    an image: <img src="..." /></p>
  <ul>
    <li>A bulleted list</li>
    <li>Bullet two</li>
  </ul>
  <table>
    <tr><th>A</th><th>B</th></tr>
    <tr><td>1</td><td>2</td></tr>
  </table>
</body>
```

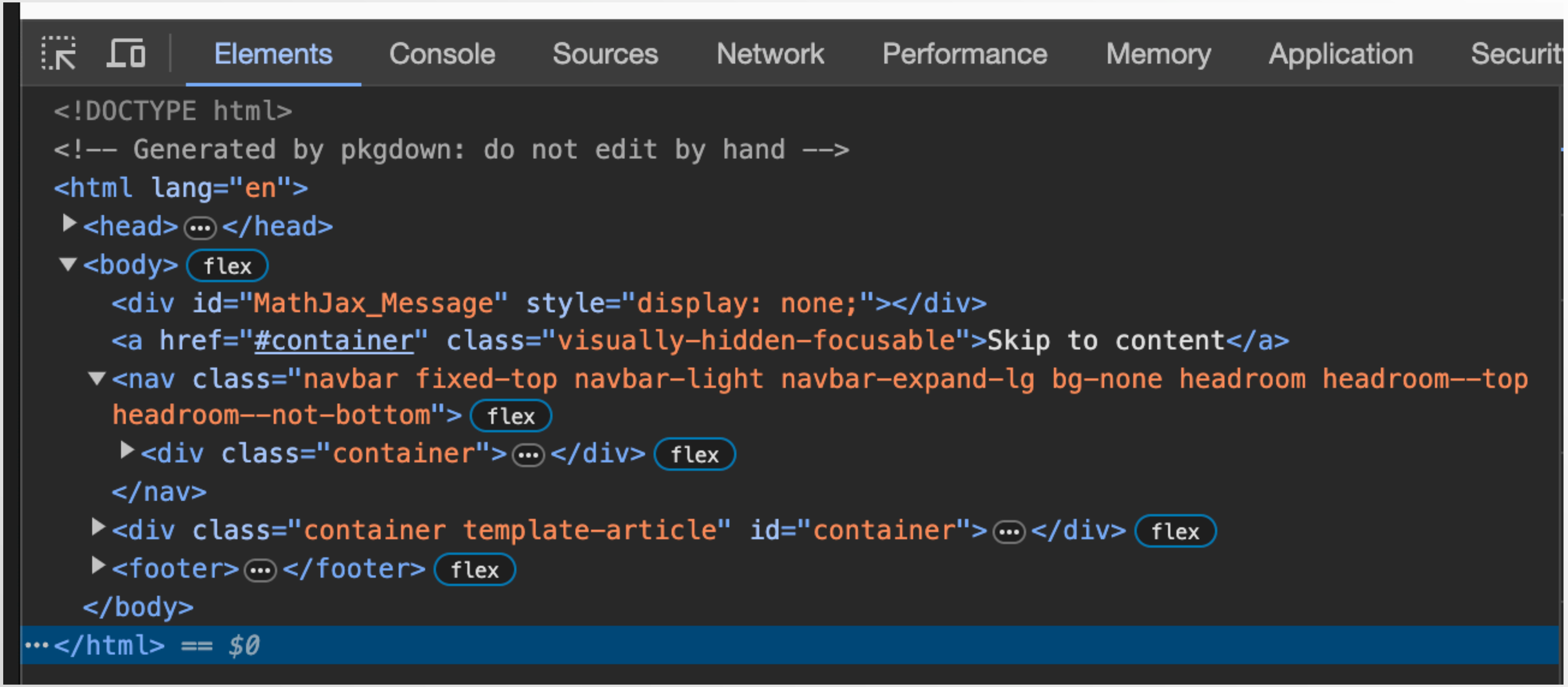
Best way to see the tree of a real page is to use DevTools



I recommend Chrome for this, even if it isn't your daily driver

Or right-click and choose inspect




Elements
Console
Sources
Network
Performance
Memory
Application
Securit
<!DOCTYPE html>
<!-- Generated by pkgdown: do not edit by hand -->
<html lang="en">
<head> ... </head>
<body> flex

Skip to content
<nav class="navbar fixed-top navbar-light navbar-expand-lg bg-none headroom headroom--top headroom--not-bottom"> flex
 ... flex
</nav>
 ... flex
<footer> ... </footer> flex
</body>
... </html> == $0

Your turn

Go to <https://rvest.tidyverse.org/articles/starwars.html>, open the developer tools, and use the "elements" view to answer the following questions:

What element contains the film titles?

What element contains the name of the director? What attributes does this element have?

How many paragraphs does the “crawl” at the start of each movie have?

Where does the table of contents live relative to the film data?

Static vs dynamic

The HTML displayed in the elements pane is usually generated from a HTML file that you can find in the sources pane.

Sometimes, however, the HTML is generated **dynamically** with Javascript. We'll come back to this later, but I wanted to illustrate the difference.

Let's look at <https://rvest.tidyverse.org/dev/articles/starwars-dynamic.html>

rvest basics

Easiest to get data from HTML if it’s already in a table

```
<table>
  <tr>
    <th>Character</th>
    <th>Role</th>
    <th>Affiliation</th>
  </tr>
  <tr>
    <td>Luke Skywalker</td>
    <td>Jedi Knight</td>
    <td>Rebel Alliance</td>
  </tr>
  <tr>
    <td>Darth Vader</td>
    <td>Sith Lord</td>
    <td>Galactic Empire</td>
  </tr>
</table>
```

Header

Row

Cell
(d = data)

Can read with html_table()

```
library(rvest)
html ← minimal_html("<table>
  <tr>
    <th>Character</th>
    <th>Role</th>
    <th>Affiliation</th>
  </tr>
  <tr>
    <td>Luke Skywalker</td>
    <td>Jedi Knight</td>
    <td>Rebel Alliance</td>
  </tr>
</table>")
html_table(html)
```

Let’s look at a more realistic example

```
# I want to get the sound track for Star Wars movie

url ← "https://en.wikipedia.org/wiki/Star_Wars_(soundtrack)"
html ← read_html(url)

html ▷
  html_table() ▷
  _[5:8]
```

Can we do better than asking for tables 5 through 8?

```
# I want to get the sound track for Star Wars movie

url ← "https://en.wikipedia.org/wiki/Star_Wars_(soundtrack)"
html ← read_html(url)

html ▷
  html_table() ▷
  _[5:8]
```

Your turn

Open <https://en.wikipedia.org/wiki/Star_Wars_(soundtrack)>

Using the developer tools, can you find something in the structure of the HTML that uniquely identifies these tables?

We can use html_elements() with CSS selectors

```
url ← "https://en.wikipedia.org/wiki/Star_Wars_(soundtrack)"
html ← read_html(url)

html ▷
  html_elements(".tracklist") ▷
  html_table()

