

# Server Farm to Table

Or, How the Internet Works.

# 0. User Enters URL



# 0. User Enters URL



# 1. IP Address Lookup

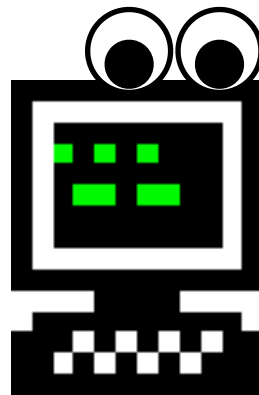
I want numbers!!!!



Client

# 1. IP Address Lookup

I want numbers!!!!



Client

# 1. IP Address Lookup

Wait, did I, like, just  
meet you?



Client

Browser checks its cache.

# 1. IP Address Lookup

Wait, did my user  
want this to go  
somewhere special?



Client

Browser checks /etc/hosts

# 1. IP Address Lookup

Wait, did I or one of  
my friends talk to you,  
somewhat recently?

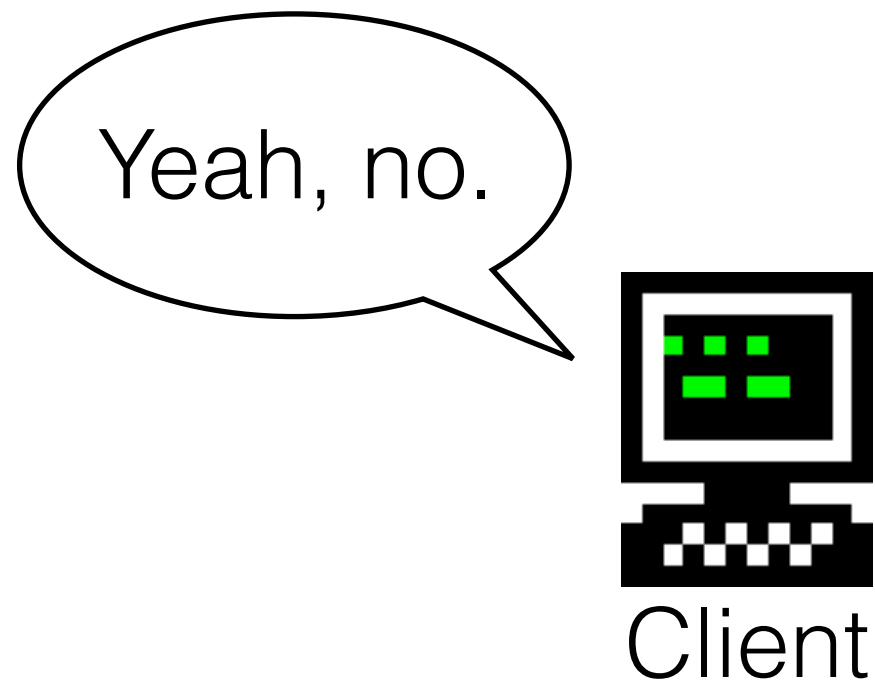


Client

Browser checks router and ISP cache

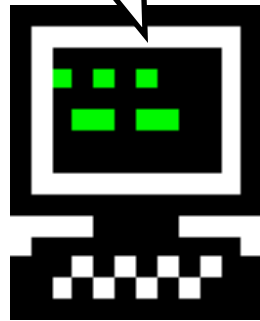


# 1. IP Address Lookup



# 1. IP Address Lookup

Do you have  
a `www.google.com`?



Client

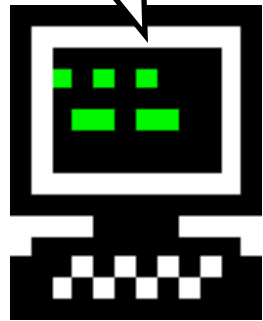


DNS Server

Browser hits DNS server which returns answer  
(...after it asks a bunch more servers)

# 1. IP Address Lookup

Do you have  
a `www.google.com`?



