# What's HTTP?

You speak my language?

Justin
Feb 17, 2020



## The TL;DR

HTTP is the **language** that your computer speaks to web servers to let you **browse the internet**. How you supplicate to the server gods that be.

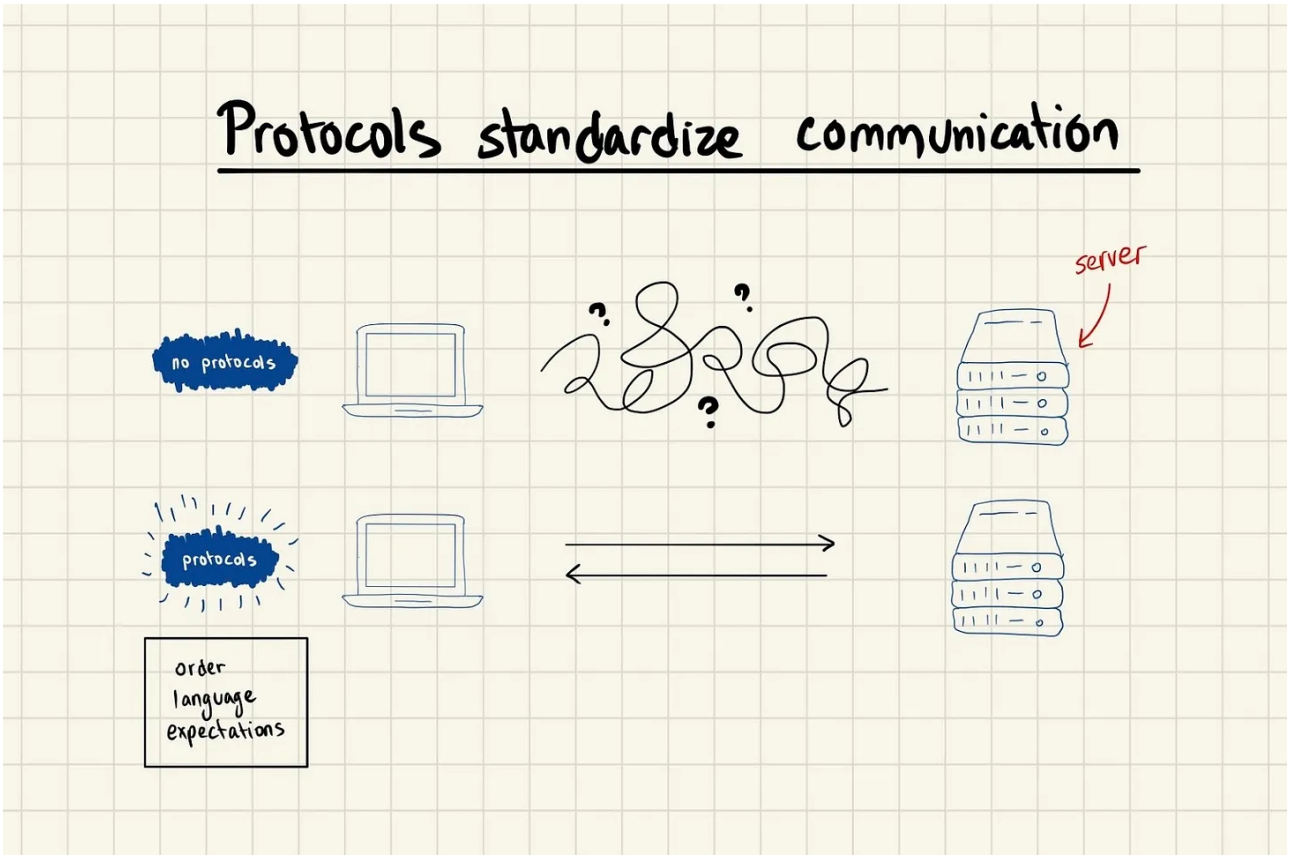
- HyperText Transfer Protocol (HTTP) is a protocol that **helps computers talk to one another** and share data
- HTTP works through the **request / response model**, kind of like a Burger King drive-thru
- Some connections are **encrypted through HTTPS**, the secure version

The HTTP protocol is the base layer for how you use pretty much anything on the web, and understanding it can be pretty practically helpful. So read this.