# .tracklist means all elements with class tracklist
```

CSS selectors

CSS = cascading style sheets

Primary purpose is to separate the visual appearance (style) from its underlying semantics.

Used to say (e.g.) “make this box blue” or “make all links green”.

It’s a domain specific language for selecting elements in the HTML tree. We’ll use it to identify the HTML elements that contain the data we care about.

Most important selectors

- `.brown` = all elements with class "brown"
- `#abc` = single element with id "abc"
- `p` = all paragraphs
- `p.important` = all paragraphs with "important" class
- `p b` = all bold elements that are descendants of a paragraph
- `p > b` = all bold elements that are children of a paragraph

Your turn

Use <https://flukeout.github.io/> to learn and practice the most important selectors.

Extracting data

1. Find the “rows” with html_elements()
2. Find the “columns” with html_element()
3. Extract the data with html_text2() or html_attr()
4. Make a tibble
5. Clean it up

Your turn

- Head back to https://rvest.tidyverse.org/articles/starwars.html
- What are the rows? How can you identify them with a css selector?
- What are the columns? How can you identify them with a css selector?

Solution

starwars.R

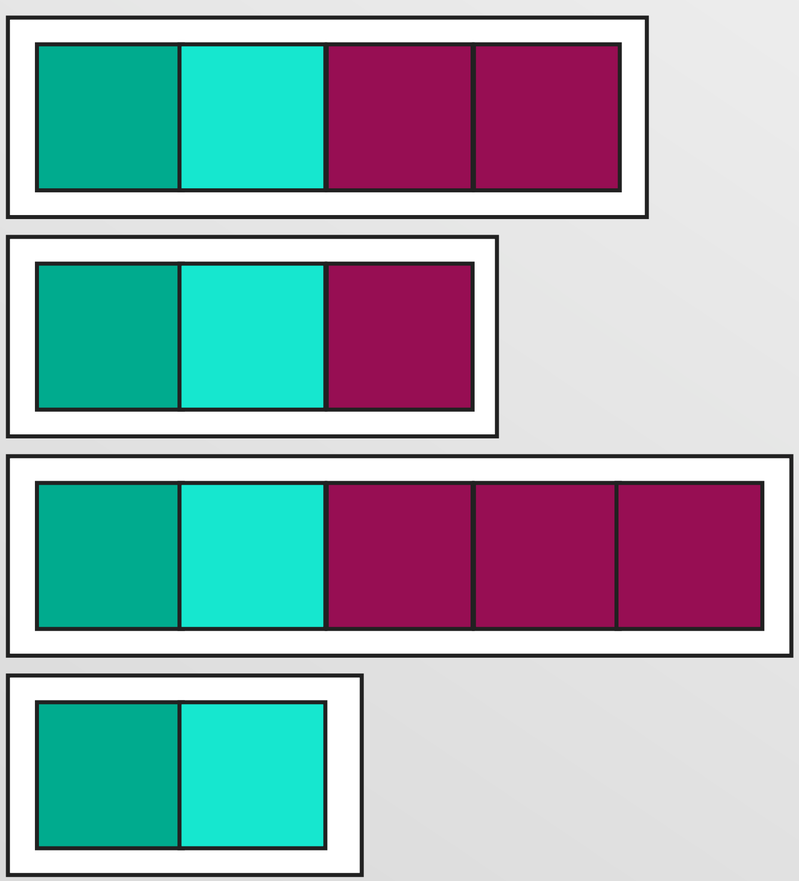
Your turn

- Go to https://quotes.toscrape.com/
- Identify the rows and columns (including the URL to the author page), and the selectors that will identify them.
- (Challenge: how might store the tags in a list column? Why might this be useful? What makes it hard?)
- Scrape into a tibble

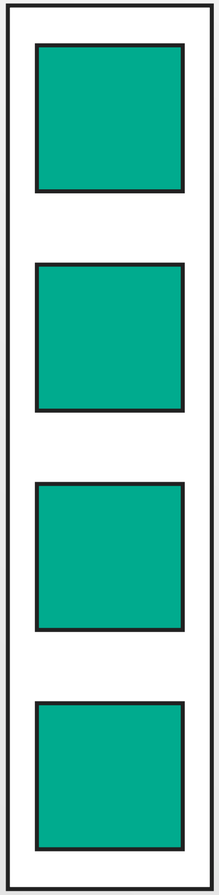
Solution

quotes.R

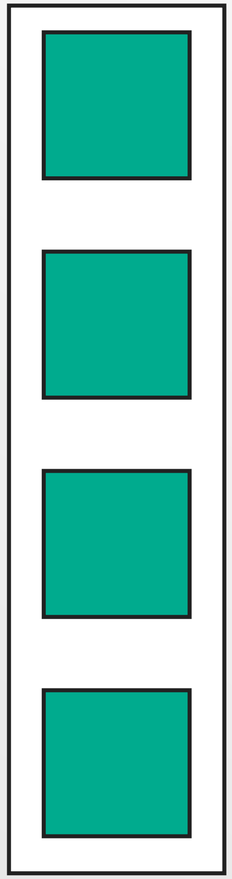
There’s no difference when there’s one match



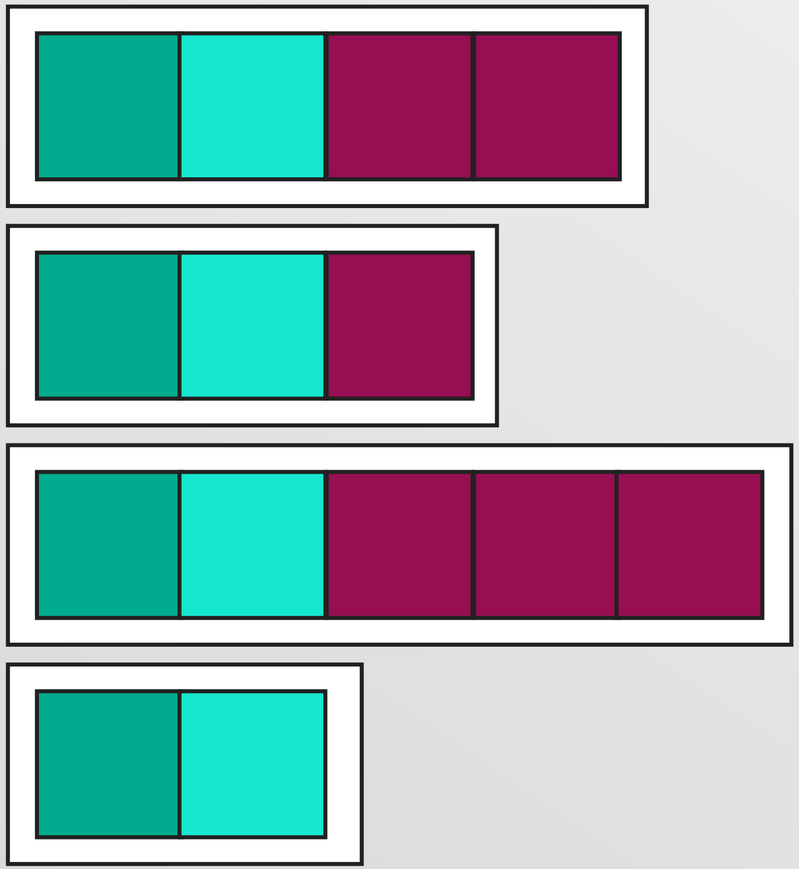
html_element()



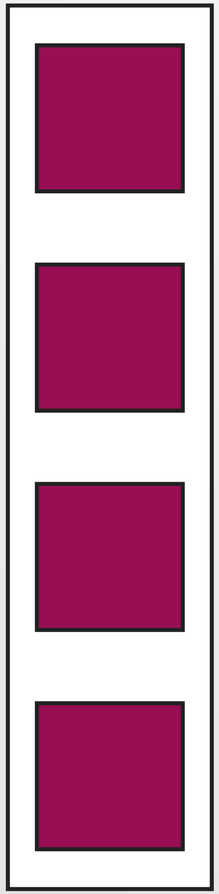
html_elements()



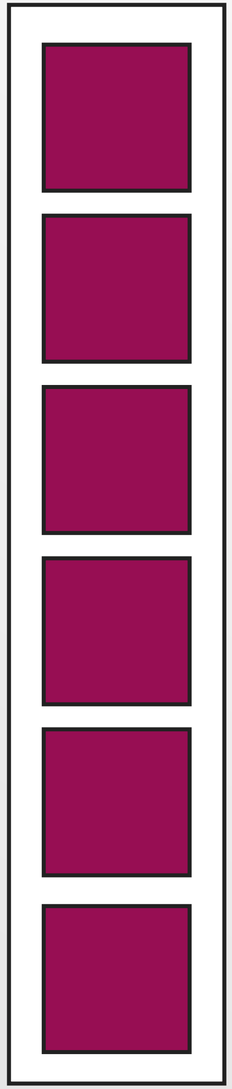
There’s a big difference when numbers of matches varies



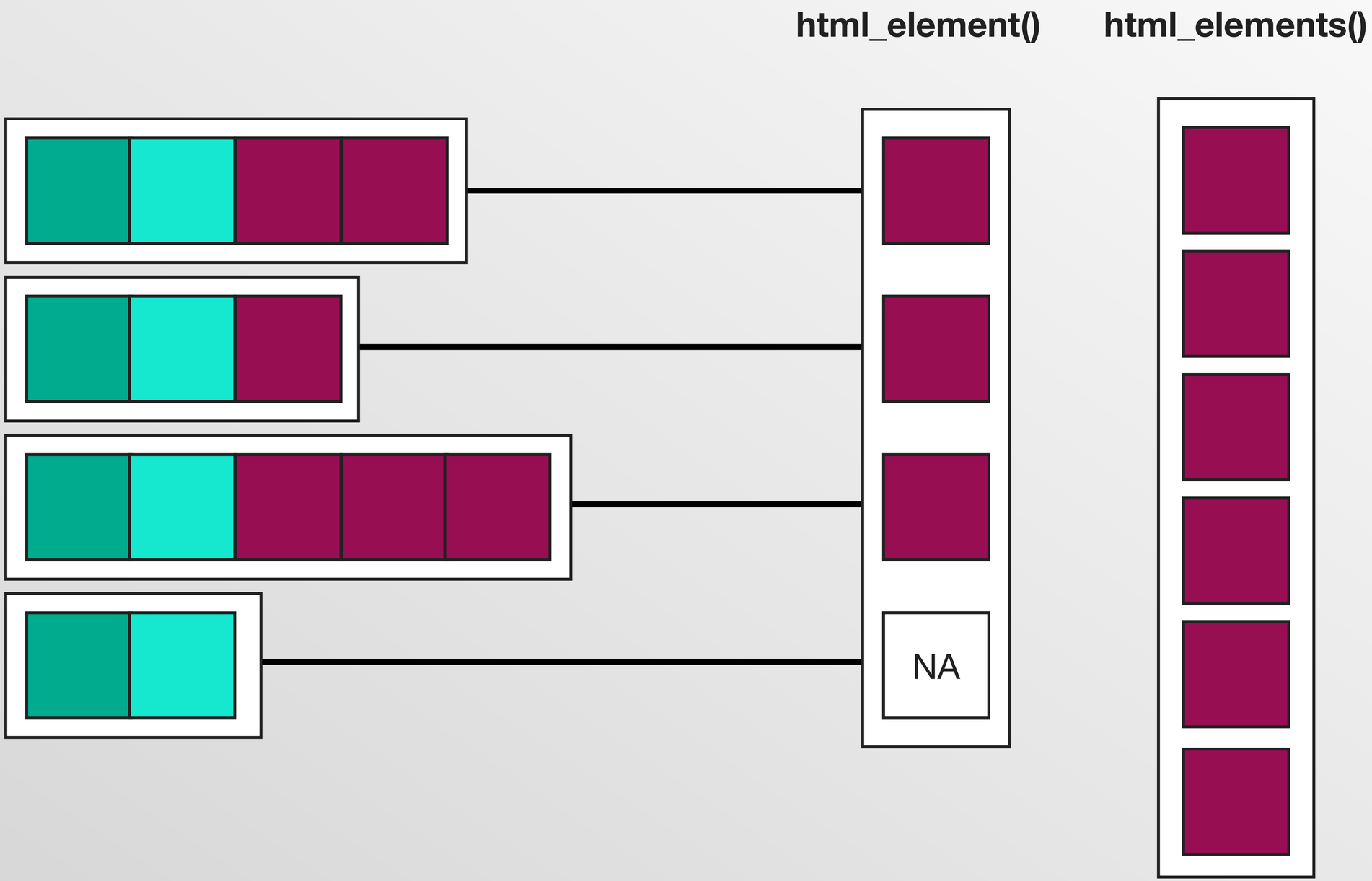
html_element()



html_elements()



html_elements() is always one-to-one



html_elements() vs html_element()

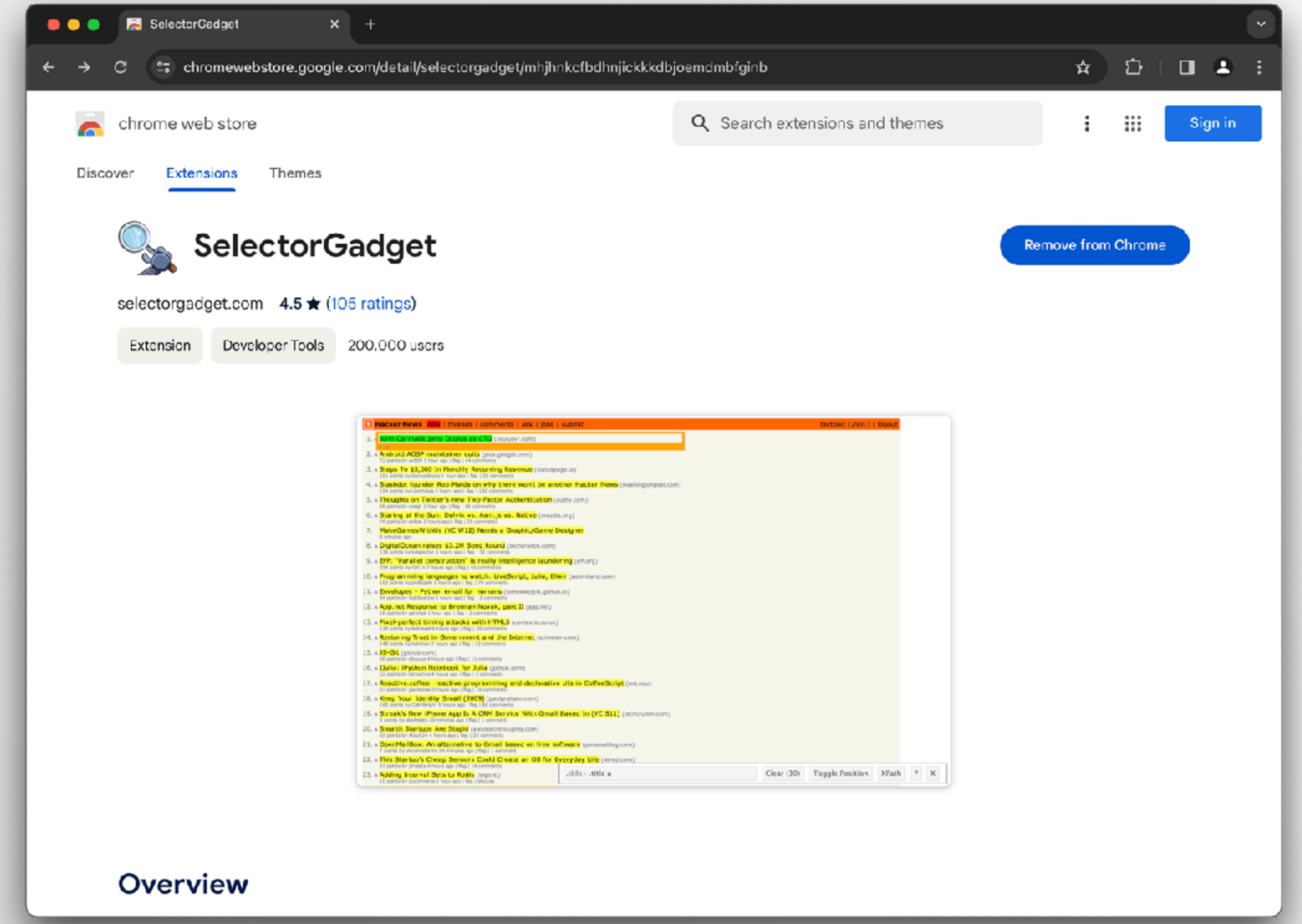
html_elements()	**html_element()**
n -> m	n -> n
length(0)	NA
Find rows	Find column in each row

Ways to find the selector

1. Directly inspecting the HTML
2. In DevTools, right-click & choose “Copy selector” (then simplify)
3. SelectorGadget

Google for selector gadget

https://chrome.google.com/webstore/detail/selectorgadget



More practice

Scrape all the books off <http://books.toscrape.com/>

Make sure to capture the (full) title, the rating, the path to the cover image and the price.

(Don’t worry about the pagination yet)

Pagination

Demo

pagination.R

Live HTML

Forbes America's Top Colleg

forbes.com/top-colleges/

Forbes

Subscribe Sign In

Forbes

America's Top Colleges 2023

Princeton University tops the 2023 list, while California public colleges continue to show their strength.

Edited by Emma Whitford with Janet Novack

Illustration by Wil Cordell For Forbes

Forbes' annual list showcases 500 of the finest U.S. colleges, ranked using data on student success, return on investment and alumni influence.

Every year, Forbes spotlights the 500 U.S. colleges that check all the boxes: impressive graduation rates, high graduate salaries, and great outcomes for low-income students, to name a few. Princeton University (#1)—with median early-career salaries north of $88,000 and a 98% six-year graduation rate—aces these criteria. But it's not just the Ivy League that delivers on the promise of a quality education.

