Set 16. Web Protocols

Skill 16.01: Describe the Web and its relationship with Internet Protocols

Skill 16.01 Concepts

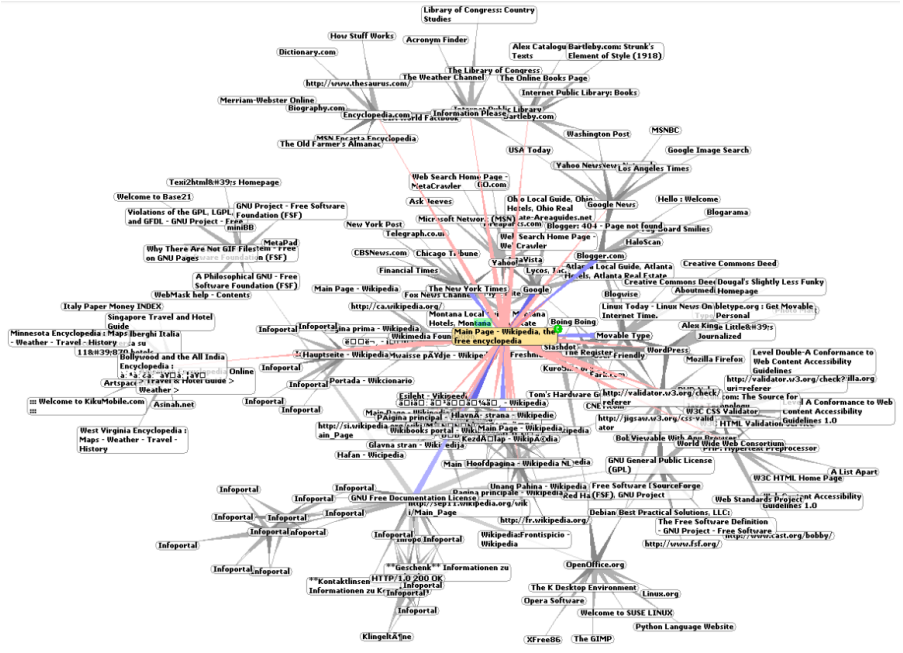
[Skill 16.01 Exercise 1]

When most of us talk about using the "Internet", we're typically talking about a specific part of the Internet: the **World Wide Web** (WWW, or simply, the **Web**).

The Web is a massive network of webpages, programs, and files that are accessible via URLs.

We call it a web because of its vast interconnectedness. Starting from one URL, such as http://wikipedia.org, we can follow links to eventually reach millions of webpages from across the globe.

Here's a tiny portion of that web from 2004:



A web browser loads a webpage using various protocols:

1. It uses the **Domain Name System (DNS) protocol** to convert a domain name into an IP address.
2. It uses the **HyperText Transfer Protocol (HTTP)** to request the webpage contents from that IP address.

It may also use the **Transport Layer Security (TLS) protocol** to serve the website over a secure, encrypted connection.
The web browser uses these protocols *on top* of the Internet protocols, so every HTTP request also uses TCP and IP.
The Web is just one of the applications built on top of the Internet protocols, but it is by far the most popular.

Skill 16.02: Explain the Domain Name System (DNS)

Skill 16.02 Concepts

IP addresses are how computers identify other computers on the Internet. IP addresses aren't particularly human-friendly, though. Who wants to memorize an address like 74.125.20.113? Or the even longer IP v6 addresses?

The **Domain Name System (DNS)** gives us humans an easy way to identify where we want to go on the Internet.
We simply type in a domain name like "www.wikipedia.org", and our computer connects us to the computers powering Wikipedia:

A **domain name** is a human-friendly address for a website, something that's easy for us to remember and type in.

Anatomy of a domain name

Each domain name is made up of parts:

[third-level-domain].[second-level-domain].[top-level-domain]

There are a limited set of top level domains (TLDs), and many websites use the most common TLDs, ".com", ".org", and ".edu".

