Web Scraping Overview

```
from requests import get
url = 'https://site-to-scrape.glitch.me/'
headers = {'User-Agent': 'Codeup Data Science Student'}
response = get(url, headers=headers)
print(response.text)
```

- Open `view-source:https://site-to-scrape.glitch.me/` or right click "View Page Source"
- Compare this text to the `response.text` from the script above.
- Congratulations! You Have a big honkin' string of HTML with all the characters!
- Browsers render HTML, but we'll need to parse that entire string to search for content.

Front-End Web Orientation

- HTML is a tree structure, like directories. Elements contain content or other elements.
- CSS is a language for selecting HTML elements then styling and laying them out.
- JS is the client side programming language of the browser. Lots of content is dynamically generated using JS. What `request` gets is the same as putting `view-source:` in front of a URL.
- Most common CSS Selectors:
    - Element selector like `p`, `h1`, `h2`, `a`, etc...
    - Class selector: Classes are for grouping. We can apply classes to multiple elements.
        - Given html of `<p class="stuff">...</p>`, `.stuff` selects all elements w/ that class.
    - Id selector. HTML id attributes are a unique for an element, like a driver's license number.
        - Given html of `<p id="content">...</p>`, `#content` selects that one element.
    - https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors for more/reference.

Best Practices

- **Scrape ethically**. When in doubt, ask yourself "What would Salas say? What would Zach do?"
- **Build a local cache of your response results**. Because scraping involves sending requests, time, servers, and bandwidth, it's important to load results if you already have them and only send requests to get fresh data. Otherwise, you risk:
    - Getting yourself or your company banned or blacklisted.
    - Losing lots of time waiting for round trips over the internet.