From Web to Web Scraping

Day 3: 3DC Introduction to Programming

Raphael Yee

How does the web work?

How do we retrieve and send information?

What tools do we use to exchange information?

How do these programs communicate?

How do humans accomplish the tasks above?

What the web pretty much boils down to,
is **an exchange of information.**

Choosing what to do with the information is entirely up to us.

First, we need to figure out how to move this information around.

Push or pull?

Are we retrieving information or are we sending information?

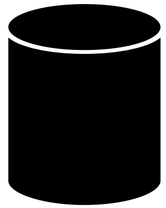
Sending an email

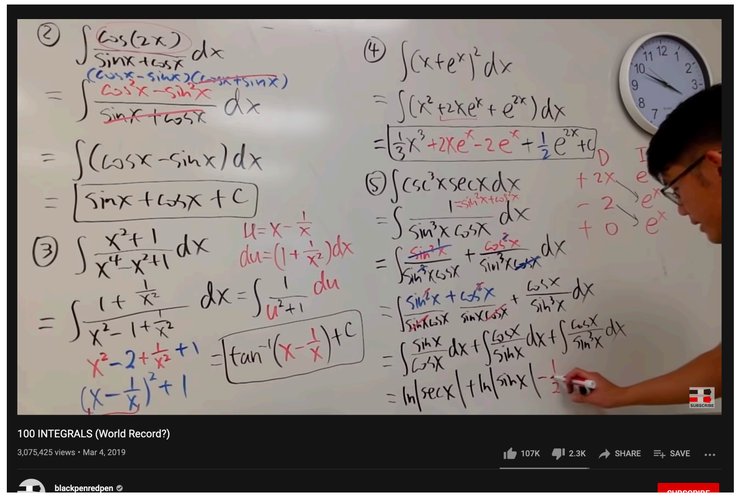
Loading a video

Think: What are the notifications you receive on your mobile phone called? Why?

Before I can watch a video, first I have to ask for the video.

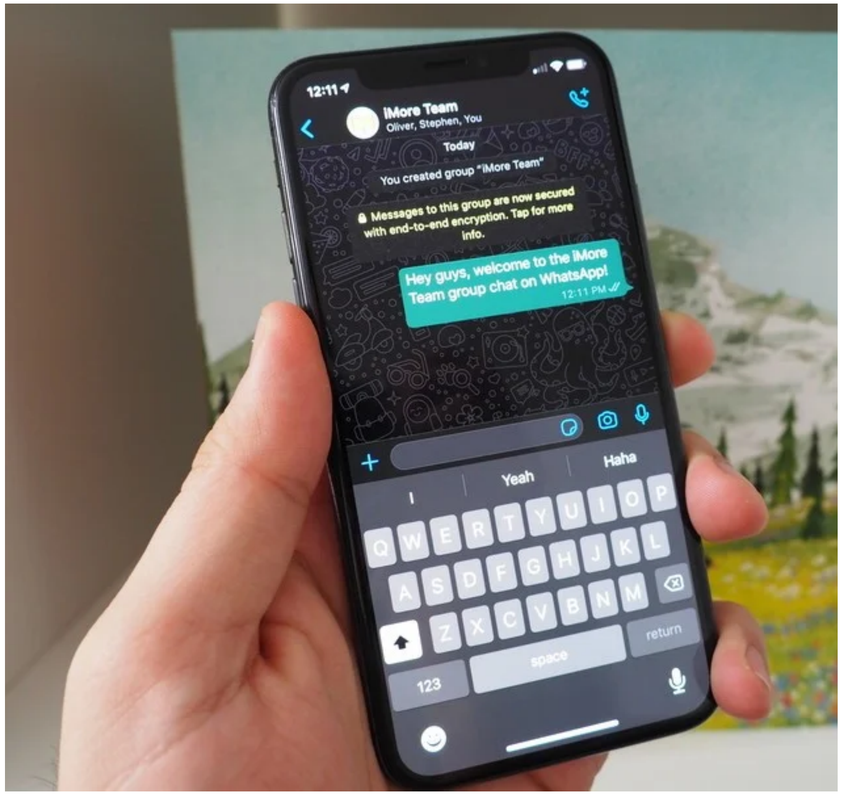
If my **request** for the video succeeds, then I can display the video.

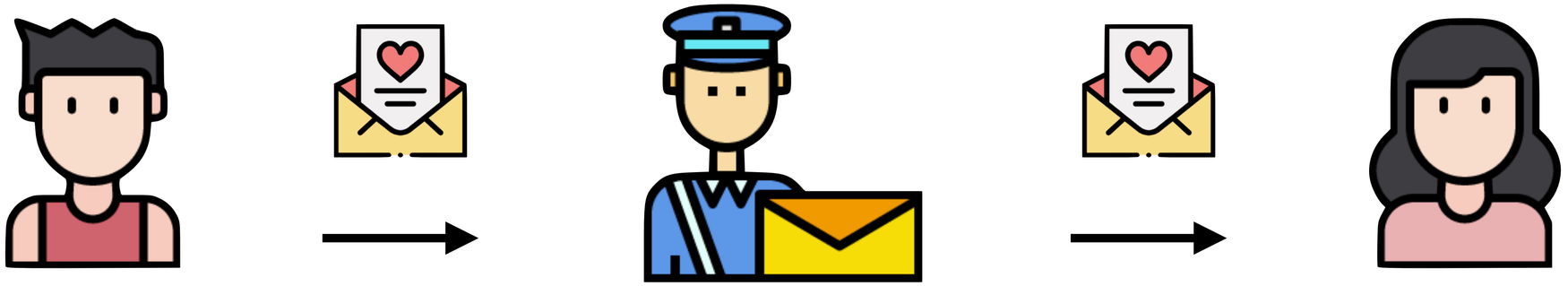




To speak to my friend remotely, I need to **send** a message.

In order to send a message, I need to know **where** to send it and **how** to send it.