Client



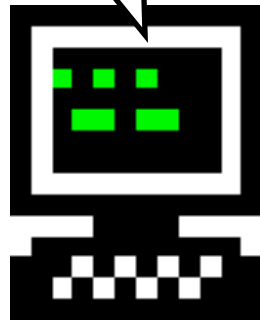
DNS Server

(Recursiveness)

Browser hits DNS server which returns answer  
(...after it asks a bunch more servers)

# 1. IP Address Lookup

Do you have  
a `www.google.com`?



Client

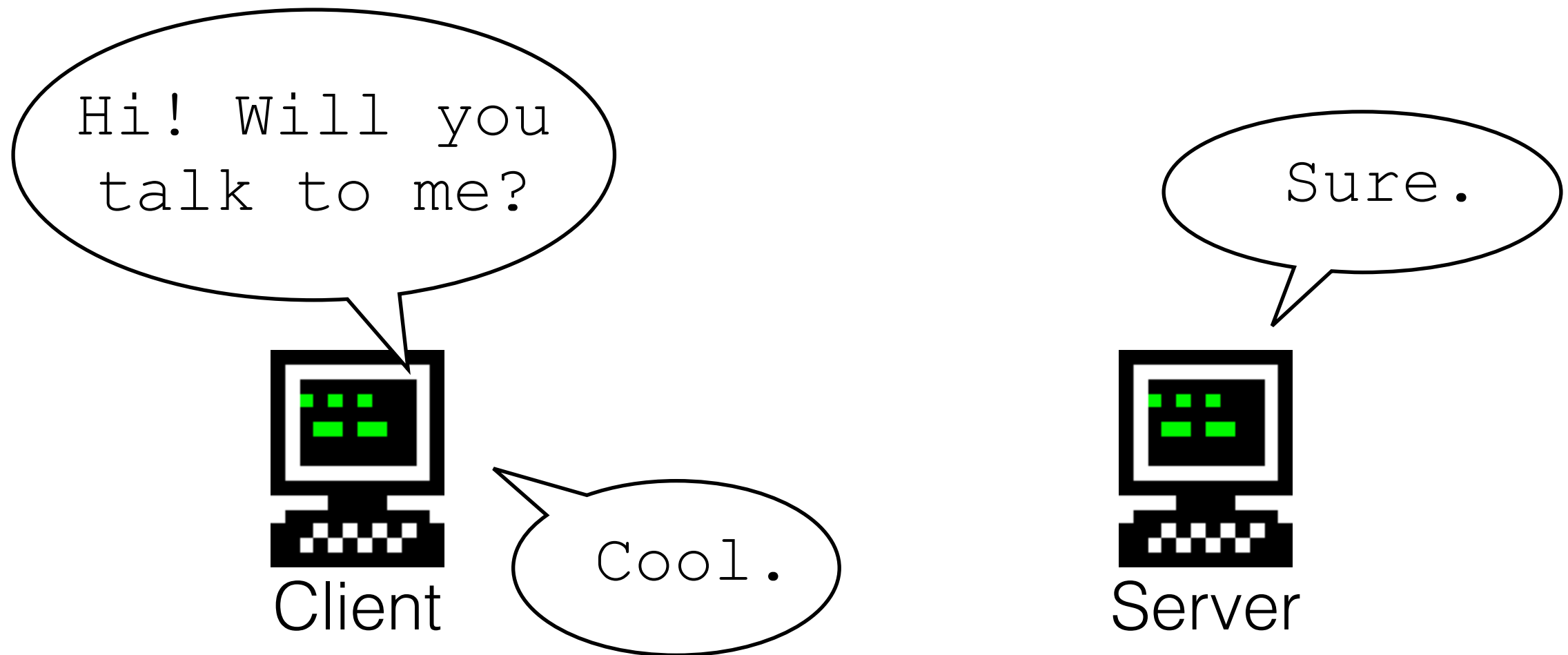
`74.125.226.1`



DNS Server

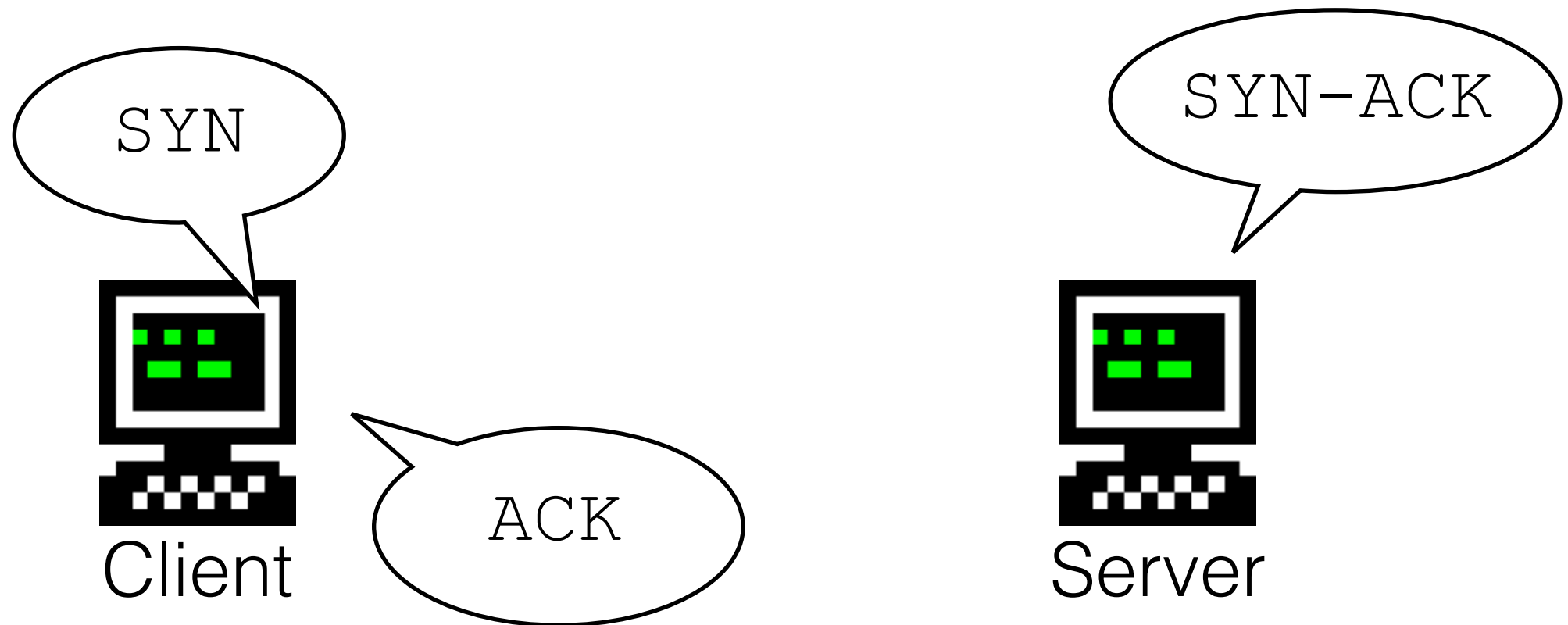
Browser hits DNS server which returns answer  
(...after it asks a bunch more servers)

## 2. TCP Handshake



Client establishes TCP connection(s) with the server

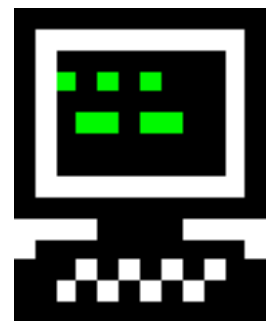
## 2. TCP Handshake



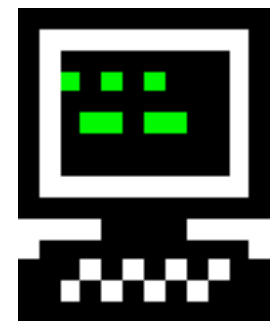
Client establishes TCP connection(s) with the server

