How the Web Works

How Web Works Exercise

Rodrigo Serrano

06/20/2022

How Web Works Exercise

Part One: Solidify Terminology

In your own terms, define the following terms:

- What is HTTP?

  HTTP are one of the set of rules of how data communication on the web must be done.

- What is a URL?

  The URL is the full identification to find a certain webpage/server. It contains the Protocol, a hostname, a Port (most time doesn't display and works with default), a Resource (specific page of the server) and a query when it applies.

- What is DNS?

  The DNS is the "nickname" given to the page for it to make it easier for us humans to find a webpage because I.P. addresses have a lot of numbers and may get confusing.

- What is a query string?

  A query string is the last part of a URL to give extra information, for example, searches, form submissions, etc.

- What are two HTTP verbs and how are they different?
  - Post: This "verb" makes it so you can make a change on the server (for example submitting a post or commenting).
  - Get: This "verb" only retrieves information from the server without modifying anything.

- What is an HTTP request?

  The request is the action that we want to be performed on the resource (URL). In other words, it's what we are asking the server to do for us (retrieve information, change a post, etc.).

- What is an HTTP response?

The response is what the server give us back depending on the information we give on the request.

- What is an HTTP header? Give a couple examples of request and response headers you have seen.

  Headers provide additional information on what you are requesting or the response from the webpage.

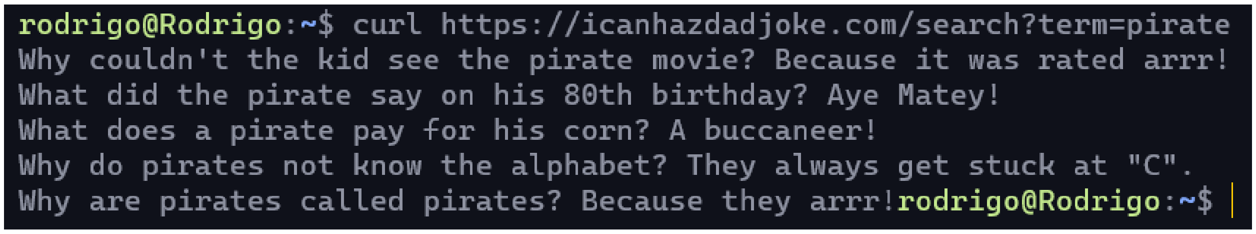
  - Date, content-type, last-modified, server, content-encoding, cookie, cache-control, accept-language.

- What are the processes that happen when you type "http://somesite.com/some/page.html" into a browser?

  1. The page looks at what type of protocol it's going to use (in this case http.
  2. Then the domain's name is turned into an IP address and tries connecting to it.
  3. Because there was no port specified it's going to connect to the default port.
  4. Finally it's going to try to get to the resource of the server (/some/page.html).

