1. **Domain Names**

[the anatomy of a URL](https://themeisle.com/blog/what-is-a-website-url/)

https://themeisle.com/

This URL contains two parts:

* **A protocol** (https**:**) is a set of guidelines that a browser follows to send a request to the server.
* The **domain**, themeisle.com, **or URL** to the main website. A domain consists of two parts:
  + the TLD (**top level domain**) which is the **.com** part (or [another domain extension](https://themeisle.com/blog/domain-extensions-guide/)), and
  + the SLD **(second level domain**), themeisle, the name that you buy from a [domain registrar](https://themeisle.com/blog/best-domain-registrars/).

A subdomain contains a second name before the SLD. For instance, if the ThemeIsle blog was hosted on https://**blog**.themeisle.com, the blog would be the **subdomain**.

A subdomain is commonly used to logically separate a website into sections. You can use a subdomain to launch a career site (careers.yoursite.com), a forum (forum.yoursite.com) or for customer support (support.yoursite.com). You may use subdomains to create blogs of different themes too. For instance, [sbnation.com](https://www.sbnation.com/) is a sports news blog. However, it uses blogs like [weaintgotnohistory.sbnation.com](https://weaintgotnohistory.sbnation.com/) and [theshortfuse.sbnation.com](https://theshortfuse.sbnation.com/) for specific teams on different subdomains

 Before you can set up any subdomain, you have to have a main domain. If you don’t have that taken care of yet, here are our guides on [choosing a domain name](https://themeisle.com/blog/how-to-choose-a-domain-name/) and [how to register a domain name](https://themeisle.com/blog/how-to-register-a-domain-name/).

Once you buy a domain, you also buy the rights for subdomains within it. Broadly, these are the steps that you must follow to create a subdomain:

* Come up with the name of the subdomain, enter it as a record in your DNS settings.
* Redirect to the server that hosts your subdomain.

When you are entering a record in your DNS settings, you would notice that www.yoursite.com points to yoursite.com. This essentially makes www a subdomain too!

<https://themeisle.com/blog/fix-broken-links-in-wordpress/>

1. The protocol – *HTTP* or *HTTPS*.
2. The domain name (including the TLD) that identifies a site.
3. The path leading to a specific web page.



Your **main domain** – also known as a primary domain or a root domain – is essentially the name of your website

In Brafton’s case, our **main domain** name is [**brafton.com**](http://brafton.com/). Notice I didn’t say [**www.brafton.com**](http://www.brafton.com/) or [**https://www.brafton.com**](https://www.brafton.com/), which are technically our site URLs.

A subdomain is a division of your website that you want to distinguish with its own unique identity and content. For instance, if Brafton wanted to create a subdomain for our blog page (we don’t), our subdomain name would be [**blog.brafton.com**](http://blog.brafton.com/).

**The structure and components of a URL**

A URL is usually made up of several parts. To understand the structure and the components, we will dismantle the following example URL:



General language use

If we talk about a URL, we usually mean the concrete path to a directory (http://www.domain.com/a-directory/) or a file (http://www.domain.com/documents/study.pdf) on a website.

Subdomain vs. Subfolder

Another common distinction you’ll need to make when it comes to subdomains is the difference between a subdomain and a subfolder.

By now, you already know what a subdomain is, but a subfolder is a bit different. With a subfolder, you’re adding a folder to your existing domain.

So, instead of creating a new subdomain for your blog like “blog.mysite.com,” you’ll use a subfolder instead “mysite.com/blog.”

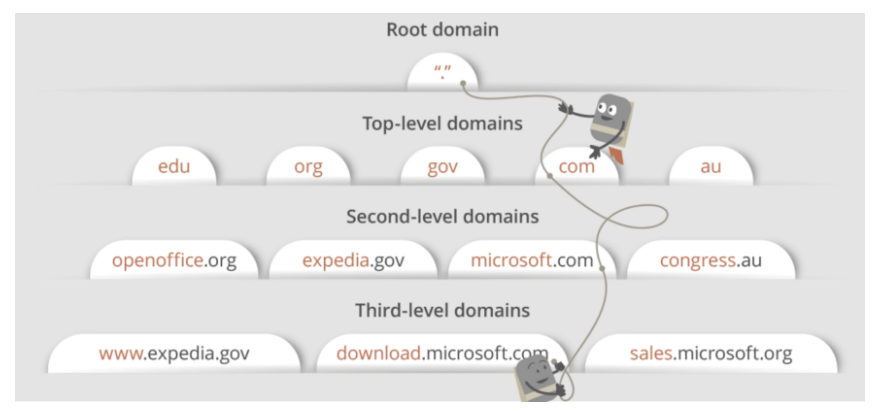
For example, here at HostGator you can access the blog by navigating to “https://www.hostgator.com/blog” , not “blog.hostgator.com”.

