Computer and Network Security

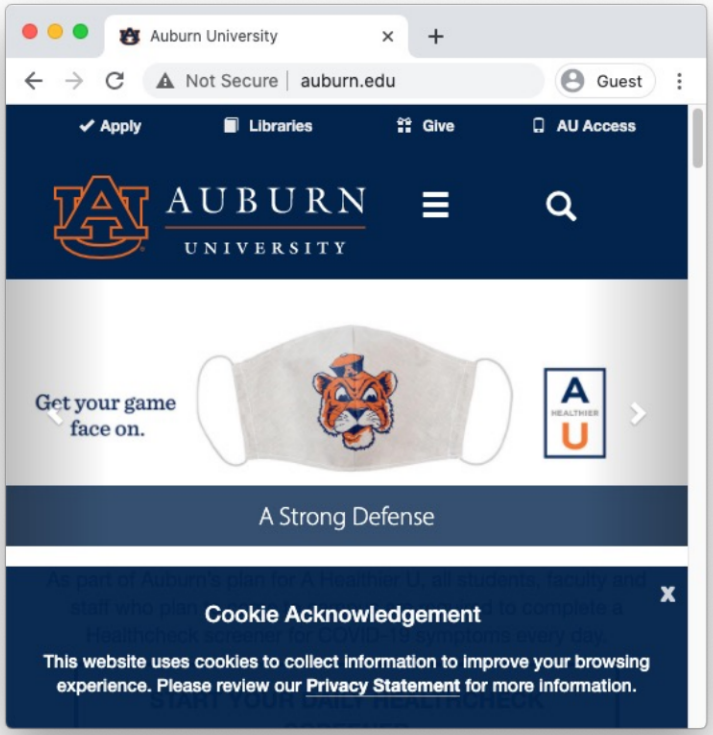
Lecture 18:
WWW & Web Attacks

COMP-5370/6370
Fall 2025

How the "Web" Works

The "web" is an array of protocols, standards, and un-written conventions for providing content via the Internet.

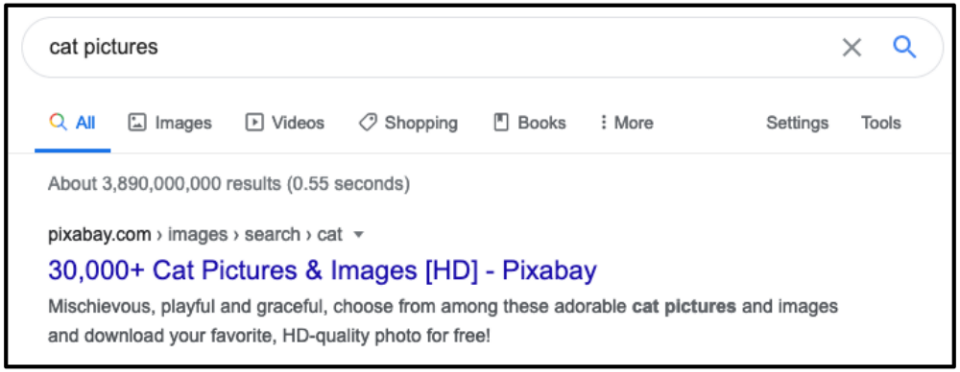
- Even though it's no longer the 90's, we're still paying for bad decisions



What is it Built To Do?

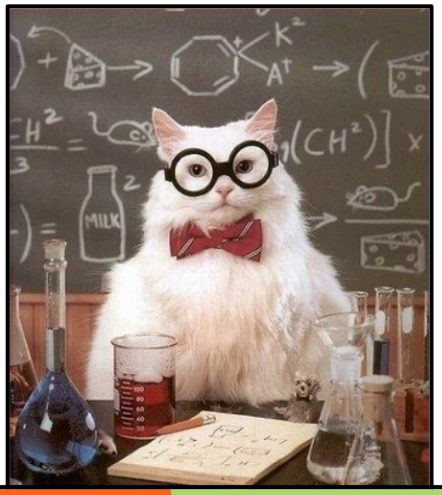
What is it Built To Do?

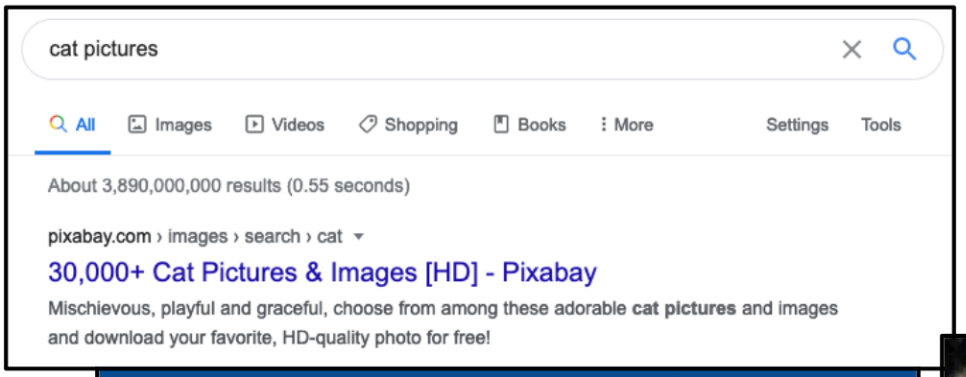
The “web” was built to serve cat pictures.

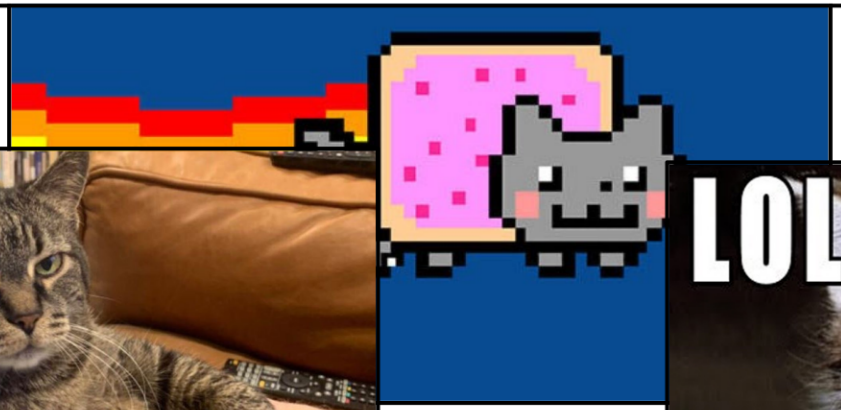


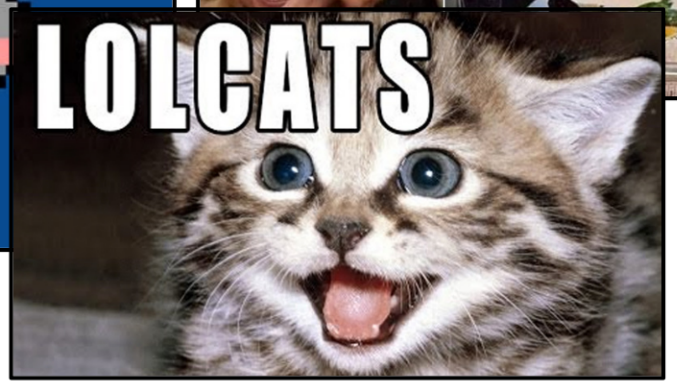
What is it Built To Do?

The “web” was built to serve cat pictures.









What is it Built To Do?

The “web” was built to serve cat pictures.

It was **NOT** built to serve cat pictures

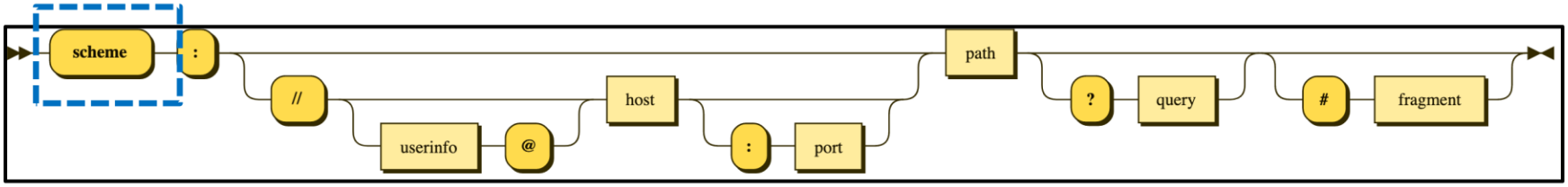
- In a secure way
- In a private way
- In a safe way
- In a trustworthy way

The “web” is insecure by-design and by-incentive.

Core Web Components

- Direct navigation

Universal Resource Identifier



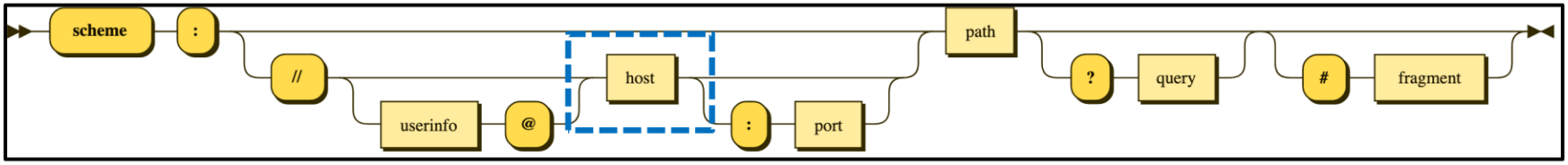
A **Universal Resource Identifier (URI)** is a structured mechanism to address content on the Internet.

'i' as in "India"

- Scheme indicates what protocol to talk
  - `ftp://, http://, https://, mongodb://, …`
- Most often seen as a "URL"

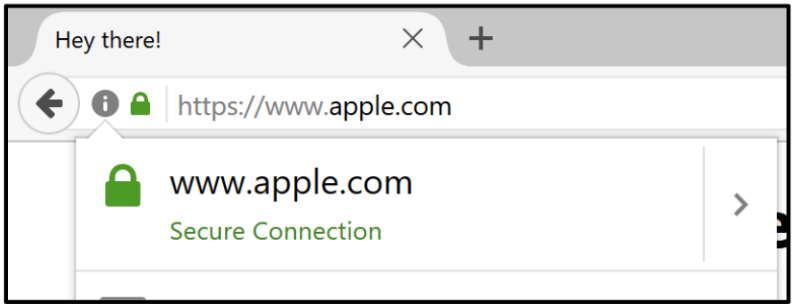
'L' as in "llama"

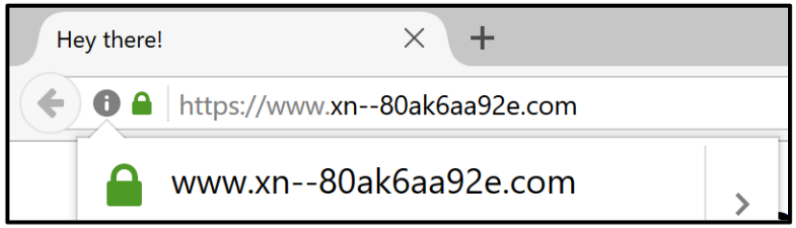
Host in URI

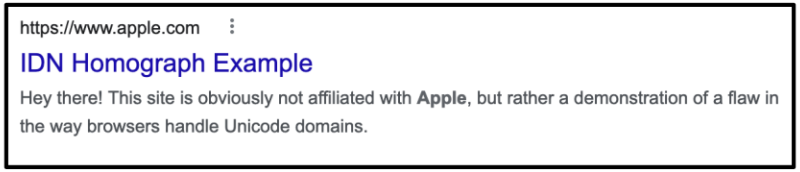


Host is the server's identity.

