# Server Farm to Table

## Or, How the Internet Works.

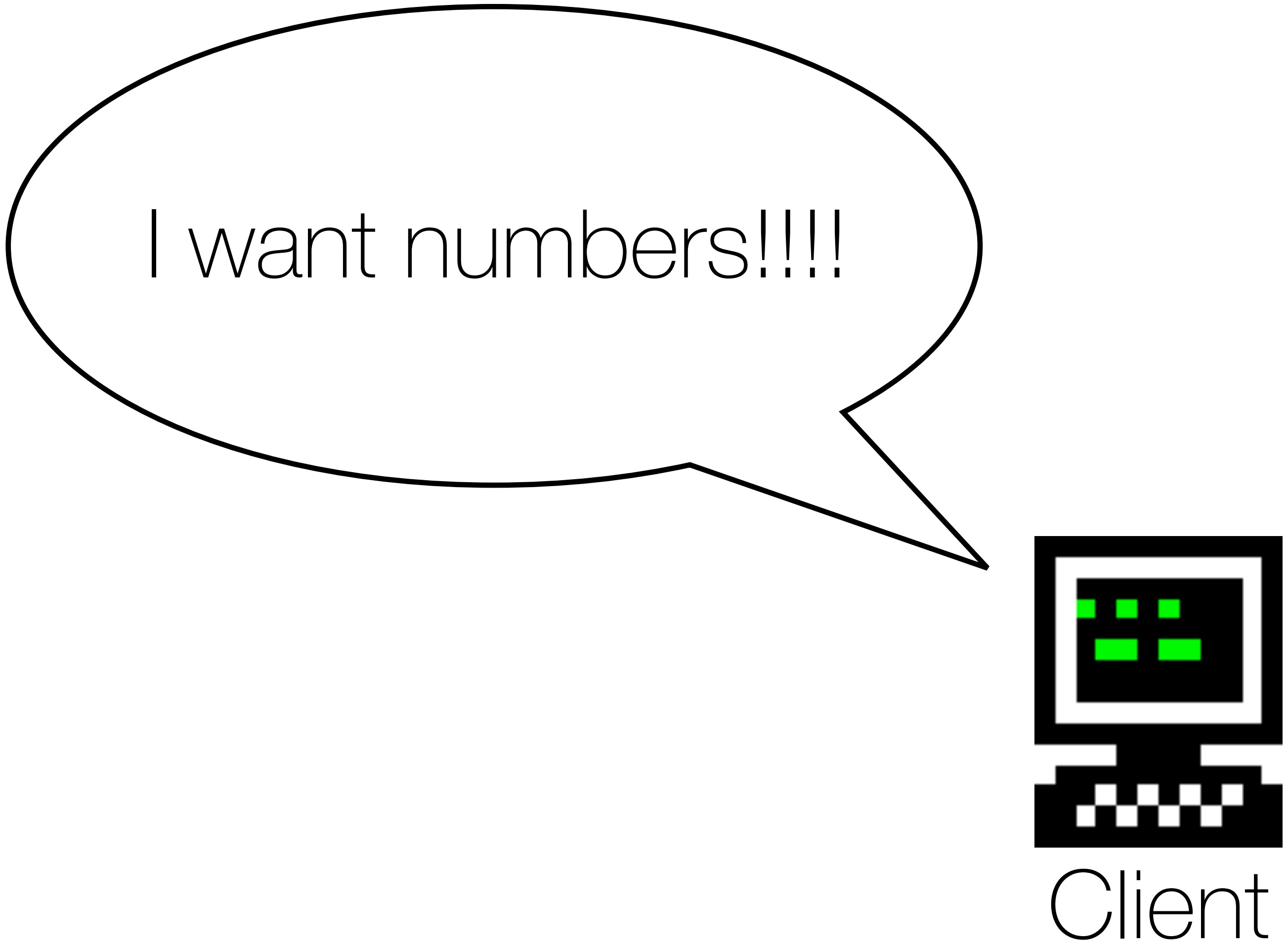