Ben

Alyssa

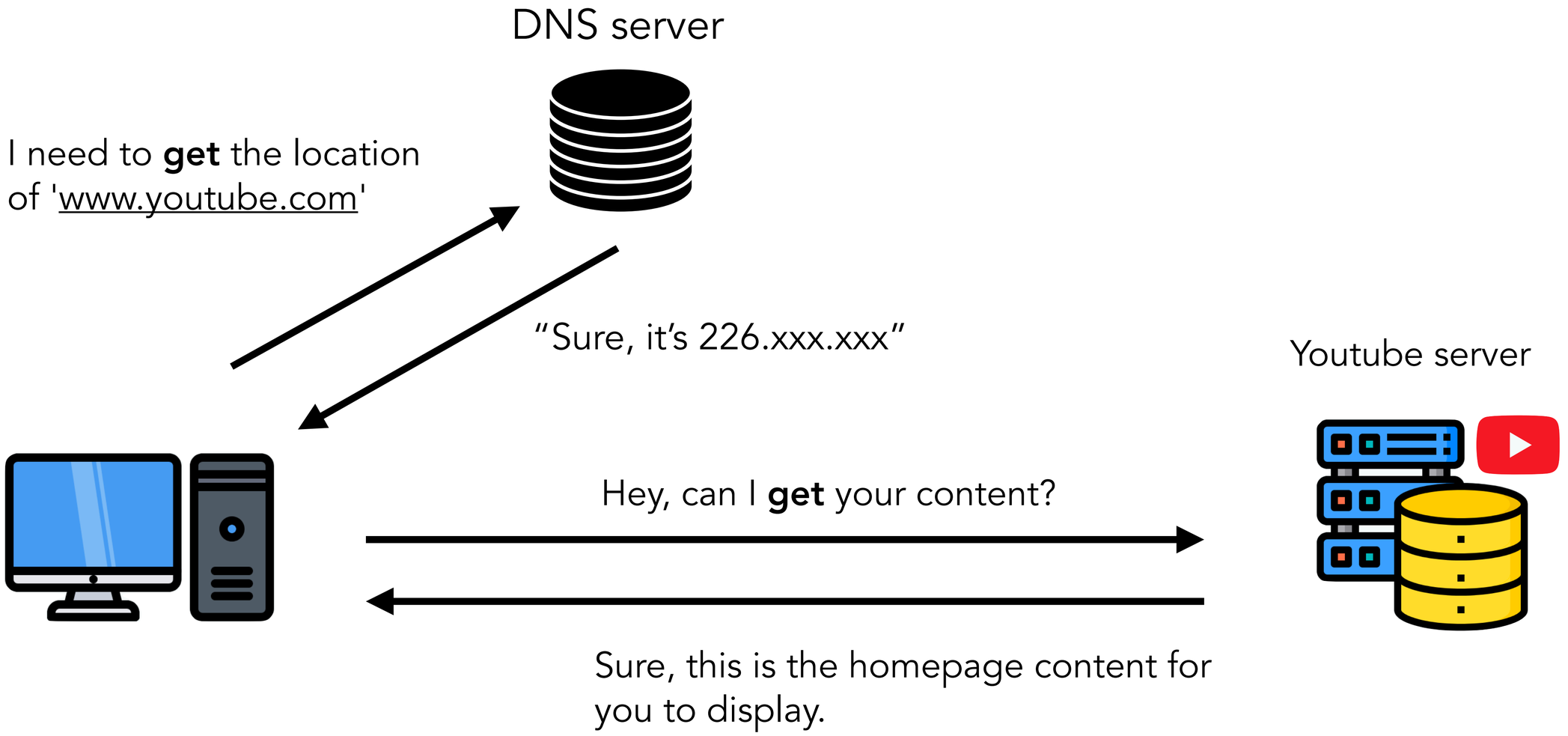
What do we need when sending/receiving mail?

Address…

Transportation…

Security…

Roads…



Think: In which order are these processes executed? Are these pull/push requests? Now do you know why it's called an 'IP address'?

Request/Response

A **request** is a question and a **response** is an answer.

The answer may have useful information or it may not. Some requests may not get an answer…

…in which case we say that the requested resource is **not found**.

Note: Responses are **not** "push" requests; you do not *initiate* a response. It is useful to think of it that once a request is sent, it "picks up" the response on its way back to the client.

Think: Why do some websites return Error 404?

A **request** is initiated by the client and a **response** is the answer to the client's request.

What makes a request?

There are different kinds of ways to move data around (transport protocols) but for the sake of simplicity we will focus on (RESTful) HTTP requests.

Type of Request

- GET
- POST
- PUT
- DELETE

Scheme

- HTTP
- HTTPS

URL

- Scheme
- Domain

… other info

Payload

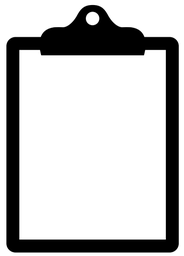
- The data you want transferred

Think: What makes a response?

An analogy

The police wants to transport important packages from one place to another.

URL
https://www.abc.com



Okay go to abc address and make sure the package secure

Scheme

HTTPS

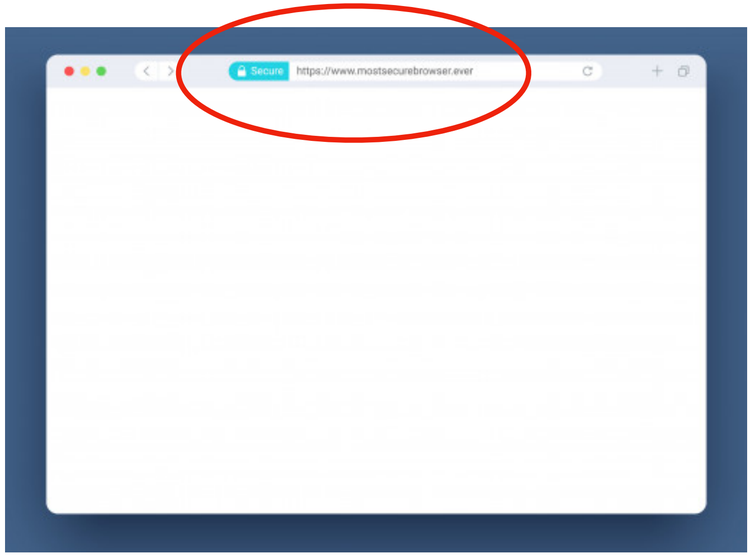
Cannot see what's inside

HTTP

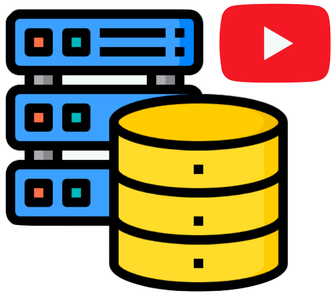
Can see what's inside

Think: What is the payload?

