# What happens when you type *holbertonschool.com* in your browser and press ‘Enter’



https://unsplash.com/photos/HOrhCnQsxnQ

So, the Internet. It’s a magical, terrible world within our world, filled with knowledge, art, and ideas too numerous to comprehend. But how does it work? How is it capable of connecting me to people on the other side of the planet, or allowing me to read nearly any book ever written, all from the comfort of my own home? Where’s the wizard behind the curtain?

As it turns out, calling it a “World-Wide Web” is appropriate. There is a massive network of interconnected computers serving as multiple layers of infrastructure, to allow for our seamless communication. Let’s take a look at how it works, step-by-step, when you enter a website’s address into your browser. In this case, we’ll use my school’s website, [holbertonschool.com](http://holbertonschool.com), as an example.

## DNS request

The first thing the browser does once the [URL](https://en.wikipedia.org/wiki/URL) (Uniform Resource Locator, aka the web address) is entered, is it checks its preloaded [HSTS](https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security) (HTTP Strict Transport Security) list. This is a list of websites that have requested to be contacted via **HTTPS** only. [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) stands for “Hypertext Transfer Protocol”, and it is the foundation of data communication for the internet. [HTTPS](https://en.wikipedia.org/wiki/HTTPS), meanwhile, adds “Secure” to it (and is also referred to as “HTTP over TLS”, or “HTTP over SSL”). It encrypts the communication signals of HTTP using [TLS](https://en.wikipedia.org/wiki/Transport_Layer_Security) (Transport Layer Security) (or, formerly, [SSL](https://en.wikipedia.org/wiki/Transport_Layer_Security#SSL_1.0,_2.0,_and_3.0) (Secure Sockets Layer)). TLS is a cryptographic protocol designed to provide communications security over a computer network. The protocol is widely used in applications such as email, instant messaging, and voice over IP, but its use in securing HTTPS remains the most publicly visible.[[1]](#endnote-2)

OK, back to our HSTS list. If the website is in the list, the browser sends its request via HTTPS instead of HTTP. Otherwise, the initial request is sent via HTTP. (Note that a website can still use the HSTS policy without being in the HSTS list. The first HTTP request to the website by a user will receive a response requesting that the user only send HTTPS requests. However, this single HTTP request could potentially leave the user vulnerable to a [downgrade attack](http://en.wikipedia.org/wiki/SSL_stripping), which is why the HSTS list is included in modern web browsers.)[[2]](#endnote-3)

The browser then checks if the address is in its cache. This is a file that stores previous responses from [web servers](https://en.wikipedia.org/wiki/Web_server), to reduce the amount of information that needs to be transmitted across the network, as information previously stored in the cache can often be re-used. If not found, the browser calls a “get host by name” library function to do the look-up.

If it does not have the address cached, then it makes a **request** to the [**DNS**](https://en.wikipedia.org/wiki/Domain_Name_System) (Domain Name System) server configured in the network stack. This is typically the local router or the ISP's caching DNS server. The DNS is a service that contains a library of websites’ URL’s (the names you and I know them as, aka google.com or holberton.com) matched with their actual [IP](https://en.wikipedia.org/wiki/IP_address) (Internet Protocol) addresses (the actual numbered address of their website. For google.com, their primary IP address is 8.8.8.8).

## TCP/IP

Once the browser receives the IP address of the destination server, it takes that, along with the given port number from the URL (the HTTP protocol defaults to port 80, and HTTPS to port 443), and it’s time for it to find the server on the Internet and establish a connection.

Using the public Internet routing infrastructure, packets from a client browser request get routed through the router, the ISP, and an internet exchange to switch ISPs or networks, to find the server with the IP address to connect to. But this is a very roundabout way to get there and it’s not efficient.