# 0. User Enters URL

http://www.google.com

# 0. User Enters URL

http://www.google.com

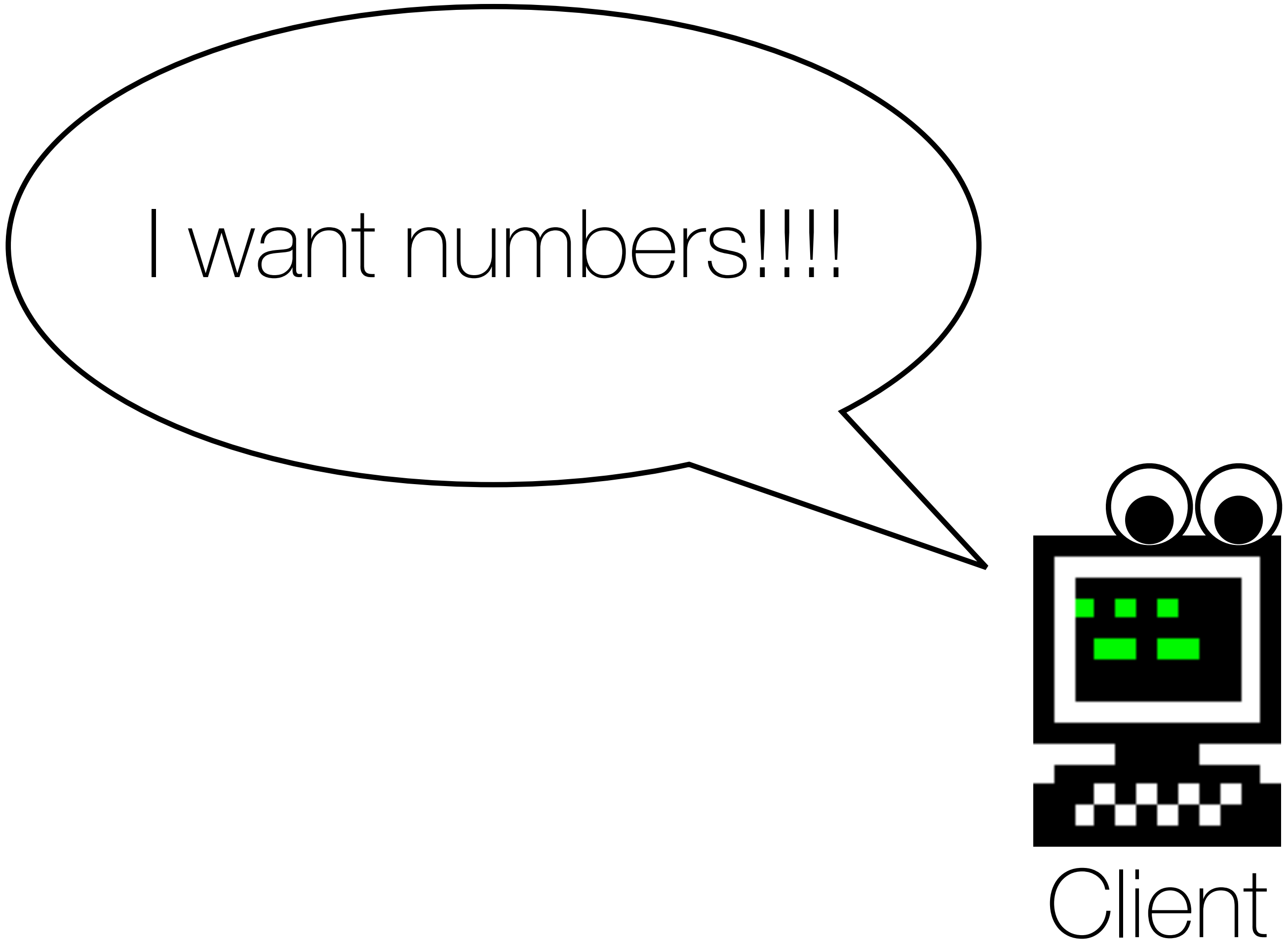
