12. How the Web Works Part 2

Blase Ur and David Cash
February 8th, 2021
CMSC 23200 / 33250

THE UNIVERSITY OF CHICAGO

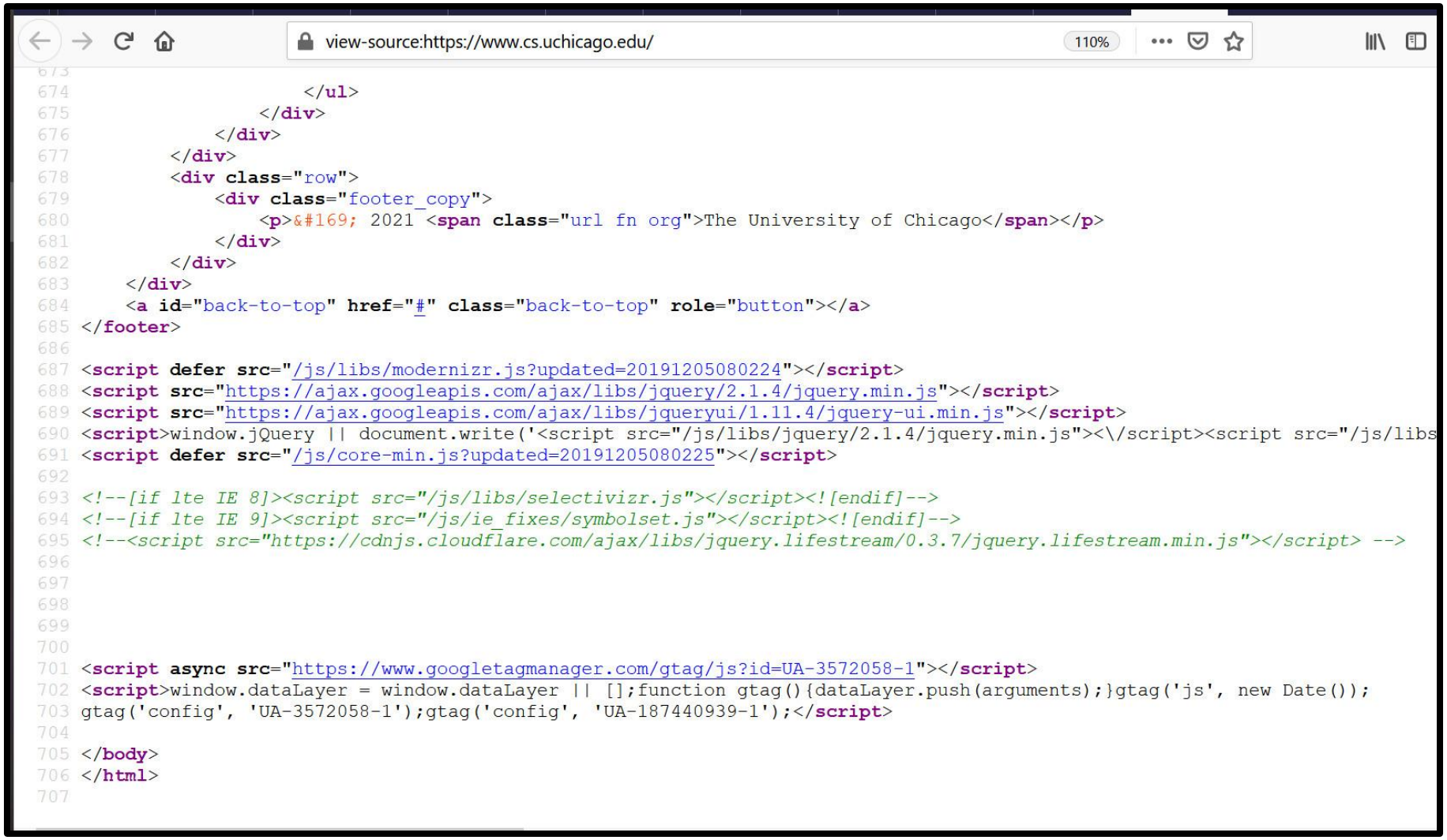
So… Interactive Pages?

- 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)

Common Javascript Libraries

- JQuery (easier access to DOM)
  - $(".test").hide() hides all elements with class="test"
- JQueryUI
- Bootstrap
- Angular / React
- Google Analytics *(sigh)*

Importing Javascript Libraries




```
                    </ul>
                </div>
            </div>
        </div>
        <div class="row">
            <div class="footer_copy">
                <p>&#169; 2021 <span class="url fn org">The University of Chicago</span></p>
            </div>
        </div>
    </div>
    <a id="back-to-top" href="#" class="back-to-top" role="button"></a>
</footer>

<script defer src="/js/libs/modernizr.js?updated=20191205080224"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.4/jquery.min.js"></script>
<script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.4/jquery-ui.min.js"></script>
<script>window.jQuery || document.write('<script src="/js/libs/jquery/2.1.4/jquery.min.js"><\/script><script src="/js/libs
<script defer src="/js/core-min.js?updated=20191205080225"></script>

<!--[if lte IE 8]><script src="/js/libs/selectivizr.js"></script><![endif]-->
<!--[if lte IE 9]><script src="/js/ie_fixes/symbolset.js"></script><![endif]-->
<!--<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery.lifestream/0.3.7/jquery.lifestream.min.js"></script> -->



<script async src="https://www.googletagmanager.com/gtag/js?id=UA-3572058-1"></script>
<script>window.dataLayer = window.dataLayer || [];function gtag(){dataLayer.push(arguments);}gtag('js', new Date());
gtag('config', 'UA-3572058-1');gtag('config', 'UA-187440939-1');</script>

</body>
</html>
```

Do You Have the Right .js File?