Forbes America's Top College

forbes.com/top-colleges/

Forbes

Subscribe Sign In

America's Top Colleges List

FILTER LIST BY:

OVERALL | STATE | PUBLIC/PRIVATE | SCHOOL SIZE | CAMPUS SETTING

SEARCH

RANK	NAME	STATE	TYPE	AV. GRANT AID	AV. DEBT	MEDIAN 10-YEAR SALARY	FINANCIAL GRADE
1	Princeton University	NJ	Private not-for-profit	$47,136	$7,216	$177,300	A+
2	Yale University	CT	Private not-for-profit	$58,715	$4,968	$163,900	A+
3	Stanford University	CA	Private not-for-profit	$56,211	$8,868	$173,800	A+
4	Massachusetts Institute of Technology	MA	Private not-for-profit	$32,562	$7,235	$182,800	A+
5	University of California, Berkeley	CA	Public	$21,406	$7,202	$161,300	
6	Columbia University	NY	Private not-for-profit	$57,726	$13,338	$150,900	A+
7	University of California, Los Angeles	CA	Public	$17,592	$5,965	$140,300	
8	University of Pennsylvania	PA	Private not-for-profit	$50,778	$10,510	$165,700	A+
9	Harvard	MA	Private not-for-	$50,840	$8,709	$169,500	A+

Top Colleges 2023: Why California's Public Universities Are So Good

It's no surprise that Princeton and Harvard made the top 25 colleges, but despite California's woes, four of its public universities did so too.

READ MORE ›

Your turn

<https://www.forbes.com/top-colleges/>

Where does the data live? What defines the rows and the columns? (Hint: it looks like a table, but it’s not)

What happens if you try to read this data into R?

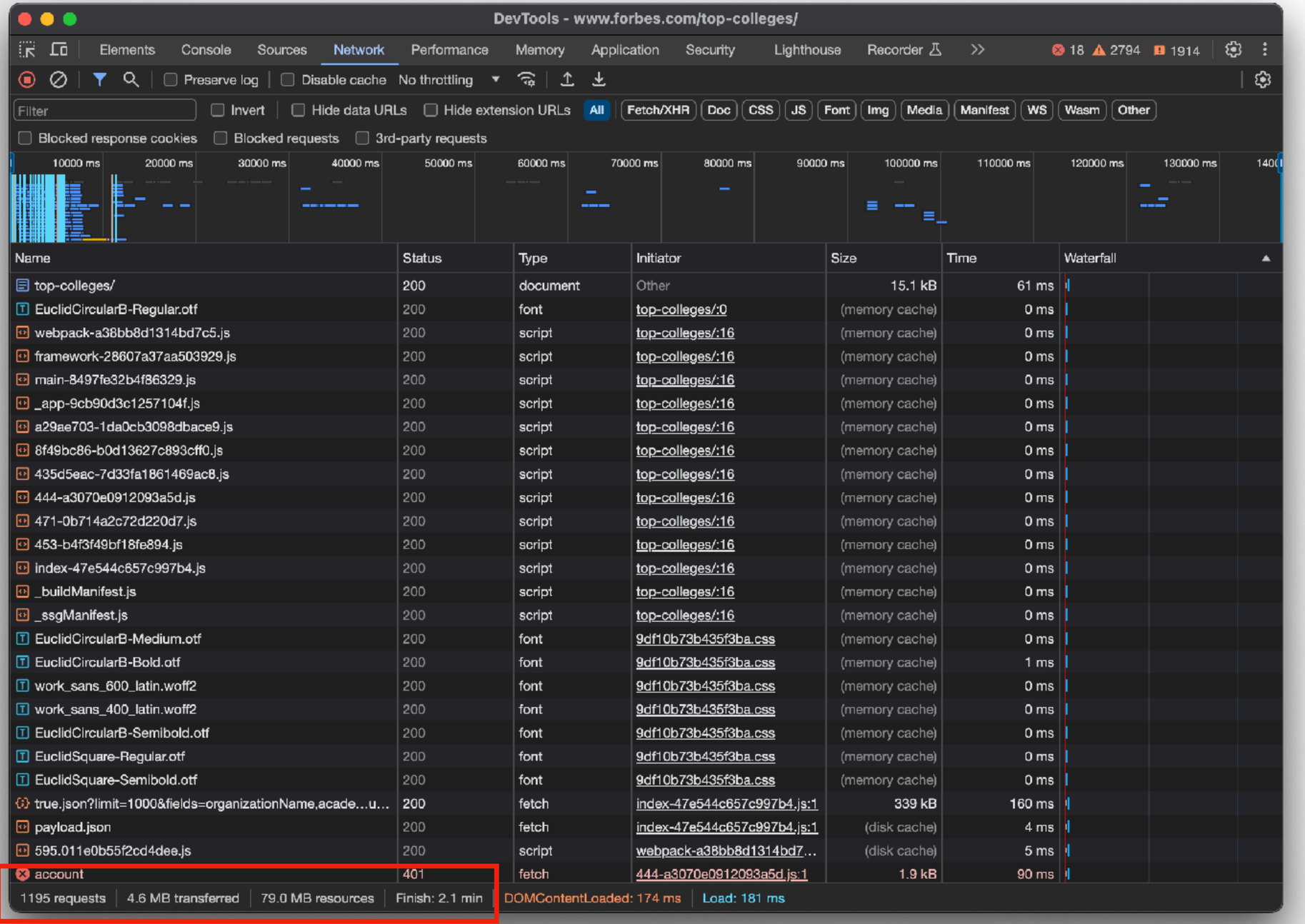
Solution

forbes-live.R

Unofficial API

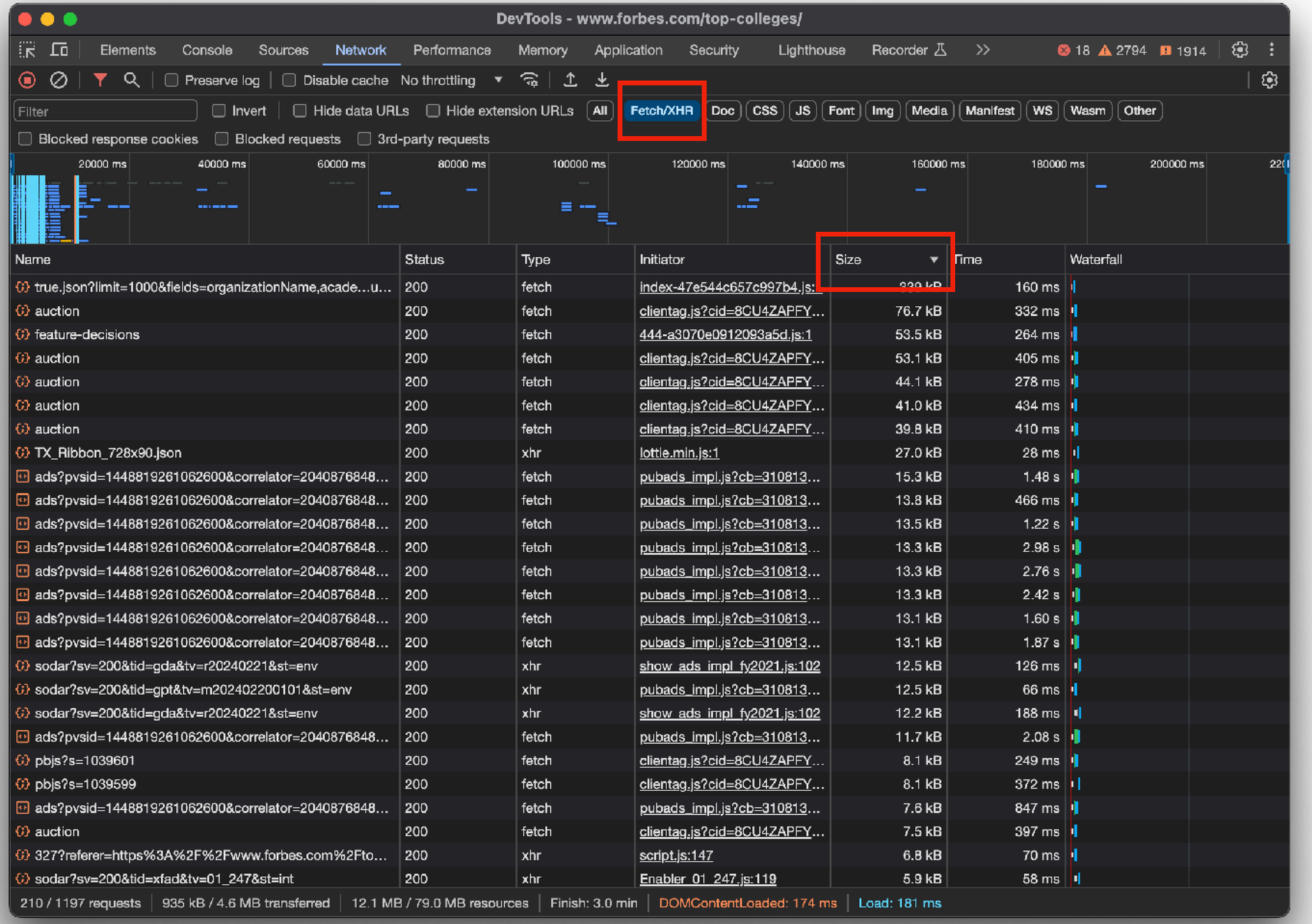
Unofficial API

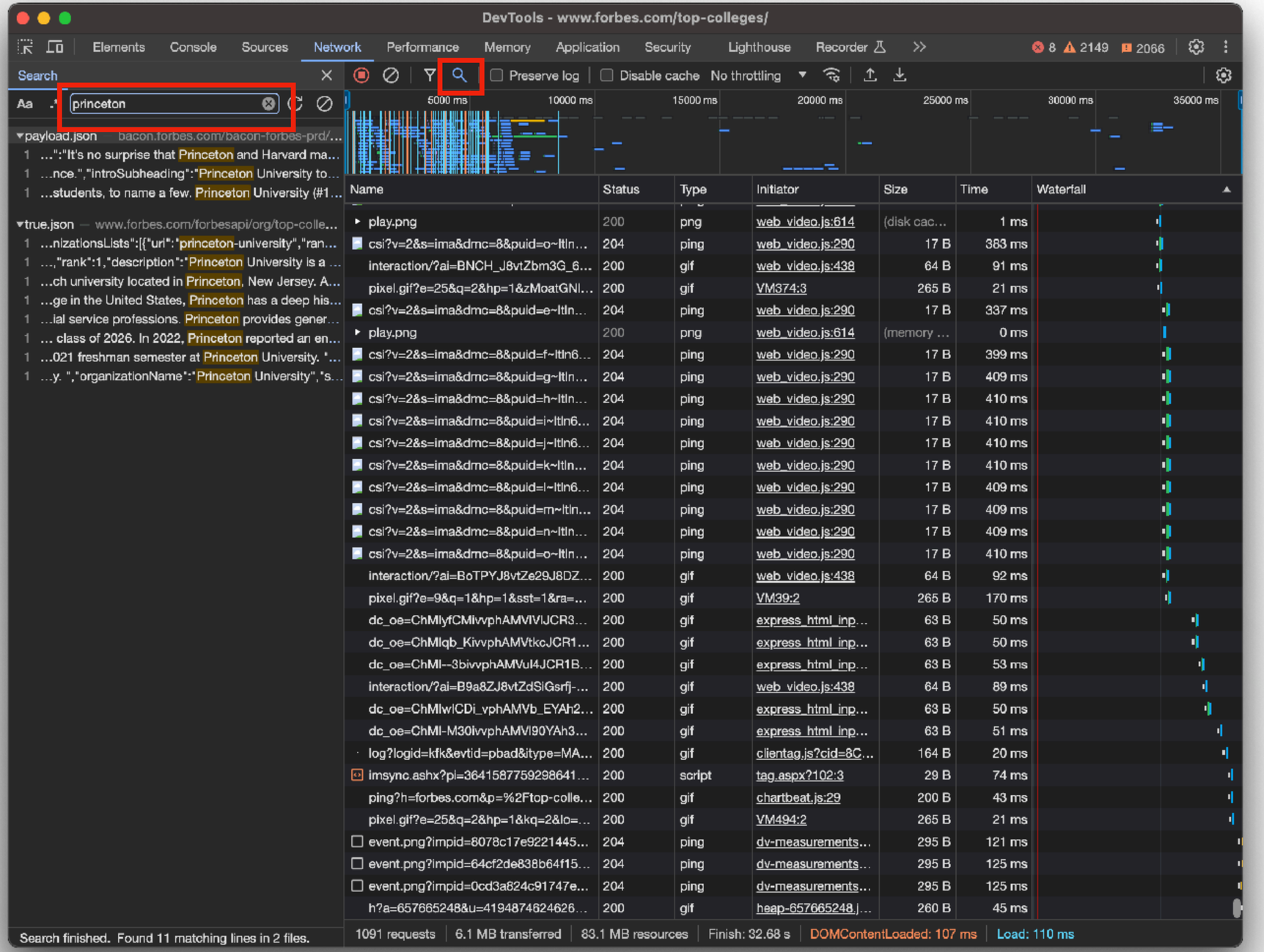
- Most websites that dynamically generate HTML do so from a JSON file.
- If you can find that JSON file you can work with it directly, making your life much easier.
- Find it with network pane in the browser developer tools.
- It’s often obvious, but even when not it’s worth spending 30 minutes on because it’ll easily save that much time.
- (I think this Forbes site is an outlier; most of the time it will be a bit easier.)
- Another useful resource is <http://inspectelement.org/apis.html>

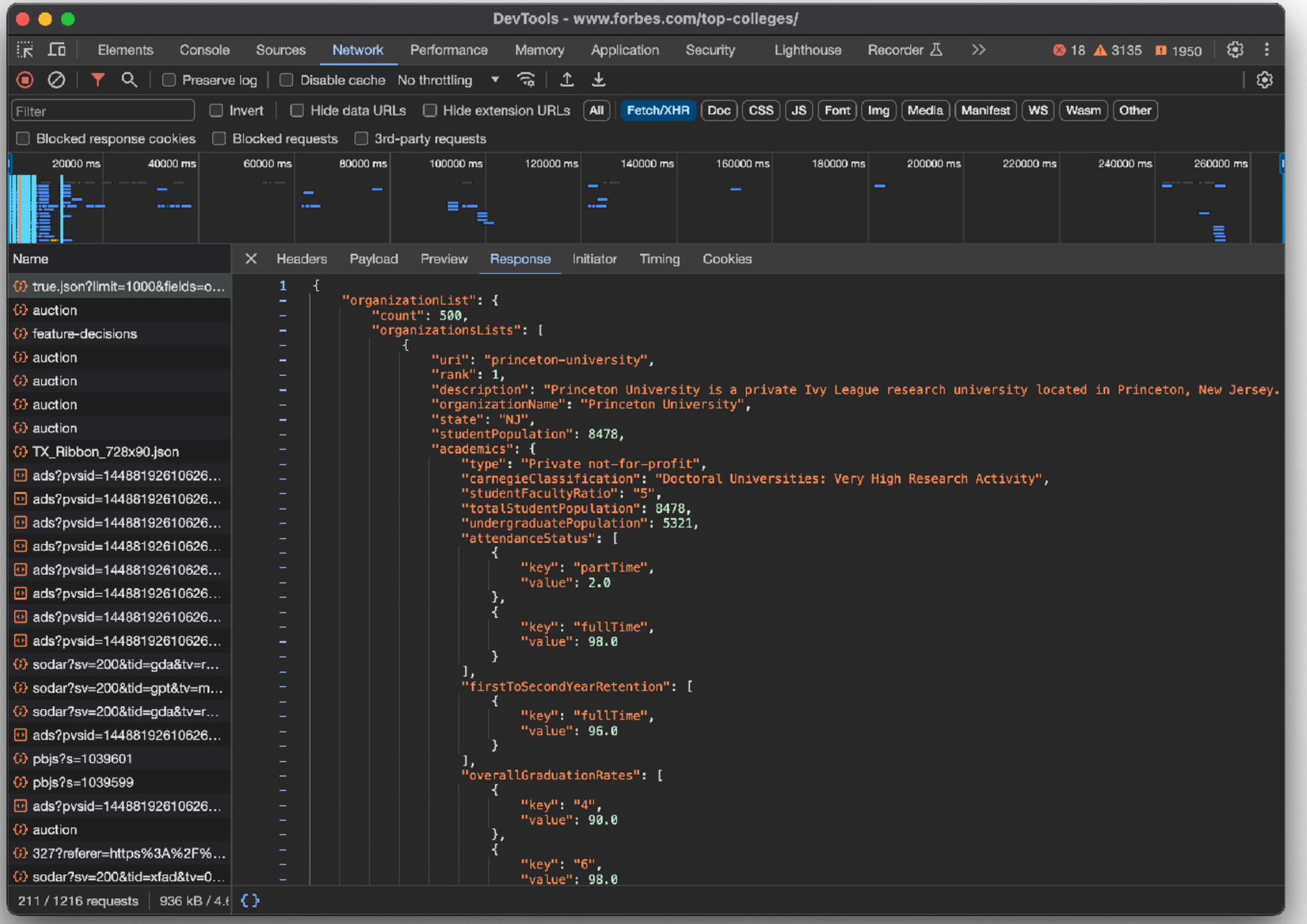

DevTools - www.forbes.com/top-colleges/
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder
18 2794 1914
Preserve log Disable cache No throttling
Filter Invert Hide data URLs Hide extension URLs All Fetch/XHR Doc CSS JS Font Img Media Manifest WS Wasm Other
Blocked response cookies Blocked requests 3rd-party requests
10000 ms 20000 ms 30000 ms 40000 ms 50000 ms 60000 ms 70000 ms 80000 ms 90000 ms 100000 ms 110000 ms 120000 ms 130000 ms
Name Status Type Initiator Size Time Waterfall
top-colleges/ 200 document Other 15.1 kB 61 ms
EuclidCircularB-Regular.otf 200 font top-colleges/:0 (memory cache) 0 ms
webpack-a38bb8d1314bd7c5.js 200 script top-colleges/:16 (memory cache) 0 ms
framework-28607a37aa503929.js 200 script top-colleges/:16 (memory cache) 0 ms
main-8497fe32b4f86329.js 200 script top-colleges/:16 (memory cache) 0 ms
_app-9cb90d3c1257104f.js 200 script top-colleges/:16 (memory cache) 0 ms
a29ae703-1da0cb3098dbace9.js 200 script top-colleges/:16 (memory cache) 0 ms
8f49bc86-b0d13627c893cff0.js 200 script top-colleges/:16 (memory cache) 0 ms
435d5eac-7d33fa1861469ac8.js 200 script top-colleges/:16 (memory cache) 0 ms
444-a3070e0912093a5d.js 200 script top-colleges/:16 (memory cache) 0 ms
471-0b714a2c72d220d7.js 200 script top-colleges/:16 (memory cache) 0 ms
453-b4f3f49bf18fe894.js 200 script top-colleges/:16 (memory cache) 0 ms
index-47e544c657c997b4.js 200 script top-colleges/:16 (memory cache) 0 ms
