15. How the Web Works (Part 2)
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">`

• view-source:https://www.cs.uchicago.edu/css/main.css?updated=20181020002547

• 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

• https://www.cs.uchicago.edu/
  – Protocol: https
  – Hostname: www.cs.uchicago.edu
  – Default file name (since none is 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

PUT /create_page HTTP/1.1
Host: localhost:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: text/html
Content-Length: 345

Body line 1
Body line 2
...

HTTP/2 stream (composed of frames)

Frame
Type=HEADERS

Frame
Type=CONTINUATION

Frame
Type=CONTINUATION

Frame
Type=DATA

Frame
Type=DATA

Frame
Type=DATA

From https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages
Sending Data to a Server

• GET request
  – Data at end of URL (following “?”)

• POST request
  – Typically used with forms
  – Data not in URL, but rather (in slightly encoded form) in the HTTP request body

• PUT request
  – Store an entity at a location
URL Parameters / Query String

• End of URL (GET request)
  – https://www.cs.uchicago.edu/?test=foo&test2=bar
HTTP Response

• Status: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
  – 200 (OK)
  – 404 (not found)
  – 301 (moved permanently)
  – 302 (moved temporarily)

• HTTP Headers

• Body
HTTP

Requests

POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh; ... ) ... Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

Start-line

HTTP/1.1 403 Forbidden
Server: Apache
Content-Type: text/html; charset=iso-8859-1
Date: Wed, 10 Aug 2016 09:23:25 GMT
Keep-Alive: timeout=5, max=1000
Connection: Keep-Alive
Age: 3464
Date: Wed, 10 Aug 2016 09:46:25 GMT
X-Cache-Info: caching
Content-Length: 220

Empty line

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 2.0//EN">

Body

(more data)
HTTPS

• Simply an HTTP request sent over TLS!
  – That is, the request and response are encrypted
• 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
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)
Interactive Pages?

• JavaScript!
  – The core idea: Let’s run (somewhat) arbitrary code on the client’s computer
• Math, variables, control structures
• Imperative, object-oriented, or functional
• Modify the DOM
• Request data (e.g., through AJAX)
• Can be multi-threaded (web workers)