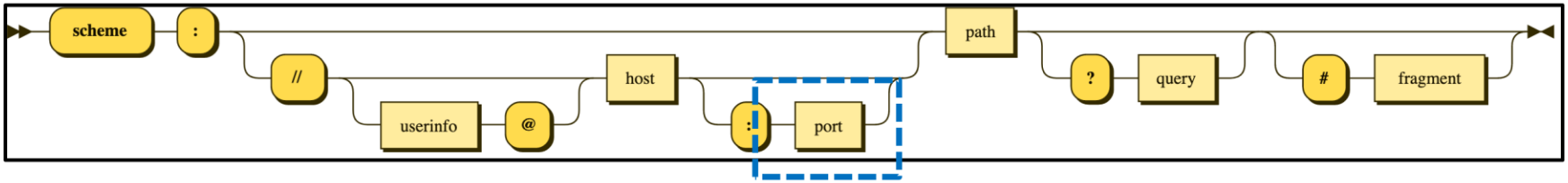
- Various ways to refer to a server
  - Domain Name: `apple.com`







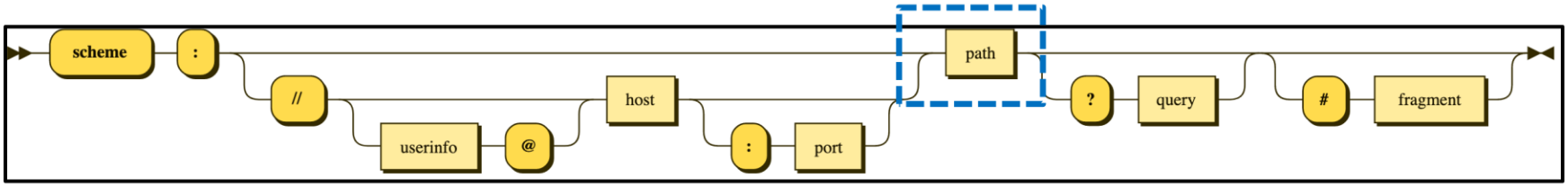
Port in URI



Port is the TCP/UDP port that should be contacted to talk to the server.

- Can be implicit or explicit
  - `http://example.com/` → 80
  - `http://example.com:8080/` → 8080

Path in URI



Path is the logical location of the resource being request in storage.

- Can be implicit (/index.html)
- File extension typically indicates the source type
- Is relative to the server's config not filesystem

```
5370-web-server> pwd
/usr/share/caddy
5370-web-server> tree
.
├── Lecture-01_Intro-and-Mindset-safe.pdf
├── Lecture-02_Intro-to-Crypto-safe.pdf
├── Lecture-03_Hashing-and-Integrity-safe.pdf
├── Lecture-04_Confidentiality-safe.pdf
├── Lecture-05_KEX-Asym-Operations-safe.pdf
├── Lecture-07_Sender-Authenticity-safe.pdf
├── Lecture-08_Authentication-safe.pdf
├── Lecture-09_Binary-Exploitation-1-safe.pdf
├── Lecture-10_Binary-Exploitation-2-safe.pdf
├── Lecture-11_Malware-and-Attacks-safe.pdf
├── Lecture-12_OS-Isolation-safe.pdf
├── Lecture-13_Hardware-Security-and-Attacks-safe.pdf
├── Lecture-14_Side-Channels-and-Review-safe.pdf
├── Lecture-15_Networking-Background-safe.pdf
├── Lecture-16_Network-Attacks-safe.pdf
├── favicon.ico
├── index.html
├── proj1a-assn.pdf
├── proj1a-framing.tar.gz
└── proj1a-spec.txt
```

Directory Traversal Vulns

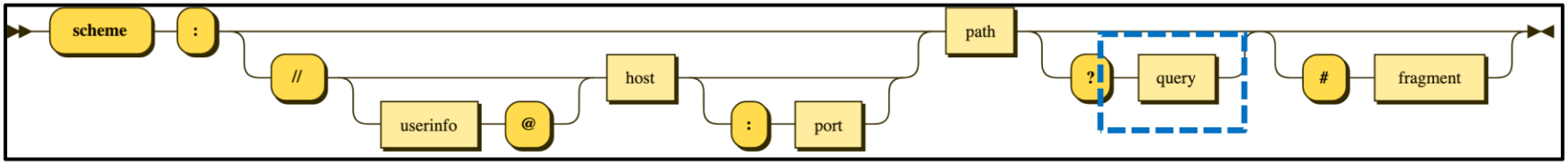
A **Directory Traversal Vulnerability** is one where a client can access the filesystem objects outside of the intended limitations.

```
/index.html
/../caddy/index.html
/../../../etc/passwd
```

```
5370-web-server> pwd
/usr/share/caddy
5370-web-server> ls -lah index.html
-r--r--r-- 1 root root 13K Oct 16 03:45 index.html
5370-web-server> ls -lah ../../../etc/passwd
-rw-r--r-- 1 root root 1.9K Aug 16 01:41 ../../../etc/passwd
5370-web-server>
```

```
5370-web-server> pwd
/usr/share/caddy
5370-web-server> tree
.
├── Lecture-01_Intro-and-Mindset-safe.pdf
├── Lecture-02_Intro-to-Crypto-safe.pdf
├── Lecture-03_Hashing-and-Integrity-safe.pdf
├── Lecture-04_Confidentiality-safe.pdf
├── Lecture-05_KEX-Asym-Operations-safe.pdf
├── Lecture-07_Sender-Authenticity-safe.pdf
├── Lecture-08_Authentication-safe.pdf
├── Lecture-09_Binary-Exploitation-1-safe.pdf
├── Lecture-10_Binary-Exploitation-2-safe.pdf
├── Lecture-11_Malware-and-Attacks-safe.pdf
├── Lecture-12_OS-Isolation-safe.pdf
├── Lecture-13_Hardware-Security-and-Attacks-safe.pdf
├── Lecture-14_Side-Channels-and-Review-safe.pdf
├── Lecture-15_Networking-Background-safe.pdf
├── Lecture-16_Network-Attacks-safe.pdf
├── favicon.ico
├── index.html
├── proj1a-assn.pdf
├── proj1a-framing.tar.gz
└── proj1a-spec.txt
```

Query in URI

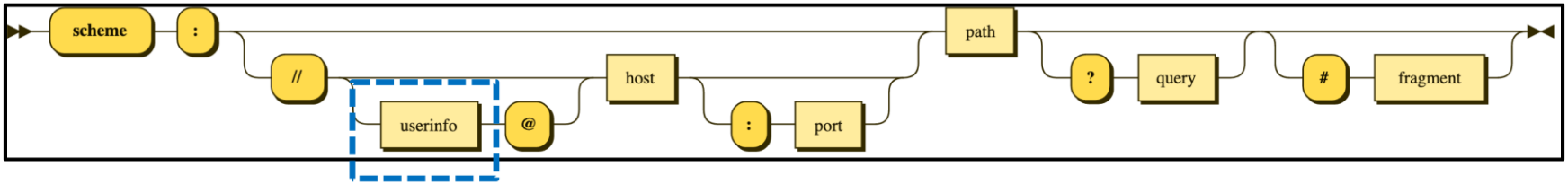


Query is a mechanism to pass arguments to the server in key-value pairs.

```
?key1=val1&key2=val2
```

- Percent-encoded keys & values
- '?' indicates the start of the query
- '&' separates the key-value pairs
- '=' separates the key and the value

User Information in URI



http://j74v:fOl24Hg49@example.com

- Userinfo allows the client to include authentication info

3.2.1. User Information

The userinfo subcomponent may consist of a user name and, optionally, scheme-specific information about how to gain authorization to access the resource. The user information, if present, is followed by a commercial at-sign ("@") that delimits it from the host.

```
userinfo    = *( unreserved / pct-encoded / sub-delims / ":" )
```

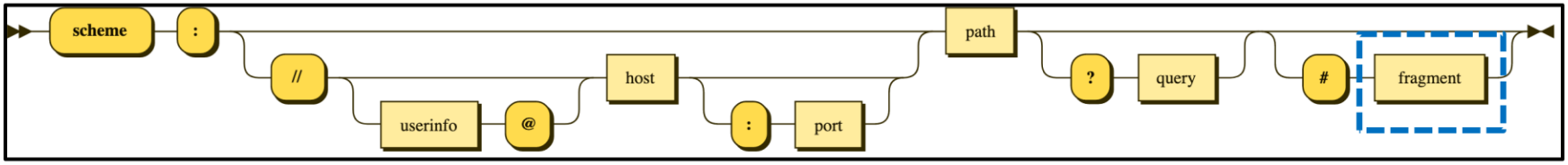
Use of the format "user:password" in the userinfo field is deprecated. Applications should not render as clear text any data after the first colon (":") character found within a userinfo subcomponent unless the data after the colon is the empty string (indicating no password). Applications may choose to ignore or reject such data when it is received as part of a reference and should reject the storage of such data in unencrypted form. The passing of authentication information in clear text has proven to be a security risk in almost every case where it has been used.

Applications that render a URI for the sake of user feedback, such as in graphical hypertext browsing, should render userinfo in a way that is distinguished from the rest of a URI, when feasible. Such rendering will assist the user in cases where the userinfo has been misleadingly crafted to look like a trusted domain name (Section 7.6).

RFC 3986

http://username:password@example.com

Fragment in URI



Fragment is a secondary resource indicator.

`http://example.com/home#faq`

- Often used by frameworks to indicate where visibility should be placed on-load

URL Breakout

http://example.com/about.html?src=home

- Protocol: HTTP
- Host: example.com's IP address
- Port: 80 (implicit via HTTP)
- Page to Display: About
- Format of Page: HTML
- Query: <based on link user clicked>

  User clicked on 'About' from the home page.

URLs in the Real-World

What you think you're watching:

https://www.youtube.com/watch?v=5BxxbTPoWWY

What you're actually watching:

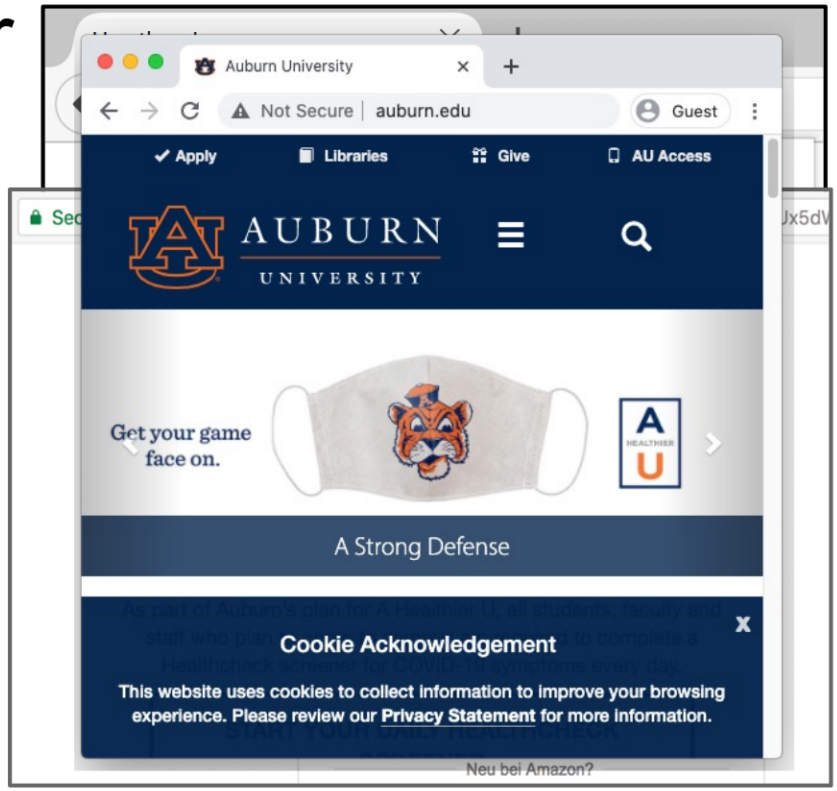
https://r3---sn-nvouixgo-5uae.googlevideo.com/videoplayback?expire=1600929540&ei=pOprX4TGB_KCo_wPn76t-AU&ip=131.204.254.86&id=o-AAyq6CfLpSHdnMuGwq6O5M2y8nATOdqXCAWBWeTqGxsa&itag=242&aitags=133%2C160%2C242%2C278&source=youtube&requiressl=yes&mh=Ig&mm=31%2C29&mn=sn-nvouixgo-5uae%2Csn-5uaezny6&ms=au%2Crdu&mv=m&mvi=3&pl=16&initcwndbps=1515000&vprv=1&mime=video%2Fwebm&gir=yes&clen=6143289&dur=254.554&lmt=1557584010890541&mt=1600907907&fvip=3&keepalive=yes&fexp=23915654&c=WEB&txp=5432432&sparams=expire%2Cei%2Cip%2Cid%2Caitags%2Csource%2Crequiressl%2Cvprv%2Cmime%2Cgir%2Cclen%2Cdur%2Clmt&lsparams=mh%2Cmm%2Cmn%2Cms%2Cmv%2Cmvi%2Cpl%2Cinitcwndbps&lsig=AG3C_xAwRQIgWCY1lyUsDChk5cl8dd6NdaC4Qqa_L5N8kRzyj3jv4MoCIQDXomwqtYYtNgZl7xx9crVCAyRTYqcZwPyw2IyjpuTYcw%3D%3D&alr=yes&sig=AOq0QJ8wRAIgVZ11DAdE5RRklf-eZPU6H81kSmatwCv9O2-OAIY4Vz8CIHe1FCoJOwSrDlpyQoGskQu_gC49rsvGfvckRdLTwVj2&cpn=meaDbj6jkGiZRZfR&cver=2.20200923.01.00&range=2388108-3189140&rn=27&rbuf=82146