# 3. HTTP request

```
> GET / HTTP/1.1  
> User-Agent: Mozilla/5.0...  
> Host: google.com  
> Accept: */*
```



Client



Server

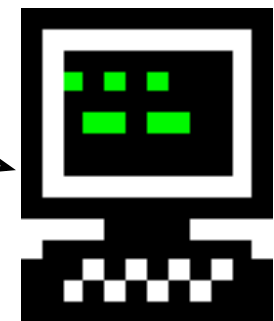
Client sends HTTP request to server

# 4. HTTP response



Client

```
< HTTP/1.1 200 OK
< Date: Thu, 07 Aug 2014 17:23:50 GMT
< Expires: -1
< Cache-Control: private, max-age=0
< Content-Type: text/html; charset=ISO-8859-1
< Server: gws
< X-XSS-Protection: 1; mode=block
< X-Frame-Options: SAMEORIGIN
< Alternate-Protocol: 80:quic
< Transfer-Encoding: chunked
<
<doctype !html><html>...</html>
```



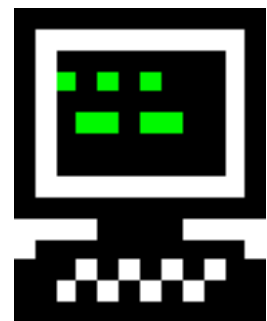
Server

Server responds to client with HTTP response containing header followed by HTML document

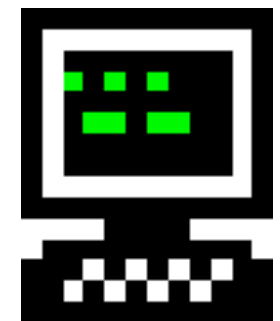
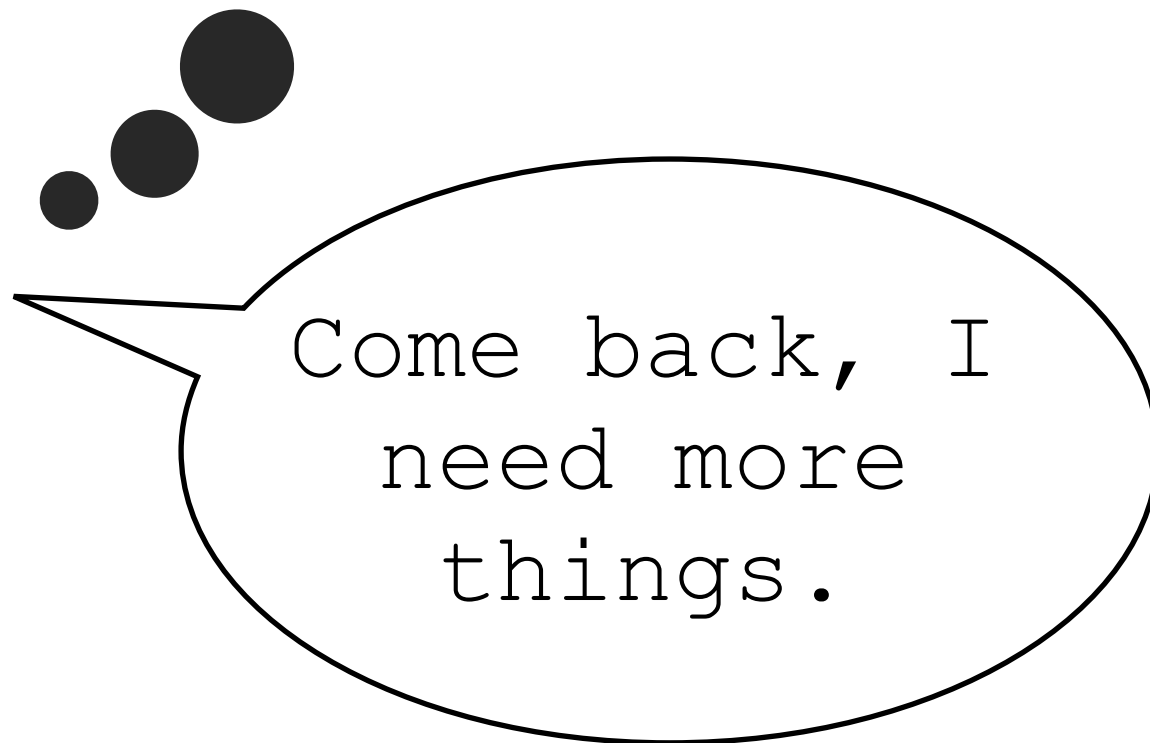


