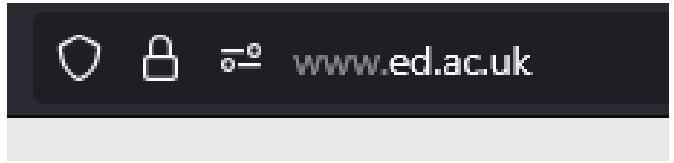
UNDERSTANDING THE WEB

To scrape, it’s good to know a bit about how the web works behind the scene.



When you open a website in your browser, you are actually making **HTTP(S) requests**. If you hit **F12** in your browser and go to the Network tab, you can see them.

Method	Domain	File	Initiator	Type
GET	www.ed.ac.uk	/	docum...	html
GET	www.ed.ac.uk	css_eLy1dYUOnHhuC5hnvhXgiNJq_2Xf4Flx2dzPNfqCZeQ.css?delta=0&language=en&theme=wpp_theme&include=eJxljeEKwyAMhF9IK6zb80ismQpplcSs-PYrHWyFwXE_7j7uCEbV7mORpb6Qh-uZE	stylesh...	css
GET	www.ed.ac.uk	css_I62W-aN2Ha9CRJ1UzWtVAZ87Frg94yn1adFTUfPHwmo.css?delta=1&language=en&theme=wpp_theme&include=eJxljeEKwyAMhF9IK6zb80ismQpplcSs-PYrHWyFwXE_7j7uCEbV7mORpb6Qh-u	stylesh...	css
GET	www.ed.ac.uk	js_3tFWa_jBT8TTPeNeylhqlonTffHgPGzttCr83gV1IIA.js?scope=footer&delta=0&language=en&theme=wpp_theme&include=eJwrLyiIL8lIzU3VT8_JT0rM0SmHC-SkpicmVwIA30ENAw	script	js
GET	www.ed.ac.uk	edgel.bundle.min.js?t447iv	script	js
GET	www.ed.ac.uk	js_Dd-2nskI5K-fWlRRkT8SXx-IWwDs3r7nNJUixzS9oKc.js?scope=footer&delta=2&language=en&theme=wpp_theme&include=eJwrLyiIL8lIzU3VT8_JT0rM0SmHC-SkpicmVwIA30ENAw	script	js
GET	www.ed.ac.uk	Study with us.png.webp?itok=XizS81fO	images...	webp
GET	www.ed.ac.uk	pg virtual open days.jpg.webp?itok=VHl0HY2a	images...	webp
GET	www.ed.ac.uk	imapct hana edinburgh award edweb.jpg.webp?itok=DcUnAx65	images...	webp
GET	www.ed.ac.uk	edweb-impact-standing-firm-power-pride.jpg.webp?itok=0lEh25Km	images...	webp
GET	www.ed.ac.uk	Latest news.jpg.webp?itok=jzxoBBIz	images...	webp
GET	www.ed.ac.uk	Art Makes Children Powerful - Bob and Roberta Smith - crop.JPG.webp?itok=IqWjmxWz	images...	webp
GET	www.ed.ac.uk	Edinburgh Global.jpg.webp?itok=J6bpw9Kz	images...	webp

As the name suggests, requests are sent to the server to retrieve data back images, scripts, …

Opening a single website lead to a **lot** of requests.

UNDERSTANDING THE WEB

There are many **request methods** (GET, POST, …).