URLs are Effectively Unusable

Were originally built to be readable and understandable but are not anymore

- Were a replacement for BBS & AOL Keywords

- **URLs require users to know where they are and where they're suppose to be**



Core Web Components

- Direct navigation
  - URLs are a poor security foundation

Core Web Components

- Direct navigation
  - URLs are a poor security foundation
- Communication protocol

HTTP Protocol

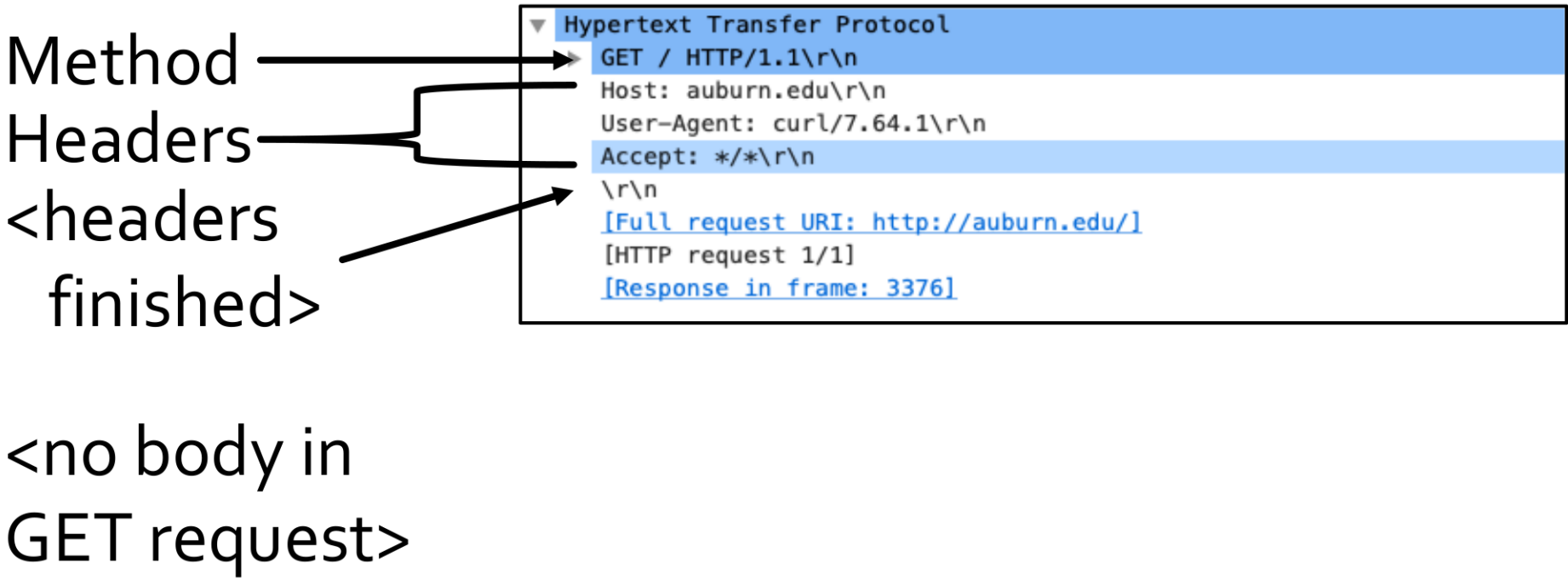
The **Hypertext Transfer Protocol (HTTP)** is the base-protocol through which web servers and web clients communicate.

- *Idea* is extremely simple
- *Implementation* is extremely complicated

HTTP Protocol Details

- Methods (often referred to as “verbs”)
  - **GET**: Fetch content from a web server
  - **POST**: Send content to a web server
  - ***others exist*** for different uses
- Passes information via a “body” and arbitrary “headers” describing the body
  - CR/LF (“\r\n”) separated key-value pairs

HTTP Request



<no body in
GET request>

User Agents Lie

- Chrome User Agent on Windows:

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/94.0.4606.81 Safari/537.36
```

```
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: auburn.edu\r\n
  User-Agent: curl/7.64.1\r\n
  Accept: */*\r
  \r\n
  [Full request
  [HTTP request
```

```
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: auburn.edu\r\n
  Accept: */*\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.81 Safari/537.36\r\n
  \r\n
  [Full request UR
  [HTTP request 1/
  [Response in fra
```

```
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: auburn.edu\r\n
  User-Agent: Wacky Waving Inflatable Arm Flailing Tubeman laptop\r\n
  Accept: */*\r\n
  \r\n
  [Full request URI: http://auburn.edu/]
  [HTTP request 1/1]
  [Response in frame: 374]
```

User Agents Lie

- Chrome User Agent on Windows:

  `Mozilla/5.0 (Windows NT 10.0; Win64; x64)`

```
$> curl 'http://auburn.edu' > /dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 56412    0 56412    0     0   114k      0 --:--:-- --:--:-- --:--:--  114k
$> curl -H "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 56412    0 56412    0     0   107k      0 --:--:-- --:--:-- --:--:--  107k
$> curl --user-agent 'Wacky Waving Inflatable Arm Flailing Tubeman laptop' 'http://auburn.edu' > /dev/null
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 56412    0 56412    0     0   124k      0 --:--:-- --:--:-- --:--:--  124k
$>
```

```
[Response in fra
```

```
User-Agent: Wacky Waving Inflatable Arm Flailing Tubeman laptop\r\n
Accept: */*\r\n
\r\n
[Full request URI: http://auburn.edu/]
[HTTP request 1/1]
[Response in frame: 374]
```

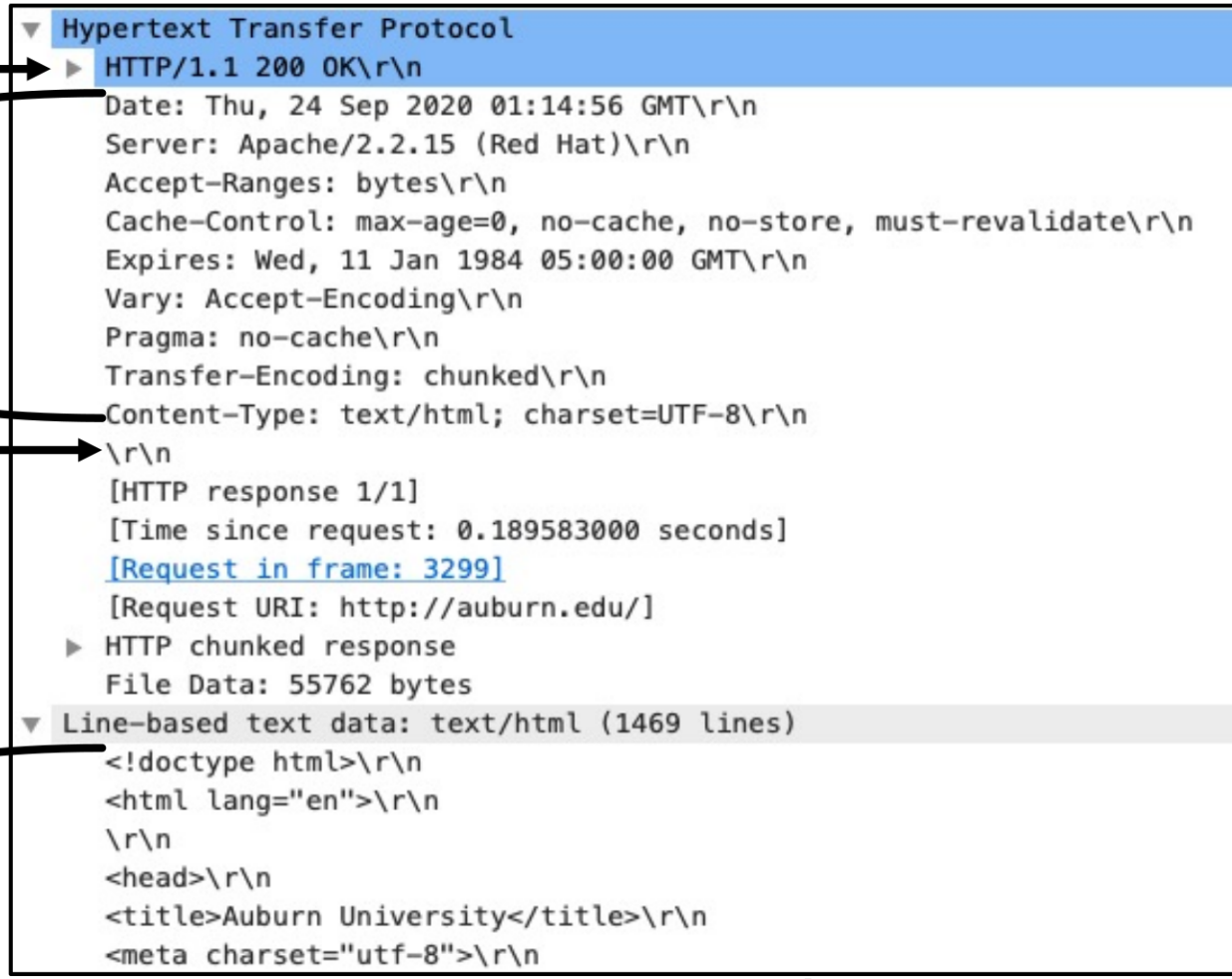
HTTP Response

Status

Headers

<headers finished>

Body



..... <truncated> .....

Common HTTP Responses

2XX – Normal

- `200` : OK
- `204` : OK(Unchanged)

3XX – Redirect

- `301` : Permanent Redirect
- `307` : Temporary Redirect

4XX – Client Error

- `400` : Bad Request
- `404` : No resource at requested path

5XX – Server Error

- `500` : *Server is on fire*
- `502` : *Corp network is on fire*

3XX Redirects

```
> curl http://auburn.edu -v
* Host auburn.edu:80 was resolved.
* IPv6: (none)
* IPv4: 131.204.138.170
*   Trying 131.204.138.170:80...
* Connected to auburn.edu (131.204.138.170) port 80
> GET / HTTP/1.1
> Host: auburn.edu
> User-Agent: curl/8.5.0
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 302 Moved Temporarily
< Location: https://auburn.edu/
< Server: BigIP
* HTTP/1.0 connection set to keep alive
< Connection: Keep-Alive
< Content-Length: 0
```