_buildManifest.js 200 script top-colleges/:16 (memory cache) 0 ms
_ssgManifest.js 200 script top-colleges/:16 (memory cache) 0 ms
EuclidCircularB-Medium.otf 200 font 9df10b73b435f3ba.css (memory cache) 0 ms
EuclidCircularB-Bold.otf 200 font 9df10b73b435f3ba.css (memory cache) 1 ms
work_sans_600_latin.woff2 200 font 9df10b73b435f3ba.css (memory cache) 0 ms
work_sans_400_latin.woff2 200 font 9df10b73b435f3ba.css (memory cache) 0 ms
EuclidCircularB-Semibold.otf 200 font 9df10b73b435f3ba.css (memory cache) 0 ms
EuclidSquare-Regular.otf 200 font 9df10b73b435f3ba.css (memory cache) 0 ms
EuclidSquare-Semibold.otf 200 font 9df10b73b435f3ba.css (memory cache) 0 ms
true.json?limit=1000&fields=organizationName,acade...u... 200 fetch index-47e544c657c997b4.js:1 339 kB 160 ms
payload.json 200 fetch index-47e544c657c997b4.js:1 (disk cache) 4 ms
595.011e0b55f2cd4dee.js 200 script webpack-a38bb8d1314bd7... (disk cache) 5 ms
account 401 fetch 444-a3070e0912093a5d.js:1 1.9 kB 90 ms
1195 requests 4.6 MB transferred 79.0 MB resources Finish: 2.1 min DOMContentLoaded: 174 ms Load: 181 ms

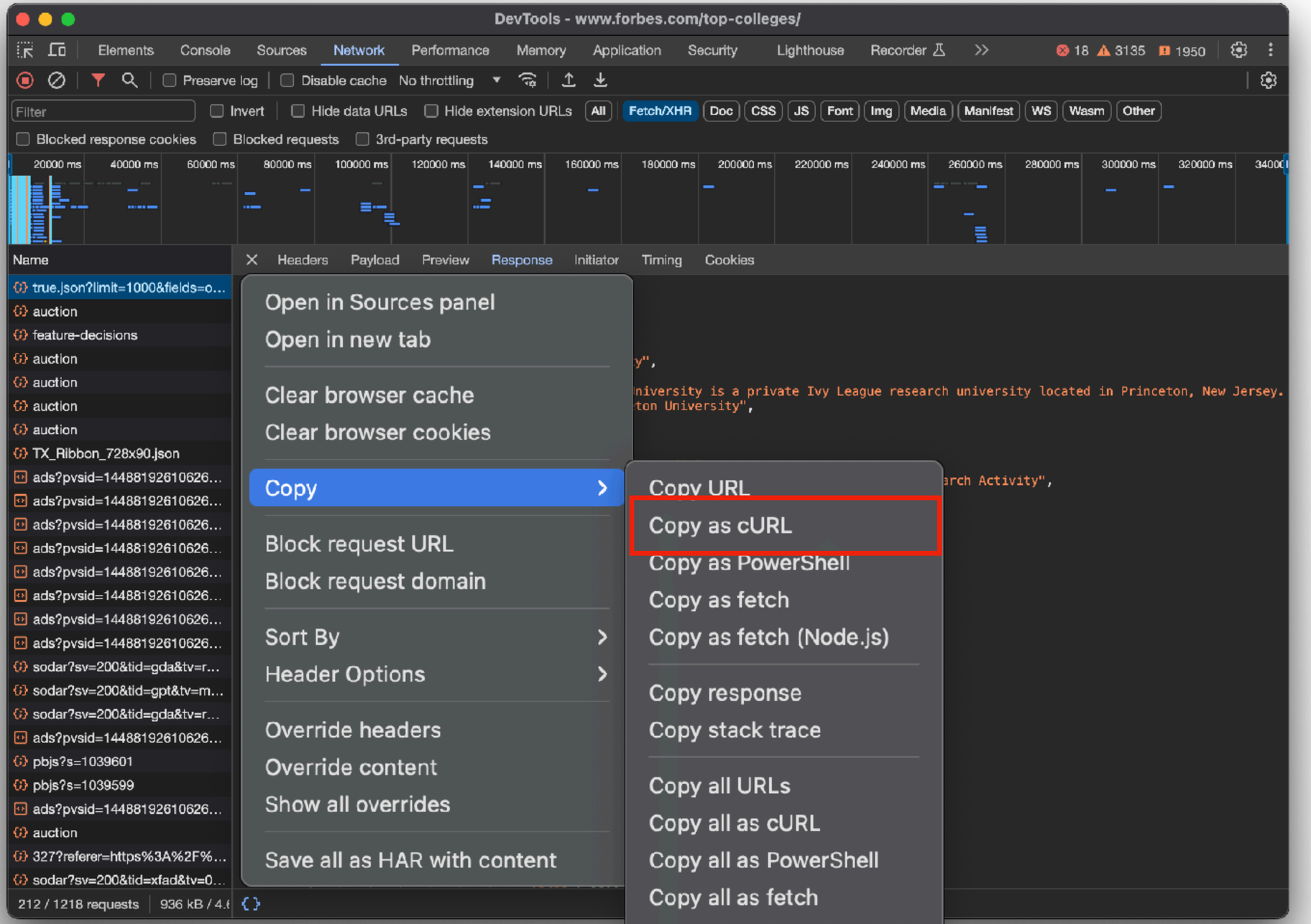
Two useful heuristics to start with

- Filter to “Fetch/XHR” + Sort by size (decreasing)
- Use search to find text that you know must be in the data


DevTools - www.forbes.com/top-colleges/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
18
2794
1914
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
Hide extension URLs
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Blocked response cookies
Blocked requests
3rd-party requests
20000 ms
40000 ms
60000 ms
80000 ms
100000 ms
120000 ms
140000 ms
160000 ms
180000 ms
200000 ms
Name
Status
Type
Initiator
Size
Time
Waterfall
true.json?limit=1000&fields=organizationName,acade...u... 200 fetch index-47e544c657c997b4.js: 160 ms
auction 200 fetch clientag.js?cid=8CU4ZAPFY... 76.7 kB 332 ms
feature-decisions 200 fetch 444-a3070e0912093a5d.js:1 53.5 kB 264 ms
auction 200 fetch clientag.js?cid=8CU4ZAPFY... 53.1 kB 405 ms
auction 200 fetch clientag.js?cid=8CU4ZAPFY... 44.1 kB 278 ms
auction 200 fetch clientag.js?cid=8CU4ZAPFY... 41.0 kB 434 ms
auction 200 fetch clientag.js?cid=8CU4ZAPFY... 39.8 kB 410 ms
TX_Ribbon_728x90.json 200 xhr lottie.min.js:1 27.0 kB 28 ms
ads?pvsid=1448819261062600&correlator=2040876848... 200 fetch pubads_impl.js?cb=310813... 15.3 kB 1.48 s
ads?pvsid=1448819261062600&correlator=2040876848... 200 fetch pubads_impl.js?cb=310813... 13.8 kB 466 ms
ads?pvsid=1448819261062600&correlator=2040876848... 200 fetch pubads_impl.js?cb=310813... 13.5 kB 1.22 s
ads?pvsid=1448819261062600&correlator=2040876848... 200 fetch pubads_impl.js?cb=310813... 13.3 kB 2.98 s
ads?pvsid=1448819261062600&correlator=2040876848... 200 fetch pubads_impl.js?cb=310813... 13.3 kB 2.76 s
ads?pvsid=1448819261062600&correlator=2040876848... 200 fetch pubads_impl.js?cb=310813... 13.3 kB 2.42 s
ads?pvsid=1448819261062600&correlator=2040876848... 200 fetch pubads_impl.js?cb=310813... 13.1 kB 1.60 s
ads?pvsid=1448819261062600&correlator=2040876848... 200 fetch pubads_impl.js?cb=310813... 13.1 kB 1.87 s
sodar?sv=200&tid=gda&tv=r20240221&st=env 200 xhr show_ads_impl_fy2021.js:102 12.5 kB 126 ms
sodar?sv=200&tid=gpt&tv=m202402200101&st=env 200 xhr pubads_impl.js?cb=310813... 12.5 kB 66 ms
sodar?sv=200&tid=gda&tv=r20240221&st=env 200 xhr show_ads_impl_fy2021.js:102 12.2 kB 188 ms
ads?pvsid=1448819261062600&correlator=2040876848... 200 fetch pubads_impl.js?cb=310813... 11.7 kB 2.08 s
pbjs?s=1039601 200 fetch clientag.js?cid=8CU4ZAPFY... 8.1 kB 249 ms