When you’re thinking about using a subdomain, you should really view it as creating a separate website. Although subdomains branch off of an existing domain name, they do take more work to build, grow, and maintain.

A subfolder is a way to organize your site more easily. Think of it as creating categories for your blog and blog posts. If you have a sports website, you could create subfolders for each sport you cover. So, you’d end up with a URL structure something like the following: “sports.com/basketball,” “sports.com/football,” “sports.com/hockey,” and on and on. Each page could operate as its own separate sports-specific blog with each page filled with unique content about that sport.

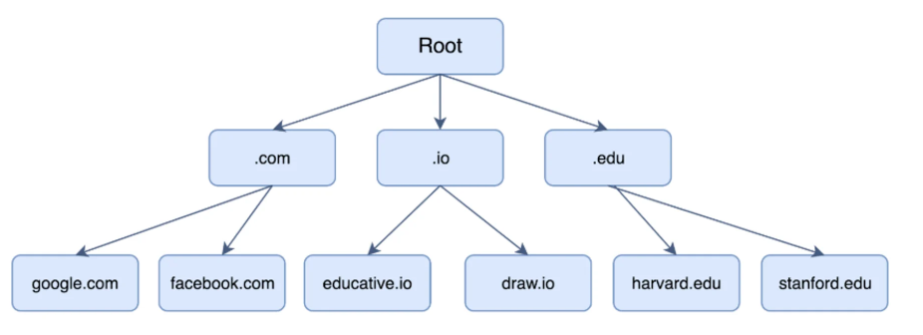
1. **What Happens When You Type in a URL**
2. You enter a URL into a web browser
3. The browser looks up the IP address for the domain name via **DNS**
4. The browser sends a HTTP *request* to the server
5. The server sends back a HTTP *response*
6. The browser begins rendering the HTML
7. The browser sends requests for additional objects embedded in HTML (images, css, JavaScript) and repeats steps 3-5.
8. Once the page is loaded, the browser sends further async requests as needed.
9. **DNS (Domain name system)**

The browser extracts the domain name from the URL.



## DNS lookup to find IP address

After hitting the URL, the first thing that needs to happen is to resolve IP address associated with the **domain name**.



There are 4 local cache to check :

1. The browser’s local cache is checked
2. The operating system’s cache is checked
3. The router is checked for the record.
4. Lastly, the query is sent to the Internet Service Provider (ISP) for it to check its cache.

Hence, if the record cannot be found locally, a full DNS resolution is conducted as follows:

1. The first point of contact for a full resolution is a root server. As of the writing of this post, 1017 instances of root servers exist.
2. The root server returns the IP address of the relevant top level domain server.
3. The top level domain returns the IP address of the second level domain server.
4. The second-level domain server contains the DNS record of the server we are looking for. The second-level domain server returns the IP address to the browser.

This is the overview, but there are **four layers** through which this domain name query goes through. Let’s understand the steps:

1. After hitting the URL, the **browser cache** is checked. As browser maintains its DNS records for some amount of time for the websites you have visited earlier. Hence, firstly, DNS query runs here to find the IP address associated with the domain name.

2. The second place where DNS query runs in **OS cache** followed by **router cache**.

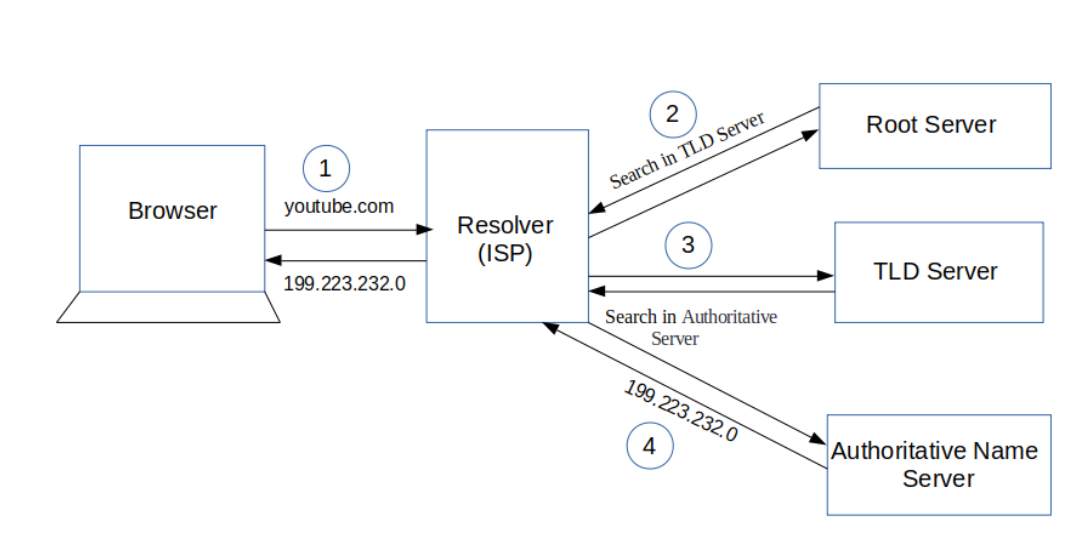
3. If in the above steps, a DNS query does not get resolved, then it takes the help of resolver server. **Resolver server** is nothing but your ISP (Internet service provider). The query is sent to ISP where DNS query runs in**ISP cache.**