# 1. IP Address Lookup



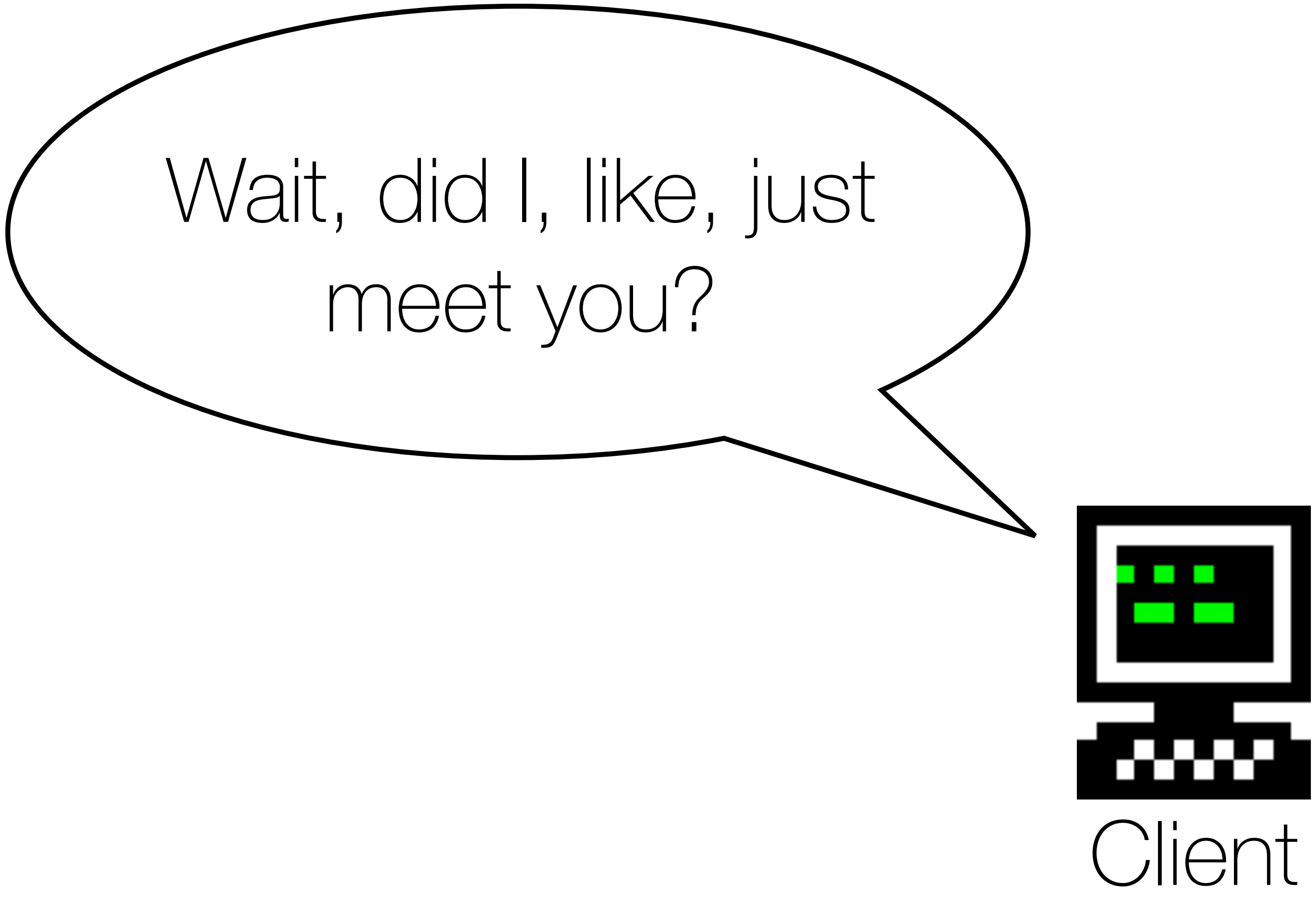
I want numbers!!!!

