

15. How the Web Works

Blase Ur and David Cash

(Some slides borrowed from Ben Zhao)

February 10th, 2023

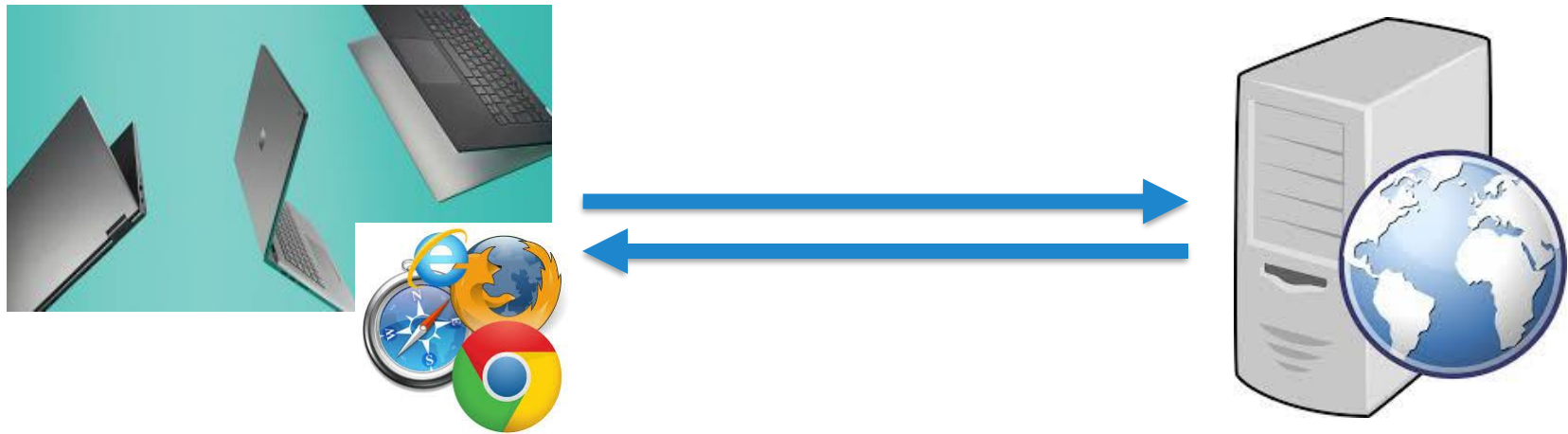
CMSC 23200 / 33250



THE UNIVERSITY OF
CHICAGO

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) →



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

Databases (MySQL)