Instead, many sites use a [CDN](https://en.wikipedia.org/wiki/Content_delivery_network) (Content Delivery Network), to cache static and dynamic content closer to the browser. A CDN is a globally distributed network of caching servers that improves the performance of a site or app by bringing the content closer to the users. Requests from the client browser get to take advantage of this private network that has ultra-low latency and high availability. Once the browser finds the server on the Internet, it establishes a TCP connection with the server and if HTTPS is being used, a TLS handshake takes place to secure the communication.[[3]](#endnote-4)

[TCP](https://en.wikipedia.org/wiki/Transmission_Control_Protocol) (Transmission Control Protocol) is one of the main protocols of the “Internet Protocol suite”, commonly referred to as [**TCP/IP**](https://en.wikipedia.org/wiki/TCP/IP). The suite itself is a framework for organizing the set of communication protocols used in the Internet according to their function and purpose. The other main protocol of the suite is [UDP](https://en.wikipedia.org/wiki/User_Datagram_Protocol) (User Datagram Protocol), and is mainly used for fast, real-time systems, such as gaming or live-streams. TCP on the other hand, has multiple reliability services that may increase latency, but help ensure correct transmission of all necessary information. As such, it is the main protocol used for most HTTP requests and internet communication in general.



https://unsplash.com/photos/FXFz-sW0uwo

Front-End

Now that the browser has a connection to the server, it follows the rules of communication for the HTTP(s) protocol. It starts with the browser sending an HTTP request to the server to request the contents of the page. The HTTP request contains a request line, headers (or metadata about the request), and a body. The request line contains information that the server can use to determine what the client (in this case, your browser) wants to do. The request line contains the following:

* a request method, which is one of GET, PUT, POST, DELETE, or a handful of other HTTP verbs
* the path, pointing to the requested resources
* the HTTP version to communicate with

When the server receives this request message, it interprets it, maps the request to a file or program on the server itself, and sends back a response. The first server it hits is typically a [**load-balancer**](https://en.wikipedia.org/wiki/Load_balancing_(computing)). This is a server dedicated for handling large amounts of incoming requests and sending them off to multiple other servers, in parallel, to actually be processed. It uses algorithms to systematically direct traffic to the various other web servers that serve the content itself. Common algorithms include:

1. Round robin — distributes traffic to groups of servers sequentially
2. Least connections — distributes traffic to servers based on which have the fewest connections at the time
3. IP hash — IP address is translated to a hash to determine which server handles the request (pseudo-random)[[4]](#endnote-5)

After the load-balancer transmits the request to one of the servers to handle it, the message is received by the [**web server**](https://en.wikipedia.org/wiki/Web_server) and processed. These 2 types of servers together are generally known as the website’s “front-end”, and pertain to the presentation of data as people typically interact with it, in their browser. The web server is in charge of serving static web content of the codebase, like the [HTML](https://en.wikipedia.org/wiki/HTML) (HyperText Markup Language), [CSS](https://en.wikipedia.org/wiki/CSS) (Cascading Style Sheets), and [JavaScript](https://en.wikipedia.org/wiki/JavaScript) pages themselves. If there is dynamic content that is requested, such as [PHP](https://en.wikipedia.org/wiki/PHP) or [Python](https://en.wikipedia.org/wiki/Python) code that specifically requests certain attributes or data, this is sent to another server. **[[5]](#endnote-6)** This may or may not be handled through an API.

API

An [API](https://en.wikipedia.org/wiki/API) (Application Programming Interface) is a popular software interface for abstracting the data transfer between a web server and application server, which allows it to hide the internal details of how a system works, exposing only those parts a programmer will find useful and keeping them consistent even if the internal details later change.[[6]](#endnote-7) An API can even be made public, allowing users to access raw data from the database in a useful format, without going through the main website front-end.

There are many different formats that an API can be designed around, but one of the most popular ones is known as [REST](https://en.wikipedia.org/wiki/Representational_state_transfer) (Representational State Transfer). It is a set of guidelines for creating stateless, reliable web APIs. An API that obeys the REST constraints is informally described as *RESTful*. By using a stateless protocol (that is, sending messages that do not refer to any previous requests or pre-existing session state, and thus each request can be understood in isolation) and standard operations, RESTful systems aim for fast performance, reliability, and the ability to grow by reusing components that can be managed and updated without affecting the system as a whole, even while it is running.