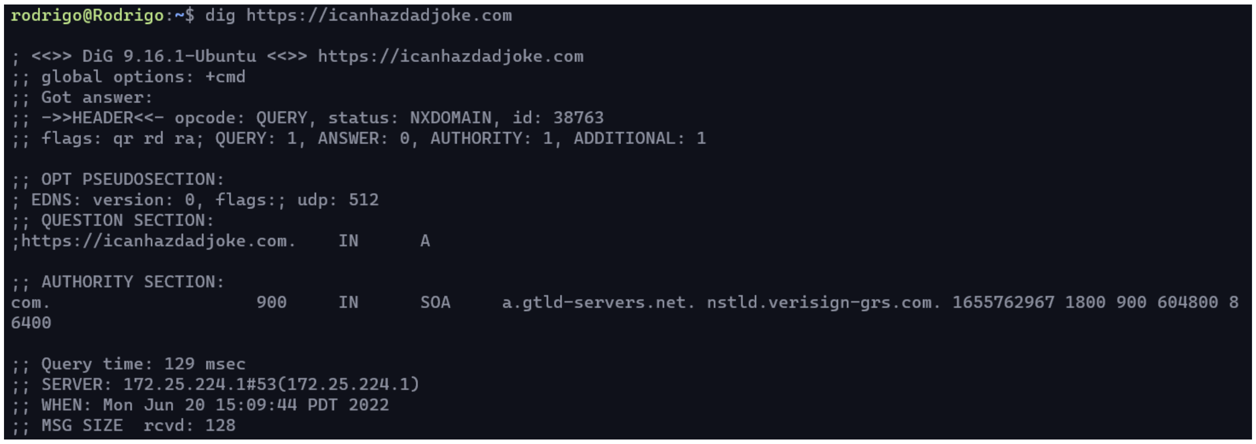
Part Two: Practice Tools

1. Using ***curl***, make a ***GET*** request to the *icanhazdadjoke.com* API to find all jokes involving the word "pirate"



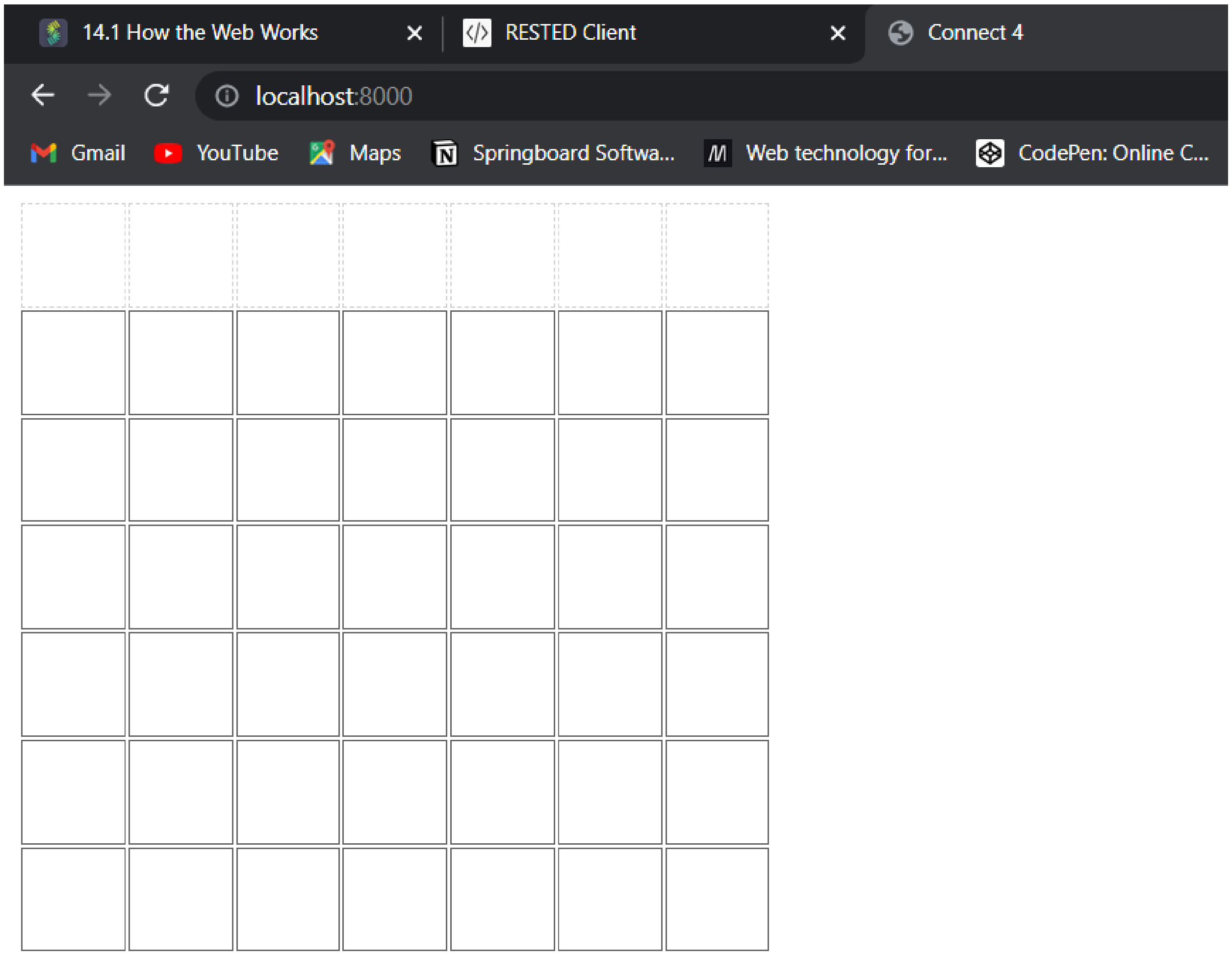
```
rodrigo@Rodrigo:~$ curl https://icanhazdadjoke.com/search?term=pirate
Why couldn't the kid see the pirate movie? Because it was rated arrr!
What did the pirate say on his 80th birthday? Aye Matey!
What does a pirate pay for his corn? A buccaneer!
Why do pirates not know the alphabet? They always get stuck at "C".
Why are pirates called pirates? Because they arrr!rodrigo@Rodrigo:~$ 
```

