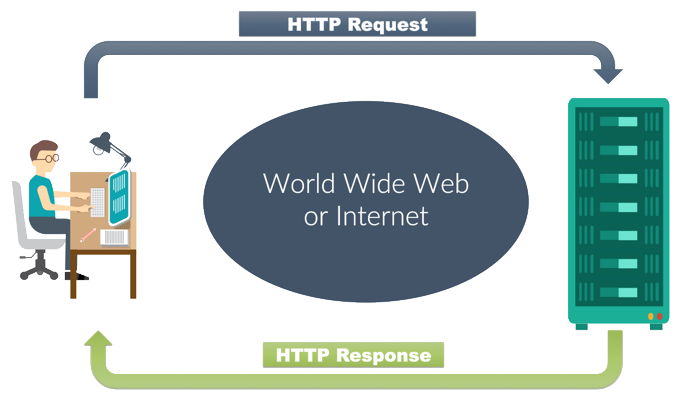
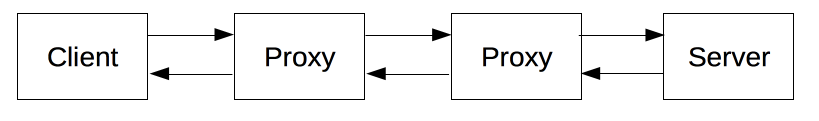
**What is HTTP?**

**HTTP** is a protocol that allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, Your server will receive requests from the browser that follows HTTP. It then responds with an HTTP response that all browsers are able to parse.



**## Components of HTTP-based systems**

Each individual request is sent to a server, which handles it and provides an answer, called the *response*. Between the client and the server there are numerous entities, collectively called proxies, which perform different operations and act as gateways or caches, for example.



**Client:**

The browser is **always** the entity initiating the request. It is never the server (though some mechanisms have been added over the years to simulate server-initiated messages).To present a Web page, the browser sends an original request to fetch the HTML document that represents the page.

**Server:**

On the opposite side of the communication channel, is the server, which *serves* the document as requested by the client. A server appears as only a single machine virtually: this is because it may actually be a collection of servers, sharing the load (load balancing) or a complex piece of software interrogating other computers (like cache, a DB server, or e-commerce servers), totally or partially generating the document on demand.

**Proxies:**

Between the Web browser and the server, numerous computers and machines relay the HTTP messages. Due to the layered structure of the Web stack, most of these operate at the transport, network, or physical levels, becoming transparent at the HTTP layer and potentially making a significant impact on performance. Those operating at the application layers are generally called **proxies**. These can be transparent, forwarding on the requests they receive without altering them in any way, or non-transparent, in which case they will change the request in some way before passing it along to the server. Proxies may perform numerous functions:

* caching (the cache can be public or private, as the browser cache)
* filtering (like an antivirus scan or parental controls)
* load balancing (to allow multiple servers to serve the different requests)
* authentication (to control access to different resources)
* logging (allowing the storage of historical information)

**## HTTP Messages**

HTTP is the language browsers speak. Every time you load a web page, you are making an HTTP request to the site's server and the server sends back an HTTP response.

**# Requests**

**URI**

When you make a request on the web, how do you know where to send it? This is done through **U**niform **R**esource **I**dentifiers or URIs. You’ve probably also heard of these as URLs. Both are fine. Let’s look at the URI we used up top.

http://sabre.com/bookings/

This URI is broken into three parts:

* http - the protocol
* sabre.com - the domain
* /bookings - the resource

The protocol is the way we're sending our request. There are several different types of internet protocols

The domain name is a string of characters that identifies the unique location of the webserver that hosts that particular website. This will be things like yahoo.com and google.com.

The resource is a particular part of the website we want to load. YouTube has millions and millions of channels and videos, so the specific resource we want is /bookings.

**HTTP Verbs**

**GET** /search?q=test HTTP/2  
Host: www.bing.com  
User-Agent: curl/7.54.0  
Accept: \*/\*

In the above request, “GET” is the HTTP method, which is a way for the request to specify *what* it *generally* wants to do.