The Anatomy of a URL

- <https://www.uchicago.edu/fun/funthings.htm?query=music&year=2022#topsection>

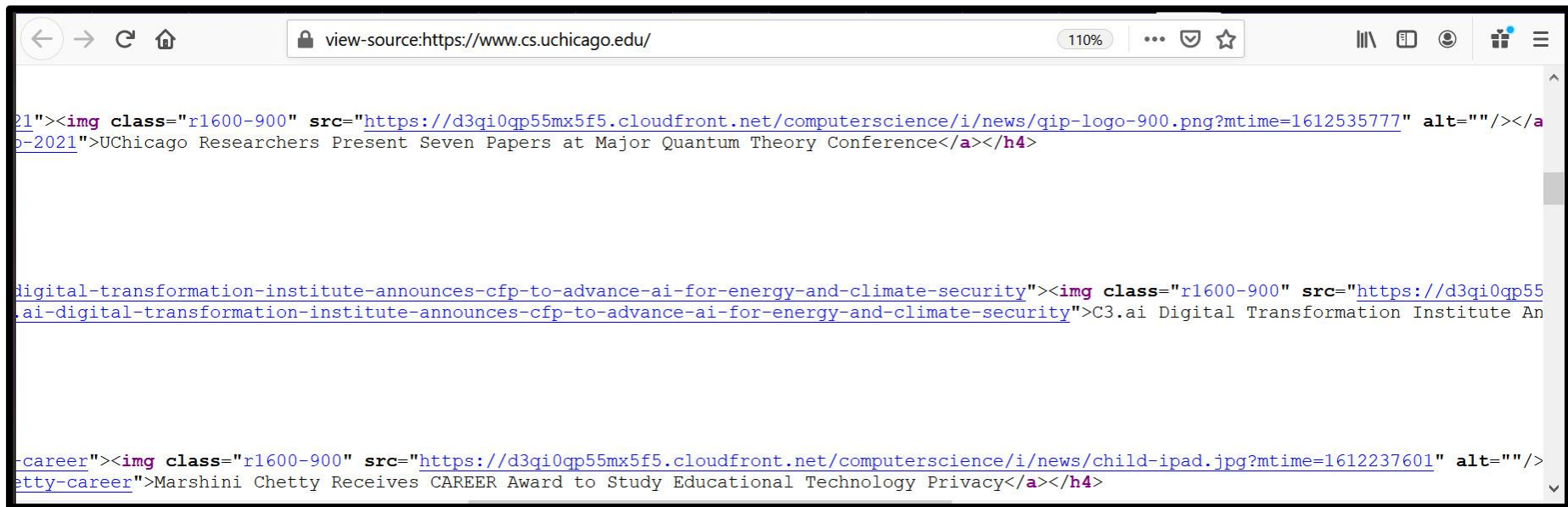
The Anatomy of a URL

- <https://www.uchicago.edu/fun/funthings.htm?query=music&year=2022#topsection>
 - Protocol: https
 - Hostname: www.uchicago.edu
 - .edu is the top level domain (TLD)
 - Path: /fun/funthings.html
 - Parameters: (key=value pairs, & delimited)
 - Named anchor: #topsection
- Some technologies (e.g., Django) parse the path differently (e.g., parameters in path)

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: ``
 - Forms
 - Audio/video

The Anatomy of a Webpage

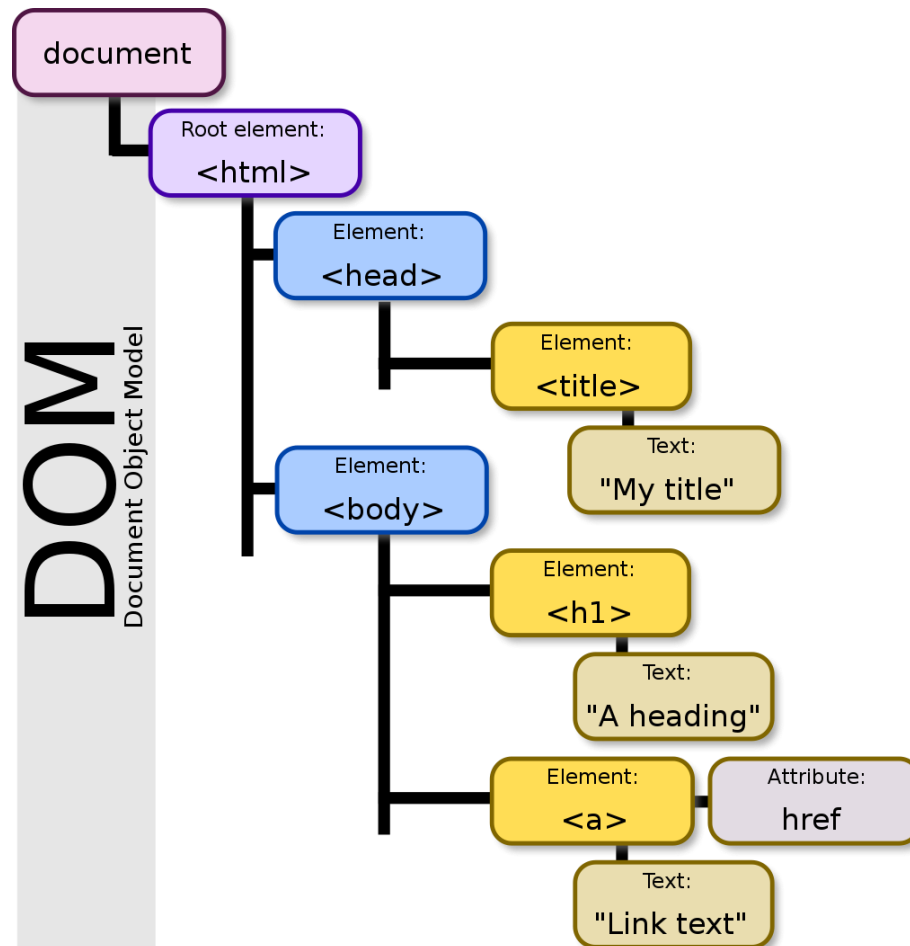


The Anatomy of a Webpage

- 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)