Putting it all together



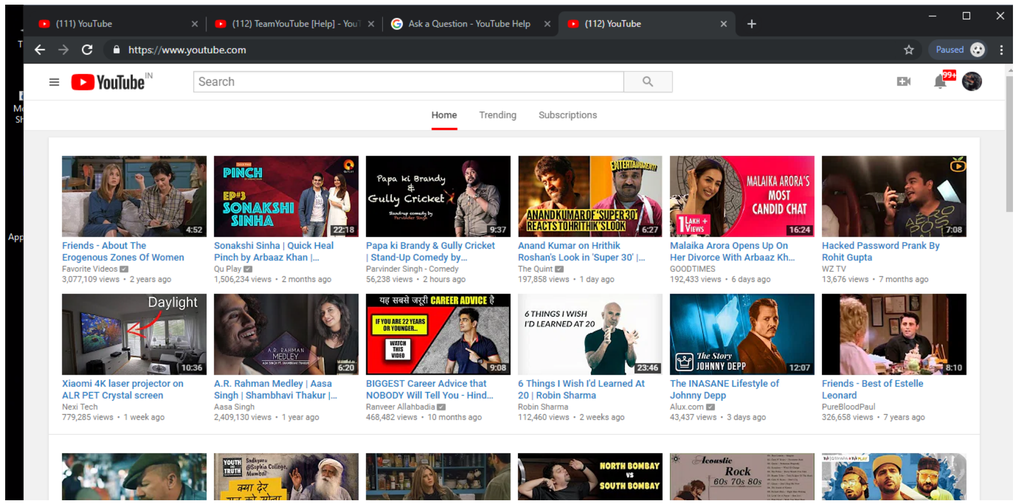
GET https://www.youtube.com



HTML, Javascript, CSS

Putting it all together

With the information the Youtube server provides us, the browser does some magic…



What is HTML?

According to W3C:

HTML (the Hypertext Markup Language) and CSS (Cascading Style Sheets) are two of the core technologies for building Web pages. HTML provides the structure of the page, CSS the (visual and aural) layout, for a variety of devices. Along with graphics and scripting, HTML and CSS are the basis of building Web pages and Web Applications.

Simply put, HTML contains the instructions on how to structure a web page and CSS provides information on styling (eg. colour, width, height...).

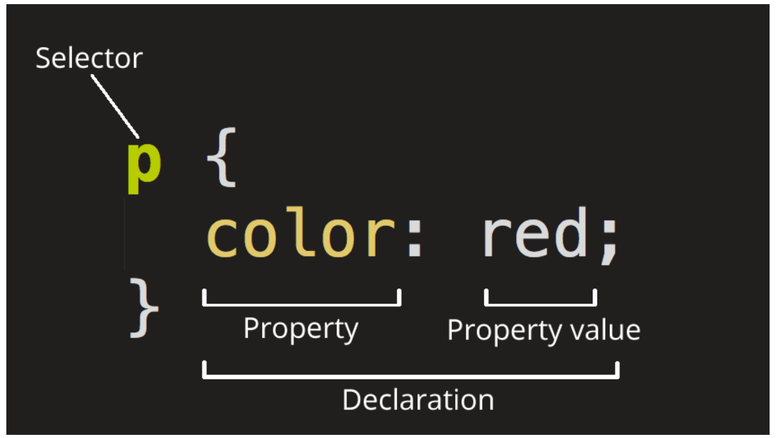
What is HTML?

```
<body>
    <p>
    Hello this is a piece of text
    </p>
</body>
```