```
> curl https://auburn.edu -v | head
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
                                 Dload  Upload   Total   Spent
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- -
* IPv6: (none)
* IPv4: 131.204.138.170
*   Trying 131.204.138.170:443
```

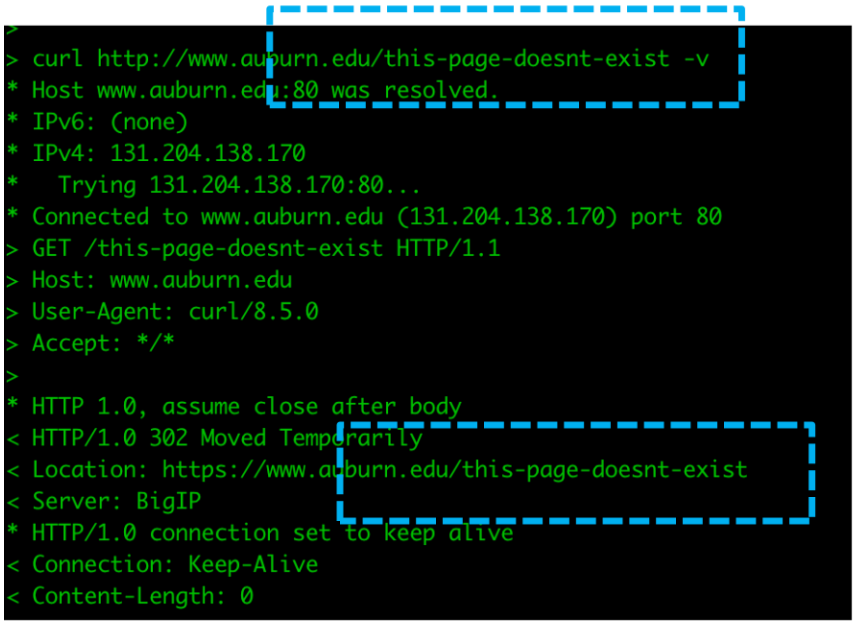
```
> GET / HTTP/1.1
> Host: auburn.edu
> User-Agent: curl/8.5.0
> Accept: */*
```

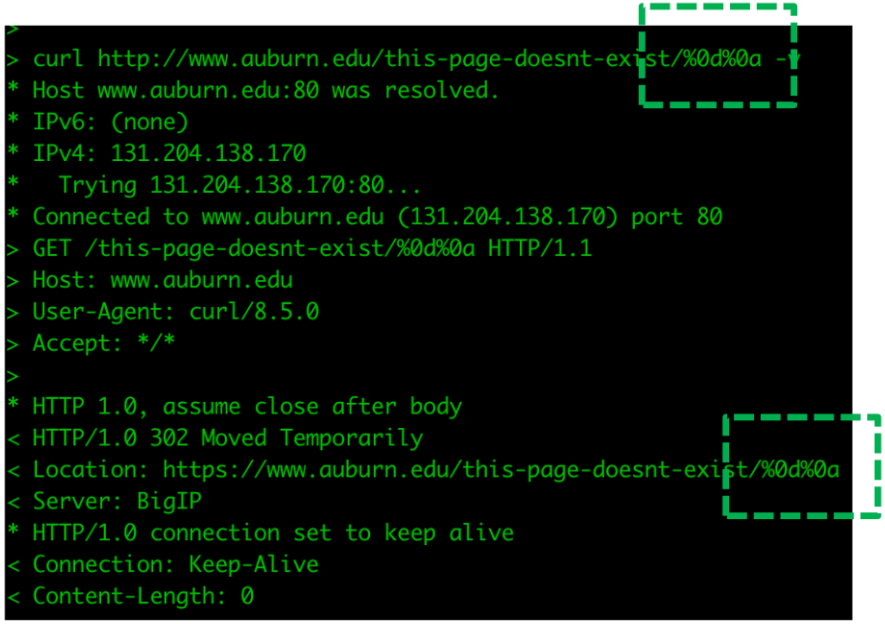
```
{ [5 bytes data]
< HTTP/1.1 200 OK
< Date: Tue, 21 Oct 2025 19:19:44 GMT
< Server: Apache/2.4.62 (Red Hat Enterprise Linux)
< X-Powered-By: PHP/8.2.27
< Cache-Control: max-age=0, no-cache, no-store, must-revalidate
< Expires: Wed, 11 Jan 1984 05:00:00 GMT
< Vary: Accept-Encoding
< Pragma: no-cache
< Transfer-Encoding: chunked
< Content-Type: text/html; charset=UTF-8
<
{ [8116 bytes data]

<!doctype html>
<html lang="en"><head>
<title>Auburn University Homepage</title>
<meta charset="utf-8">
```

Open Redirect Attacks

Secure Handling





Open Redirect Attacks

Insecure Handling

```
> curl http://www.auburn.edu/this-page-doesnt-exist -v
* Host www.auburn.edu:80 was resolved.
* IPv6: (none)
* IPv4: 131.204.138.170
*   Trying 131.204.138.170:80...
* Connected to www.auburn.edu (131.204.138.170) port 80
> GET /this-page-doesnt-exist HTTP/1.1
> Host: www.auburn.edu
> User-Agent: curl/8.5.0
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 302 Moved Temporarily
< Location: https://www.auburn.edu/this-page-doesnt-exist
< Server: BigIP
* HTTP/1.0 connection set to keep alive
< Connection: Keep-Alive
< Content-Length: 0
```

```
> curl http://www.auburn.edu/this-page-doesnt-exist/%0d%0a -v
* Host www.auburn.edu:80 was resolved.
* IPv6: (none)
* IPv4: 131.204.138.170
*   Trying 131.204.138.170:80...
* Connected to www.auburn.edu (131.204.138.170) port 80
> GET /this-page-doesnt-exist/%0d%0a HTTP/1.1
> Host: www.auburn.edu
> User-Agent: curl/8.5.0
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 302 Moved Temporarily
< Location: https://www.auburn.edu/this-page-doesnt-exist/

< Server: BigIP
* HTTP/1.0 connection set to keep alive
< Connection: Keep-Alive
```

Open Redirect Attacks

```
http://auburn.edu/%0d%0a
Location%3A%20http%3A%2F%2Fua.edu
```

```
>
* HTTP 1.0, assume close after body
< HTTP/1.0 302 Moved Temporarily
< Location: https://www.auburn.edu/
< Location: http://ua.edu
< Server: BigIP
* HTTP/1.0 connection set to keep alive
< Connection: Keep-Alive
< Content-Length: 0
```

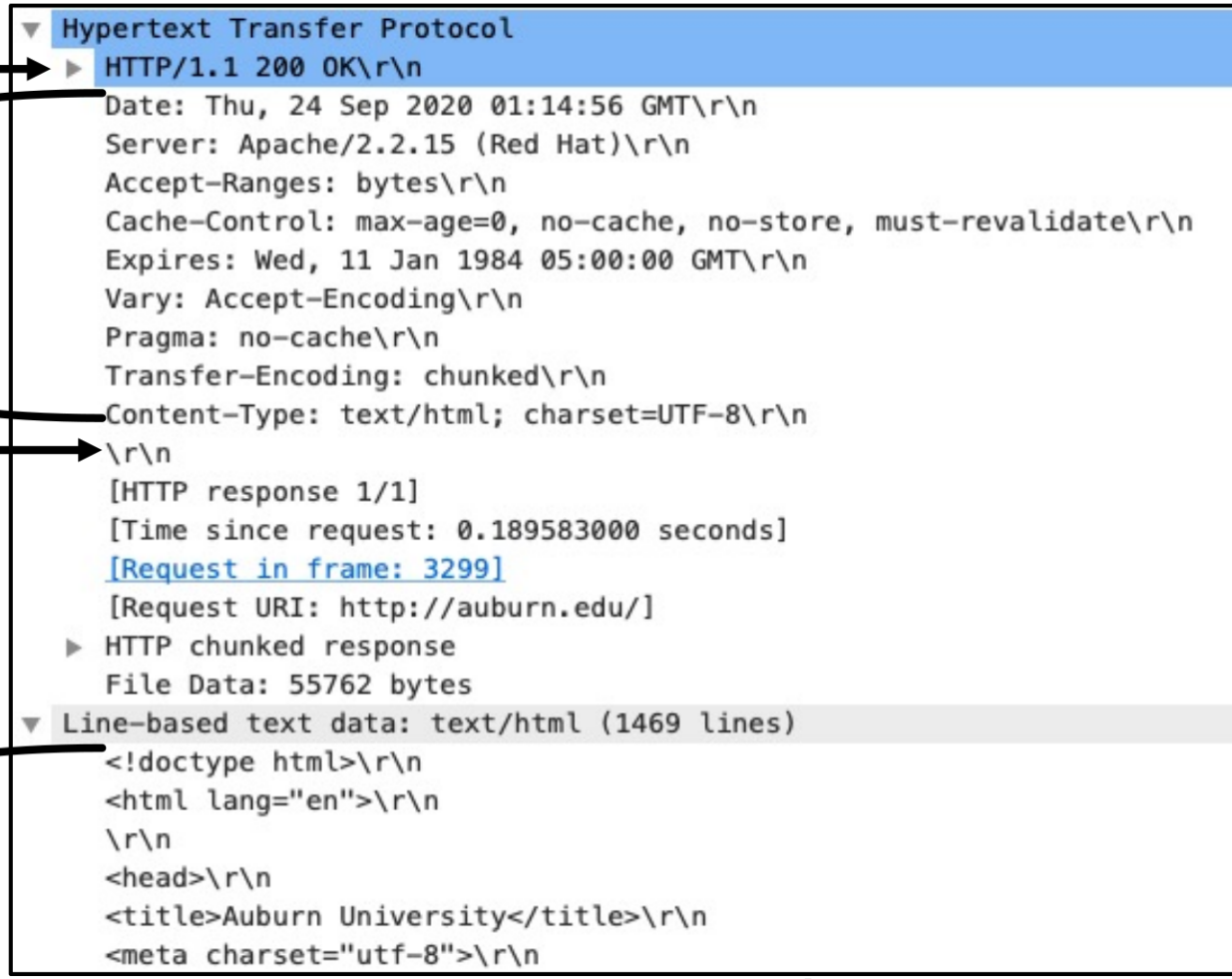
HTTP Response

Status

Headers

<headers finished>

Body



..... <truncated> .....

Core Web Components

- Direct navigation
  - URLs are a poor security foundation
- Communication protocol
  - Fundamental abstractions == fundamental assumptions

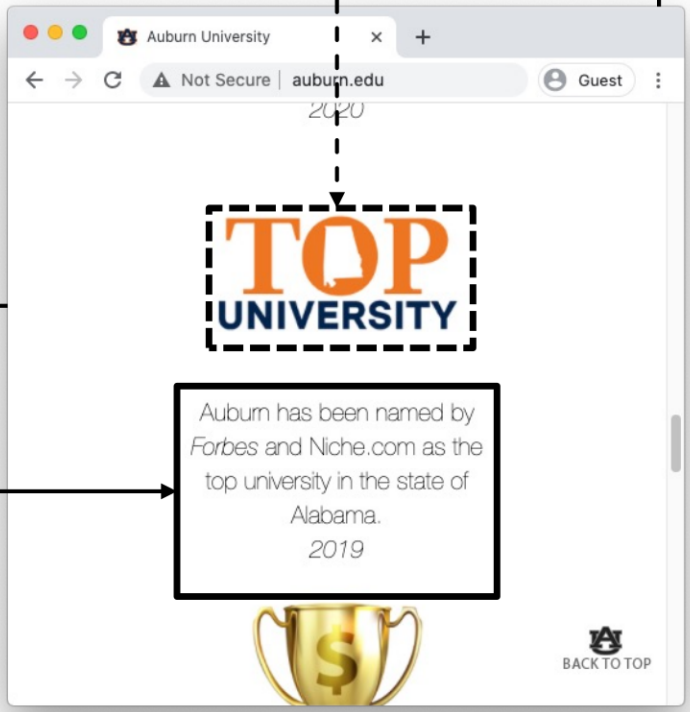
Core Web Components

- Direct navigation
  - URLs are a poor security foundation
- Communication protocol
  - Fundamental abstractions == fundamental assumptions
- Content rendering

Hypertext Markup Language (HTML)

The **Hypertext Markup Language (HTML)** is root-mechanism for all websites' content.

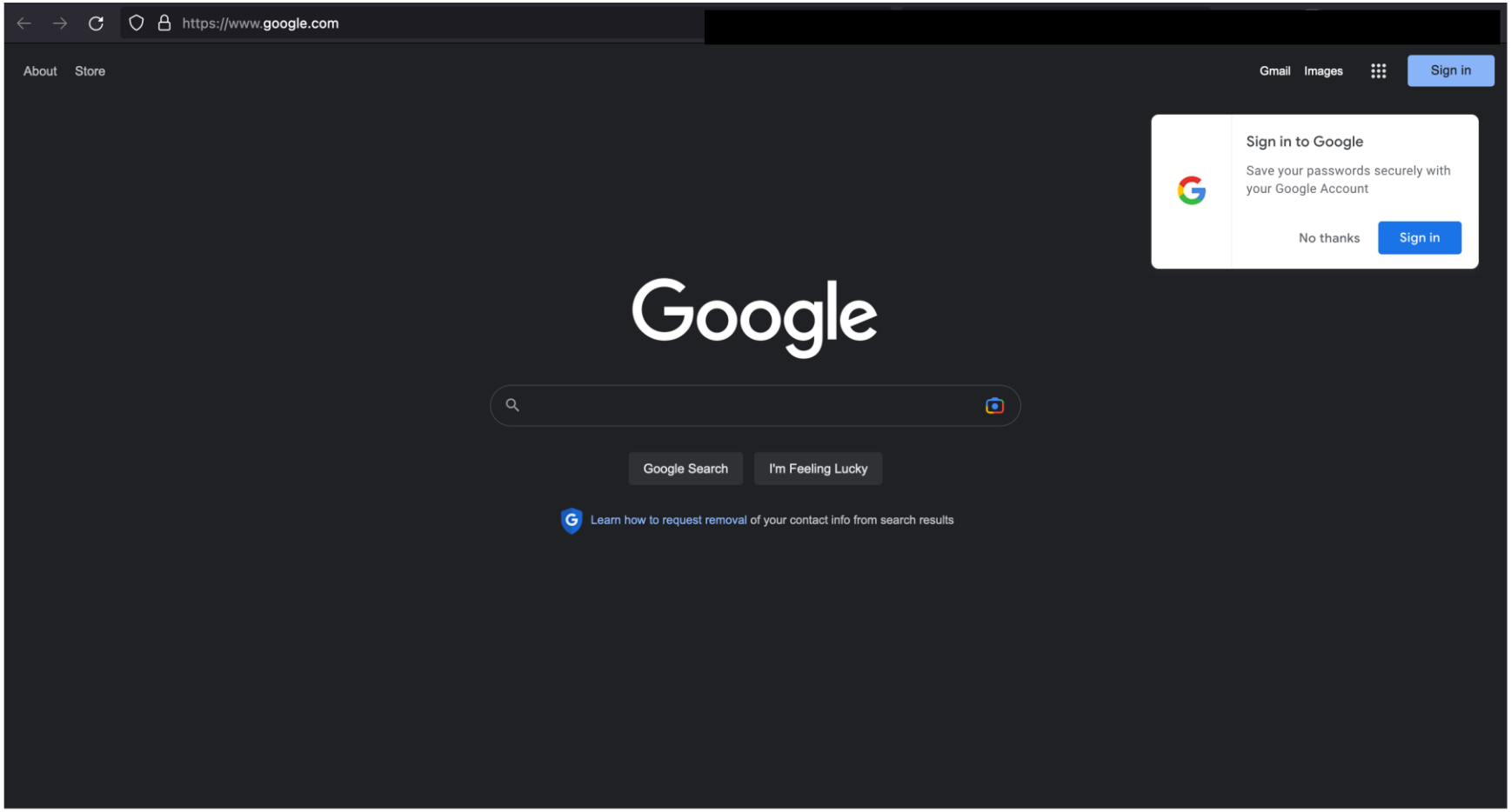
```
<div class="item active">
  <div class="image">
    <img alt="Doctor bag with paw print on it" src="//www.auburn.edu/template/2016/_assets/images/graphic-topu.png">
  </div>
  <div class="text2">
    "Auburn has been named by "
    <em>Forbes</em>
    " and Niche.com as the top university in the state of Alabama."
    <br>
    <em>2019</em>
  </div>
</div>
```



HTML Tags