We **will mostly care about GET.**

Method	Domain
GET	www.ed.ac.uk
GET	www.ed.ac.uk
GET	www.ed.ac.uk
GET	www.ed.ac.uk
GET	www.ed.ac.uk

```
HTTP/2 200
cache-control: max-age=31622400
content-encoding: gzip
content-type: application/x-javascript
etag: W/"68ee1724-2c76"
expires: Thu, 15 Oct 2026 09:26:00 GMT
last-modified: Tue, 14 Oct 2025 09:25:56 GMT
server: nginx
strict-transport-security: max-age=300
x-pantheon-styx-hostname: styx-fe3fe4-c-5545684fbf-pgrh2
x-styx-req-id: cbff3def-a8df-11f0-9038-b66ddd1a1815
age: 1269723
accept-ranges: bytes
via: 1.1 varnish, 1.1 varnish, 1.1 varnish, 1.1 varnish
date: Wed, 29 Oct 2025 02:08:04 GMT
x-served-by: cache-ams21063-AMS, cache-lcy-eglc8600081-LCY, cache-lcy
x-cache: MISS, HIT, MISS, MISS
x-cache-hits: 0, 24129, 0, 0
x-timer: S1761703685.805060,VS0,VE8
vary: Accept-Encoding
content-length: 3747
X-Firefox-Spdy: h2
```

Server sends back a **response**, gives requested data (e.g., image) and metadata (e.g. type of content) called **headers**.

As a scraper we mostly won't care about the headers, except **status code.**

Status 200

UNDERSTANDING THE WEB

- *1xx informational response* – the request was received, continuing process
- *2xx successful* – the request was successfully received, understood, and accepted
- *3xx redirection* – further action needs to be taken in order to complete the request
- *4xx client error* – the request contains bad syntax or cannot be fulfilled
- *5xx server error* – the server failed to fulfil an apparently valid request

https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

If the status code is **200 OK** you're good. If it's **4xx** or **5xx**, it's bad. (404 Not Found is a classic one). That's all we really need to know for now!

UNDERSTANDING THE WEB

Usually, these requests are made for you automatically by browsers (Google Chrome, Firefox, Edge, etc.)

But when we get to scraping, we have to make requests **manually** in our code!

And that's why sometimes you might see people talking about "requests" when they do web scraping.

(Don't worry. People have made this process easy so it's easier than you think.)

HTML BASICS

Typically, a website (intended for human use, not an API) will be a HTML page. If you go to your favorite website and hit Ctrl + U, you'll see its HTML.

```
<header role="banner" class="masthead">
  <div class="container-masthead">
    <div class="row">
      <div class="col-md-7 col-lg-8">
        <a href="https://www.ed.ac.uk/">
          <img class="masthead-logo" src="/themes/upstream/wpp_theme/images
        </a>
      </div>
            <div class="col-md-5 col-lg-4">
        <div class="masthead-text">
          <ul class="list-inline">
            <li class="list-inline-item">
              <a href="https://www.ed.ac.uk/schools-departments">Schools
            </li>
            <li class="list-inline-item">
              <a href="https://www.myed.ed.ac.uk/">MyEd</a>
            </li>
          </ul>
        </div>
```

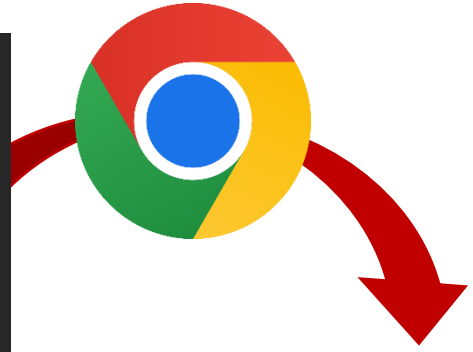
This is what the server actually sends back.

Your browser turns HTML into the webpage that you actually see.

HTML BASICS

Your browser turns HTML into the webpage that you actually see.

```
<p>
<strong>
        This is bold text.
        <font color="red">
                I'm red and bold!
        </font>
</strong>
<font color="red">
        I'm red but not bold
</font>
</p>
```



This is bold text. **I'm red and bold!** I'm red but not bold

(You can try this out. If you open Notepad or TextEdit and type these and save it as "test.html". You can open it with your browser.)

HTML BASICS

HTML stands for **H**yper**T**ext **M**arkup **L**anguage.

Basically, it's a text file that has **HTML tags** in it. They look like this:

```
<strong>some text</strong>
<a href="https://google.com">google</a>
```

Different tags are assigned meaning by your browser. For example, `<strong>` is interpreted as "turning the text in between into boldfaced text". So that will get rendered as **some text**.