4. If in 3rd steps as well, no results found, then request sends to **top or root server** of the DNS hierarchy. There it never happens that it says no results found, but actually it tells, from where this information you can get. If you are searching IP address of the top level domain (.com,.net,.Gov,. org). It tells the resolver server to search **TLD server** (Top level domain).

5. Now, resolver asks TLD server to give IP address of our domain name. TLD stores address information of domain name. It tells the resolver to ask it to **Authoritative Name server.**

6. The authoritative name server is responsible for knowing everything about the domain name. Finally, resolver (ISP) gets the IP address associated with the domain name and sends it back to the browser.

After getting an IP address, **resolver stores it in its cache** so that next time, if the same query comes then it does not have to go to all these steps again. It can now provide IP address from their cache.



## TCP connection initiates with the server by Browser

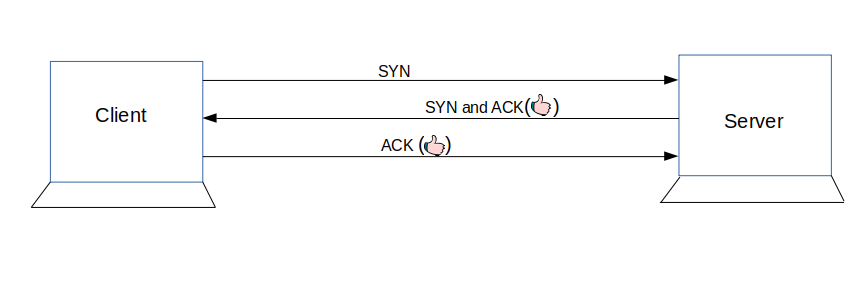
Once the **IP address** of the computer (where your website information is there) is**found**, it **initiates connection** with it. To communicate over the network,**internet protocol** is followed. **TCP/IP** is most common protocol. A connection is built between two using a process called **‘TCP 3-way handshake’**. Let’s understand the process in brief:

1. A client computer sends a **SYN message** means, whether second computer is open for new connection or not.

2. Then**another computer**, if open for new connection, it sends **acknowledge message** with SYN message as well.

3. After this, **first computer** receives its message and acknowledge by **sending** an**ACK message.**

To better  understand, look below diagram.



## What is HTTP?

These requests follow a ‘protocol’ or ‘rules of communication’ called **HyperText Transfer Protocol (HTTP)**. This protocol dictates the format of the messages, when what message is sent, appropriate responses, and how messages are interpreted. **HTTP messages** are of two types: **request** and **response**.

An HTTP **request message** consists of a **request line** and **headers**. The message starts with a request line and is followed by headers. Here’s a sample HTTP request:

GET /path/to/file/index.html HTTP/1.1  
  
Host: www.educative.io  
  
Connection: close  
  
User-agent: Mozilla/5.0  
  
Accept-language: fr  
  
Accept: text/html

The **request line** consists of a **request method**, a **path**, and the **HTTP version**.

The request method, GET, in the example above tells the server what to do. GET, for example, tells the server that the client wants to get the resource found at the given file path.

Other examples of request methods include DELETE, which tells the server to delete a resource at the given path, and PUT, which tells the server to put a supplied resource at the given path. The HTTP version is also specified to cater for the differences between each.

Next come the HTTP **headers**. Headers allow the client to communicate extra information such as the server type and the date. Each header is on a seperate line and contains a name and value, separated by a colon.

There are many headers which provide different functions. For example, the connection header indicates whether user is on a HTTP connection type.

The server then sends an HTTP **response message**. Here’s a sample response message:

HTTP/1.1 200 OK  
  
Connection: close  
  
Date: Tue, 18 Aug 2015 15: 44 : 04 GMT  
  
Server: Apache/2.2.3 (CentOS)  
  
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT   
  
Content-Length: 6821  
  
Content-Type: text/html  
   
[The object/file that was requested]

Response messages consist of a **status line** to start with, followed by a number of **headers**, followed by a blank line and ends with a resource if any was requested.

The **status line** consists of the **HTTP version** and a **status code**. There are a few types of status codes. A common example is the infamous 404 Not Found status code.