```
▼<div class="item active">
  ▼<div class="image">
      <img alt="Doctor bag with paw print on it" src="//www.auburn.edu/template/2016/_assets/images/graphic-topu.png">
    </div>
  ▼<div class="text2">
      "Auburn has been named by "
      <em>Forbes</em>
      " and Niche.com as the top university in the state of Alabama."
      <br>
      <em>2019</em>
    </div>
  </div>
</div>
```

- `<div>` Logical division
- `<a>` Link to another website
- `<img>` Load and display an image
- `<script>` Execute JavaScript client-side


https://www.google.com
About
Store
Gmail
Images
Sign in
Sign in to Google
Save your passwords securely with your Google Account
No thanks
Sign in
Google
Google Search
I'm Feeling Lucky
Learn how to request removal of your contact info from search results

view-source:https://www.google.com/

```
var Ud=function(a){_.G.call(this,a)};_.w(Ud,_.G);_.Wd=function(){return _.F(_.Vd,_.Cc,1)};_.Xd=function(){return _.F(_.Vd,_.Dc,5)};
var Yd;window.gbar_&&window.gbar_.CONFIG?Yd=window.gbar_.CONFIG[0]||{}:Yd=[];_.Vd=new Ud(Yd);
var Dd,Ed,Od,Pd,Nd;Dd=_.F(_.Vd,_.vd,3)||new _.vd;Ed=_.Wd()||new _.Cc;_.K=new Fd;Od=_.Wd()||new _.Cc;Pd=_.Xd()||new _.Dc;Nd=_.F(_.Vd,Ld,4)||new Ld;_.Zd=new Qd;
_.A("gbar_._DumpException",function(a){_.K?_.K.log(a):console.error(a)});
_.$d=new Fc(_.K);
_.Zd.log(8,{m:"BackCompat"==document.compatMode?"q":"s"});_.A("gbar.A",_.Rd);_.Rd.prototype.aa=_.Rd.prototype.then;_.A("gbar.B",_.I);_.I.prototype.ba=_.I.prototype.si;_.I.prototype.bb=_
var be=_.Xd()||new _.Dc;window.__PVT=_.u(_.D(be,8));_.zd("eq",_.$d);

}catch(e){_._DumpException(e)}
try{
var ce=function(a){_.G.call(this,a)};_.w(ce,_.G);
var de=function(){_.H.call(this);this.o=[];this.j=[]};_.w(de,_.H);de.prototype.A=function(a,b){this.o.push({features:a,options:b})};de.prototype.init=function(a,b,c){window.gapi={};var
var ee=_.F(_.Vd,_.Gc,14)||new _.Gc,fe=_.F(_.Vd,_.Hc,9)||new _.Hc,ge=new ce,he=new de;he.init(ee,fe,ge);_.zd("gs",he);

}catch(e){_._DumpException(e)}
})(this.gbar_);
// Google Inc.
</script><style>h1,ol,ul,li,button{margin:0;padding:0}button{border:none;background:none}body{background:#202124}body,input,button{font-size:14px;font-family:arial,sans-serif;color:#bdc
try{
_.ie=function(a,b,c){if(!a.o)if(c instanceof Array){c=_.Ya(c);for(var d=c.next();!d.done;d=c.next())_.ie(a,b,d.value)}else{d=(0,_.z)(a.F,a,b);var e=a.B+c;a.B++;b.setAttribute("data-eqid

}catch(e){_._DumpException(e)}
try{
/*

 Copyright The Closure Library Authors.
 SPDX-License-Identifier: Apache-2.0
*/
_.je=function(){if(!_.n.addEventListener||!Object.defineProperty)return!1;var a=!1,b=Object.defineProperty({},"passive",{get:function(){a=!0}});try{_.n.addEventListener("test",function(
_.ke=_.Fb?"webkitTransitionEnd":"transitionend";

}catch(e){_._DumpException(e)}
try{
var le=document.querySelector(".gb_z .gb_A"),me=document.querySelector("#gb.gb_Jc");le&&!me&&_.ie(_.$d,le,"click");

}catch(e){_._DumpException(e)}
try{
var Xh=function(a){_.H.call(this);this.C=a;this.A=null;this.o={};this.D={};this.j={};this.B=null};_.w(Xh,_.H);_.Yh=function(a){if(a.A)return a.A;for(var b in a.j)if(a.j[b].Ne()&&a.j[b].
_.k.Ch=function(a){return!this.D[a.Rc()]};_.k.Jg=function(a){this.j[a]&&(_.Yh(this)&&_.Yh(this).Rc()==a||this.j[a].Hd(!0))};_.k.Wa=function(a){this.B=a;for(var b in this.j)this.j[b].Ne(

}catch(e){_._DumpException(e)}
try{
var zj=document.querySelector(".gb_Ma .gb_A"),Aj=document.querySelector("#gb.gb_Jc");zj&&!Aj&&_.ie(_.$d,zj,"click");

}catch(e){_._DumpException(e)}
})(this.gbar_);
// Google Inc.
```

Almost No One Uses HTML

In 2025, almost no one builds websites in HTML anymore due to time/skill required.

- Frameworks are almost universally used for building
  - ColdFusion, NodeJS, TrueForms, ASP.NET Laravel, Django, PHP, Ruby on Rails, Flask

It's 2019 and I Still Make Websites with my Bare Hands

Matt Holt Follow
Dec 27, 2018 · 10 min read

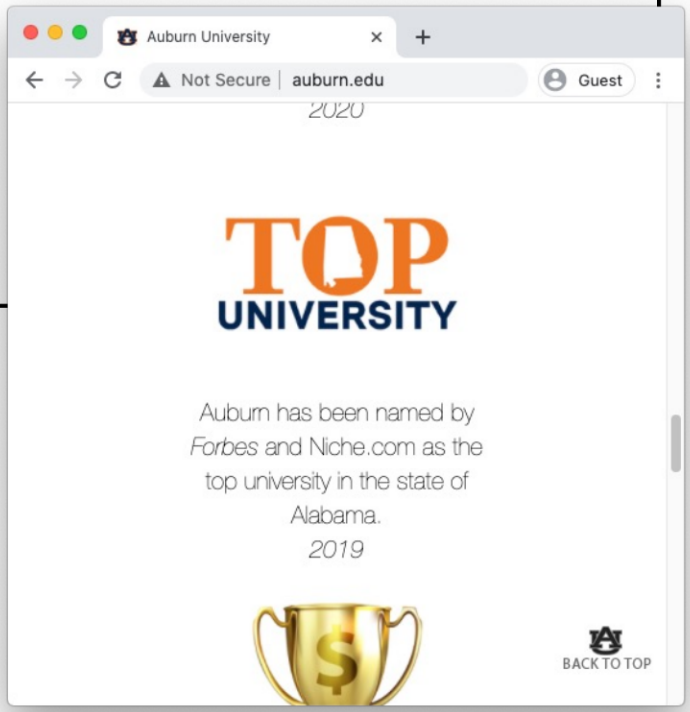
I have no idea how to make a website the way the cool kids do today.

All I know is that our frontend team spent about a day laying the foundation for our new website, and the next day my `git pull` landed this thing (after a post-merge hook):

Hypertext Markup Language (HTML)

The **Hypertext Markup Language (HTML)** is root-mechanism for all websites' content.

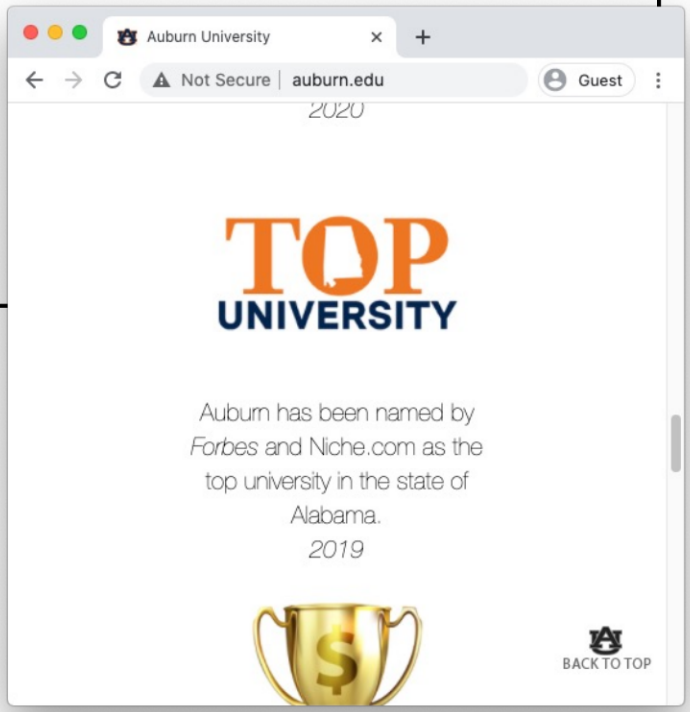
```
<div class="item active">
  <div class="image">
    <img alt="Doctor bag with paw print on it" src="//www.auburn.edu/template/2016/_assets/images/graphic-topu.png">
  </div>
  <div class="text2">
    "Auburn has been named by "
    <em>Forbes</em>
    " and Niche.com as the top university in the state of Alabama."
    <br>
    <em>2019</em>
  </div>
</div>
```



Hypertext Markup Language (HTML)

The **Hypertext Markup Language (HTML)** is root-mechanism for all websites' content.