The second level domain is unique to the company or organization that registers it, like "wikipedia" or "code".
The third level domain is also called a subdomain, because it's owned by the same group and that URL often directs you to a subset of the website, like "m.wikipedia.org" (mobile-optimized Wikipedia) or "studio.code.org" (the learning management site for Code.org).

Domains ↔ IP Addresses

Behind the scenes, each domain name maps to an IP address. When we type a URL in the address bar of our browser, the computer has to figure out its IP address.

The computer can't store a database of more than 300 million domain names locally, so it goes through a multi-step process to find out the IP address.

Step 1: Check the local cache

If you've visited a website once, there's a fairly good chance you'll visit it again. That's why computers keep their own local cache of domain name to IP mappings. The cache stays small, because it kicks out domains you haven't visited in a while or domains that send down expiration dates.

Below is a snippet from my cache,

```
www.google.com
----------------------------------------
Record Name . . . . . : www.google.com
Record Type . . . . . : 1
Time To Live  . . . . : 1657
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 216.239.38.120
```

If you are on a windows machine, you can access this by typing *ipconfig /displaydns* in the command prompt.

Step 2: Ask the ISP cache

Every ISP (Internet Service Provider) provides a domain name resolving service and keeps its own cache. Perhaps you've never visited a particular website but your neighbor just did, so the ISP can lookup the domain name mapping from their visit.

If it's not in the ISP's cache, then it's off to the next step.

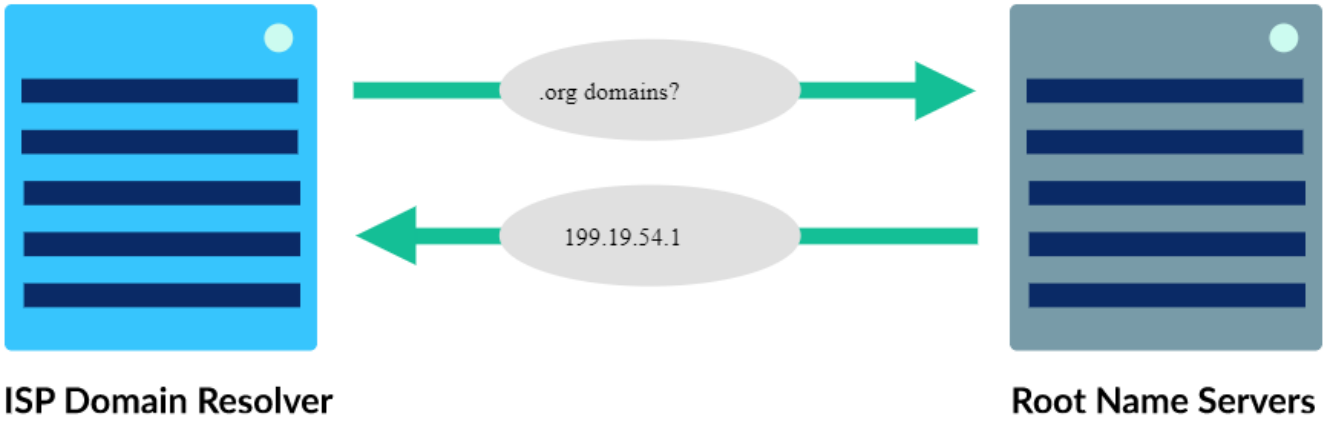
Step 3: Ask the name servers

There are domain name servers scattered around the globe that are responsible for keeping track of a subset of the millions of domain names.