2. Use ***dig*** to find what the IP address is for *icanhazdadjoke.com*



```
rodrigo@Rodrigo:~$ dig https://icanhazdadjoke.com

; <<>> DiG 9.16.1-Ubuntu <<>> https://icanhazdadjoke.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 38763
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;https://icanhazdadjoke.com.    IN      A

;; AUTHORITY SECTION:
com.                    900     IN      SOA     a.gtld-servers.net. nstld.verisign-grs.com. 1655762967 1800 900 604800 8
6400

;; Query time: 129 msec
;; SERVER: 172.25.224.1#53(172.25.224.1)
;; WHEN: Mon Jun 20 15:09:44 PDT 2022
;; MSG SIZE  rcvd: 128
```

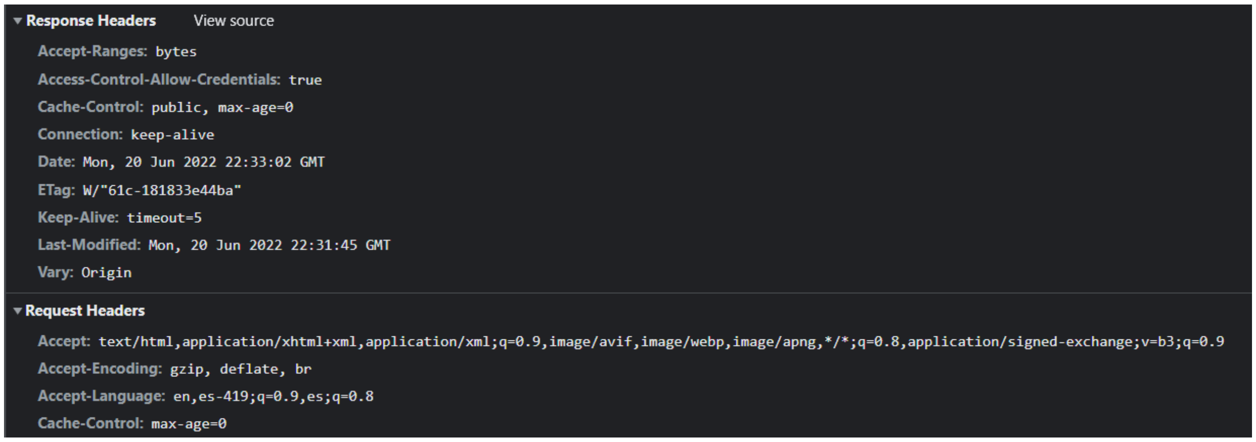
3. Make a simple web page and serve it using ***python3 -m http.server***. Visit the page in a browser.