HTML BASICS

An HTML document will consist of **nested** HTML tags:

```
<p>
   <strong>
        This is bold text.
        <font color="red">
             I'm red and bold!
        </font>
   </strong>
   <font color="red">
        I'm red but not bold
   </font>
</p>
```

Most tags will come in pairs (`<start tag>` … `</end tag>`) and control what is going on inside them.

But some don't need an end tag, like `<img>`, which is for images.

HTML BASICS

You don’t need to know exactly how to write HTML. But you do need to know the structure.

You already know about **start** and **end tags.** Another thing you should know is that a tag can have **attributes** and **values**

```
<img src="dog.jpg">
```

The src attribute of the `<img>` tag specifies the image location. The browser will make a GET request and (attempt) show this image.

HTML BASICS

Paired tags can also have attribute and values.

```
<div class="test">ABCDEFG</div>
```

Some attributes require a value to work, but a few do not. For example, the `disabled` attribute, which disables a button, does not need an explicitly defined value.

```
<button disabled>Don't click me</button>
```

Don't click me

FINDING INFORMATION FROM TAGS

When we scrape, we will be extracting information by first identifying **which tag(s) contains the information we want**, then **get the content inside** (or get the value of an attribute)**.**

FINDING INFORMATION FROM TAGS

We usually identify these by a combination of **tag name**, **attribute** and **value** that gets us **all and only** the tags surrounding target content.

```
<p>
   <strong>
      This is bold text.
      <font color="red">
            I'm red and bold!
      </font>
   </strong>
   <font color="blue">
      I'm not bold and blue.
   </font>
</p>
```

This is bold text. I'm red and bold! I'm not bold and blue.

If I want extract the text "I'm not bold and blue.", the tag that I need is "the tag *<font>* that has the attribute *color* with the value *blue*"

LET'S TRY IT

Consider HTML below, which produces the table on the right. **Describe the combination of tag name, attribute and value** (Do you need all three?) that will allow you to extract the brands of the cars.

```html
<table>
    <tr>
        <th>Brand</th>
        <th>Color</th>
    </tr>
    <tr>
        <td class="brand">Toyota</td>
        <td>Red</td>
    </tr>
    <tr>
        <td class="brand">Honda</td>
        <td>Black</td>
    </tr>
</table>
```

Brand	Color
Toyota	Red
Honda	Black

LET'S TRY IT

Consider HTML below, which produces the table on the right. **Describe the combination of tag name, attribute and value** (Do you need all three?) that will allow you to extract the brands of the cars.

```html
<table>
    <tr>
        <th>Brand</th>
        <th>Color</th>
    </tr>
    <tr>
        <td class="brand">Toyota</td>
        <td>Red</td>
    </tr>
    <tr>
        <td class="brand">Honda</td>
        <td>Black</td>
    </tr>
</table>
```

"Any <td> tag that has the attribute *class* with the value *brand*"

(We don't actually need to specify which tag it is!)

LET'S TRY IT

Can you describe the combination of tag name, attribute and value that would get us the brands of the cars? Why (not?)

```html
<table>
    <tr>
        <th>Brand</th>
        <th>Color</th>
    </tr>
    <tr>
        <td>Toyota</td>
        <td>Red</td>
    </tr>
    <tr>
        <td>Honda</td>
        <td>Black</td>
    </tr>
</table>
```

LET'S TRY IT

Can you describe the combination of tag name, attribute and value that would get us the brands of the cars? Why (not?)

```
<table>
    <tr>
        <th>Brand</th>
        <th>Color</th>
    </tr>
    <tr>
        <td>Toyota</td>
        <td>Red</td>
    </tr>
    <tr>
        <td>Honda</td>
        <td>Black</td>
    </tr>
</table>
```

You need extra information (e.g., about where the tag is).

It is still possible to scrape! Just a bit more complicated to do.

