codecademy

My Path
Catalog

Codecademy Resources > web

## HTTP Requests

Understand the basics of how your web browser communicates with the internet

## Background:

This page is generated by a web of HTML, CSS, and Javascript, sent to you by Codecademy via the internet. The internet is responsible for delivering several types of content and is constantly evolving. Have you ever thought of what needs to happen for you to be able to view this page in your browser?

This article focuses on one very small, but important piece of the internet puzzle: HTTP.

## What is HTTP?

HTTP stands for Hypertext Transfer Protocol and is the default way of getting all this HTML to render in your browser. It requires data to be transferred from one point to another over the network over TCP (Transmission Control Protocol).

TCP manages the channels between you and the server you want to connect to (in this case, Codecademy.com). TCP is used to manage many types of internet connections in which one computer or device wants to send or receive something from another. HTTP is the command language that the devices on both sides of the connection must follow in order to communicate.

## HTTP & TCP: How it Works

When you type an address such as www.codecademy.com in your browser, you are commanding it to open a TCP channel to the server that responds to that URL. (URL stands for Uniform Resource Locator, which you can read more about on Wikipedia). A URL is like your home address or phone number because it describes how to reach you.

In this situation, your computer, which is making the request, is called the client. The URL you are requesting is the address that belongs to the server.

Once the TCP connection is established, the client sends an HTTP request to the server.

A client can send several types of requests to a server over HTTP. It can request content from a server, send data to a server, or to send commands to a server. The most common types of requests are *GET*, *POST*, *PUT* and *DELETE*, which perform the following actions:

- *GET* sends a request for data from the server.
- *POST* and *PUT* are commands, usually accompanied by a data payload, to have the server accept data from the client. They're accompanied by that data.
- *DELETE* is used to request that a server remove data.

Let's explore an example of using the GET request (the most common type of request).

Suppose you want to check out the latest course offerings from Codecademy.com. After you type the URL into your browser, it will initiate a GET request to the server which contains the URL and optionally a data payload. The GET request contains the following text:

```
GET / HTTP/1.1
Host: www.codecademy.com
```

This identifies the type of request, the path on www.codecademy.com (in this case, "/") and the protocol "HTTP/1.1." The second line contains the address of the server which is "www.codecademy.com." There may be additional lines as well depending on what data your browser chooses to send.

The server might respond with the following in the header:

```
HTTP/1.1 200 OK
Date: Thu, 01 Oct 2015 19:00 GMT
Content-Type: text/html
```

Which is followed by the content requested below it.

The first line is confirmation that the server understands the protocol that the client wants to communicate with, and that the resource was found on the server using an HTTP status code. The third line shows the type of content that it will be sending to the client.

If the server is not able to locate the path requested by the client, it will respond with the header:

```
HTTP/1.1 404 Not Found
```

In this case, the server identifies that it understands the HTTP protocol, but the 404 HTTP status code signifies that the specific piece of content requested was not found. This might happen if the content was moved or if you typed in the URL path incorrectly. You can read more about the 404 status code, commonly called a 404 error, here.

After the server has sent the response, it closes the TCP connection. If you open the website in your browser again, or if your browser automatically requests something from the server, a new connection is opened which follows the same process described above.

## An Analogy:

It can be tricky to understand how HTTP functions because it's difficult to examine what your browser is actually doing. (And perhaps also because we explained it using acronyms that may be new to you.) Let's review what we learned by using an analogy that could be more familiar to you.

Imagine the internet is a town. Houses in the town represent your connection to the internet. Your address determines where you can be reached. The houses are where all the people live (like yourself, assuming you’re a person). Business in town, such as Codecademy.com, serve requests that are sent to them. This town also has a crazy fast mail service, an army of mail delivery staff that can travel on trains that move at the speed of light.

Suppose you want to read the morning newspaper while eating breakfast and sipping coffee. In order to retrieve it, you write down what you need in a language called HTTP and ask your local mail delivery staff agent to retrieve it from a specific business. The mail delivery person agrees and builds a railroad track (connection) between your house and the business nearly instantly, and rides the train labeled "TCP" to the address of the business you provided.

Upon arriving at the business, she asks the first of several free employees ready to fulfill the request. The employee searches for the page of the newspaper that you requested but cannot find it and communicates that back to the mail delivery person.

The mail delivery person returns on the light speed train, ripping up the tracks on the way back, and tells you that there was a problem "404 Not Found." After you check the spelling of what you had written, you realize that you misspelled the newspaper title. You correct it and provide the corrected title to the mail delivery person.

This time the mail delivery person is able to retrieve it from the business. You can now read your newspaper in peace until you decide you want to read the next page.

## What’s HTTPS?:

Since your HTTP request (or, in our analogy, the piece of paper that represents your request) can be read by anyone at certain network junctures, it might not be a good idea to deliver information such as your credit card or password using this protocol. Fortunately, many servers support HTTPS, short for HTTP Secure, which allows you to encrypt data that you send and receive. You can read more about HTTPS on Wikipedia.

HTTPS is important to use when passing sensitive or personal information to and from websites. However, it is up to the businesses maintaining the servers to set it up. In order to support HTTPS, the business must use and pay for a certificate from a Certificate Authority.

codecademy

Teaching the world how to code.

**Company**

- About
- Stories
- We're hiring
- Blog

**Resources**

- Articles
- Schools

**Learn To Code**

- Make a Website
- Make an Interactive Website
- Learn Rails
- Ruby on Rails Authentication
- Learn AngularJS
- Learn the Command Line
- Learn SQL
- SQL: Analyzing Business Metrics
- Learn Java
- Learn Git

- HTML & CSS
- JavaScript
- jQuery
- PHP
- Python
- Ruby
- Learn APIs