```
▼<div class="item active">
  ▼<div class="image">
      <img alt="Doctor bag with paw print on it" src="//www.auburn.edu/template/2016/_assets/images/graphic-topu.png">
    </div>
  ▼<div class="text2">
      "Auburn has been named by "
      <em>Forbes</em>
      " and Niche.com as the top university in the state of Alabama."
      <br>
      <em>2019</em>
    </div>
  </div>
</div>
```



Almost No One Uses HTML

In 2020, almost no one builds websites in HTML anymore due to time/skill required.

- Frameworks are almost universally used for building
  - ColdFusion, NodeJS, TrueForms, ASP.NET Laravel, Django, PHP, Ruby on Rails, Flask
- **WORDPRESS SHOULD DIE IN A FIRE**

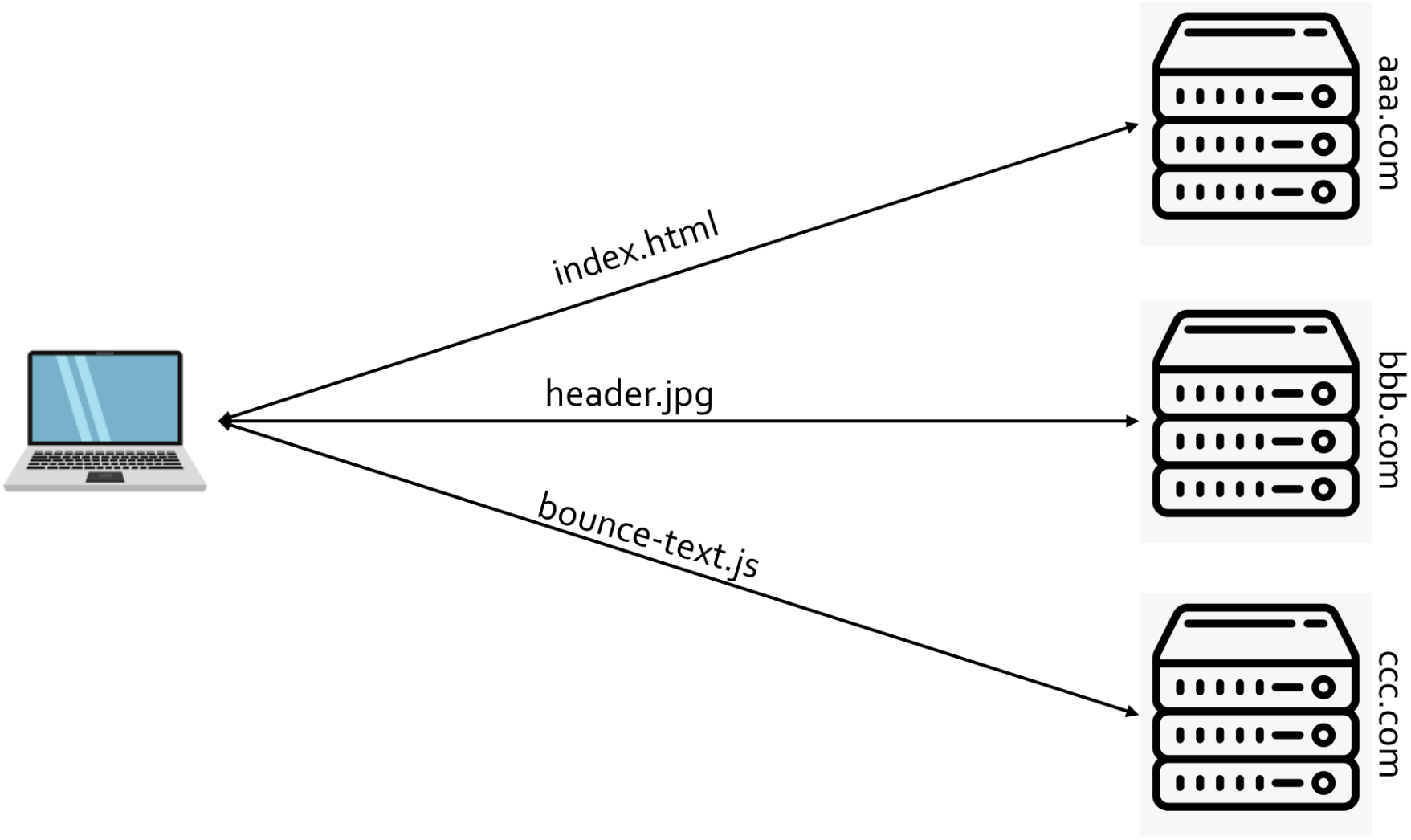
It's 2019 and I Still Make Websites with my Bare Hands

Matt Holt Follow
Dec 27, 2018 · 10 min read

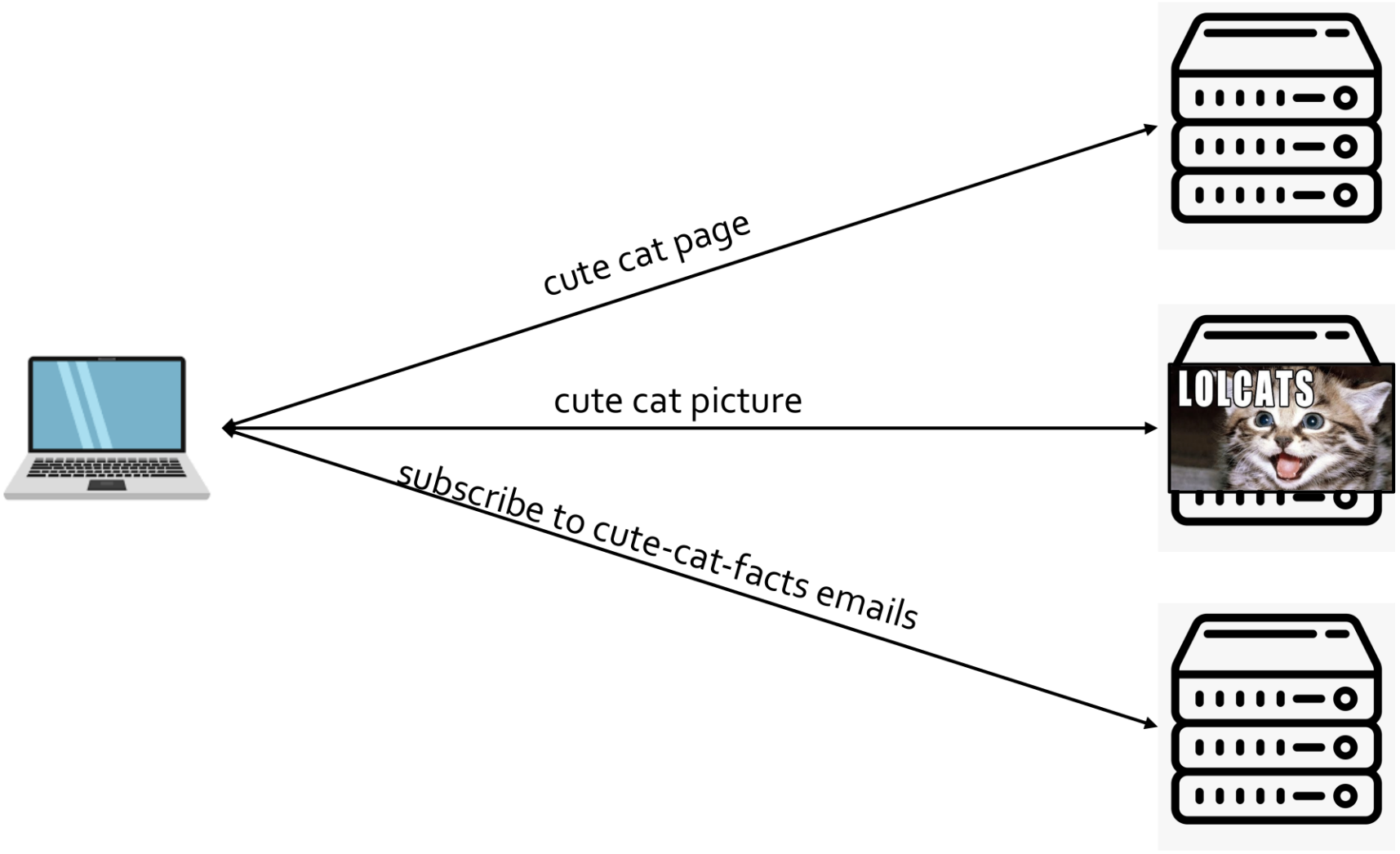
I have no idea how to make a website the way the cool kids do today.

All I know is that our frontend team spent about a day laying the foundation for our new website, and the next day my `git pull` landed this thing (after a post-merge hook):

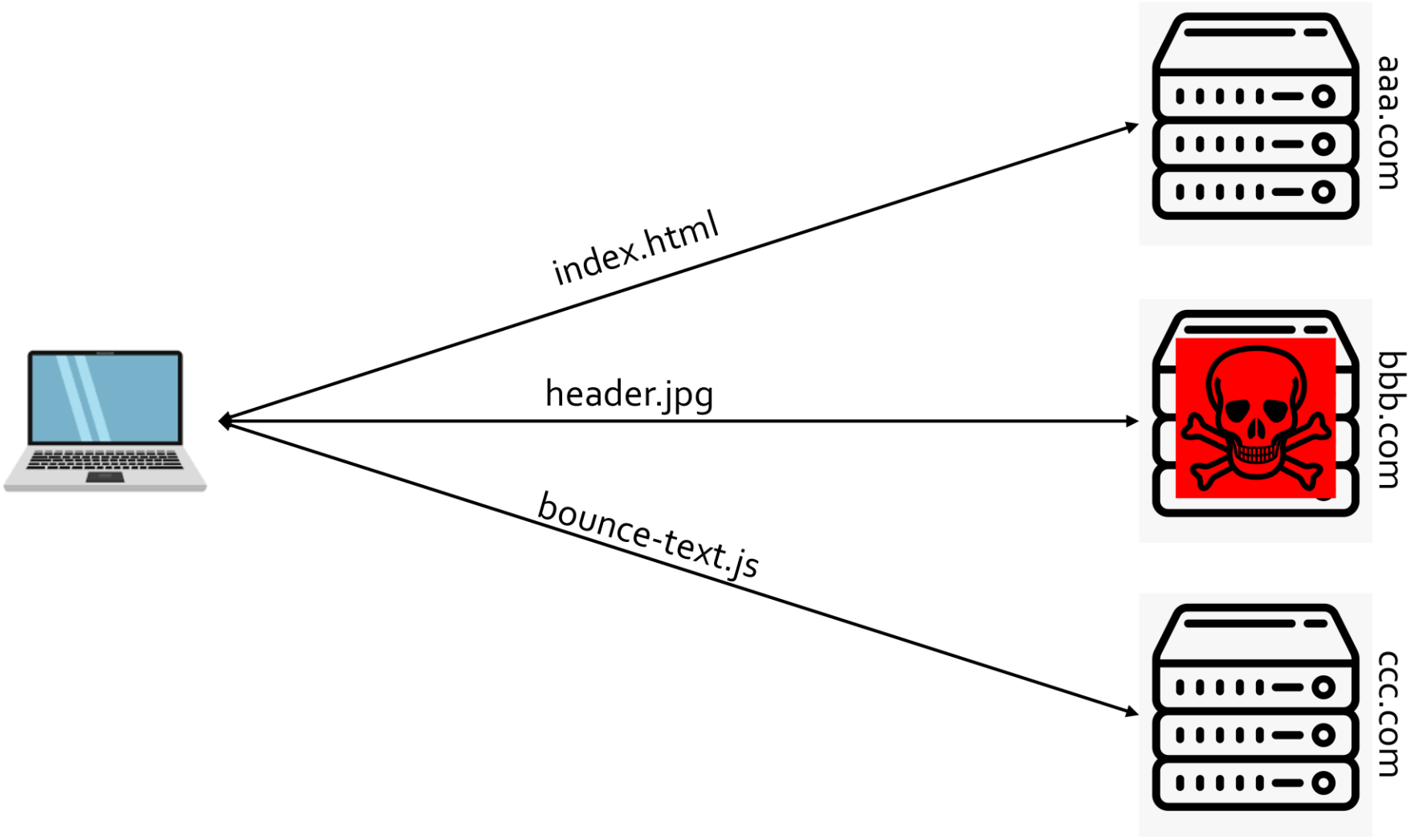
HTML Resource Fetching



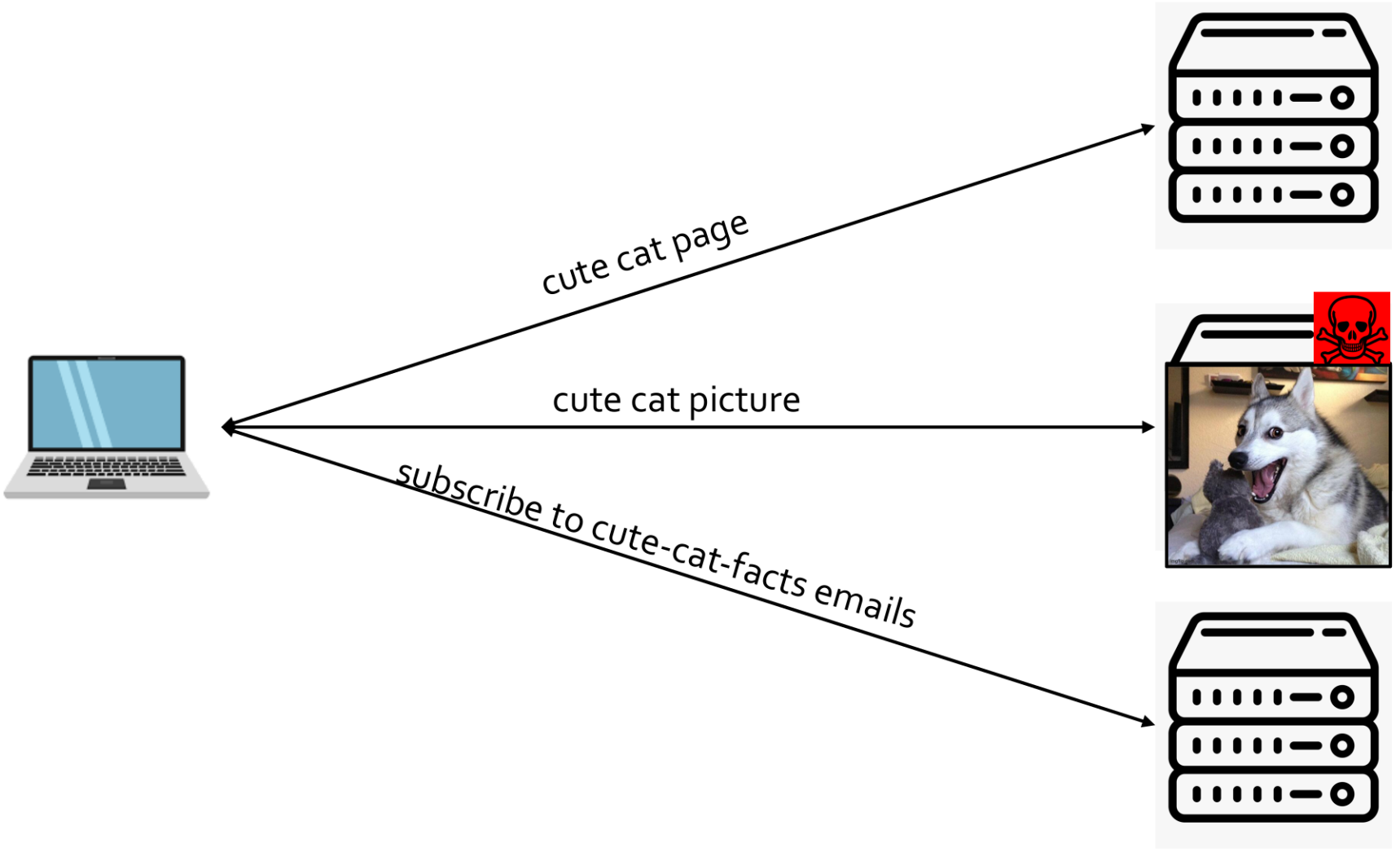
HTML Resource Fetching



Domain Take-Over Attacks



Domain Take-Over Attacks



Core Web Components

- Direct navigation
  - URLs are a poor security foundation
- Communication protocol
  - Fundamental abstractions == fundamental assumptions
- Content rendering
  - Weak dependencies via frameworks/infrastr.

Core Web Components

- Direct navigation
  - URLs are a poor security foundation
- Communication protocol
  - Fundamental abstractions == fundamental assumptions
- Content rendering
  - Weak dependencies via frameworks/infrastr.
- Content execution

Server-Side Exec vs Client-Side Exec

Server-Side

- Server stores code locally
- Server executes to create/modify HTML
- Client receives finalized HTML

EX: PHP, ASP.NET, Ninja, …

Server-Side Exec vs Client-Side Exec

Server-Side

- Server stores code locally
- Server executes to create/modify HTML
- Client receives finalized HTML

EX: PHP, ASP.NET, Ninja, …

Client-Side Exec

- Server gives code to every client
- Clients execute to create/modify HTML

EX: Javascript, WebAssembly (WASM) …

JavaScript is Stupid