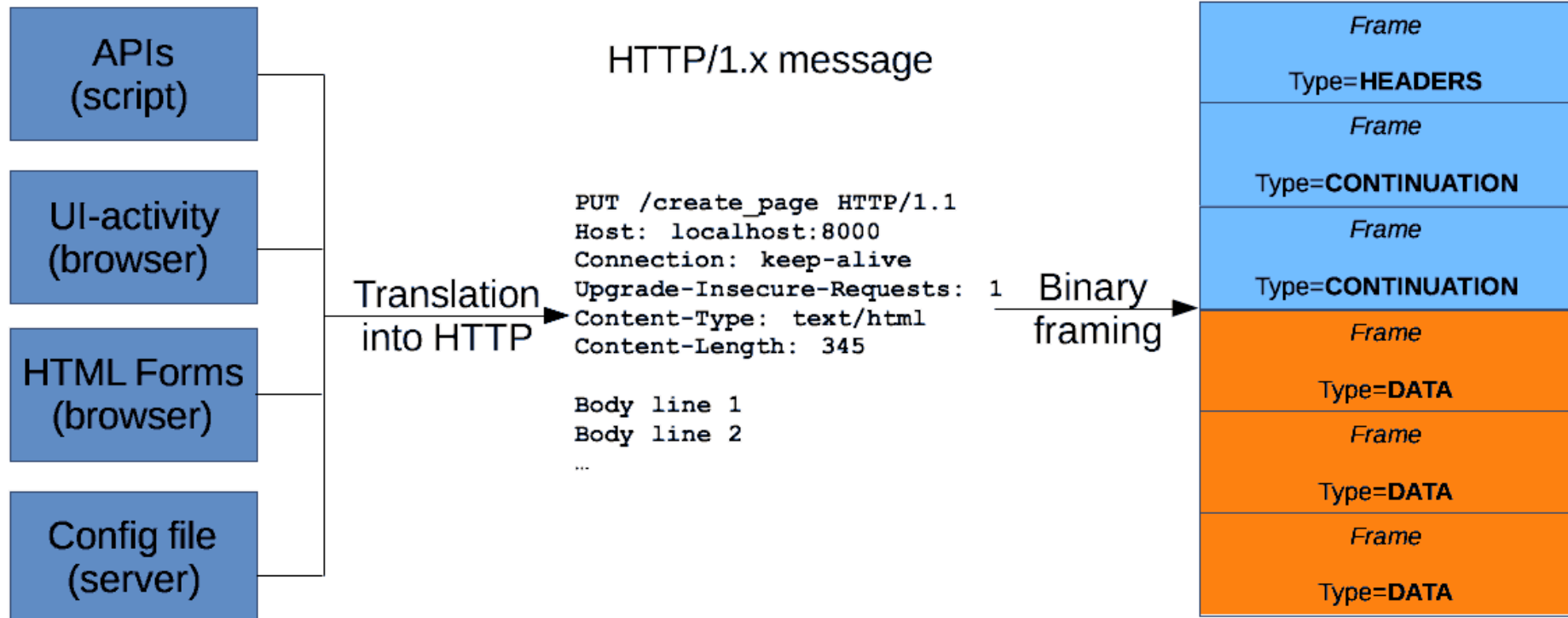
HTTP Request

- 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
 - <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>
- Body (not needed for GET, etc.)
- In Firefox: F12, “Network” to see HTTP requests

HTTP Request

- GET /index.html HTTP/1.1

Activity initiation



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>

The screenshot shows a web browser displaying the University of Chicago Department of Computer Science website. The URL in the address bar is <https://www.cs.uchicago.edu/?test=foo&test2=bar>. The browser's developer tools are open, showing the Network tab. The first request is highlighted, showing a GET request to <https://www.cs.uchicago.edu/?test=foo&test2=bar> with a status of 200. The right-hand pane of the developer tools shows the 'Params' tab, displaying the query string parameters: `test: foo` and `test2: bar`.

Status	Method	F...	Domain	Cause	Type	Transferred	Size
200	GET	/?test=...	www.cs.uchi...	document	html	6.76 KB	23.87 KB
302	GET	fonts.css	cloud.typogr...	stylesheet	css	154.58 KB	205.03 KB
200	GET	main.cs...	www.cs.uchi...	stylesheet	css	cached	189.57 KB
200	GET	moder...	www.cs.uchi...	script	js	cached	5.65 KB
200	GET	jquery....	ajax.googlea...	script	js	cached	0 B
200	GET	jquery-...	ajax.googlea...	script	js	cached	0 B

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,..., */*;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
```

```
-12656974
(more data)
```

Responses

```
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
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML
2.0//EN">
(more data)
```

start-
line

HTTP headers

empty
line

body

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)