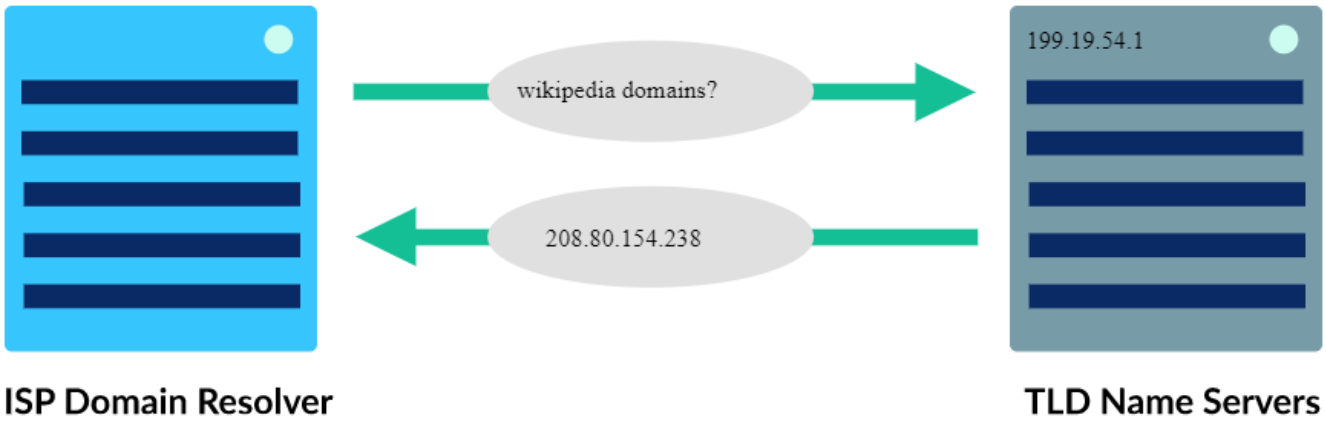
The servers are ordered in a hierarchy:

Root name servers → TLD (Top Level Domain) name servers → Host name servers.

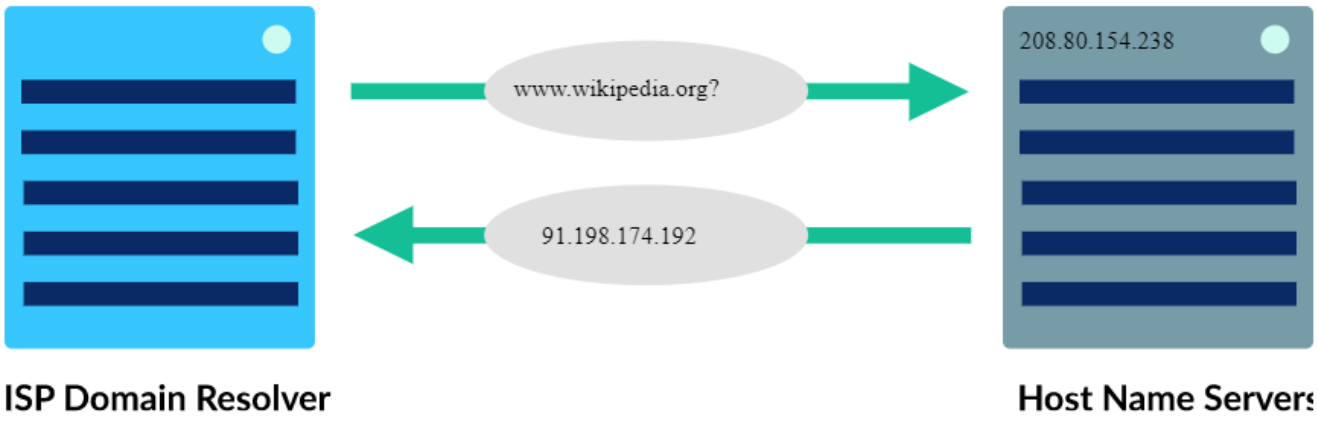
The ISP starts by asking the root name servers: "hey, which name server knows about .org domains?" The root name server responds with the IP address of a TLD name server that tracks ".org" domains.