# 5. HTML Parsing

```
1 <!doctype html>
2 <html>
3   <head>
4     <script type="text/javascript" src="library.js"></script>
5     <link rel="stylesheet" type="text/css" href="styles.css">
6     <script type="text/javascript" src="website.js"></script>
7   </head>
8   <body>
9     <div>Let's pretend this is all of google</div>
10  </body>
11 </html>
```



Client



Server

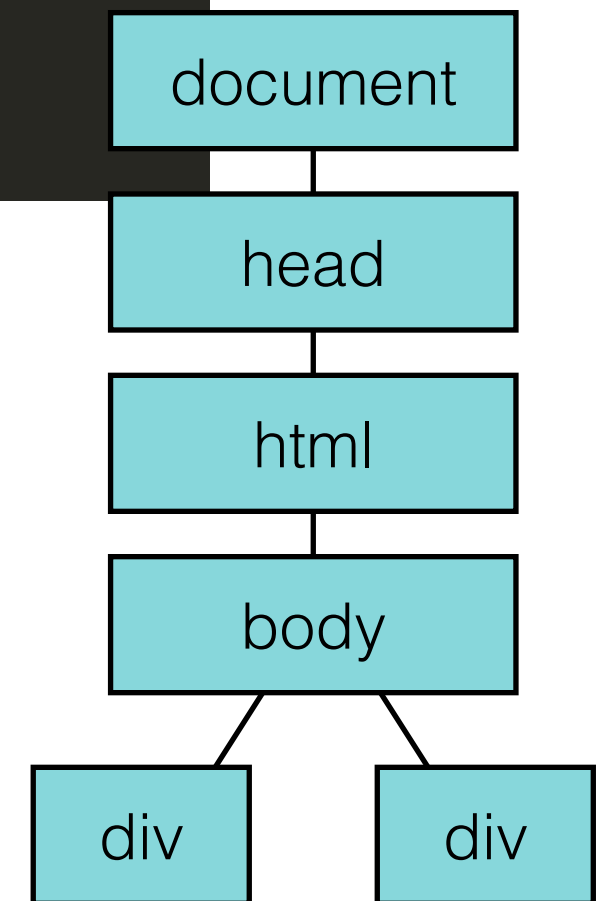
1. Speculative parser looks ahead for assets it can fetch

# 5. HTML Parsing



Client

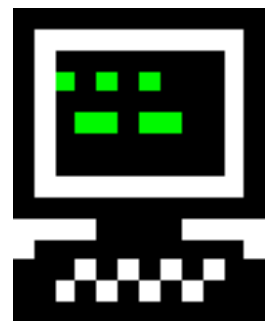
```
1  <!doctype html>
2  <html>
3    <head>
4      <script type="text/javascript" src="library.js"></script>
5      <link rel="stylesheet" type="text/css" href="styles.css">
6      <script type="text/javascript" src="website.js"></script>
7    </head>
8    <body>
9      <div>Let's pretend this is all of google</div>
10   </body>
11  </html>
```



