How The World Wide Web Works

HTTP

Hyper Text Transfer Protocol.

HTTP is a protocol, which is to say a set of rules and formats, that allows one computer to get content from another computer.

Client and Server

HTTP is assymetric. One computer requests the content, while the other computer serves the content.

The computer requesting the content is the "client."

The computer serving the content is the "server."

You can think of HTTP as similar to the transaction that happens in a restaurant. Information needs to flow both ways between client and server, but the two parties have very different roles in the transaction. Thus it is an assymetric operation.

Visiting a web page

When you visit twitter.com:

- Your browser (i.e. Firefox) is the client.
- The server is some custom-built software running in Twitter's cloud infrastructure somewhere.

Note that the client and the server are just software, they don't need to be (but usually are) running on separate physical hardware.

Request/Response

- The format that the client uses to send information to the server is called a "request".
- The format that the server uses to send information to the client is called a "response".

When you visit twitter.com, your browser (the client) makes an HTTP request, and Twitter's server returns an HTTP response

HTTP Methods

There are several types of requests the client can make. The type of the request is called the "method." There are several HTTP methods the client can choose from, but for now we will focus on one method:

GET

GET requests are the most common type of request when browsing the internet. It's a way for your client to say "give me some content."

Requests

GET requests consist of:

- URL
- Additional Metadata

Responses consist of:

- Body (content)
- Status Code
- Additional Metadata

URLs

A URL (Universal Resource Locator) is a type of URI (Universal Resource Identifier).

A URL is meant to be a unique identifier for a "resource", or a piece of content, that can be accessed via the world wide web.

A URL consists of:

protocol	[subdomain.]	domain	[:port]	[/path]	[?query]
http://	blog.	science.com	:80	/foo	?bar=baz

Protocol

HTTP is unencrypted.

HTTPS is encrypted.

Domains

Every computer connected to the internet lives at a certain address (domain/subdomain pair).

Ports

Ports are like doors.

Every computer connected to the internet has thousands of potential ports.

A server is a piece of software that runs on a computer, and "listens" for HTTP requests on a certain port.

Default Ports

- HTTP requests default to port 80
- HTTPS requests default to port 443

Most web pages serve content on the default ports, and as such, we drop the port from the URL.

Servers

It's the internet's job to direct the HTTP request to the right address.

It's the computer's job to direct the HTTP request to the right port.

Then, it's up to the server listening on that port to send a response.

Path

From one server, we often want to serve lots of different content.

Like the menu of a restaraunt.

A path is a unique identifier for each piece of content the server can provide (each item on the menu)

Paths are trees! Just like menus are organized categorically by their creators (burritos, tacos, tostadas), content within the server is organized with a taxonomy that is created by whoever wrote the server software.

Query String

In a restaraunt, the food might be like:

- Burritos
  -- Chimichangas
  -- Grande
  -- Children's

While each burrito might have the following options:

Meat: Asada, Chorizo, Lengua, Soyrizo, Tofu
Cheese: Queso Fresco, Vegan Cheese

Query String

Sometimes we want to allow options for each piece of content. A query string is a collection of key-value pairs (like a Python dictionary) that describe the content requested.

A Grande Burrito with Soyrzo and Vegan Cheese would look like this in a URL:

https://yum.yes/burritos/grande?meat=soyrizo&cheese=vegan

Servers

Paths and query strings are nothing but a way for the server to organize its content for the client. There are no strict rules, as long as the client can learn the format and the server is happy with the organization.

Headers

All metadata sent in HTTP requests and responses lives in the "headers".

The headers are just a collection of key-value pairs.

Status codes live in the headers. The key is "status" and the value is the code number!

Another common header in HTTP Responses exists under the key "Content-Type". This tells the client what type of content is in the body, and thus how to decode it.

Content-Types

There are 3 main content types you should know about:

- Plain text
- HTML
- JSON

Plain Text

Plain text is the simplest form of content. It is not used very often in actual applications.

JSON

JavaScript Object Notation.

An "object" in JavaScript is similar to a Dictionary in python. It's an associative data structure, a collection of key-value pairs!

JSON is the most common format for sending data over HTTP when the data is meant to be consumed by a computer program, rather than presented for a human to view.

API's (application programming interfaces) commonly use JSON to send data.

JSON

An example of JSON:

```
{
    "id": "b4vd345s45gd",
    "tweets": [12543, 9878945, 90384],
    "profile": {
        "name": "Man Onthe Moon",
        "location": "moon"
    }
}
```

HTML

Hyper Text Markup Language.

HTML is a format used to encode content, so that it can be displayed for humans to read in a web browser.

HTML tells the browser what to display, and how to display it.

For example: if you have a heading (title) followed by two paragraphs. You need to tell the browser not only about the order of the text, but to make the heading larger and bolder, and to separate the paragraphs with a new line!

HTML

HTML is a tree. It organizes all the content for the browser into a hierarchical taxonomy.

```
                   |-- meta qux
        |-- head --|
        |          |-- script baz
html --|
        |          |-- div.foo
        |-- body --|
                   |-- div.bar
```

HTML

The root node is called "html", which has only two possible child nodes, "head" and "body." Those two nodes can have unlimited children.

```
<html>
    <head>
        ...
    </head>
    <body>
        <div class="foo"></div>
        <div class="bar"></div>
    </body>
</html>
```

HTML or JSON???

JSON and HTML are used in two very different contexts:

- HTML is used to create a "UI", a user interface, for consumption by human eyes.
- JSON is used in an "API", an application programming interface, for consumption by other computer programs.

It should be clear that, in general, you should prefer APIs for getting data, wherever they are available. Getting data out of HTML, gotten from websites, is referred to as "scraping".