Hello this is a piece of text

CSS examples

Hello this is a piece of red text



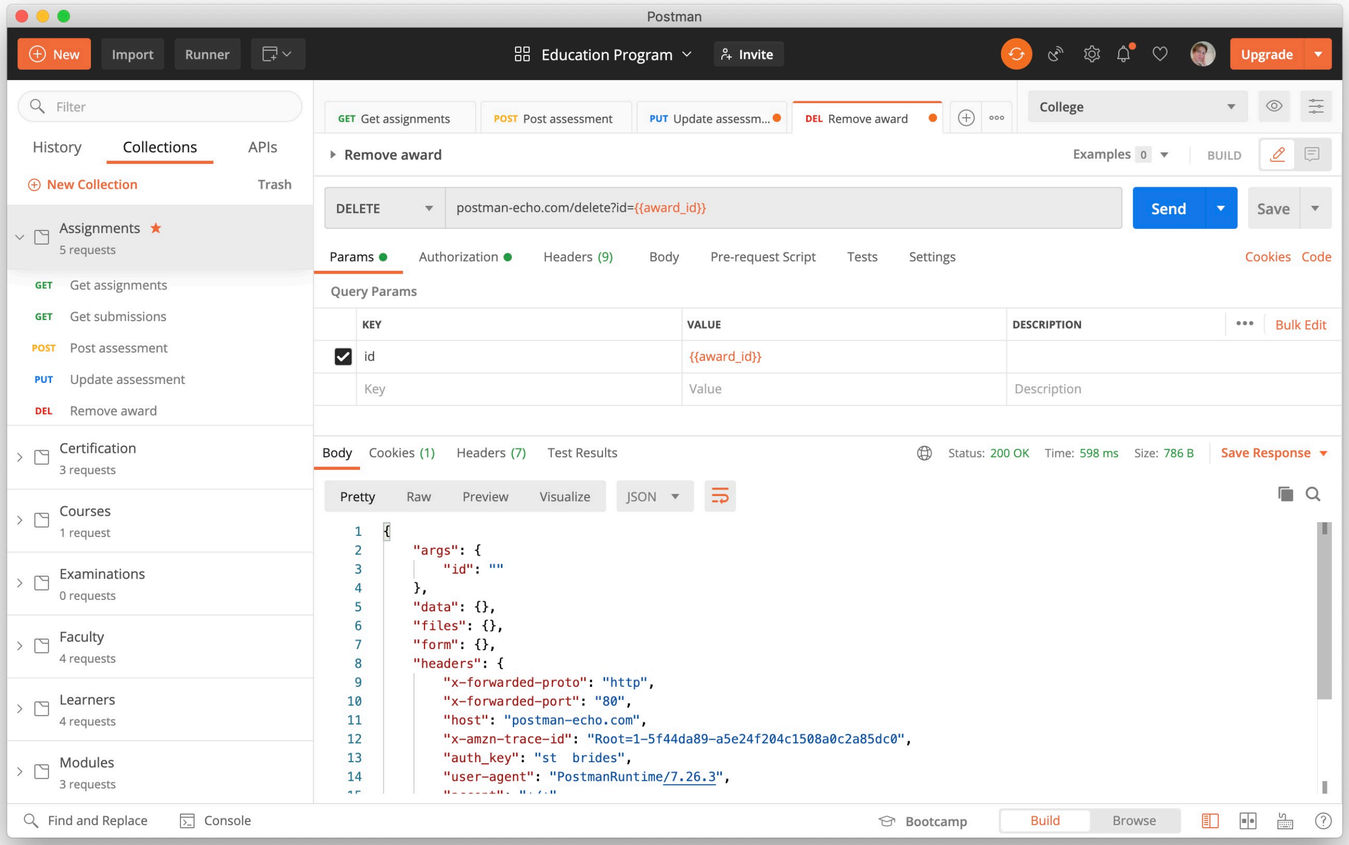
Let’s play with requests

Activity #1

What is Postman?

Postman is an API client that provides a GUI interface to make server requests.

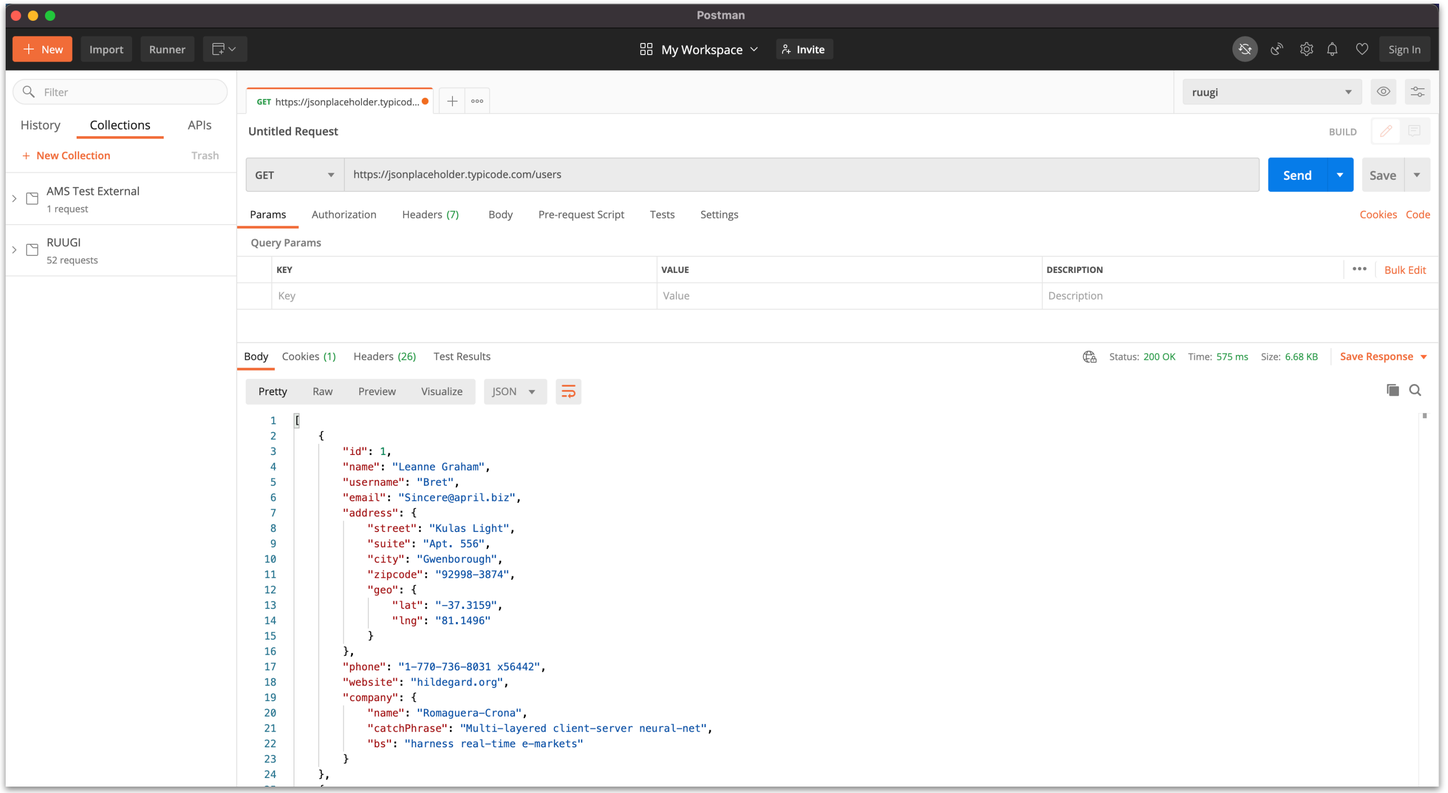
Get Postman here:
https://www.postman.com/downloads/



