11. How the Web Works
Part 1
Your interface to the web

- Your web browser contacts a web server
A 10,000 Foot View of Technologies

• Where things run:

  HTML / CSS

  JavaScript
  (Angular/React)

  Browser Extensions

  HTTP(S)

  Python (Django) / CGI (Perl) / PHP / Node.js / Ruby on Rails

  Databases (MySQL)
The Anatomy of a Webpage

- view-source:https://www.cs.uchicago.edu/
- HTML (hypertext markup language)
  - Formatting of a page
  - All sorts of formatting: `<div><p>Hi</p></div> <br />`
  - Links: `<a href="blaseur.com">Click here</a>`
  - Pictures: `<img src="unicorn.jpg" />`
  - Forms
- HTML 5 introduced many media elements
The Anatomy of a Webpage
The Anatomy of a Webpage

- CSS (cascading style sheets)
  - `<link href="/css/main.css?updated=20181020002547" rel="stylesheet" media="all">`

- id (*intended to be unique*)
- class (not intended to be unique)
The Anatomy of a Webpage

- DOM (document object model)
Typing Something into a Browser:

- DNS (domain name service)
  - www.cs.uchicago.edu resolves to IP address 128.135.164.125

- URL (uniform resource locator)
  - https://www.cs.uchicago.edu/test.html
    - Protocol: https
    - Hostname: www.cs.uchicago.edu
    - Filename: test.html
    - Default file name if none listed: index.html (and similar)
HTTP Request

• HTTP = Hypertext Transfer Protocol
• Start line: method, target, protocol version
  – GET /index.html HTTP/1.1
  – Method: GET, PUT, POST, HEAD, OPTIONS
• HTTP Headers
  – Host, User-agent, Referer, many others
• Body (not needed for GET, etc.)
• In Firefox: F12, “Network” to see HTTP requests
HTTP Request

- GET /index.html HTTP/1.1

Activity initiation

HTTP/1.x message

Translation into HTTP

Binary framing

HTTP/2 stream (composed of frames)
HTTP Response

• Status
  – 200 (OK)
  – 404 (not found)
  – 302 (redirect)

• HTTP Headers

• Body
HTTPS

- Simply an HTTP request sent over TLS!
  - That is, the request and response are encrypted
Keeping State Using Cookies

• Cookies enable persistent state
• Set-Cookie HTTP header
• Cookie HTTP header
  – Cookie: name=value; name2=value2; name3=value3
• Cookies are automatically sent with all requests your browser makes
• Cookies are bound to an origin (only sent to the origin that set them)
Keeping State Using Cookies

• Session cookies (until you close your browser) vs. persistent cookies (until the expiration date)

• *Secure* cookies = only sent over HTTPS, not HTTP

• *HTTPonly* cookies are not accessible to JavaScript, etc.

• View cookies: “Application” tab in Chrome developer tools, “Storage” in Firefox
Authorization Tokens = Cookies

- You log into a website, and it presents you an authorization token (typically a hash of some secret)
- Subsequent HTTP requests automatically embed this authorization token
Other Ways to Keep State

• Local storage
• Flash cookies
• (Many more)
HTTPS

• An extension of HTTP over TLS (i.e., the request/response itself is encrypted)

• Which CAs (certificate authorities) does your browser trust?
  – Firefox: Options → Privacy & Security → (all the way at the bottom) View Certificates