Client

# 1. IP Address Lookup



# 1. IP Address Lookup



Browser checks its cache.

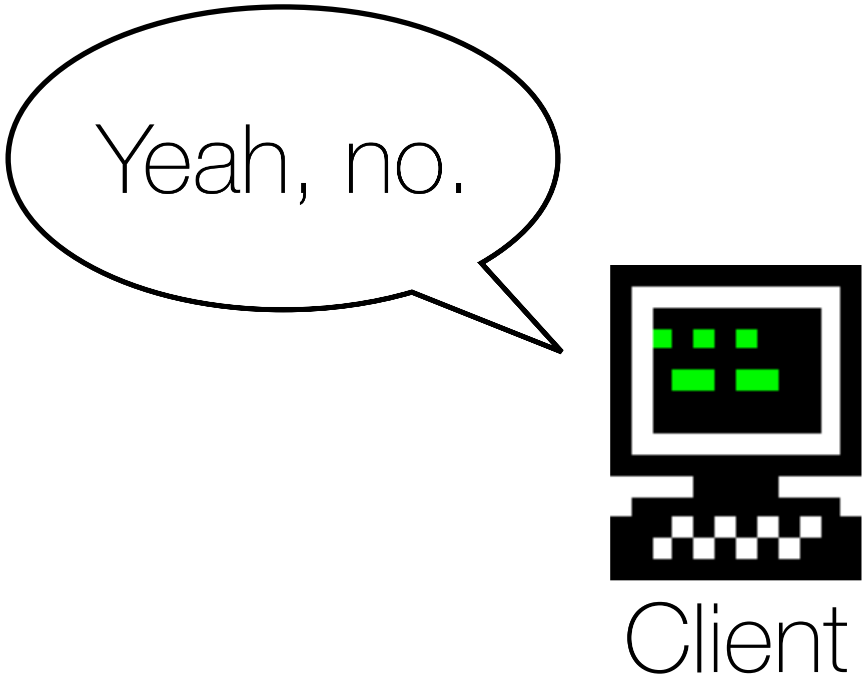
# 1. IP Address Lookup

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