- Subresource integrity: https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity
- <script src=https://example.com/example-framework.js integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQho1wx4JwY8wC“ crossorigin="anonymous"></script>
- cat FILENAME.js | openssl dgst -sha384 -binary | openssl base64 –A

Same-Origin Policy

- Prevent malicious DOM access
- Origin = URI scheme, host name, port
- Only if origin that loaded script matches can a script access the DOM
  - Not where the script ultimately comes from, but what origin ***loads*** the script
- Frames / iframes impact origin
- CORS (Cross-Origin Resource Sharing)

Same-Origin Policy (SOP)

https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy

Definition of an origin

Two URLs have the *same origin* if the protocol, port (if specified), and host are the same for both. You may see this referenced as the "scheme/host/port tuple", or just "tuple". (A "tuple" is a set of items that together comprise a whole — a generic form for double/triple/quadruple /quintuple/etc.)

The following table gives examples of origin comparisons with the URL `http://store.company.com/dir/page.html`:

URL	Outcome	Reason
`http://store.company.com/dir2/other.html`	Same origin	Only the path differs
`http://store.company.com/dir/inner /another.html`	Same origin	Only the path differs
`https://store.company.com/page.html`	Failure	Different protocol
`http://store.company.com:81/dir/page.html`	Failure	Different port (`http://` is port 80 by default)
`http://news.company.com/dir/page.html`	Failure	Different host

CORS (Relaxes SOP)

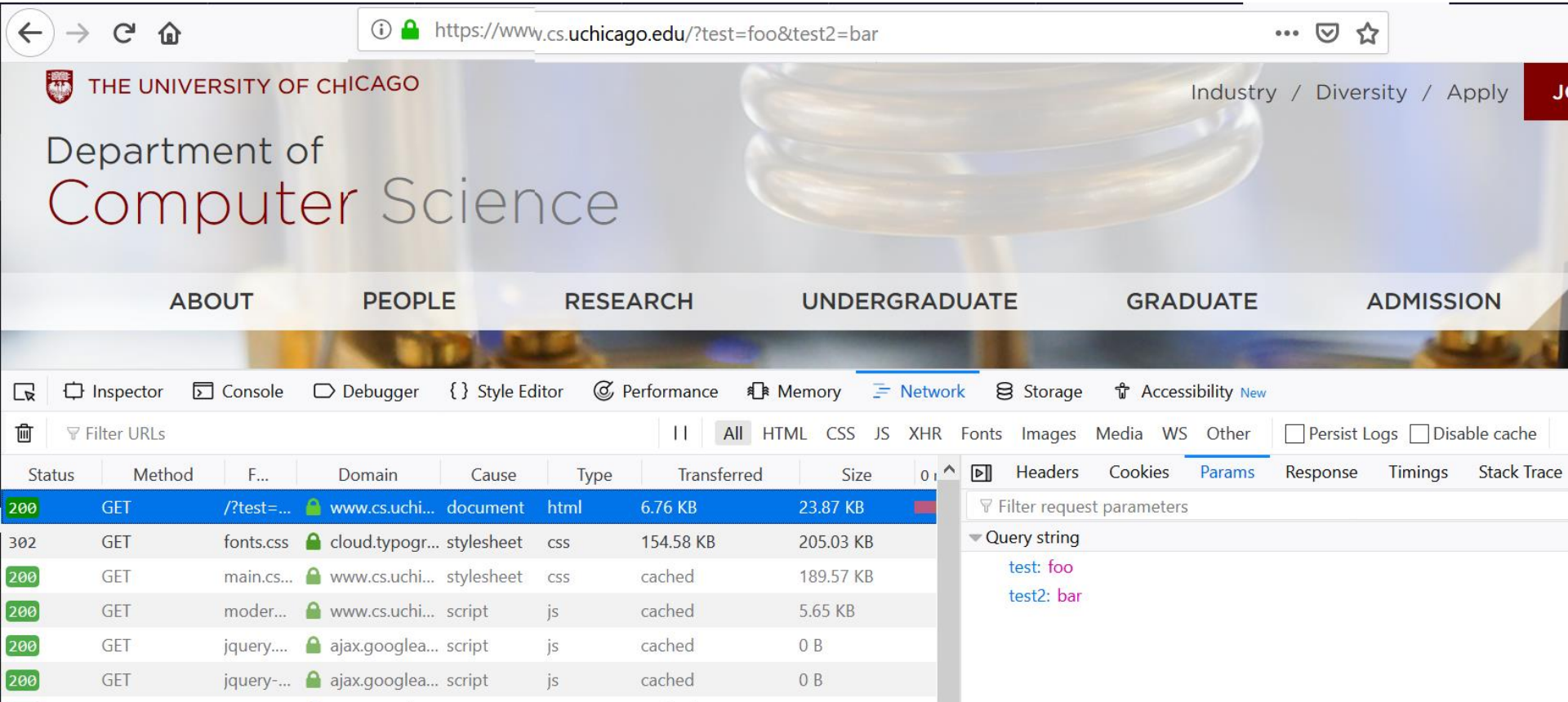
- Cross-Origin Resource Sharing
  - Specifies when specific other origins can make a request for data on a different origin
- https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