Make a GET request

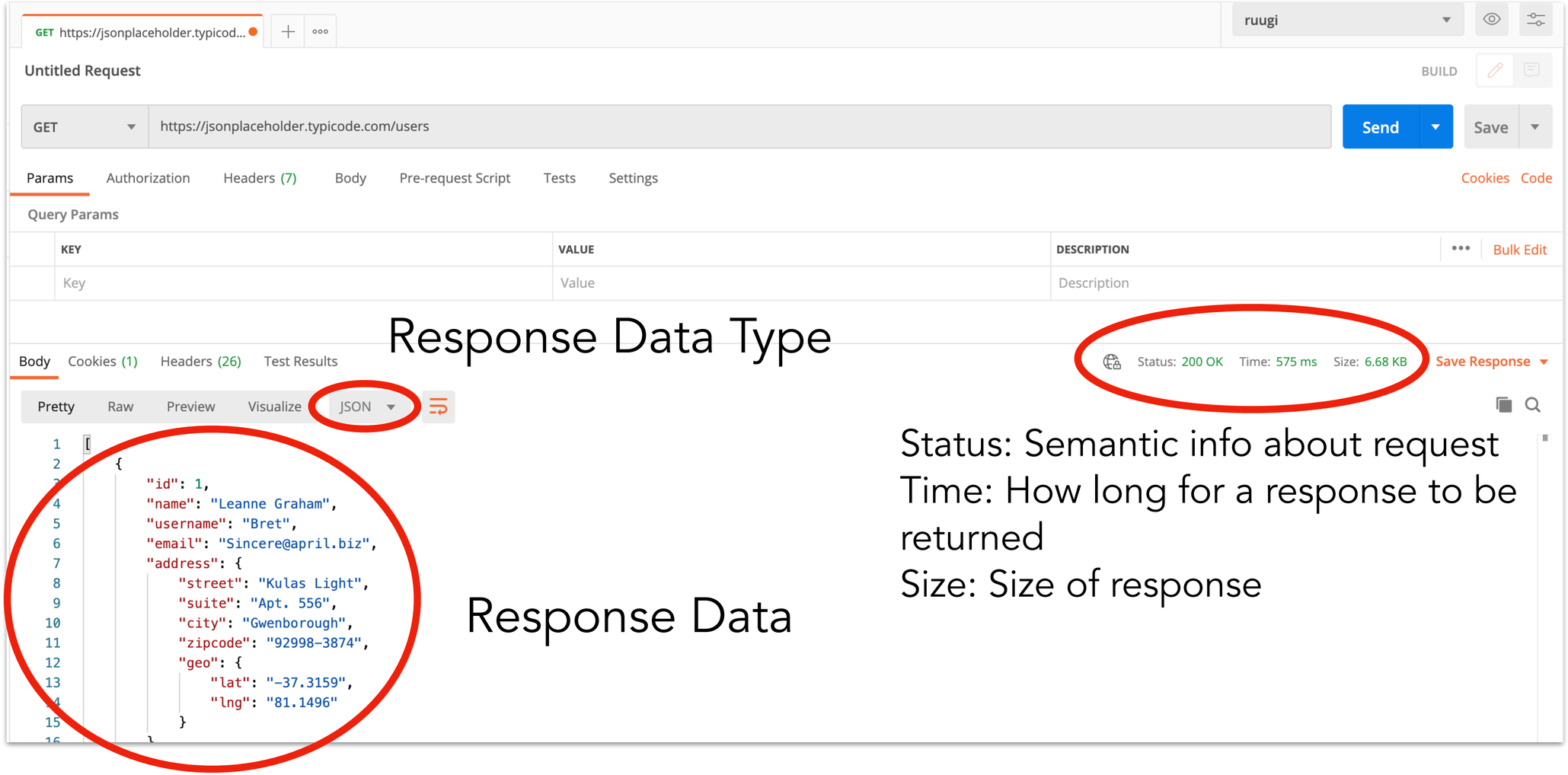
Let’s get some user information.

GET
https://jsonplaceholder.typicode.com/users



Breaking down the response

GET https://jsonplaceholder.typicode.com/users



Response Data Type

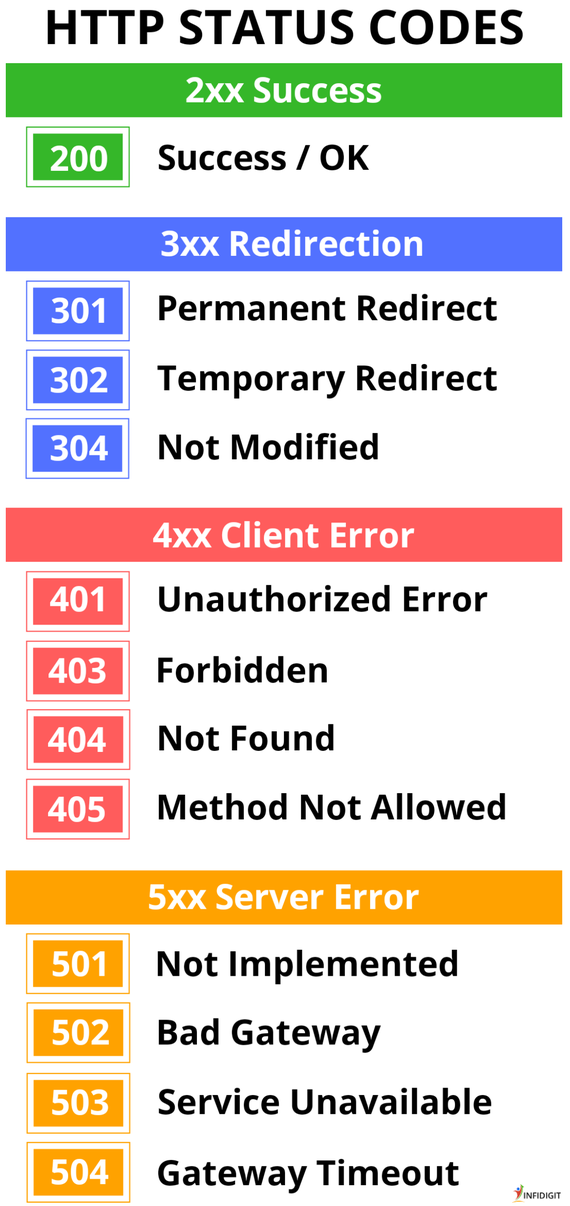
Response Data

Status: Semantic info about request
Time: How long for a response to be returned
Size: Size of response

What are HTTP status codes?