CSS AND CSS SELECTORS

Here's one good way of describing exactly what we want: the <td> tag **that is the first node (tag) within its parent tags** (in this case, <tr>)**.**

When scraping, what we will be using to precisely specify this kind of thing is **CSS selectors**. They are things like this:

`font[color="red"]` =
<font> tag where the attribute color has the value red

`td:first-child` =
<td> tag that is the first child node of its parent

CSS AND CSS SELECTORS

Originally, CSS selectors were meant to be used in CSS, which is basically code that adjusts the appearances of HTML (like colours and widths of boxes). But we don’t have to care about that today.

Instead, we will just learn how CSS selectors are written and work!

CSS SELECTOR BASICS

CSS selectors are very powerful. Mostly, you will have to know only a few basic things to make use of them.

If you can identify the tag name/attribute/value combo, you can translate it into a CSS selector like so:

```
     tagname[attribute="value"]
->   img[src="google.jpg"]
```

<img> tag where the attribute src has the value “google.jpg”

CSS SELECTOR BASICS

The attributes *class* and *id* are special and you can write a shorthand.

```
a[id="x"] = a#x              = <a id="x">…</a>
a[class="test"] = a.test = <a class="test"></a>
```

You can also just omit the tag name and the CSS selector will target any tag with that class/id.

`#x` = any tag that has the id "x"
`.test` = any tag that has the class "test"

CSS SELECTOR BASICS

Note that an ID is unique to a single HTML node/tag in a document (so most of the time you won't need the tag name at all.)

A class can be applied across elements, and a single tag can have multiple classes (e.g., `<a class="link link-main">link</a>` has two classes, `link` and `link-main`).

CSS SELECTOR BASICS

Finally, we can be a bit more precise and specify **where** the tag is relative to another tag. This is done by adding the tag it is nested in in front.

`strong a =` An `<a>` tag that is inside a `<strong>` tag somewhere.

So it will target

```
<strong><a>here</a></strong>
```

as well as

```
<strong><font color="red"><a>here</a></font></strong>
```

CSS SELECTOR BASICS

We can repeat this process…

```
div strong.test a
```

selects any <a> that is inside with the class *test,* which in turn is inside

CSS SELECTOR BASICS

“But what about things like :first-child? I saw you did that earlier.”

