HTTP Basic Structure

Requests and responses and headers and…

Noah Gibbs, March 2023

codefolio@ruby.social

Help, I Hate Taking Notes

Good news: these slides are online.

https://codefol.io/speaking/hwg/http101.pdf

The URL is on the print-outs.

Put your energy into questions.

Wait, What's HTTP?

Web browsers and servers need to move data around. HTTP is the protocol they use.

The client asks. The server sends back what they asked for, or a reason why it can't have it.

What Gets Sent?

What's It Look Like?

Here's a really simple one, from Curl*.

```
GET /index.html HTTP/1.1
Host: localhost:4321
User-Agent: curl/7.79.1
Accept: */*
```

* Curl is a really common browser-testing tool. You can think of it as a browser impersonator if you like. There are lots of tools that "speak" HTTP. It's simple (for a protocol) so people build lots of good tools.

Structure...

```
GET /index.html HTTP/1.1
Host: localhost:4321
User-Agent: curl/7.79.1
Accept: */*
```

The first line is special:

METHOD URL HTTP/VERSION

GET /index.html HTTP/1.1

Structure...

```
GET /index.html HTTP/1.1
Host: localhost:4321
User-Agent: curl/7.79.1
Accept: */*
```

Later lines are "header" lines:

HEADERNAME: VALUE

Host: localhost:4321

Structure...

```
GET /index.html HTTP/1.1
Host: localhost:4321
User-Agent: curl/7.79.1
Accept: */*
```

Finally, two newlines to say "all done."

Newlines are invisible on the page, so please imagine them.

This Seems Too Simple

This isn't everything. This is the format for GET and HEAD requests.

If you have to upload a file or send form data, there's a payload of data after the two newlines.

Usually POST, PUT, etc have payloads, but technically any request can if it says it does.

This Seems Too Short

Curl sends a *tiny* request. Browser requests have a *lot* more headers. Let's look at one, briefly.

What About For Real?

Here's a small one from Chrome.

```
GET /root.html HTTP/1.1
Host: localhost:4321
Connection: keep-alive
sec-ch-ua: "Chromium";v="110", "Not A(Brand";v="24",
"Google Chrome";v="110"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
DNT: 1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/110.0.0.0 Safari/537.36
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=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

What Gets Sent Back?

The Response

This was all about HTTP requests. The server sends back an HTTP response, which has a pretty similar structure.

More of them usually have payloads, since they're usually carrying a chunk of data, like a web page, back to the client.

What's It Look Like?

This is a tiny response from a tiny web server.

```
HTTP/1.1 200 OK
Content-Type: text/plain

Hello World!
```

A lot like the request, right?
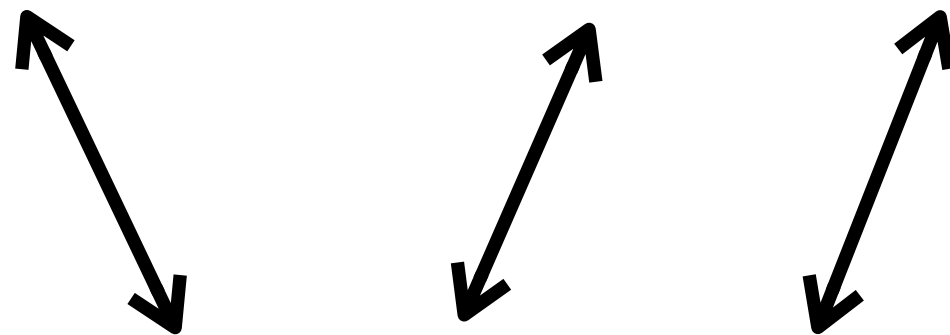
Structure...

```
HTTP/1.1 200 OK
Content-Type: text/plain

Hello World!
```

The first line:

HTTP/VERSION STATUS MSG

HTTP/1.1 200 OK

Structure...

```
HTTP/1.1 200 OK
Content-Type: text/plain

Hello World!
```

Header line(s).

Structure...

```
HTTP/1.1 200 OK

Content-Type: text/plain


Hello World!
```

Two newlines.

Structure...