pbjs?s=1039599 200 fetch clientag.js?cid=8CU4ZAPFY... 8.1 kB 372 ms
ads?pvsid=1448819261062600&correlator=2040876848... 200 fetch pubads_impl.js?cb=310813... 7.6 kB 847 ms
auction 200 fetch clientag.js?cid=8CU4ZAPFY... 7.5 kB 397 ms
327?referer=https%3A%2F%2Fwww.forbes.com%2Fto... 200 xhr script.js:147 6.8 kB 70 ms
sodar?sv=200&tid=xfad&tv=01_247&st=int 200 xhr Enabler_01_247.js:119 5.9 kB 58 ms
210 / 1197 requests
935 kB / 4.6 MB transferred
12.1 MB / 79.0 MB resources
Finish: 3.0 min
DOMContentLoaded: 174 ms
Load: 181 ms


DevTools - www.forbes.com/top-colleges/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Search
princeton
Preserve log
Disable cache
No throttling
payload.json
true.json
Name
Status
Type
Initiator
Size
Time
Waterfall
Search finished. Found 11 matching lines in 2 files.
1091 requests
6.1 MB transferred
83.1 MB resources
Finish: 32.68 s
DOMContentLoaded: 107 ms
Load: 110 ms


DevTools - www.forbes.com/top-colleges/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
18
3135
1950
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
Hide extension URLs
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Blocked response cookies
Blocked requests
3rd-party requests
20000 ms
40000 ms
60000 ms
80000 ms
100000 ms
120000 ms
140000 ms
160000 ms
180000 ms
200000 ms
220000 ms
240000 ms
260000 ms
Name
Headers
Payload
Preview
Response
Initiator
Timing
Cookies
true.json?limit=1000&fields=o...
auction
feature-decisions
auction
auction
auction
auction
TX_Ribbon_728x90.json
ads?pvsid=14488192610626...
ads?pvsid=14488192610626...
ads?pvsid=14488192610626...
ads?pvsid=14488192610626...
ads?pvsid=14488192610626...
ads?pvsid=14488192610626...
ads?pvsid=14488192610626...
ads?pvsid=14488192610626...
sodar?sv=200&tid=gda&tv=r...
sodar?sv=200&tid=gpt&tv=m...
sodar?sv=200&tid=gda&tv=r...
ads?pvsid=14488192610626...
pbjs?s=1039601
pbjs?s=1039599
ads?pvsid=14488192610626...
auction
327?referer=https%3A%2F%...
sodar?sv=200&tid=xfad&tv=0...
211 / 1216 requests
936 kB / 4.
{
"organizationList": {
"count": 500,
"organizationsLists": [
{
"uri": "princeton-university",
"rank": 1,
"description": "Princeton University is a private Ivy League research university located in Princeton, New Jersey.
"organizationName": "Princeton University",
"state": "NJ",
"studentPopulation": 8478,
"academics": {
"type": "Private not-for-profit",
"carnegieClassification": "Doctoral Universities: Very High Research Activity",
"studentFacultyRatio": "5",
"totalStudentPopulation": 8478,
"undergraduatePopulation": 5321,
"attendanceStatus": [
{
"key": "partTime",
"value": 2.0
},
{
"key": "fullTime",
"value": 98.0
}
],
"firstToSecondYearRetention": [
{
"key": "fullTime",
"value": 96.0
}
],
"overallGraduationRates": [
{
"key": "4",
"value": 90.0
},
{
"key": "6",
"value": 98.0


DevTools - www.forbes.com/top-colleges/
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
Hide extension URLs
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Blocked response cookies
Blocked requests
3rd-party requests
Name
Headers
Payload
Preview
Response
Initiator
Timing
Cookies
Open in Sources panel
Open in new tab
Clear browser cache
Clear browser cookies
Copy
Block request URL
Block request domain
Sort By
Header Options
Override headers
Override content
Show all overrides
Save all as HAR with content
Copy URL
Copy as cURL
Copy as PowerShell
Copy as fetch
Copy as fetch (Node.js)
Copy response
Copy stack trace
Copy all URLs
Copy all as cURL
Copy all as PowerShell
Copy all as fetch
Copy all as fetch (Node.js)
Copy all as HAR
212 / 1218 requests

This gives a giant curl call