## What's HTTP?

HTTP is a *protocol* – a word thrown around a lot in tech, but one that took me a while to understand. What exactly are we dealing with here?

The web is all about communication: web servers out in the cloud have information you need (web pages and applications) and your computer needs to ask them for that information. To communicate well though, computers need a shared language, just like humans: protocols put *standards* around the language and format that computers use to connect and exchange information.




With that in mind, it's a bit easier to understand what HTTP is: it's one of the core protocols of the web, and governs how computers download files (web pages, usually HTML) from web servers. To make things a little more concrete, here are a few of the *rules* that the HTTP protocol requires computers to follow:

- There's an **order** for communication called the request / response model
- There's a **language** for specifying where files are called Uniform Resource Locator (URL)
- There's **directions** for where to connect, using specific ports (80 and 443)

Don't sweat the specifics: the point is that there needs to be *some* standard for how computers communicate, and HTTP is the series of them that became the default for the web. So how exactly does it work?

# The request / response model

What's happening when you put a URL into your web browser?

HTTP and most of the web works through the *request / response model*, where clients – like you on your laptop – *request* files from a web server in the cloud, and that server sends a *response* to you. When you head over to twitter dot com,

you're making a request (send me the files for my feed!) from Twitter's web servers, and they're responding with those files.

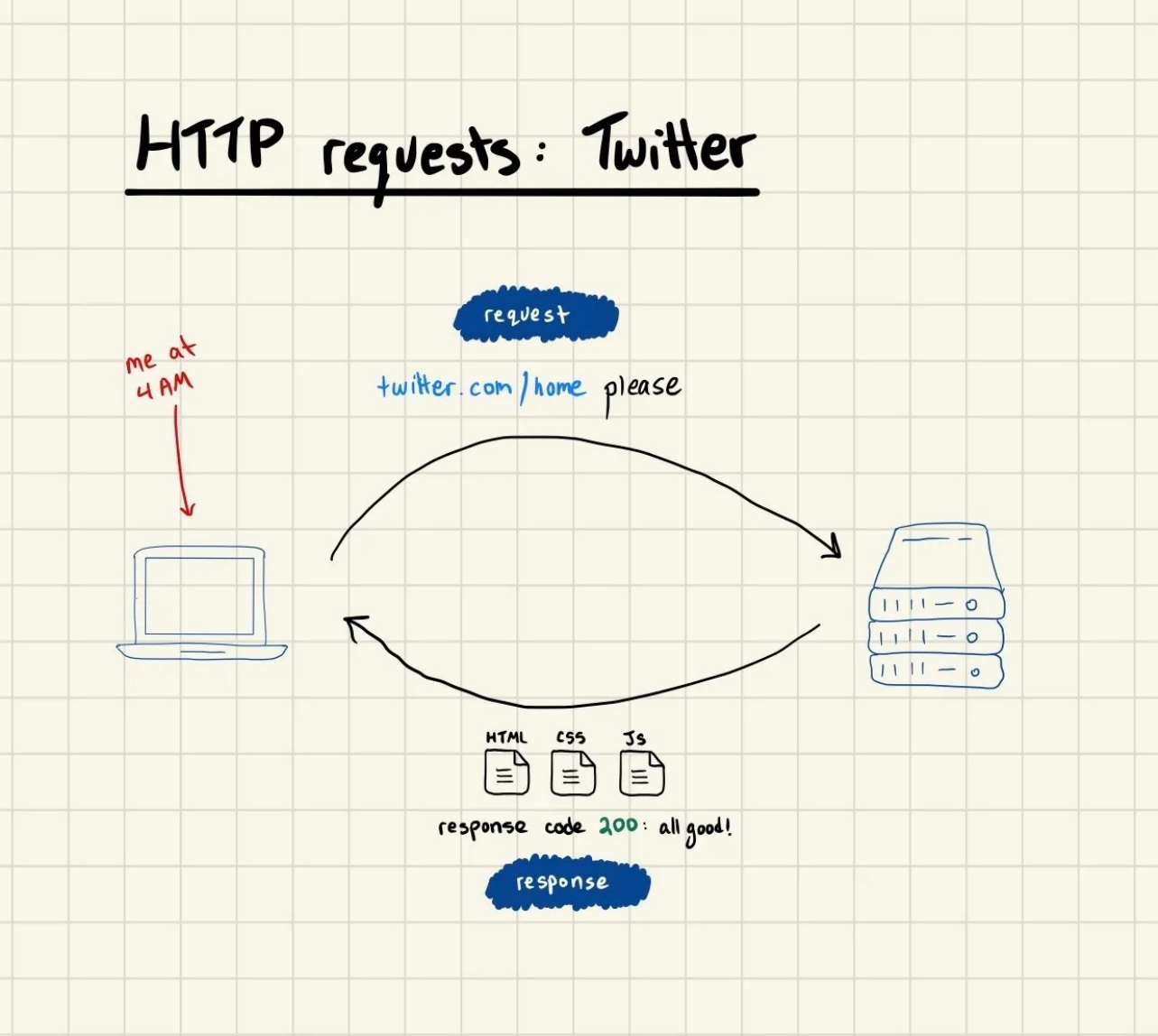
This model for communication is pretty popular across tech, so it's kind of important to understand. The basics:

- The **client** is the device requesting files: usually you on your laptop, searching for pictures of that pair of sneakers (that you've been eyeing for 4 months but don't have the guts to buy them. Buy them)
- The **server** is a web server in the cloud with the files you need: in this case, Google's servers that respond when you make a search
- The client makes a **request** to the server for the files it wants and the server sends back a **response** telling the client how it went

Request / response is kind of like a drive-thru: you tell the server what you want, and the server does their best to get that information to you. If something goes wrong – like if Burger King is (gasp!) out of Whoppers, the staff will tell you that and ask you to order something else. You're literally just asking for files: **a URL tells the server exactly which files you're requesting**.

The request / response isn't limited to just HTTP (it's popular elsewhere too), but the HTTP protocol puts some *specific guidelines* around exactly how the client (you) makes those requests to the server, and what kinds of responses you get back. For example, HTTP specifies certain *types* of requests you can make, like GET and POST.

Ever try to load a website, but get a **404 error**? That's another HTTP specification called a **response code**: they tell the client *what happened* with their request. If things went well and the server was able to send the requested files to the client, you'll see a 200. If you requested a file or page that wasn't found on the server, you'll see that old familiar 404. These response codes are standardized, and you can usually tell what *kind* they are based on the first number. And now you know what it means when you see them in your browser!




Without HTTP (or whatever protocol would have developed in its place), the internet would be complete nonsense: computers would have no standards for communicating with each other. It would be like the whole Tower of Babel situation, or when Americans try to speak basic Spanish – a complete mess.

> **Workplace Example**
>
> 404 errors – the response a server gives when it can't find the files you specified in your URL – are really common. That's why a lot of companies will create special HTML pages just for when the server encounters a 404, like this cute Star Wars themed one from Github.
>
> **Workplace Example**

# HTTPS (whoa very secure!)

When you're loading a web page that looks the same for everyone, like the homepage of the New York Times, there's not much security involved: if some evil third party were to get their hands on the data you're requesting from the NYT server, it wouldn't matter. It's public information. But if you log in, or you're working with company data, it's critical for your connections to web servers to be *secure*, and that's where HTTPS comes in.

HTTPS is (literally) the secure version of HTTP (thank you for the clever naming scheme, internet gods that be). The protocol requires that connections to servers be *encrypted* so that nobody other than the client that made the request can understand the response. Encryption is it's own topic, but it basically garbles up the data into a nonsense code that *only you* have the key to understanding.

Over the past few years, HTTPS has gotten incredibly popular, and a bunch of hosting providers even require you to use it. Google search gives higher rankings to HTTPS sites than regular HTTP ones. And if a site ever *doesn't* use HTTPS, your browser might give you a warning telling you that the site isn't secure (you've probably gotten one of these, and they're really annoying).

# Terms and concepts covered

`HTTP`

`Request / response`

`HTTP response codes`

`HTTPS`

# Further reading

- HTTP standardizes certain “methods” or types of requests you can make: there’s the popular GET and POST, as well as things like HEAD and CONNECT
- There’s an entire series of standards around URLs and how to structure them
- The major originator of HTTP was Tim Berners-Lee, who’s like the OG of internet shit, we’ll be coming back to him

---

## Comments

Write a comment...

---