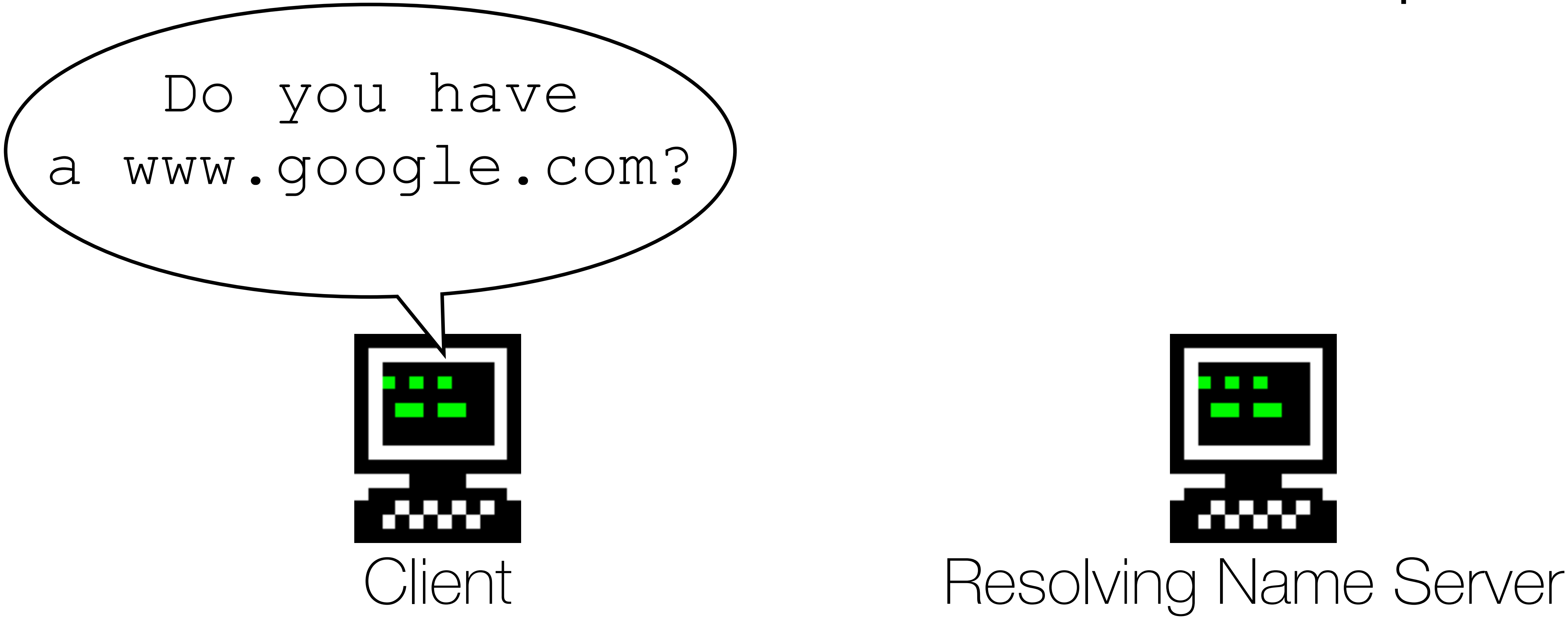
Client

Browser checks /etc/hosts

# 1. IP Address Lookup



# 1. IP Address Lookup



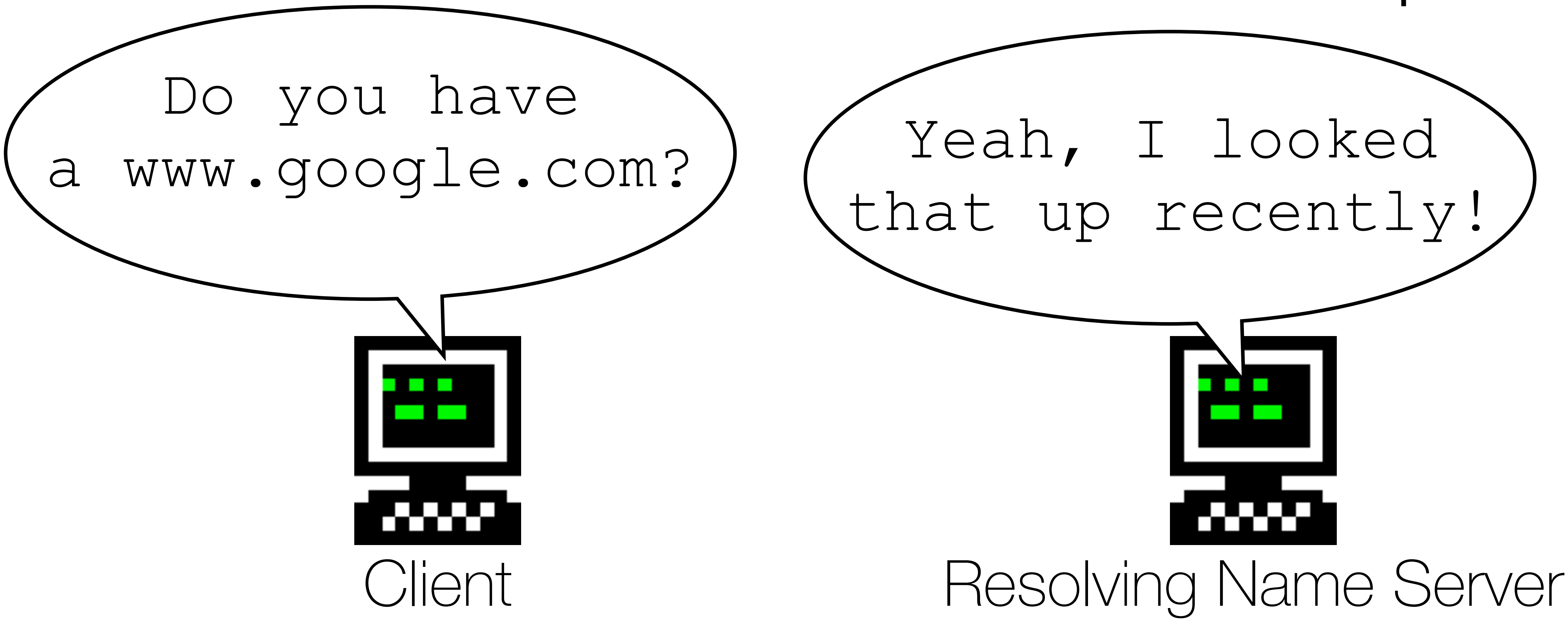
Do you have
a www.google.com?