Back-End

The next server the data is sent to is known as the [**application server**](https://en.wikipedia.org/wiki/Application_server).This is a server dedicated to “business logic”, which is where most of the algorithms and processing of data happens. It will process the incoming request data, and retrieve the necessary information from the [**database**](https://en.wikipedia.org/wiki/Database), usually on its own server. The database is where all the important, permanent data is organized and stored, such as user emails, passwords, profile information, etc. This data is sent back to the application server which then sends it to the web server to incorporate into its response. The application server and database server are collectively known as the “back-end” of a website.

All this occurs assuming that a [**firewall**](https://en.wikipedia.org/wiki/Firewall_(computing)) allows incoming and outgoing traffic through port 80 (for HTTP), or port 443 (for HTTPS), of the server side of the TCP connection. Firewalls are computer programs or hardware that can be configured to block incoming and outgoing connections from a network. The software can be installed on any server in the process. In this case, if a firewall is installed on the server carrying the load-balancer or the servers containing the pages that were requested, and it is configured to allow incoming and outgoing traffic on the requested port, the HTTP request and its subsequent interactions with the web server occur unhindered. If, however, the firewall is configured to block incoming and outgoing connections on that port, the TCP connection step of this process would have failed and there would have been no connection between the browser and the server when the URL was typed. This is also true if only incoming traffic is blocked. If only outgoing traffic is blocked, the server attempts to send back the HTTP response after processing, but will be unsuccessful in doing so.[[7]](#endnote-8)

The server sends back an HTTP response after it interprets and handles the browser’s request. This response contains the status code followed by a series of optional headers that define and contain information about the content that is returned. After the status code and the headers, the actual response body/content that was requested is contained. If an HTML page was requested, the body will contain this content. The status code is important as it contains the status of the response.

1. **1xx: Informational:**  It means the request was received and the process is continuing.
2. **2xx: Success:**  It means the action was successful.
3. **3xx: Redirection:**  It means further action must be taken in order to complete the request. It may redirect the client to some other URL.
4. **4xx: Client Error:**  It means some sort of error in the client’s part.
5. **5xx: Server Error:**  It means there is some error on the server-side.

## Finally

Now the browser gets the response and the HTML web page is rendered in phases. First, it gets the HTML structure and then it sends multiple GET requests to get the embedded links, images, CSS, JavaScript files, etc. The web page will be rendered and in this case, the Holberton web page will, at last, be displayed. [[8]](#endnote-9)



https://unsplash.com/photos/Q1p7bh3SHj8

And that’s it. I know, not so magical when you see all the cogs and gears behind the scenes, but It does give one an appreciation for all the ingenuity and work that has gone into such a complex design. Truly, the Internet is the 8th Wonder of the World!

1. <https://en.wikipedia.org/wiki/Transport_Layer_Security> [↑](#endnote-ref-2)
2. <https://github.com/alex/what-happens-when#the-g-key-is-pressed> [↑](#endnote-ref-3)
3. <https://aws.amazon.com/blogs/mobile/what-happens-when-you-type-a-url-into-your-browser/> [↑](#endnote-ref-4)
4. <https://medium.com/@dkwok94/how-the-web-works-what-happens-when-you-type-a-url-and-press-enter-1958634f35bf> [↑](#endnote-ref-5)
5. <https://medium.com/@dkwok94/how-the-web-works-what-happens-when-you-type-a-url-and-press-enter-1958634f35bf> [↑](#endnote-ref-6)
6. <https://en.wikipedia.org/wiki/API> [↑](#endnote-ref-7)
7. <https://medium.com/@dkwok94/how-the-web-works-what-happens-when-you-type-a-url-and-press-enter-1958634f35bf> [↑](#endnote-ref-8)
8. <https://afteracademy.com/blog/what-happens-when-you-type-a-url-in-the-web-browser/> [↑](#endnote-ref-9)