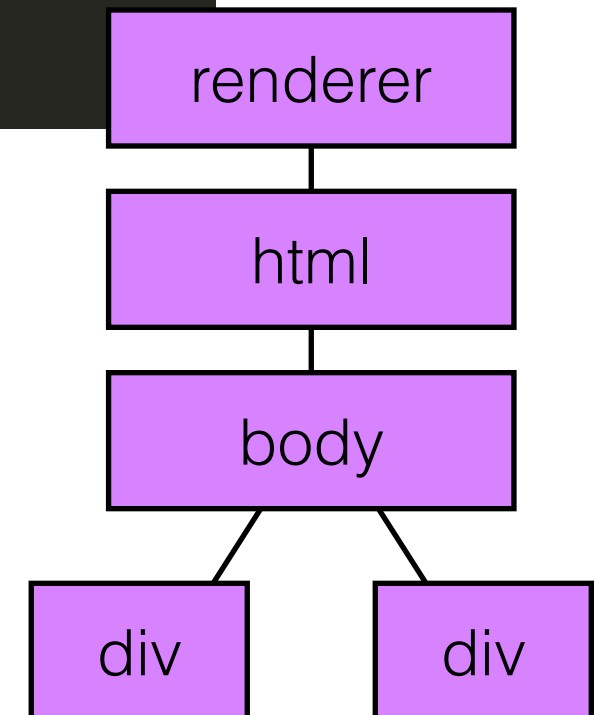
2. Main parser parses the HTML document, building a DOM tree out of the (likely broken...) HTML

# 5. HTML Parsing

```
1  <!doctype html>
2  <html>
3    <head>
4      <script type="text/javascript" src="library.js"></script>
5      <link rel="stylesheet" type="text/css" href="styles.css">
6      <script type="text/javascript" src="website.js"></script>
7    </head>
8    <body>
9      <div>Let's pretend this is all of google</div>
10   </body>
11 </html>
```



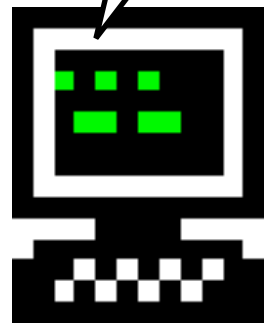
Client



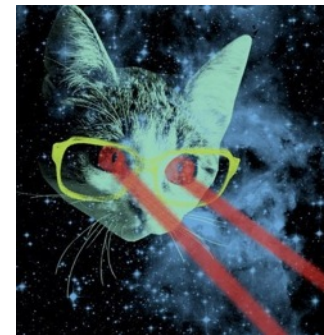
3. Render tree is also made based on the DOM, containing only things that get rendered (so, not `<head>` or `display: none`)

# 5. HTML Parsing

Just doing that  
thing that browsers  
do, ya know?



Client

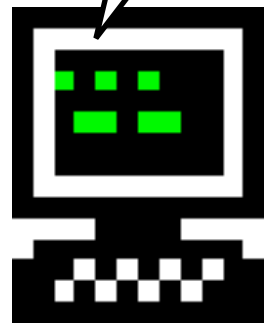


```
<script src="library.js"></script>  
<link href="styles.css">  
<script src="website.js"></script>  
<body></body>
```

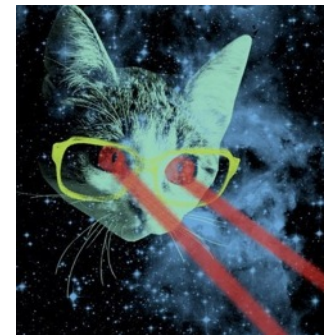
Client requests, parses, and executes all inlined  
assets (images, scripts, stylesheets).