```
» 10-1
← 9
```

```
» 10-1
← 9
» 'ten'-1
← NaN
```

```
» 10-1
← 9
» 'ten'-1
← NaN
» '10'-1
← 9
```

```
» 10-1
← 9
» 'ten'-1
← NaN
» '10'-1
← 9
» '10'-'1'
← 9
```

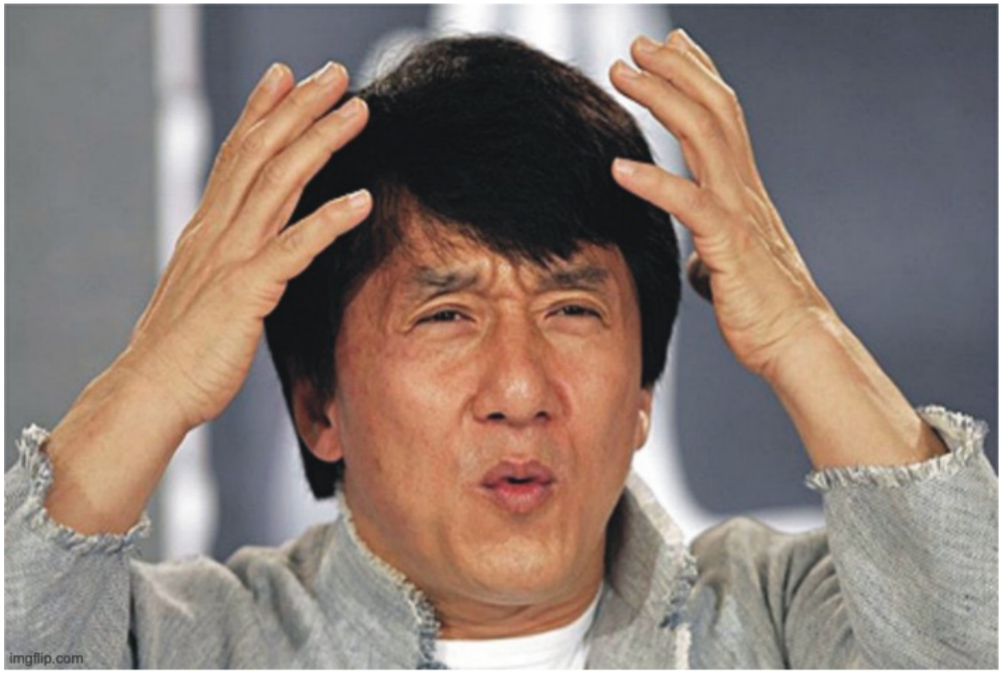
JavaScript is Stupid

```
» 10-1
← 9
```

```
» 10-1
← 9
» 'ten'-1
← NaN
```

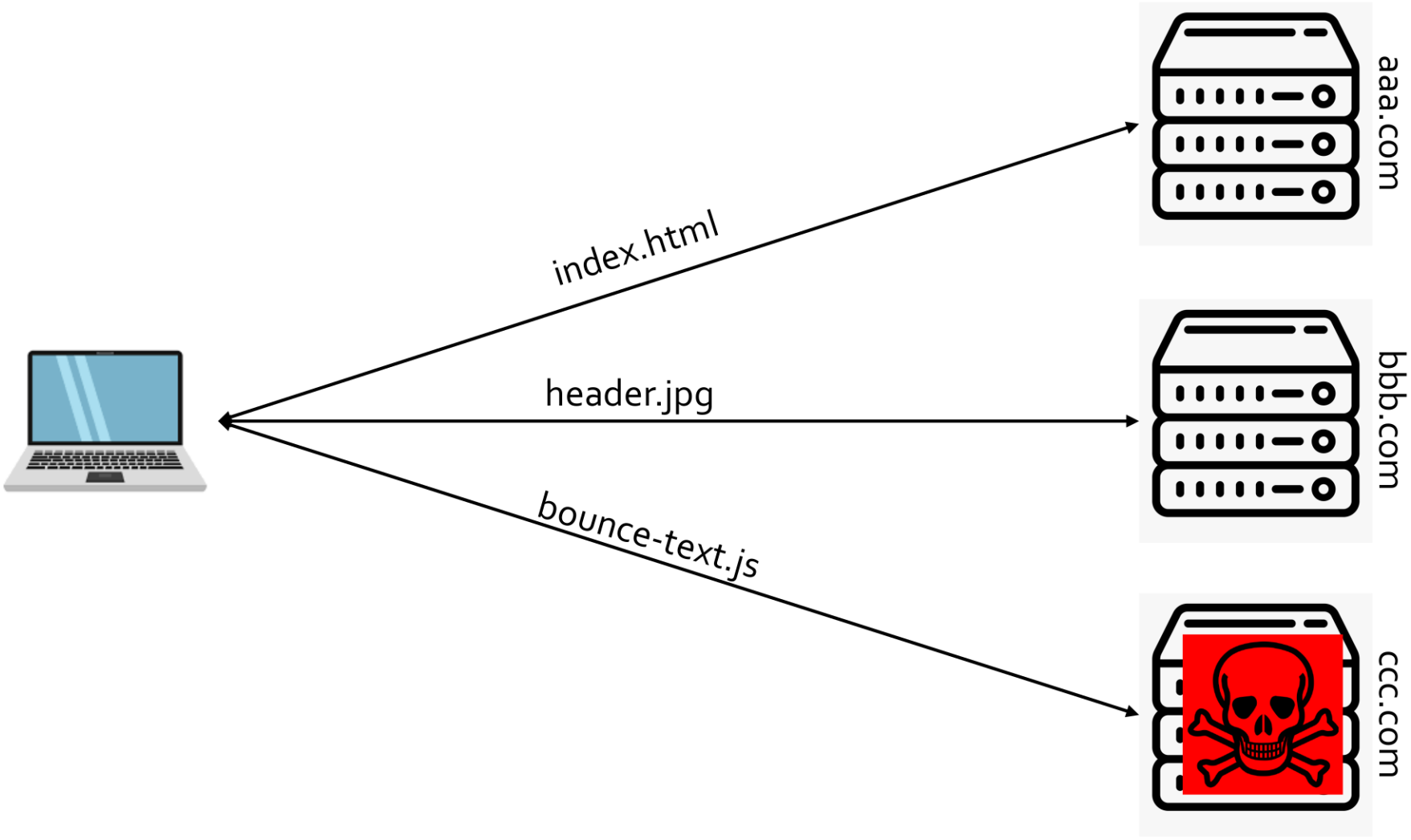
```
» 10-1
← 9
» 'ten'-1
← NaN
» '10'-1
← 9
```