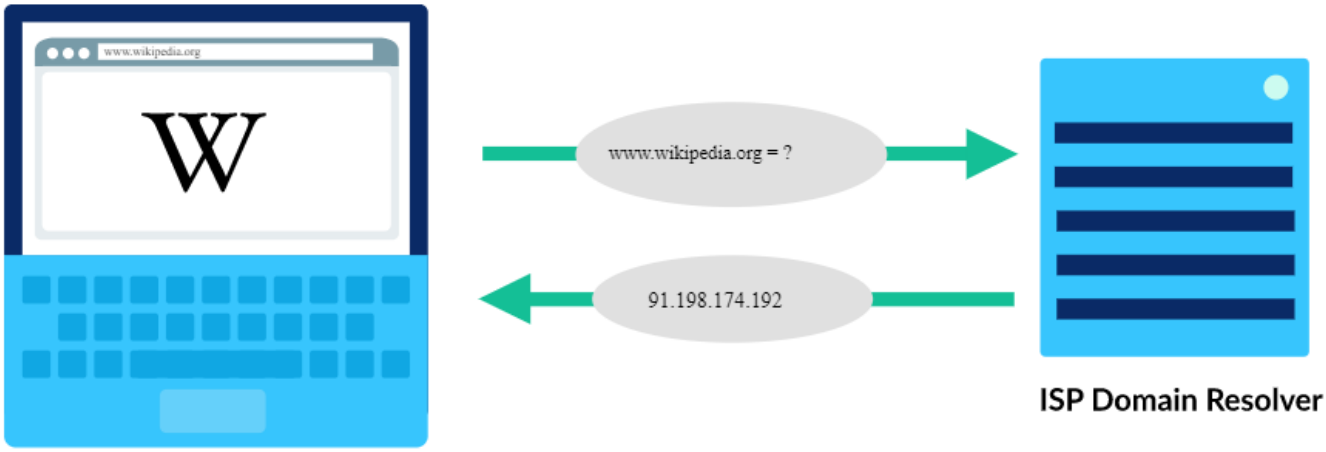
Next, the ISP asks the TLD name server: "so, who knows about wikipedia domains?" The TLD name server responds with the IP address of a host name server that contains the "wikipedia" records.



Finally, the ISP asks the host name server: "okay, so where's www.wikipedia.org?" The host name server responds with an exact IP address.



The ISP sends the IP address back to the requesting computer, and now our computer can successfully connect with the computer powering that domain.

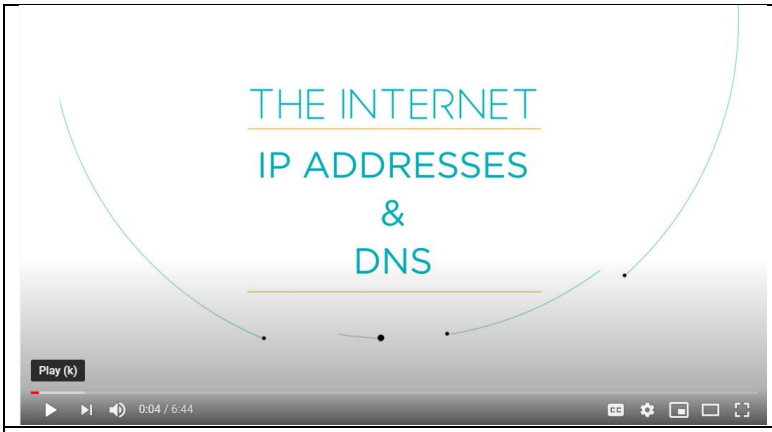


If that sounds like quite a process: yes, it is! But don't worry, it's not done that often. A lot of information is cached along the way, so it's rare that a DNS lookup has to go through so many steps.

When a lookup does have to go through all the steps, there are multiple name servers that can answer each question, so a computer doesn't have to wait too long for a response or worry about a name server going down.

We've had the domain name system since 1985, and it's scaled impressively to match the growth of the Internet, thanks to its hierarchy, redundancy, and caching.