# 5. HTML Parsing

Just doing that  
thing that browsers  
do, ya know?



Client

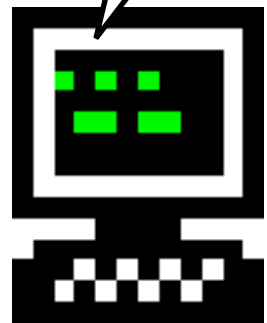


```
<script src="library.js"></script>  
<link href="styles.css">  
<script src="website.js"></script>  
<body></body>
```

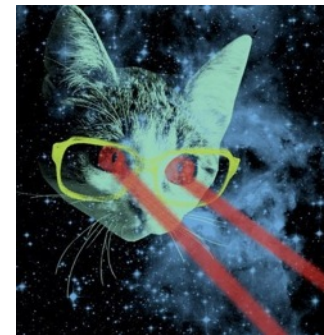
Client requests, parses, and executes all inlined  
assets (images, scripts, stylesheets).

# 5. HTML Parsing

Just doing that  
thing that browsers  
do, ya know?



Client



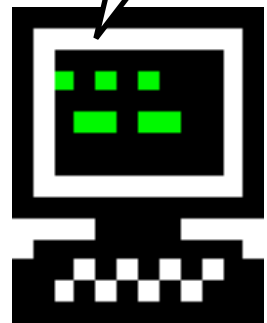
```
<script src="library.js"></script>  
<link href="styles.css">✓  
<script src="website.js"></script>  
<body></body>
```

Client requests, parses, and executes all inlined  
assets (images, scripts, stylesheets).

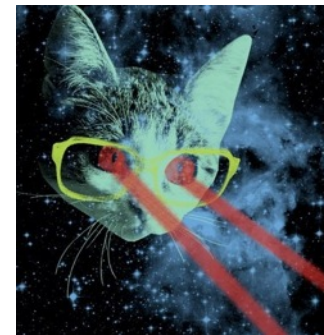


# 5. HTML Parsing

Just doing that  
thing that browsers  
do, ya know?



Client

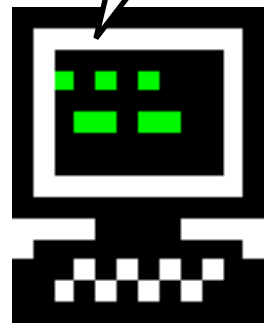


```
<script src="library.js"></script> ✓  
<link href="styles.css"> ✓  
<script src="website.js"></script>  
<body></body>
```

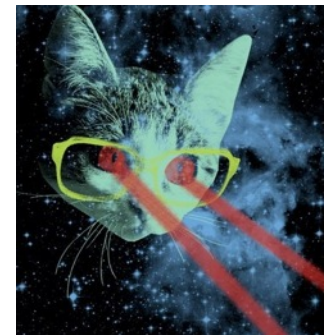
Client requests, parses, and executes all inlined  
assets (images, scripts, stylesheets).

# 5. HTML Parsing

Just doing that  
thing that browsers  
do, ya know?



Client



```
<script src="library.js"></script> ✓  
<link href="styles.css"> ✓  
<script src="website.js"></script> ✓  
<body></body>
```

Client requests, parses, and executes all inlined  
assets (images, scripts, stylesheets).



# 5. HTML Parsing

```
1  <!doctype html>
2  <html>
3    <head>
4      <script type="text/javascript" src="library.js"></script>
5      <link rel="stylesheet" type="text/css" href="styles.css">
6      <script type="text/javascript" src="website.js"></script>
7    </head>
8    <body>
9      <div>Let's pretend this is all of google</div>
10   </body>
11 </html>
```



Client

Whew. Finally got to  
that `</html>`

When client finishes parsing, DOMInteractive is fired and page is “ready”, and deferred scripts are downloaded, after which the load event is fired





**THE INTERNET!!!!**