That’s part of something more complicated, which we will not cover today…

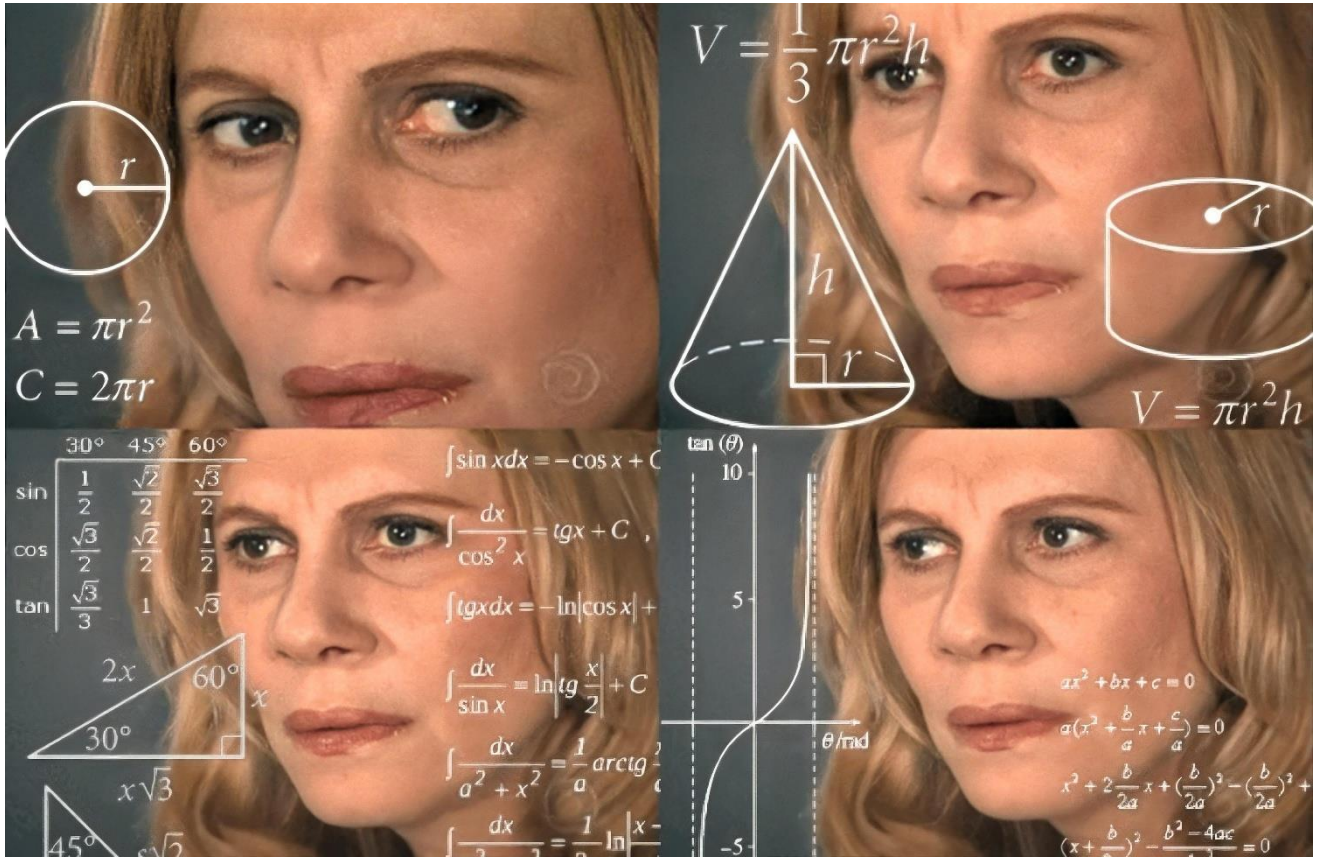
CSS SELECTOR BASICS

In general, CSS selectors can be extremely precise, for example:

```
table#price > tr a.test[href*=“google”]:nth-child(3)
```

The <a> tag that has *class=“test”* and attribute *href* whose value contains the word “google”, and also is the third child of its immediate parent node, and is nested in a <tr> tag, which in turn is an immediate child of the <table> tag that has *id=“price”*’

HUH?


$A = \pi r^2$
$C = 2\pi r$
$V = \frac{1}{3}\pi r^2 h$
$V = \pi r^2 h$
30° 45° 60°
sin
cos
tan
$\int \sin x dx = -\cos x + C$
$\int \frac{dx}{\cos^2 x} = tgx + C$
$\int tgx dx = -\ln|\cos x| +$
$\int \frac{dx}{\sin x} = \ln\left|tg\frac{x}{2}\right| + C$
$\int \frac{dx}{a^2 + x^2} = \frac{1}{a} arctg$
tan (θ)
θ/rad
$ax^2 + bx + c = 0$
$a(x^2 + \frac{b}{a}x + \frac{c}{a}) = 0$
$x^2 + 2\frac{b}{2a}x + (\frac{b}{2a})^2 - (\frac{b}{2a})^2 +$
$(x + \frac{b}{2a})^2 - \frac{b^2 - 4ac}{4a^2} = 0$

SELECTOR GADGET

You can get away with knowing how to write CSS selectors on a surface level because there are tools that can help you!

Selector Gadget is a browser extension that allow you to click on the elements that you want on a page, and it will create a CSS selector for you, and highlight all elements (tags) selected by the selector.

ACTIVITY

1. Uses Chrome, Edge, or any other Chromium-based browser
2. Download and install **Selector Gadget** from the Chrome store. Chrome Web Store
3. Play with it on https://www.connosr.com/. Try to figure out selectors for various pieces of information. E.g. What selector would you get all and only the names of Japanese distilleries?