The video below reviews IP addresses and how DNS is used to associated these addresses with useful names.



https://www.youtube.com/watch?v=5o8CwafCxnU

Skill 16.02 Exercises 1 & 2

Skill 16.03: Explain Hypertext Transfer Protocol (HTTP)

Skill 16.03 Concepts

Whenever you visit a page on the web, your computer uses the **Hypertext Transfer Protocol (HTTP)** to download that page from another computer somewhere on the Internet.

Let's step through that process.

Step 1: Direct browser to Uniform Resource Locator (URL)

When we want to browse the web, we can use many types of computers (like laptops, desktops, and phones), as long as the computer has a **browser** application installed.

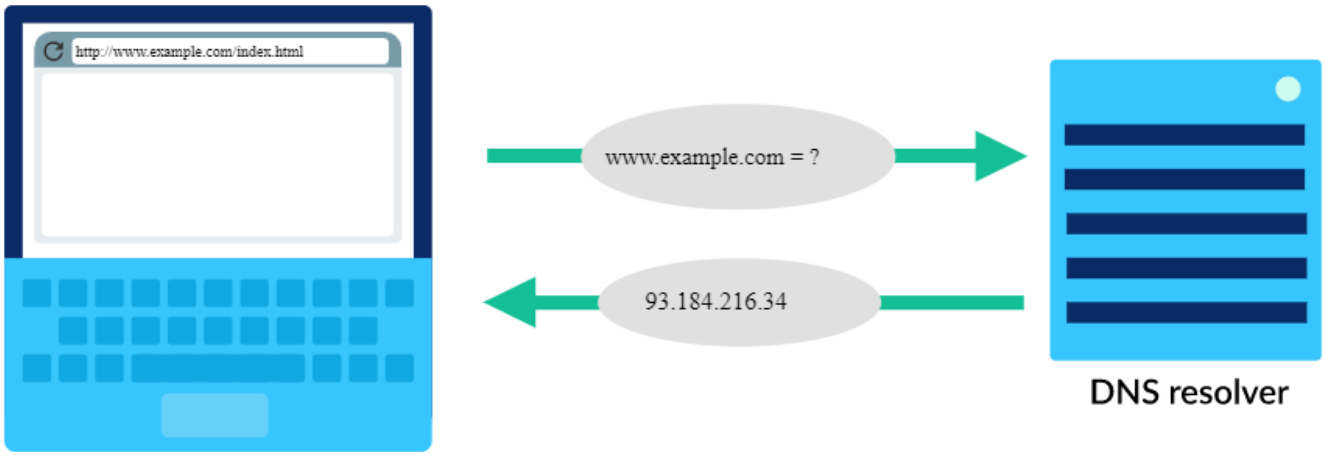
The user either types a Uniform Resource Locator (URL) in the browser or follows a link from an already opened page:

Notice something about that URL: it starts with "http". That's a signal to the browser that it needs to use HTTP to fetch the document for that URL.

Step 2: Browser looks up IP

We typically type nice human-friendly URLs into browsers, like "code.org" and "wikipedia.org". Those domain names map to IP addresses, the true location of the domain's computers. That's handled by the Domain Name System.

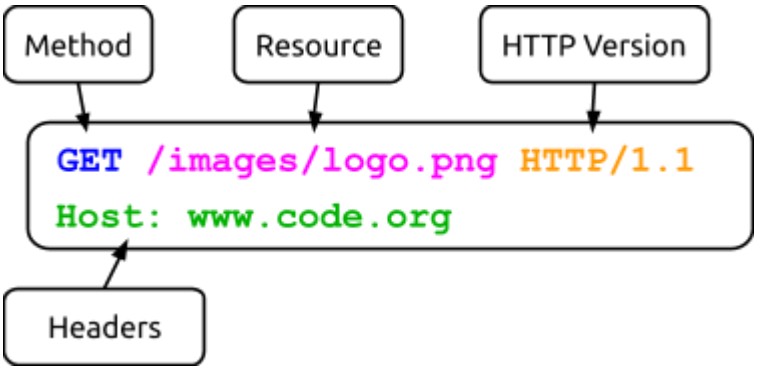
The browser uses a DNS resolver to map the domain to an IP address:



Step 3: Browser sends HTTP request

Once the browser identifies the IP address of the computer hosting the requested URL, it sends an **HTTP request**.