```
curl 'https://www.forbes.com/forbesapi/org/top-colleges/2023/position/true.json?
limit=1000&fields=organizationName,academics,state,financialAid,rank,medianBaseSalary,campusSetting,studentPopulation,squareImage,uri,description,grade' \
-X 'GET' \
-H 'Accept: */*' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Cookie: _ga_DLD85VJ5QY=GS1.1.1706542437.3.1.1706542642.60.0.0; VWO=27.700;
_ketch_consent_v1_=eyJiZWhhdmlvcmFsX2FkdmVydGlzaW5nIjp7InN0YXR1cyI6ImdyYW50ZWQiLCJjYW5vbmljYWxQdXJwb3NlcyI6WyJiZWhhdmlvcmFsX2FkdmVydGlzaW5nIl19LCJhbmFseXRpY3MiOnsic3RhdHVzIjoiZ3JhbnRlZCIsImNhbm9uaWNhbFB1cnBvc2VzIjpbImFuYWx5dGljcyJdfSwiZnVuY3Rpb25hbCI6eyJzdGF0dXMiOiJncmFudGVkIiwiY2Fub25pY2FsUHVycG9zZXMiOlsicHJvZF9lbmhhbmNlbWVudCIsInBlcnNvbmFsaXphdGlvbiJdfSwicmVxdWlyZWQiOnsic3RhdHVzIjoiZ3JhbnRlZCIsImNhbm9uaWNhbFB1cnBvc2VzIjpbImVzc2VudGlhbF9zZXJ2aWNlcyJdfX0%3D;
_swb_consent_=eyJlbnZpcm9ubWVudENvZGUiOiJwcm9kdWN0aW9uIiwiaWRlbnRpdGllcyI6eyJfZ29vZ2xlQW5hbHl0aWNzQ2xpZW50SUQiOiJHQTEuMi4zNTcxMjI4NjguMTcwNTI1OTE4OCIsInN3Yl93ZWJzaXRlX3NtYXJ0X3RhZyI6IjU3YzY4NDNhLWQ4ZTktNDBjMy05YmMxLThjYWNkZWU3ZDE2OSJ9LCJqdXJpc2RpY3Rpb25Db2RlIjoidXNfZ2VuZXJhbCIsInByb3BlcnR5Q29kZSI6IndlYnNpdGVfc21hcnRfdGFnIiwicHVycG9zZXMiOnsiYW5hbHl0aWNzIjp7ImFsbG93ZWQiOiJ0cnVlIiwibGVnYWxCYXNpc0NvZGUiOiJkaXNjbG9zdXJlIn0sImJlaGF2aW9yYWxfYWR2ZXJ0aXNpbmciOnsiYWxsb3dlZCI6InRydWUiLCJsZWdhbEJhc2lzQ29kZSI6ImRpc2Nsb3N1cmUifSwiZnVuY3Rpb25hbCI6eyJhbGxvd2VkIjoidHJ1ZSIsImxlZ2FsQmFzaXNDb2RlIjoiZGlzY2xvc3VyZSJ9fSwiY29sbGVjdGVkQXQiOjE3MDY1NDI1ODl9; _fbp=fb.1.1706121217898.164324328; _ga=GA1.2.357122868.1705259188;
_gcl_au=1.1.1366914331.1705344922; _gid=GA1.2.412386523.1706542438; us_privacy=1---; usprivacy=1---; AWSALB=tnC2CAJGdokK0IsOKzBarRIkeT8r4YS5/
wR6+n+A2VQX2Gfxa3lgP2KrEv6bGPRZyGIhGFvSCrNCftxLR8EXyMI3eDVjI5kBMLcIv9BthsZsyM9Vp0eTKR5yeR/H; AWSALBCORS=tnC2CAJGdokK0IsOKzBarRIkeT8r4YS5/
wR6+n+A2VQX2Gfxa3lgP2KrEv6bGPRZyGIhGFvSCrNCftxLR8EXyMI3eDVjI5kBMLcIv9BthsZsyM9Vp0eTKR5yeR/H; BCSessionID=2ae07b14-70be-40b1-9e90-ed5d91b1be4f; ki_r=;
ki_t=1706121218201%3B1706542438105%3B1706542587089%3B2%3B12; client_id=14d486f22e5f65b3107348cd0ffeaa50923; lux_uid=170654243670875530; AMP_TOKEN=%24NOT_FOUND; _swb=57c6843a-
d8e9-40c3-9bc1-8cacdee7d169; rbzid=CXwv4IPKOa5WMsN9vGfFx/FfHDJXwSL2pHJ/swUFnuvHRpgd1qGWgsSHDvkfxbe6A3W7IDkaJbmIeiPvd/wC7NLDIMS6nZ4N6B7HWA1lvWFqWciQ/+GZ1Bm7YCvrGGhXO59tvv6mZYAXbx6MHiL6+/
BVKFl8m2Z5gx0M2r0M2j9D/QiW7bH4TR8S/oJmWPQi7TJT5F+3SMGf6SjtfG64f74hLDwf+zEy5y95JETCHlw7Og8eg5QO5AALhKK+rhPV4z7F29XqrM2aGXgKHz//+Q==; rbzsessionid=da3b925ab2cf238acdfa18f6e699045b;
_ga_HY3LZWHH6W=GS1.1.1705344921.1.1.1705345703.0.0.0; _uetvid=a2f10940b3d711ee8c1881f6a3138e42; amp_9c5697=N1292876525...1hk77ksom.1hk77kvfo.2.2.4; notice_behavior=implied,us; fadve2etidvcnt=2;
_clck=10sfr1j%7C2%7Cfif%7C0%7C1474; fadve2etid=N1292876525; fadvfpuid=FA77ce891e24882d9a03e9d2bc5bf16cf3; _ga_0Y2Y7WWQP1=GS1.1.1705259187.1.1.1705259215.0.0.0;
_ga_JFZ3B3QM86=GS1.1.1705259187.1.1.1705259215.0.0.0; cmapi_cookie_privacy=permit 1,2,3; fadvuke2etid=N255829421; blaize_session=d72df655-7d08-4673-bbd3-beadbe316b40;
blaize_tracking_id=418dd2e2-817f-48c7-9792-e36d6ce05bf9' \
-H 'Sec-Fetch-Dest: empty' \
-H 'Accept-Language: en-GB,en-US;q=0.9,en;q=0.8' \
-H 'Sec-Fetch-Mode: cors' \
-H 'Host: www.forbes.com' \
-H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.2.1 Safari/605.1.15' \
-H 'Referer: https://www.forbes.com/top-colleges/' \
-H 'Accept-Encoding: gzip, deflate, br' \
```

And you can translate to R with httr2::curl_translate()

