Intro to WWW

created for CSCI 445

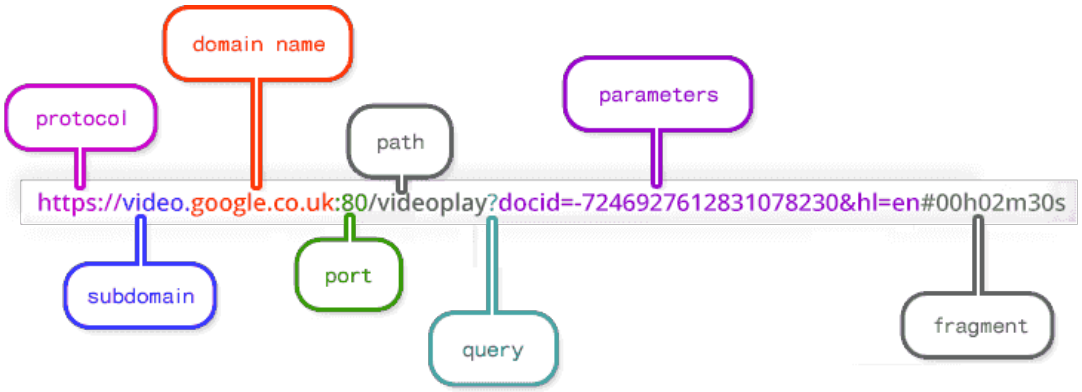
What's in a URL?*

- URL standards for Uniform Resource Locator.
  - Uniform – consistent usage across browsers etc.
  - Resource – something you want to access, like a webpage
  - Locator – all the info to find the resource among the vast array available
- You've all used URLs, either typed into the browser's address bar or by clicking on a link. Example:
  - http://cs-courses.mines.edu/csci445/
  - A *protocol* specifies the structure (i.e., rules) for communication between a client and server. Most URLs use http (not secure) or https (secure) as the protocol.
  - A *domain* is the **unique** identifier for a site, which in the example is mines.edu. This must be unique so that routers can find it (routers within the domain will forward messages to the subdomains – that's beyond our scope).
  - The *top-level domain* is the last part of the domain name, edu in this example. The most common TLDs are .edu, .org, .net and .gov.
  - Domains are often organized into subdomains, which are often stored as directories on a server. The subdomain in the example is cs-courses.
  - Within the subdomain there can be a number of files and/or directories, which are the path. In the example, csci445 is part of the path (with **index.html** implied – i.e., if no file is specified, the user will see index.html).

* some material from: https://doepud.co.uk/blog/anatomy-of-a-url

More complex URLs

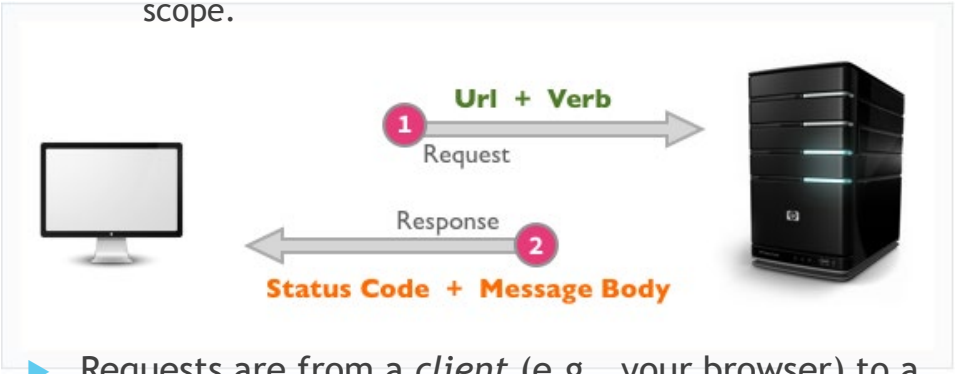
- Data can be passed from the webpage to a server via a query string.
  - The query string is separated from the rest of the URL by a ?
  - Each parameter is listed as an id=[value]
  - In the example below, docid=72469…
  - Parameters are separated by &
- It's good to know what a query string is, but we won't deal with them until the backend/PHP modules.
- Port (communication endport) is rarely used. Most http requests use port 80, https (secure) requests use port 443.



* some material from: https://doepud.co.uk/blog/anatomy-of-a-url

So what is HTTP?

- HyperText Transfer Protocol
- Structures request/response traffic
  - technically uses TCP (Transmission Control Protocol) to establish bit-level communication. That's beyond our scope.