Client

Resolving Name Server

Browser hits DNS server which…

# 1. IP Address Lookup



Do you have
a www.google.com?

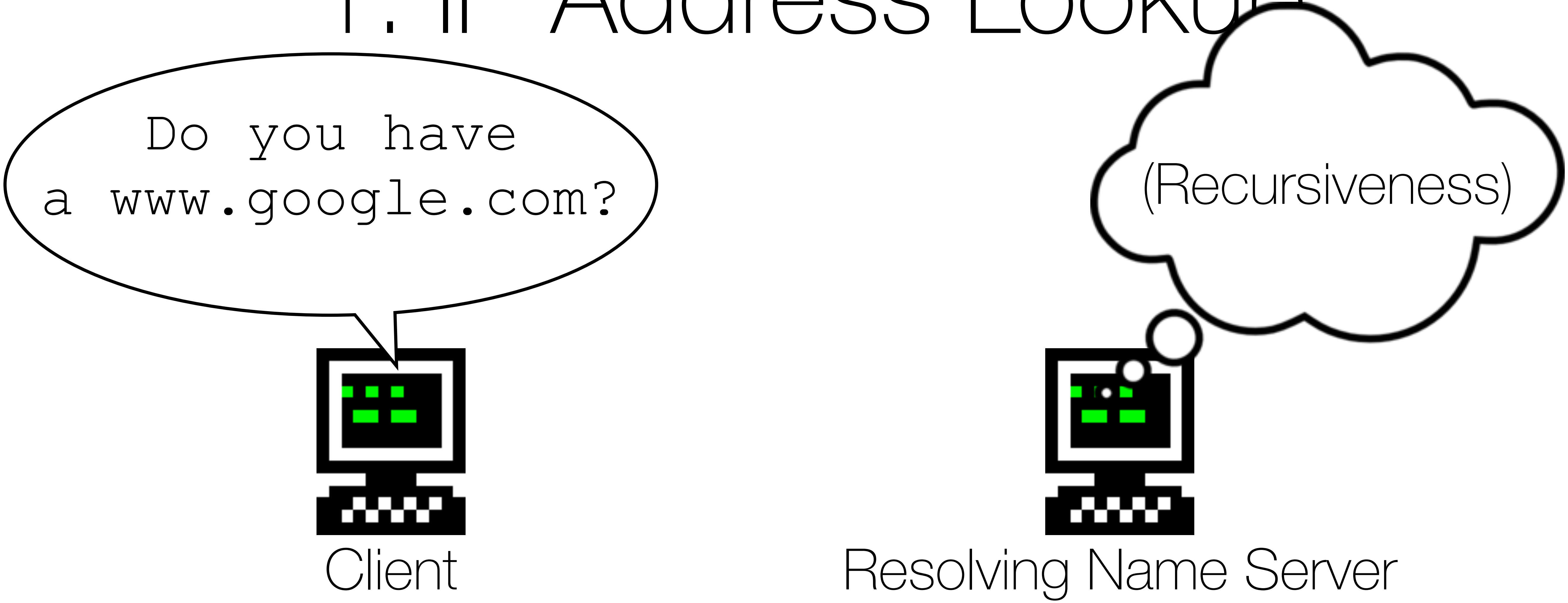
Yeah, I looked
that up recently!