```
request("https://www.forbes.com/forbesapi/org/top-colleges/2023/position/true.json") ▷
  req_method("GET") ▷
  req_url_query(
    limit = "1000",
    fields = "organizationName,academics,state,financialAid,rank,medianBaseSalary,campusSetting,studentPopulation,squareImage,uri,description,grade",
  ) ▷
  req_headers(
    Accept = "*/*",
    Cookie = "_ga_DLD85VJ5QY=GS1.1.1706542437.3.1.1706542642.60.0.0; VWO=27.700;
_ketch_consent_v1_=eyJiZWhhdmlvcmFsX2FkdmVydGlzaW5nIjp7InN0YXR1cyI6ImdyYW50ZWQiLCJjYW5vbmljYWxQdXJwb3NlcyI6WyJiZWhhdmlvcmFsX2FkdmVydGlzaW5nIl19LCJhbmFseXRpY3MiOnsic3RhdHVzIjoiZ3JhbnRlZCIsImNhbm9uaWNhbFB1cnBvc2VzIjpbImFuYWx5dGljcyJdfSwiZnVuY3Rpb25hbCI6eyJzdGF0dXMiOiJncmFudGVkIiwiY2Fub25pY2FsUHVycG9zZXMiOlsicHJvZF9lbmhhbmNlbWVudCIsInBlcnNvbmFsaXphdGlvbiJdfSwicmVxdWlyZWQiOnsic3RhdHVzIjoiZ3JhbnRlZCIsImNhbm9uaWNhbFB1cnBvc2VzIjpbImVzc2VudGlhbF9zZXJ2aWNlcyJdfX0%3D;
_swb_consent_=eyJlbnZpcm9ubWVudENvZGUiOiJwcm9kdWN0aW9uIiwiaWRlbnRpdGllcyI6eyJfZ29vZ2xlQW5hbHl0aWNzQ2xpZW50SUQiOiJHQTEuMi4zNTcxMjI4NjguMTcwNTI1OTE4OCIsInN3Yl93ZWJzaXRlX3NtYXJ0X3RhZyI6IjU3YzY4NDNhLWQ4ZTktNDBjMy05YmMxLThjYWNkZWU3ZDE2OSJ9LCJqdXJpc2RpY3Rpb25Db2RlIjoidXNfZ2VuZXJhbCIsInByb3BlcnR5Q29kZSI6IndlYnNpdGVfc21hcnRfdGFnIiwicHVycG9zZXMiOnsiYW5hbHl0aWNzIjp7ImFsbG93ZWQiOiJ0cnVlIiwibGVnYWxCYXNpc0NvZGUiOiJkaXNjbG9zdXJlIn0sImJlaGF2aW9yYWxfYWR2ZXJ0aXNpbmciOnsiYWxsb3dlZCI6InRydWUiLCJsZWdhbEJhc2lzQ29kZSI6ImRpc2Nsb3N1cmUifSwiZnVuY3Rpb25hbCI6eyJhbGxvd2VkIjoidHJ1ZSIsImxlZ2FsQmFzaXNDb2RlIjoiZGlzY2xvc3VyZSJ9LCJyZXF1aXJlZCI6eyJhbGxvd2VkIjoidHJ1ZSIsImxlZ2FsQmFzaXNDb2RlIjoiZGlzY2xvc3VyZSJ9fSwiY29sbGVjdGVkQXQiOjE3MDY1NDI1ODl9; _fbp=fb.1.1706121217898.164324328; _ga=GA1.2.357122868.1705259188;
_gcl_au=1.1.1366914331.1705344922; _gid=GA1.2.412386523.1706542438; us_privacy=1---; usprivacy=1---; AWSALB=tnC2CAJGdokK0IsOKzBarRIkeT8r4YS5/
wR6+n+A2VQX2Gfxa3lgP2KrEv6bGPRZyGIhGFvSCrNCftxLR8EXyMI3eDVjI5kBMLcIv9BthsZsyM9Vp0eTKR5yeR/H; AWSALBCORS=tnC2CAJGdokK0IsOKzBarRIkeT8r4YS5/
wR6+n+A2VQX2Gfxa3lgP2KrEv6bGPRZyGIhGFvSCrNCftxLR8EXyMI3eDVjI5kBMLcIv9BthsZsyM9Vp0eTKR5yeR/H; BCSessionID=2ae07b14-70be-40b1-9e90-ed5d91b1be4f; ki_r=;
ki_t=1706121218201%3B1706542438105%3B1706542587089%3B2%3B12; client_id=14d486f22e5f65b3107348cd0ffeaa50923; lux_uid=170654243670875530; AMP_TOKEN=%24NOT_FOUND; _swb=57c6843a-
d8e9-40c3-9bc1-8cacdee7d169; rbzid=CXwv4IPKOa5WMsN9vGfFx/FfHDJXwSL2pHJ/swUFnuvHRpgd1qGWgsSHDvkfxbe6A3W7IDkaJbmIeiPvd/wC7NLDIMS6nZ4N6B7HWA1lvWFqWciQ/+GZ1Bm7YCvrGGhXO59tvv6mZYAXbx6MHiL6+/
BVKFl8m2Z5gx0M2r0M2j9D/QiW7bH4TR8S/oJmWPQi7TJT5F+3SMGf6SjtfG64f74hLDwf+zEy5y95JETCHlw7Og8eg5QO5AALhKK+rhPV4z7F29XqrM2aGXgKHz//+Q==; rbzsessionid=da3b925ab2cf238acdfa18f6e699045b;
_ga_HY3LZWHH6W=GS1.1.1705344921.1.1.1705345703.0.0.0; _uetvid=a2f10940b3d711ee8c1881f6a3138e42; amp_9c5697=N1292876525...1hk77ksom.1hk77kvfo.2.2.4; notice_behavior=implied,us; fadve2etidvcnt=2;
_clck=10sfr1j%7C2%7Cfif%7C0%7C1474; fadve2etid=N1292876525; fadvfpuid=FA77ce891e24882d9a03e9d2bc5bf16cf3; _ga_0Y2Y7WWQP1=GS1.1.1705259187.1.1.1705259215.0.0.0;
_ga_JFZ3B3QM86=GS1.1.1705259187.1.1.1705259215.0.0.0; cmapi_cookie_privacy=permit 1,2,3; fadvuke2etid=N255829421; blaize_session=d72df655-7d08-4673-bbd3-beadbe316b40;
blaize_tracking_id=418dd2e2-817f-48c7-9792-e36d6ce05bf9",
    `Accept-Language` = "en-GB,en-US;q=0.9,en;q=0.8",
    Host = "www.forbes.com",
    `User-Agent` = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/17.2.1 Safari/605.1.15",
    `Accept-Encoding` = "gzip, deflate, br",
  ) ▷
```

Then simplify to the essentials