Part Three: Explore Dev Tools

Build a very simple HTML form that uses the GET method (it can use the same page URL for the action) when the form is submitted.

Add a field or two to the form and, after submitting it, explore in Chrome Developer tools how you can view the request and response headers.



```
▾ Response Headers      View source
    Accept-Ranges: bytes
    Access-Control-Allow-Credentials: true
    Cache-Control: public, max-age=0
    Connection: keep-alive
    Date: Mon, 20 Jun 2022 22:33:02 GMT
    ETag: W/"61c-181833e44ba"
    Keep-Alive: timeout=5
    Last-Modified: Mon, 20 Jun 2022 22:31:45 GMT
    Vary: Origin
▾ Request Headers
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
    Accept-Encoding: gzip, deflate, br
    Accept-Language: en,es-419;q=0.9,es;q=0.8
    Cache-Control: max-age=0
```

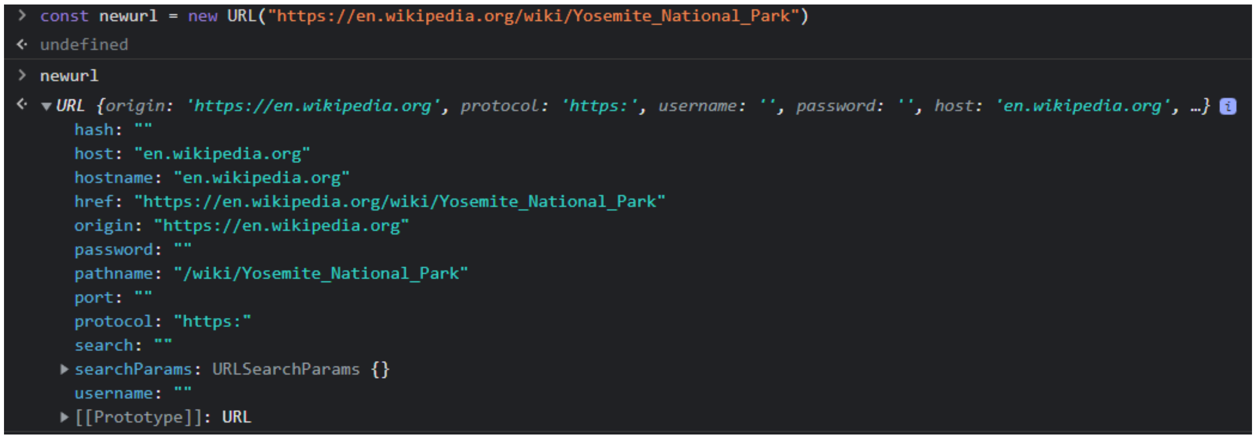
Edit the page to change the form type to POST, refresh in the browser and re-submit. Do you still see the field in the query string? Explore in Chrome how you can view the request and response headers, as well as the form data.

Part Four: Explore the URL API

At times, it's useful for your JavaScript to look at the URL of the browser window and change how the script works depending on parts of that (particularly the query string).

Read about the URL API

Try some of the code examples in the Chrome Console so that you can get comfortable with the basic methods and properties for instances of the URL class.



```
> const newurl = new URL("https://en.wikipedia.org/wiki/Yosemite_National_Park")
<· undefined
> newurl
<· ▼URL {origin: 'https://en.wikipedia.org', protocol: 'https:', username: '', password: '', host: 'en.wikipedia.org', …} i
      hash: ""
      host: "en.wikipedia.org"
      hostname: "en.wikipedia.org"
      href: "https://en.wikipedia.org/wiki/Yosemite_National_Park"
      origin: "https://en.wikipedia.org"
      password: ""
      pathname: "/wiki/Yosemite_National_Park"
      port: ""
      protocol: "https:"
      search: ""
    ▶ searchParams: URLSearchParams {}
      username: ""
    ▶ [[Prototype]]: URL
```