```
» 10-1
← 9
» 'ten'-1
← NaN
» '10'-1
← 9
» '10'-'1'
← 9
```



JavaScript can be Dangerous

Domain Take-Over Attacks



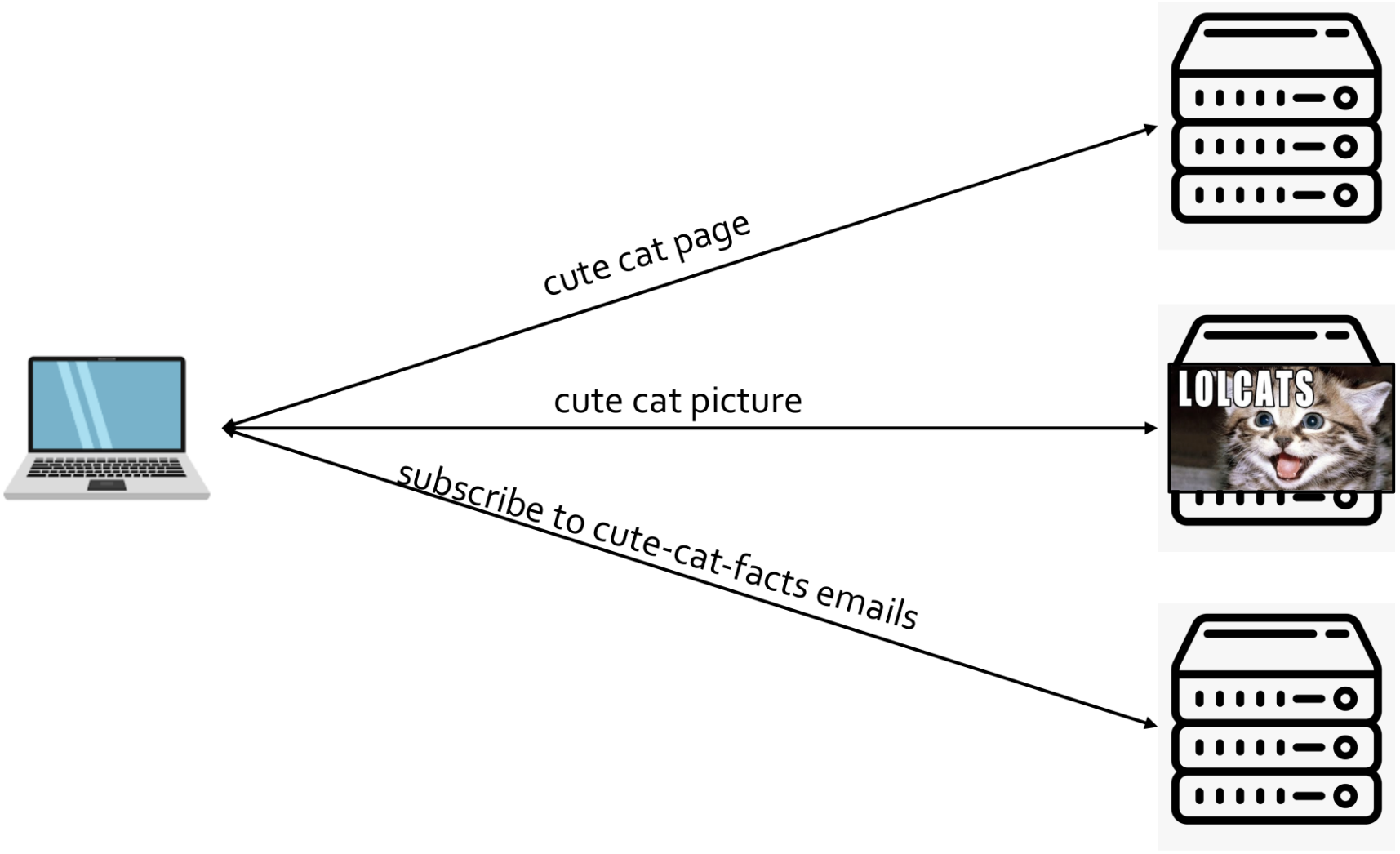
Core Web Components

- Direct navigation
  - URLs are a poor security foundation
- Communication protocol
  - Fundamental abstractions == fundamental assumptions
- Content rendering
  - Weak dependencies via frameworks/infrastr.
- Content execution
  - Poor design/architecture

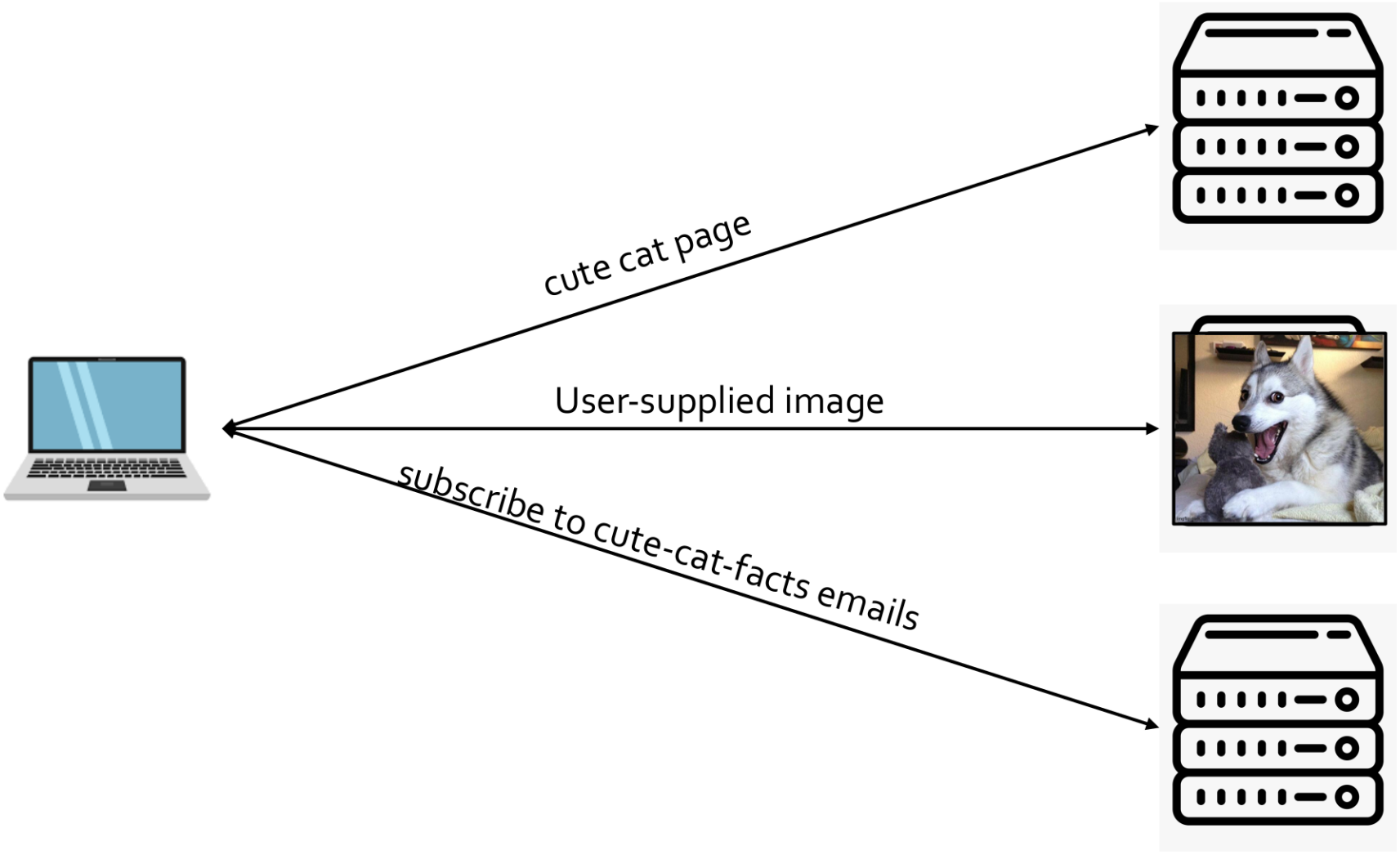
Core Web Components

- Direct navigation
  - URLs are a poor security foundation
- Communication protocol
  - Fundamental abstractions == fundamental assumptions
- Content rendering
  - Weak dependencies via frameworks/infrastr.
- Content execution
  - Poor design/architecture
- Untrusted data dependencies

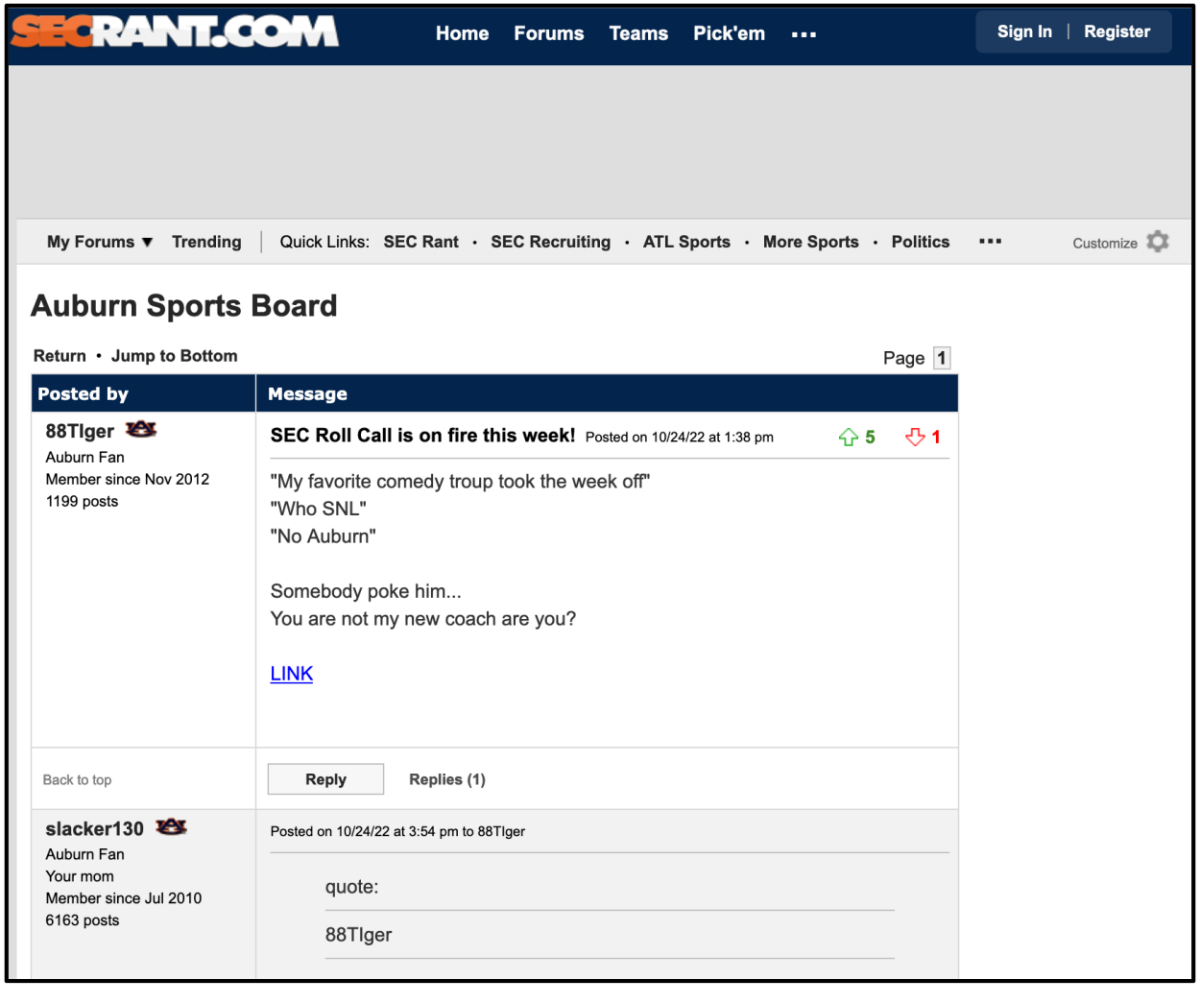
HTML Resource Fetching



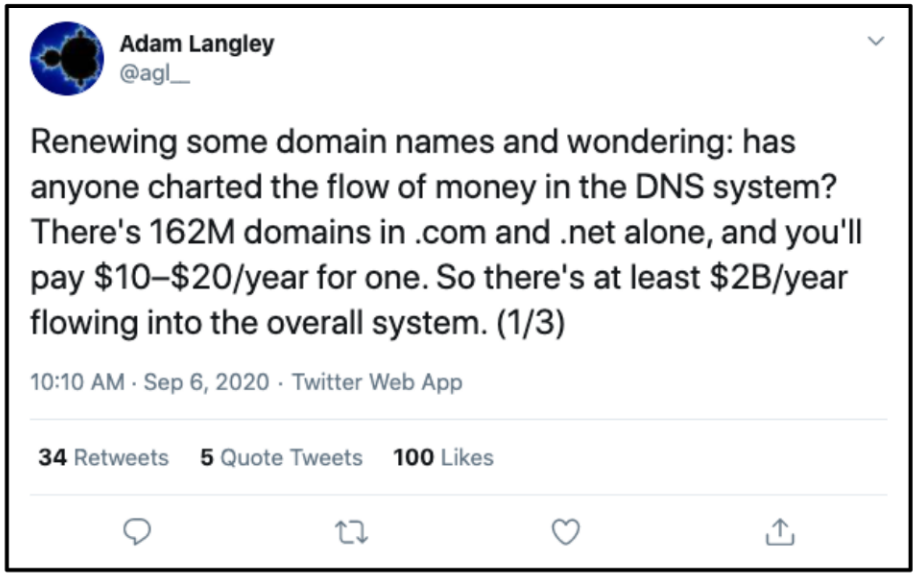
HTML Resource Fetching



User-Supplied Data Relays



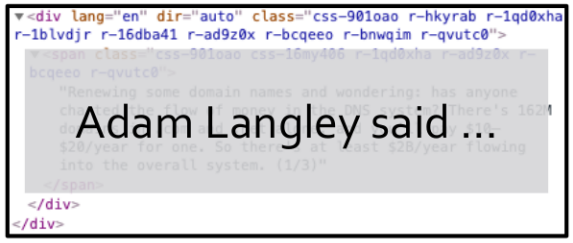
Content Injection Concepts



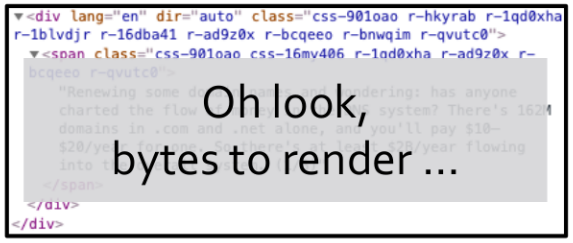

```
▼<div lang="en" dir="auto" class="css-901oao r-hkyrab r-1qd0xha
r-1blvdjr r-16dba41 r-ad9z0x r-bcqeeo r-bnwqim r-qvutc0">
  ▼<span class="css-901oao css-16my406 r-1qd0xha r-ad9z0x r-
  bcqeeo r-qvutc0">
      "Renewing some domain names and wondering: has anyone
      charted the flow of money in the DNS system? There's 162M
      domains in .com and .net alone, and you'll pay $10–
      $20/year for one. So there's at least $2B/year flowing
      into the overall system. (1/3)"
    </span>
  </div>
</div>
```

Server Interpretation

Human Interpretation



Browser Interpretation



Computer and Network Security

Lecture 18:
WWW & Web Attacks

COMP-5370/6370
Fall 2025