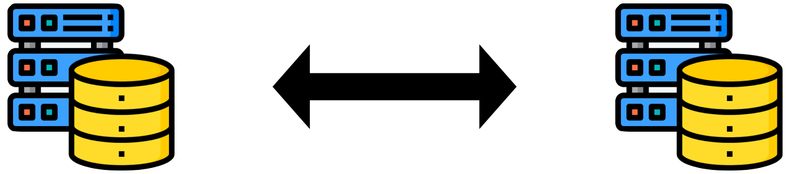
Status codes allow us to gather semantic information about the state of a request/response.

Assuming the server provides accurate status codes, we can use this in our code logic to decide whether or not to take a certain action, retry the request or abort the operation.

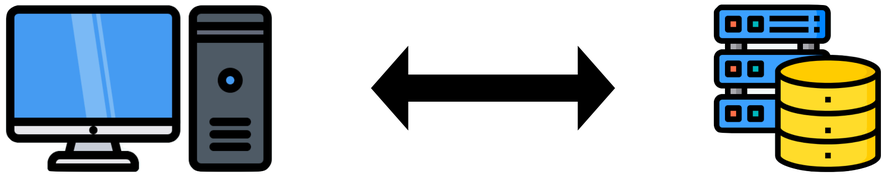


Response Data Formats

Different data types will result in different client-side behaviour.



Some APIs are intended to allow server-server communication and do not need instructions for how to *display* data. In such cases, JSON is a popular data format.

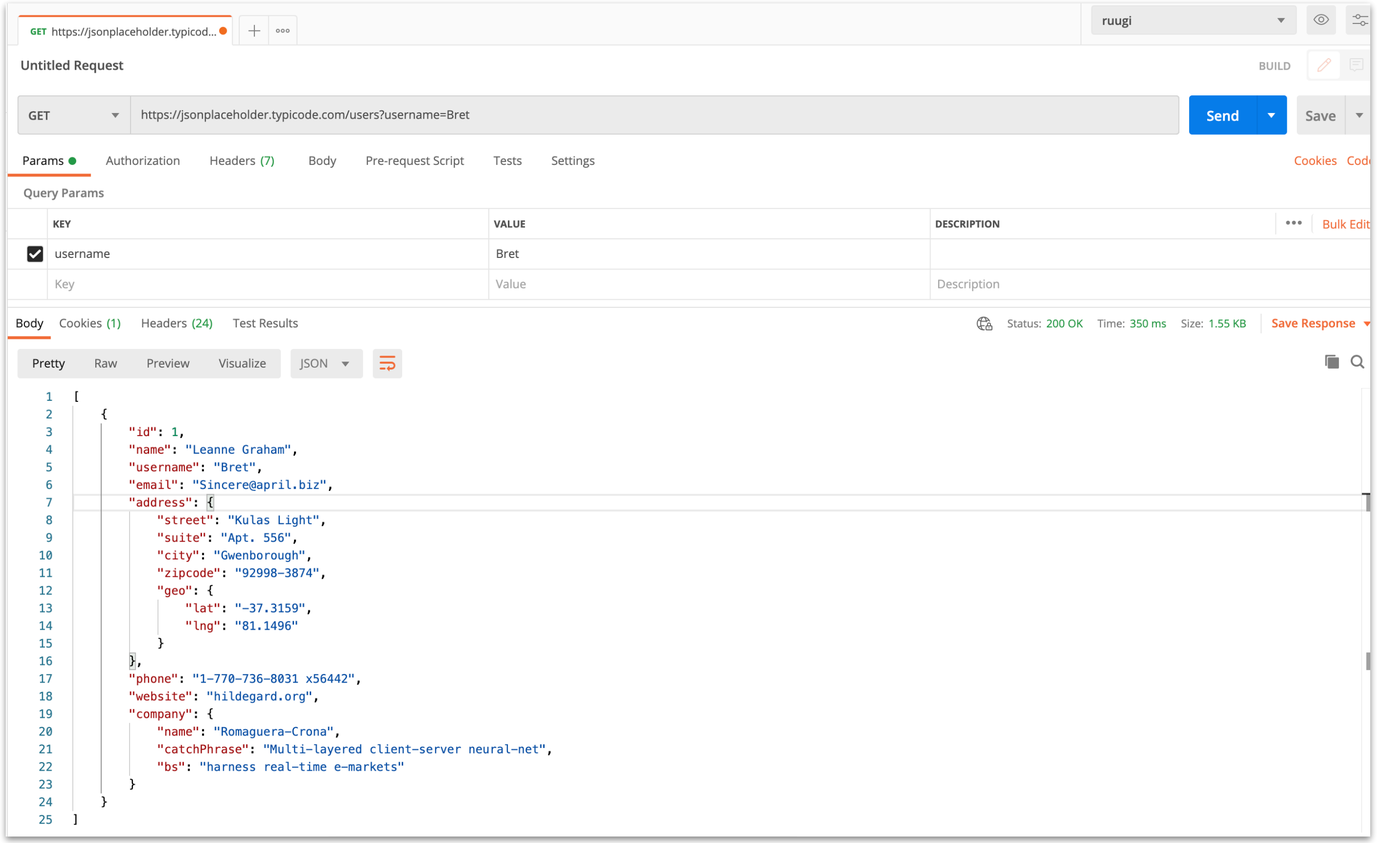


In the case of web frontend APIs, since the purpose of the browser is to display information for humans, GET requests for web pages almost always return HTML.