An HTTP request can be as short as two lines of text:



- **Method:** An HTTP request will begin with a method, which indicates what the client wants the server to do. The two most common methods are:

Method	Description
GET	Requests a specified web page or other data
POST	Submits some data for the server to accept or process

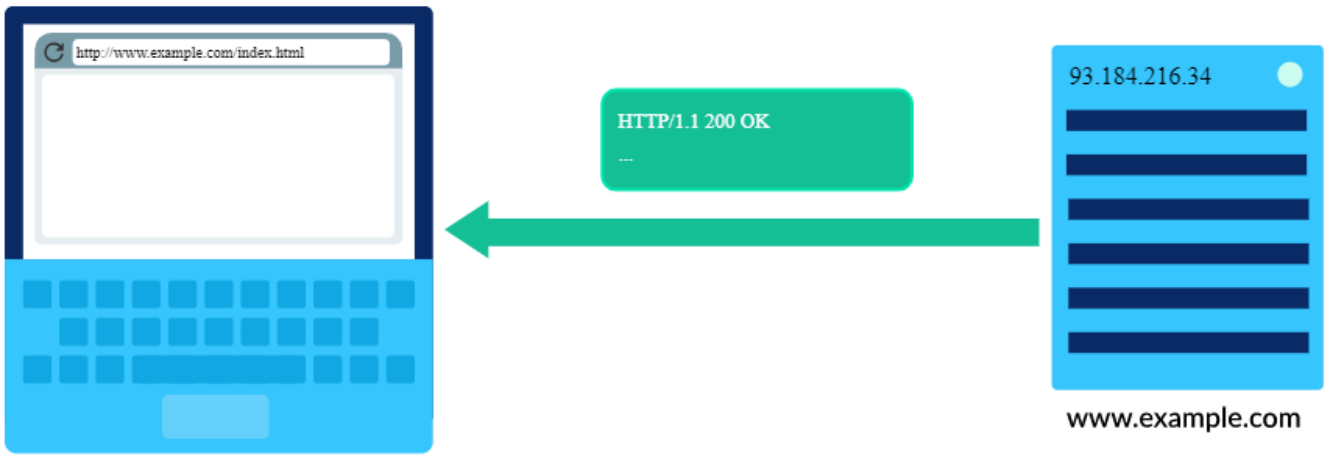
- **Resource:** The name of the resource you wish to access. In the example above, the request is for a .png image called "logo" located in the folder "images."

- **HTTP Version**: An indication of the version of HTTP being used; in this example it is HTTP 1.1.

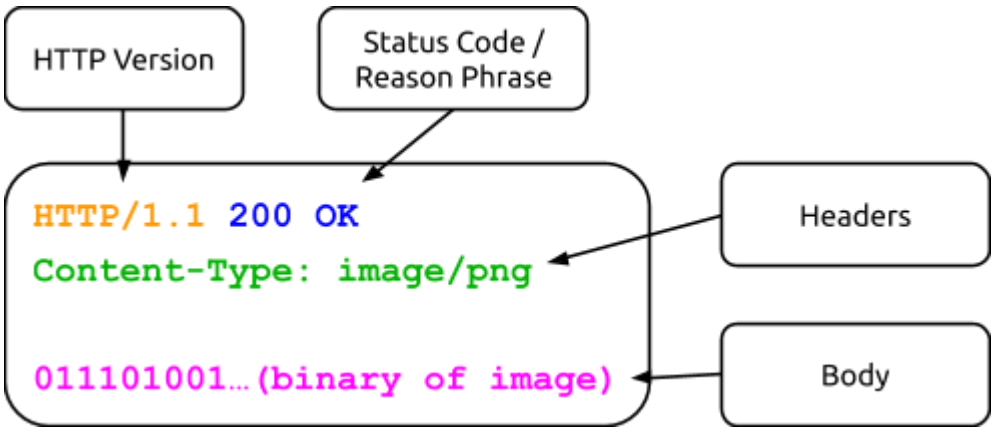
- **Headers:** Additional information included to help the server interpret the request. In the above example, "Host" is included, but many more can be added by placing them on additional lines.