```
url ← "https://www.forbes.com/forbesapi/org/top-colleges/2023/
position/true.json"

req ← request(url) ▷
  req_url_query(limit = "1000") ▷
  req_perform()
```

Demo

forbes-api.R

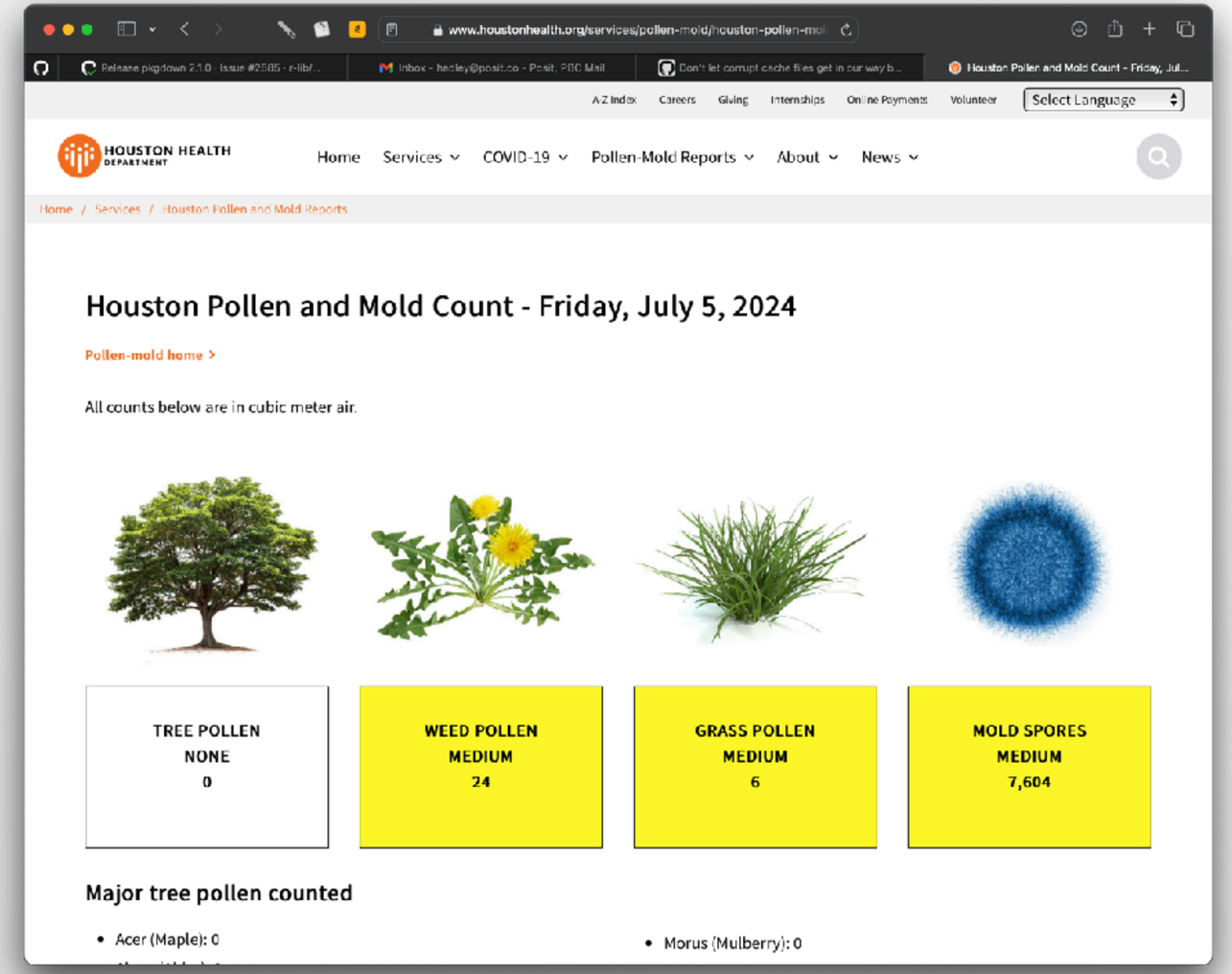
If the data is stored as JSON in the HTML

This is a pain, but is fortunately relatively rare.

I have had some luck in the past with using the V8 R package to run the javascript code and then extract the JSON object back into R.

LLMs

Houston pollen and mold counts



https://www.houstonhealth.org/services/pollen-mold

The following ranges are based on the data reported by all counting stations for weeds, grasses, trees and mold according to the National Allergy Bureau.

Weed pollen is measured per cubic meter of air.

- 1-9 pollen per cubic meter of air or less = Low
- 10 to 49 = Medium
- 50 to 499 = Heavy
- Greater than 500 = Extremely Heavy

Grass pollen is measured per cubic meter of air.

- 1-4 pollen per cubic meter of air or less = Low
- 5 to 19 = Medium
- 20 to 199 = Heavy
- Greater than 200 = Extremely Heavy

Tree pollen is measured during the tree pollen season of mid-January through mid-April. Cedar Elm tree pollen is measured in September and October. All tree pollen types are reported by HDHHS laboratory.

- 1 to 14 pollen per cubic meter of air = Low
- 15 to 89 = Medium
- 90 to 1499 = Heavy
- Greater than 1500 = Extremely Heavy

Mold spores are measured per cubic meter of air.

- 1 to 6499 = Low
- 6500 to 12999 = Medium
- 13000 to 49999 = Heavy
- Over 50000 = Extremely Heavy

The HHD laboratory reports all major types of mold spore irritants.

Prompt

Take the following data and covert it in to a call to tibble::tribble, like the following two example rows for pollen

```
thresholds <- tibble(
  ~category, ~low, ~high, ~value,
  "weed", 1, 9, "low",
  "weed", 500, Inf, "extremely high"
)
```

Here's the data ...