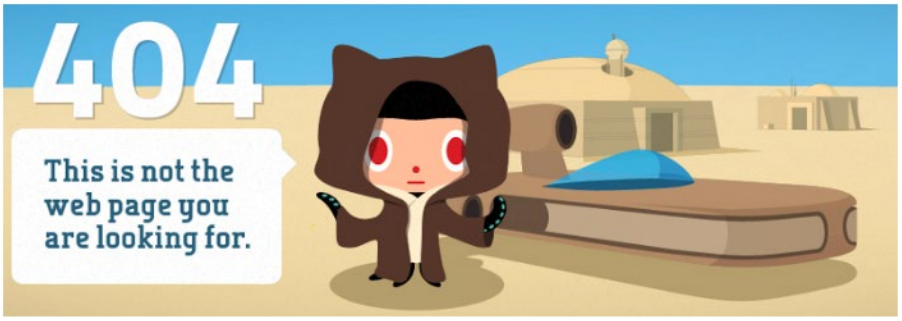


- Requests are from a *client* (e.g., your browser) to a *server* (where the web pages etc. are stored).
  - In our PHP module, the school will provide a server named Luna that we can access. You can also set up a server on your own computer.
  - For the front-end part of the course, we'll just load files directly from our file systems into the browser.
- The previous slides discussed URL (finds the resource within a particular directory on a server)
- What does it do when it finds it? That's the **verb**.
  - GET. Simple browser requests, like getting info about products
  - POST. Data from submitted forms.
  - PUT and DELETE. We won't cover.
- And what does the server return?
  - A status code (next slide)
  - and message body (varies based on request)

Image from:
https://medium.com/@jen_strong/the-request-response-cycle-of-the-web-[illegible]6e9047

HTTP Status Codes

- A server can return a number of responses back to the client. They fall into five categories:
  - 1xx - Informational Responses
  - 2xx - Success
  - 3xx - Redirection
  - 4xx - Client Errors
  - 5xx - Server Errors
- You've probably all seen 404. This is a client error (wrong url… although it might not be the user's fault, if someone has deleted page)



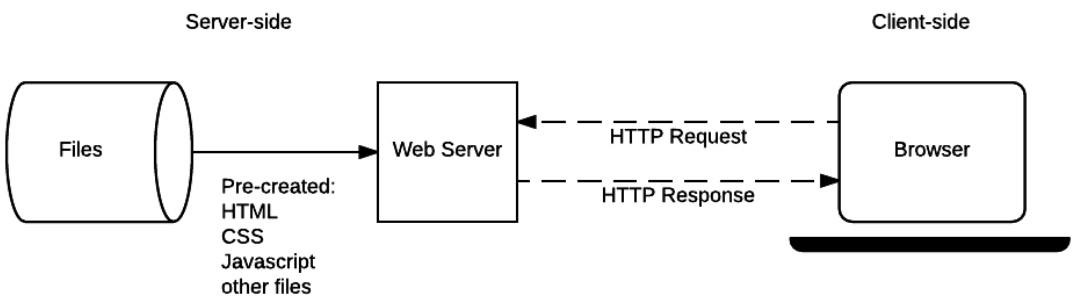
- What we hope for is Success. When we do Ajax later this semester, we'll be checking for a success return.
- More detail about status codes: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Server Response

- Example return from an HTTP get for a webpage*
  - Status code is on the first line
  - Various other negotiated parameters (beyond our scope)
  - Followed by the actual HTML code, which the browser knows how to display
- Similar process when retrieving CSS etc. (see figure)

```
HTTP/1.1 200 OK
Server: Apache
X-Backend-Server: developer1.webapp.scl3.mozilla.com
Vary: Accept,Cookie, Accept-Encoding
Content-Type: text/html; charset=utf-8
Date: Wed, 07 Sep 2016 00:11:31 GMT
Keep-Alive: timeout=5, max=999
Connection: Keep-Alive
X-Frame-Options: DENY
Allow: GET
X-Cache-Info: caching
Content-Length: 41823
```

```
<!DOCTYPE html>
<html lang="en-US"  >
<head prefix="og: http://ogp.me/ns#">
```



Example from:
https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview

Scripts – quick overview

- Scripts are small chunks of code that add functionality to our website
- Client-side scripts – FRONT END
  - typically written in JavaScript or jQuery
  - can't access resources on the server (although we'll see with Ajax that we can call a server-side script)
  - often respond to user input, such as validating data before a form is sent for processing (e.g., make sure an email is formatted correctly), performing calculations, etc.
  - typically faster than back end, because no need to communicate with a server
- Server-side scripts – BACK END
  - various languages can be used, some examples are PHP and Python
  - often interact with resources on the server, such as the database
  - typically handle HTTP request (e.g., POST a form) and generate the HTTP response (e.g., "Order processed")

- You may want to read more about front-end, back-end and full-stack: https://blog.udacity.com/2014/12/front-end-vs-back-end-vs-full-stack-web-developers.html

Some fun history (optional) *

- Take a look at the world's first web page and learn about the birth of the World Wide Web
- The first webpage was 2.17 KB. To compare, CNN is 137.91 KB, Facebook is 144.35 KB, and ESPN is 358.07 KB. Be conscious of the amount of data your page needs to transfer!
- To fully understand the role of HTML in the context of the WWW, use the line-mode browser .
- Glad we have more complex browsers and CSS? I sure am!

* Thanks to Dr. Paone for finding these tidbits