Step 4: Host sends back HTTP response

Once the host computer receives the HTTP request, it sends back a response with both the content and metadata about it.



The response will be sent entirely in ASCII-text and must be correctly formatted. Here's a sample HTTP Response:



An HTTP Response has multiple components:

- **HTTP Version:** An indication of the version of HTTP being used; in this example it is HTTP 1.1.

- **Status Code and Reason Phrase:** A number and phrase indicating how the server responded to the request. Some common Status Codes / Reason Phrases are below.

  Since users have a habit of typing URLs incorrectly, 404s happen frequently, so websites often have fun 404 webpages. Check out these, https://www.gog.com/error/404, https://www.cloudsigma.com/404-error/ , https://pureemaison.com/en/blah

Status Code / Reason Phrase	Meaning
200 Ok	Request was handled successfully
404 Not Found	Server could not find anything matching request
301 Moved Permanently	Requested data was permanently moved
302 Moved Temporarily	Requested data was temporarily moved
500 Internal Server Error	Error by the server prevented fulfilling request

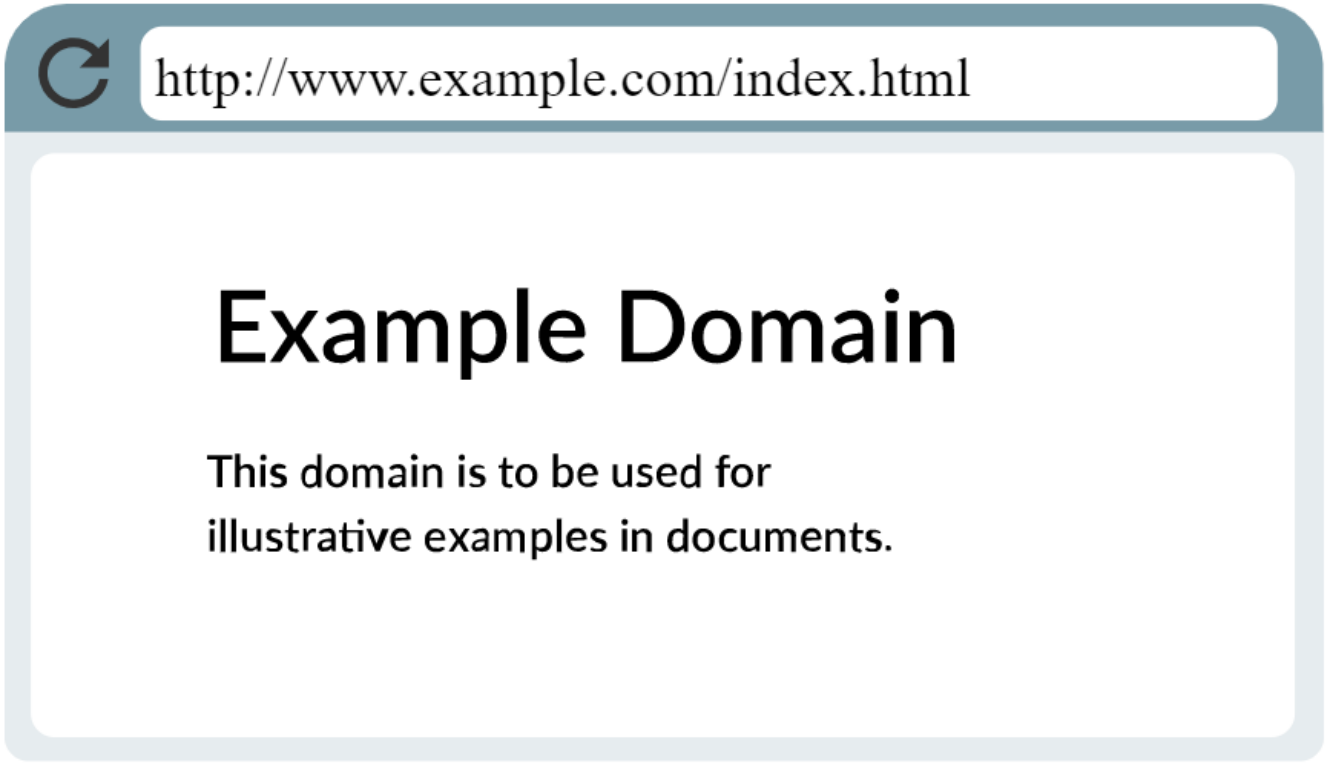
- **Headers:** Additional information about the response. The example above includes the header “Content-Type,” indicating the type of content included in the response. A common type on the web is "text/html", as all webpages are HTML text files. Other types are possible, like images ("image/png"), videos ("video/mpeg"), script ("application/javascript") and anything else you can load in your browser.