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



Processing Data on the Server

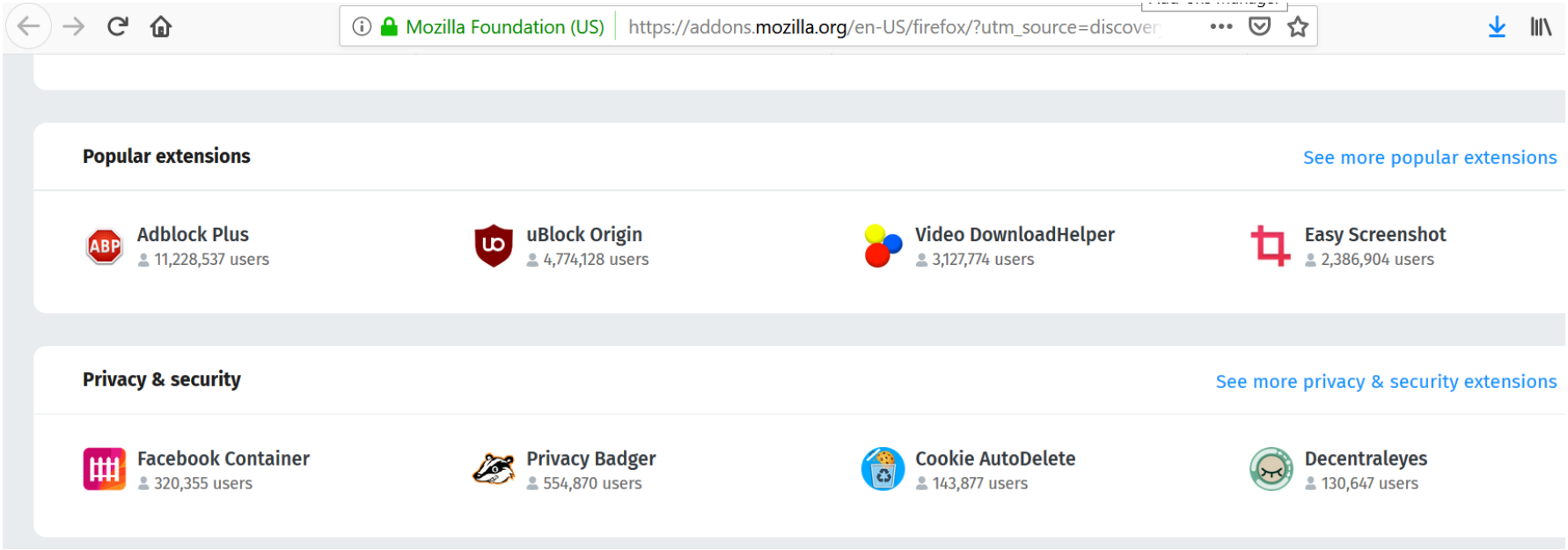
- Javascript is client-side
- Server-side you find Perl (CGI), PHP, Python (Django)
- Process data on the server
- What happens if this code crashes?

Storing Data on the Server

- Run a database on the server
- MySQL, SQLite, MongoDB, Redis, etc.
- You probably don’t want to allow access from anything other than *localhost*
- You definitely don’t want human-memorable passwords for these

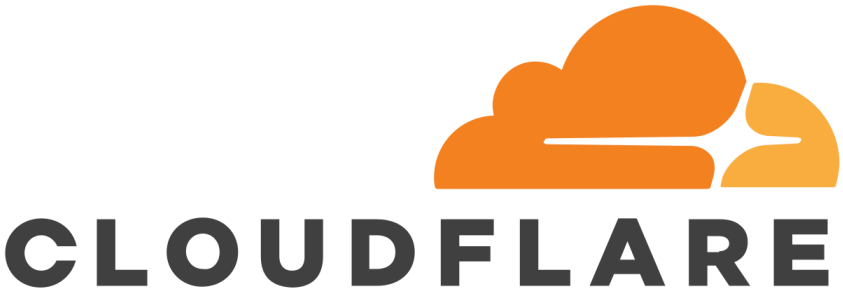
Browser Extensions

- Can access most of what the browser can
- Requires permissions system
- Malicious extensions!



What If You Get Lots of Traffic?

- CDNs (content delivery networks)



What If You Don’t Want To Code?

- CMS (content management system)
  - WordPress (PHP + MySQL), Drupal