Here are some common HTTP methods and what they are intended for:

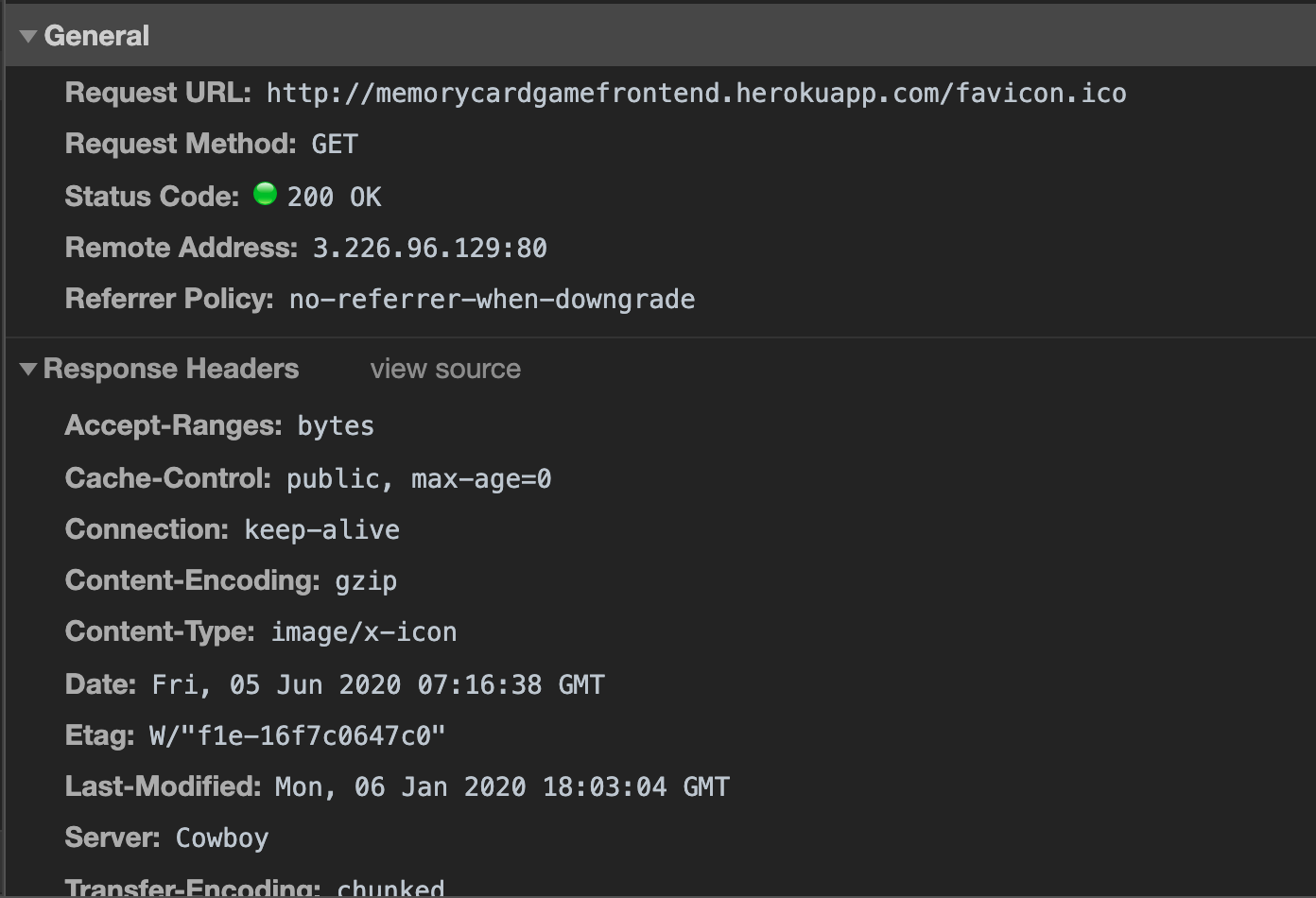
* GET— retrieves data, like a blog article.
* POST— creates data, like a new blog article.
* PUT— replaces data, like an existing blog article.
* DELETE— deletes data, like an existing blog article.

So, our example request using the GET method, says: “Please retrieve some data for me.”

There are other HTTP methods that can be seen in these MDN web docs.

**# Responses**

Once your server receives the request, it will do some processing (run code you wrote!) and then send a response back. The server’s response headers look something like this:



The server’s response is separated into two sections, the headers and the body. The headers are all of the metadata about the response. This includes things like content-length (how big is my response) and what type of content it is. The headers also include the status code of the response. The *body* of the response is what you see rendered on the page. It is all of that HTML/CSS that you see! Most of the data of a response is in the body, not in the headers.

**Status Codes**

Every time there is a successful response (you’ll know it’s successful because the page will load without any errors), it’s a status code of 200, but there are other status codes and it's good to get familiar with them. You've probably seen the second most popular status code, 404. This means "file not found." Status codes are separated into categories based on their first digit. Here are the different categories:

* 100’s — informational
* 200’s — success
* 300’s — redirect
* 400’s — error
* 500’s — server error

HTTP is a text-based protocol, used heavily by browsers to communicate with websites.