GET specific information

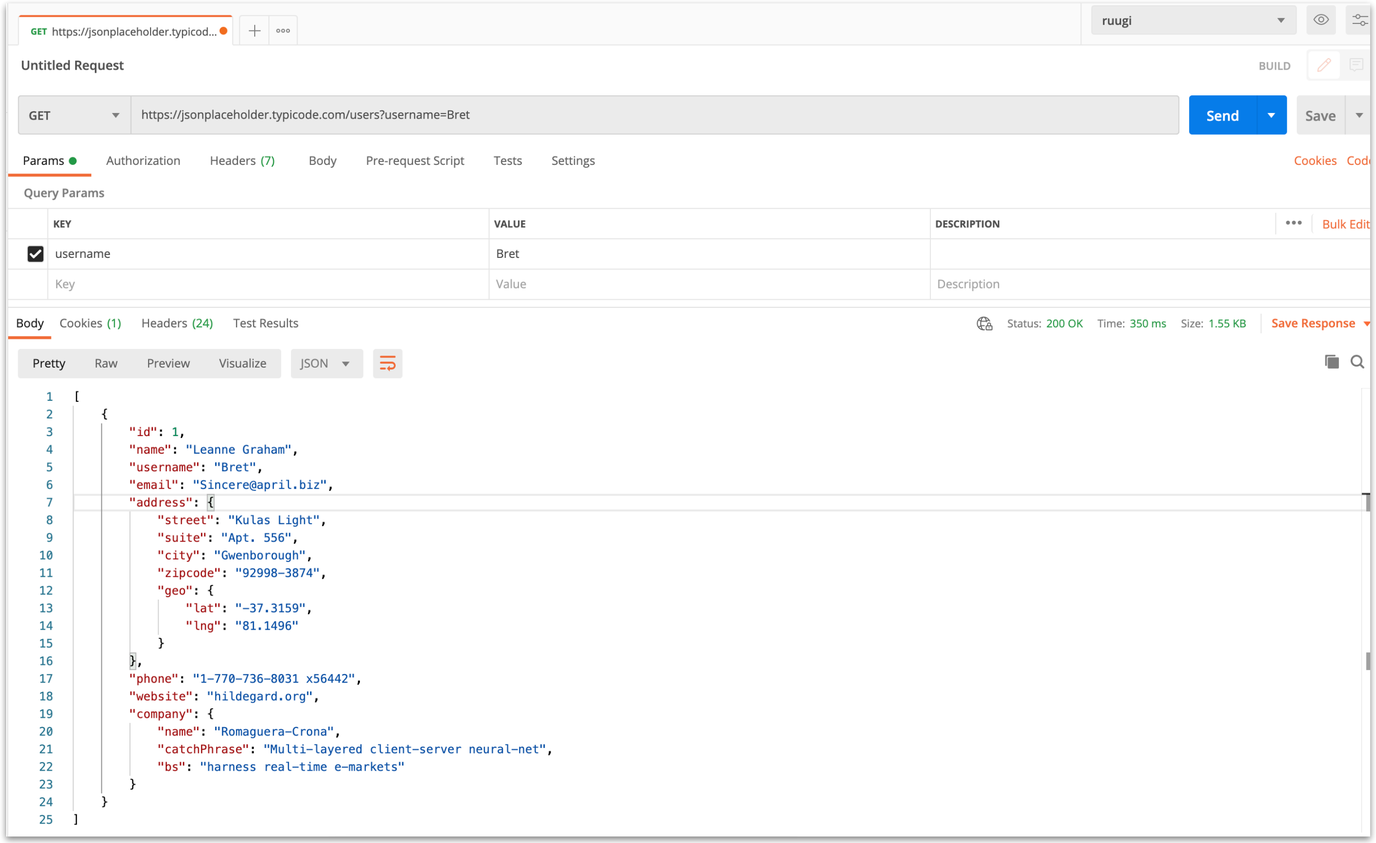
Let’s get a specific user

GET https://jsonplaceholder.typicode.com/users?username=Bret



Breaking it down

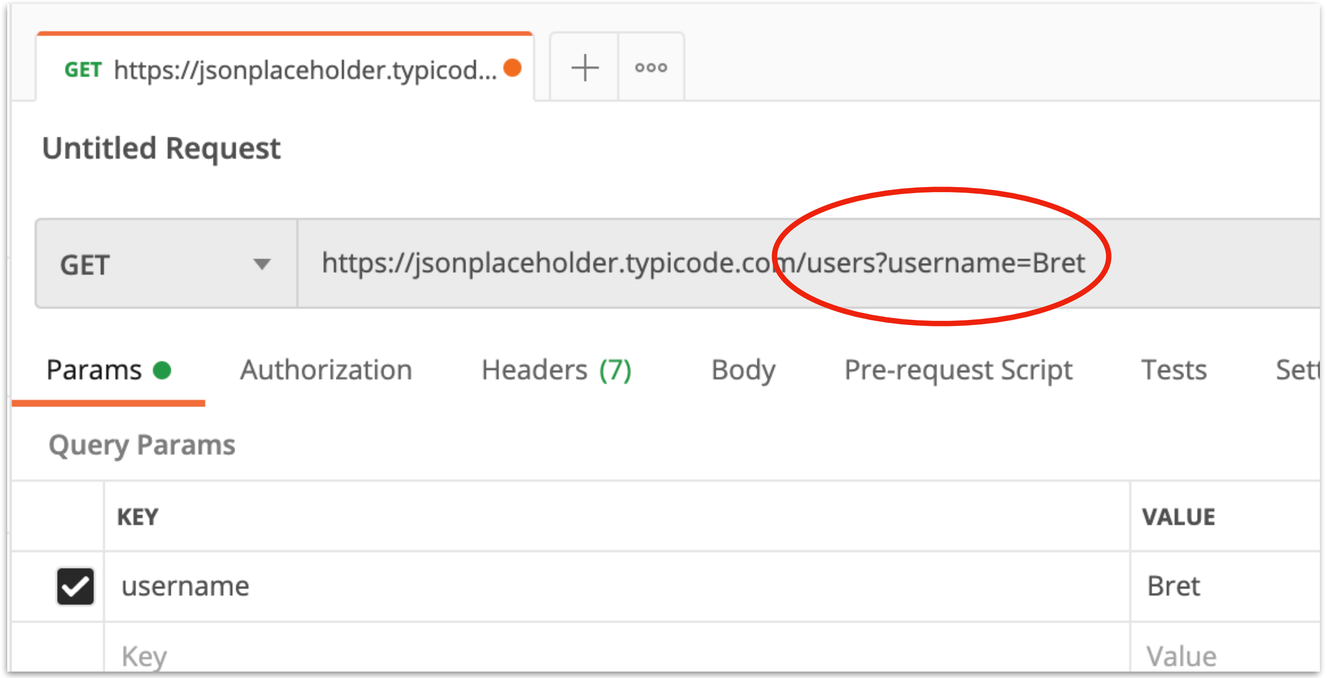
GET https://jsonplaceholder.typicode.com/users?username=Bret