- **Body:** Following the headers, the response may include some data that was requested. In this example the actual binary representation of the image would be included in the response, so this response could be quite long. If the response were text/HTML, HTTP would write out the actual document requested. Like this simple HTML file below,

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example Domain</title>
  </head>
  <body>
    <h1>Example Domain</h1>
    <p>This domain is to be used for illustrative examples in
documents.</p>
  </body>
</html>
```

Step 5: The browser renders the response

The browser now has all the information it needs to render the document requested.



[Skill 16.03 Exercises 1 thru 3]

Skill 16.04: Explain the relationship between HTTP and IP/TCP

Skill 16.04 Concepts

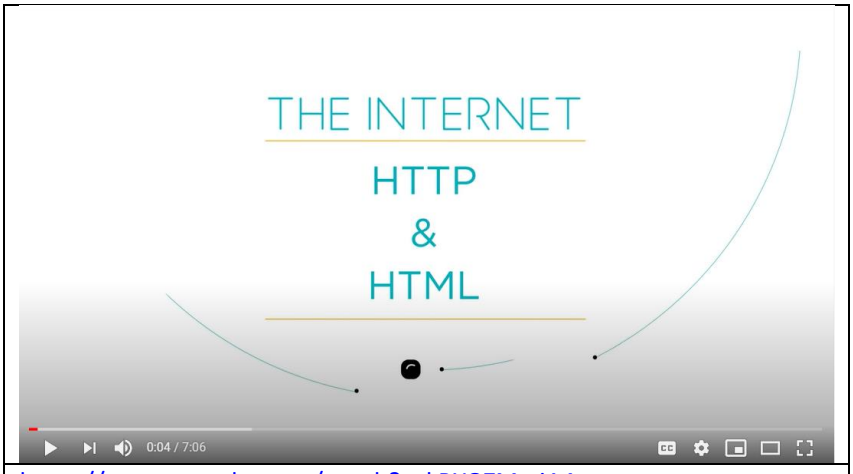
HTTP is a protocol that's built *on top* of the TCP/IP protocols.

Each HTTP request is inside an IP packet, and each HTTP response is inside another IP packet--or more typically, multiple packets, since the response data can be quite large.

There are many other protocols built on top of TCP/IP, like protocols for sending email (SMTP, POP, IMAP) and uploading files (FTP).

All of these protocols enable us to use the Internet to connect with other computers in useful ways, and to communicate and collaborate across wide distances.

The video below recaps everything we just learned today!



https://www.youtube.com/watch?v=kBXQZMmiA4s