```
HTTP/1.1 200 OK

Content-Type: text/plain

Hello World!
```

Payload. Here it's a text file.

What About Other Data?

Depending on the Content-Type (MIME type) of the data, there can be all kinds of things there, both for requests and responses.

HTML, images, whatever. A single HTTP request generally returns one file, so you need six for a page with five images, for instance.

Winding Up...

So It's HTML, Sort Of?

Nope. HTTP can send all sorts of things around. Text, images, binary files, RTF (rich text). Anything you want, pretty much.

Browsers often request HTML. But nothing about HTTP requires it.

So It's Browserspeak?

It's the *main* thing browsers speak.

A bunch of other tools also use it. Browser tools like Curl, yes. Also APIs that don't use a browser at all.

HTTP has good tools, so it's used for lots of things.
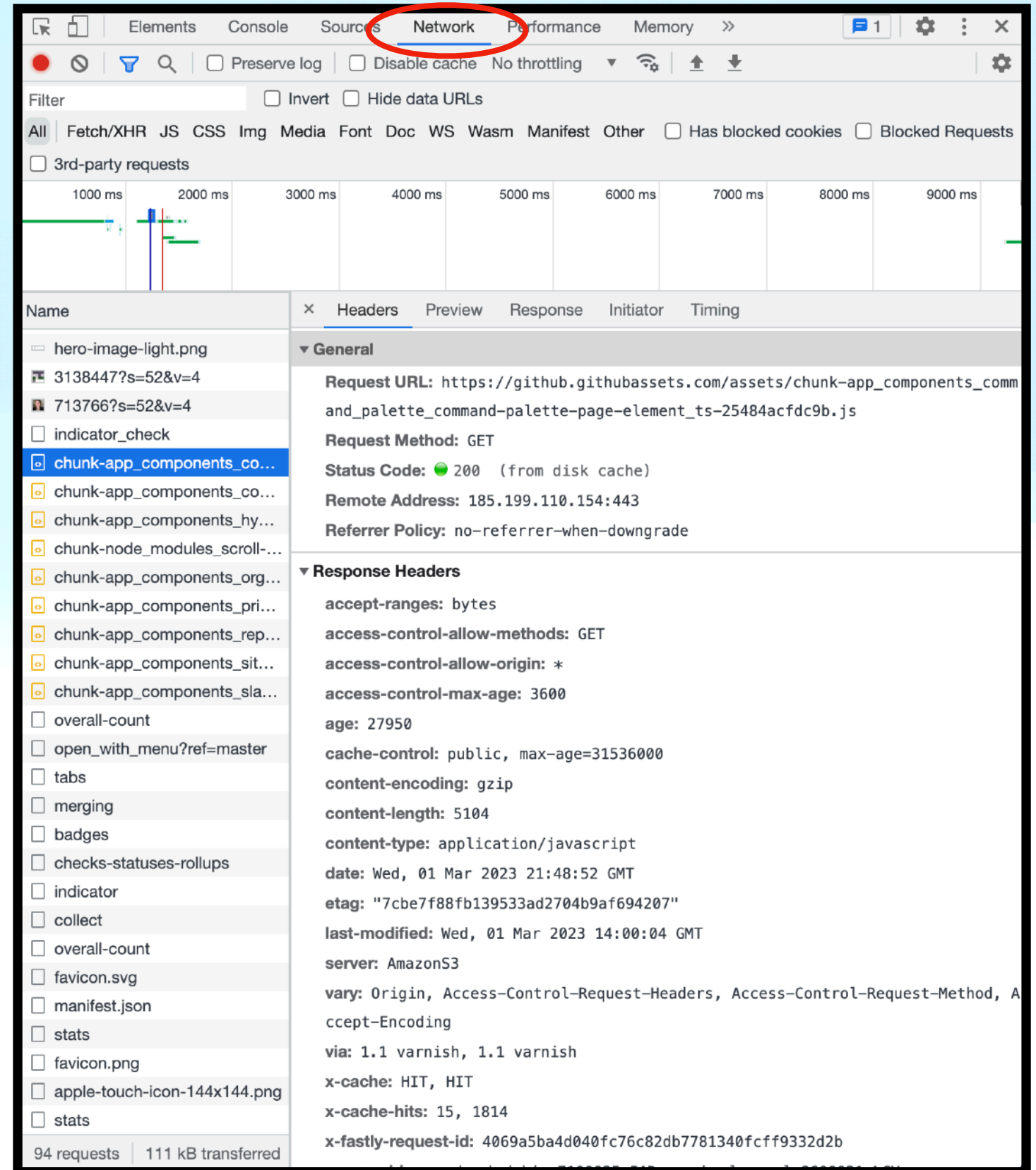
You Are Not To Be Trusted

How do I check this myself?

```
curl -v http://exmpl.com/url
```
prints everything it sends and receives.

```
nc -l localhost 4321
```
listens to port 4321 and prints data.

Chrome Dev Tools

You can also ask Chrome for low-level HTTP. It's not the full raw request, but it's most of it.

All Done.
Questions?