Again, we have the same response properties as before but we have a new set of data.

Breaking it down

GET https://jsonplaceholder.typicode.com/users?username=Bret



Query Parameters

They allow us to specify certain conditions we want in the final set of data.

GET /users**?**username=Bret
GET users **where** username equals Bret

Try: Can you set multiple query params?

Note: This feature does not exist for all APIs, it depends on how the developer has designed the server to interact. More on this in Day 4.

Let’s make a web scraper!

Activity #2

Getting started

We will be using a few external libraries for this activity that you should have installed.


```
pip install requests
```

Beautiful Soup

```
pip install beautifulsoup4
```

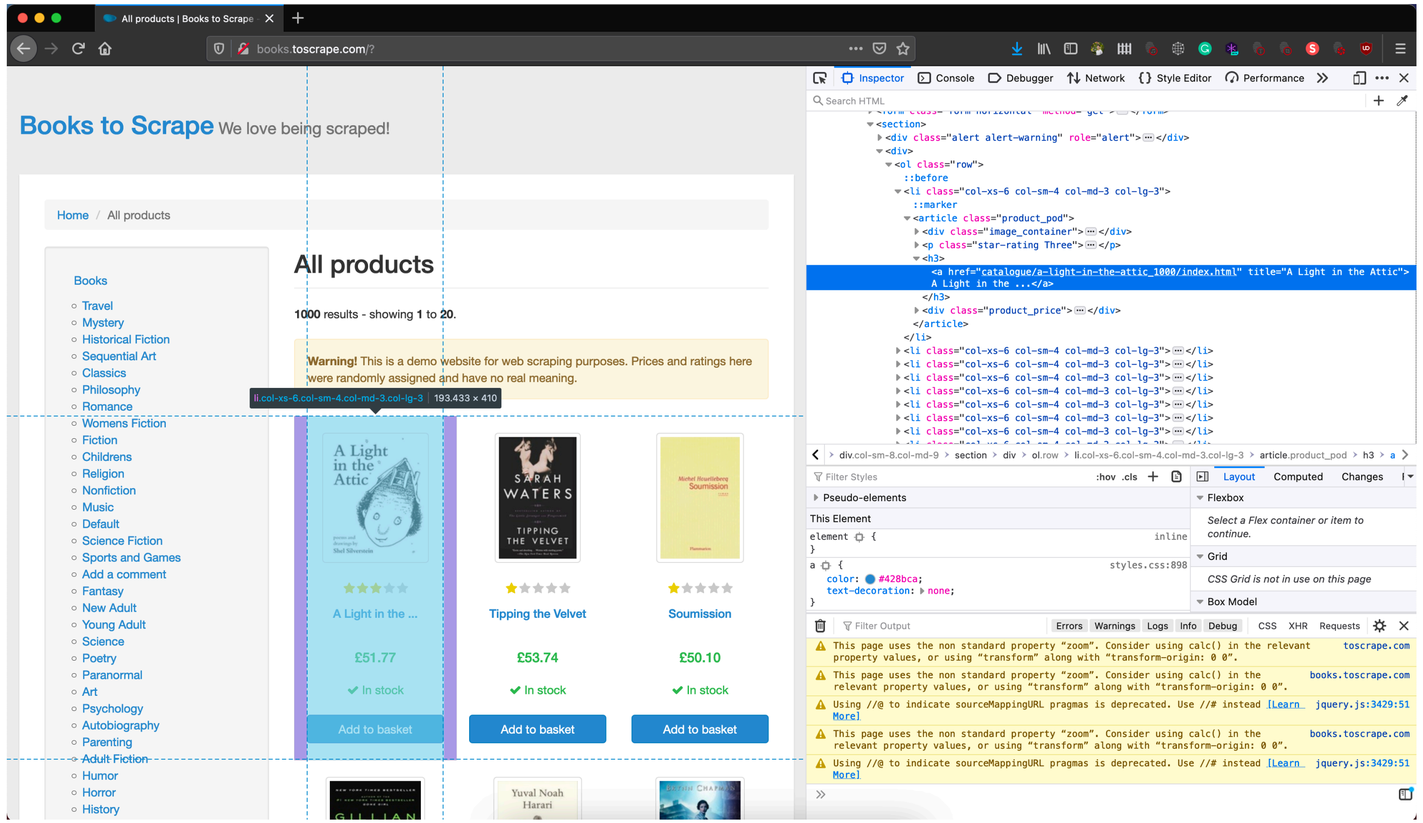

```
pip install jupyter
```

What is the Inspector Tool?

Inspecting a web page allows us to view the underlying HTML.

Think of the HTML code as the *frame* of the web page. It tells the browser what to render in what place; it is the skeleton of the page.

This is useful when we want to see how information is organised on a page.



It’s a wrap!

What have we learnt today?

- A high-level understanding of how the web works
- How information is passed around over the network
- What are HTTP Requests
- Practical use cases + light web scraping!