Client

Resolving Name Server

…first looks in it’s cache…

# 1. IP Address Lookup



Do you have
a www.google.com?

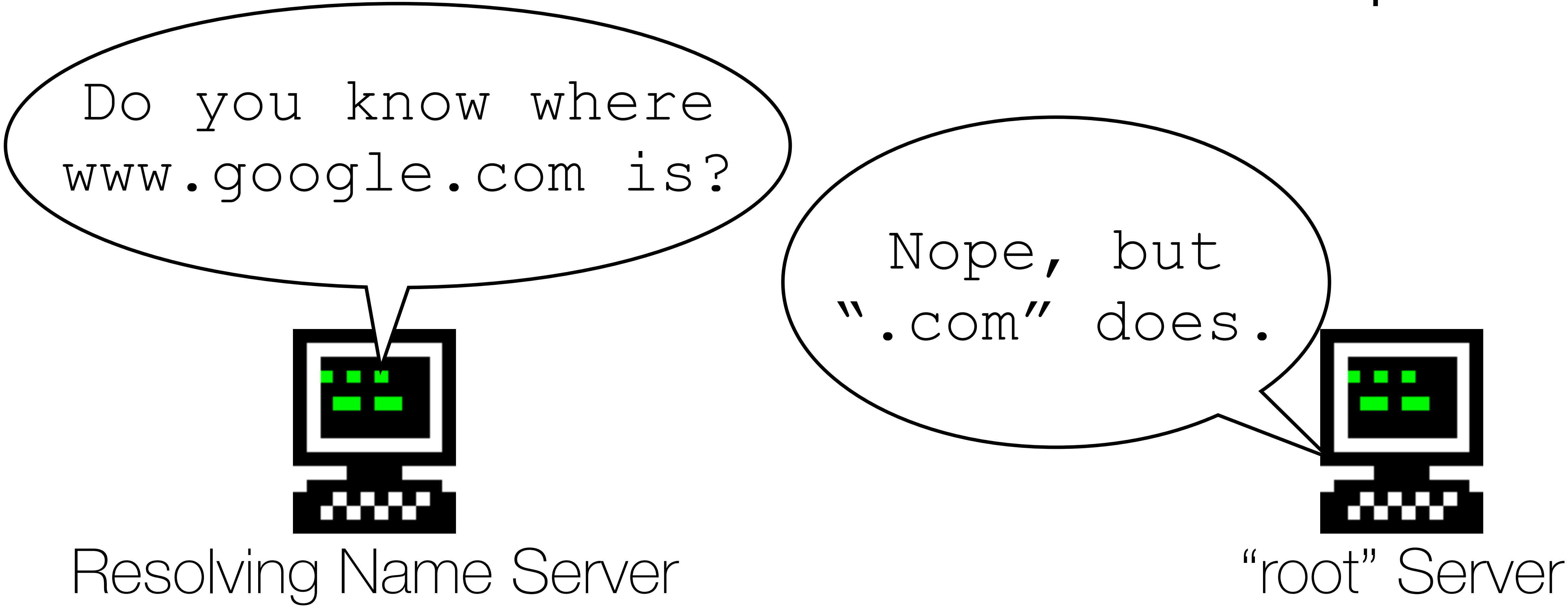
(Recursiveness)

Client

Resolving Name Server

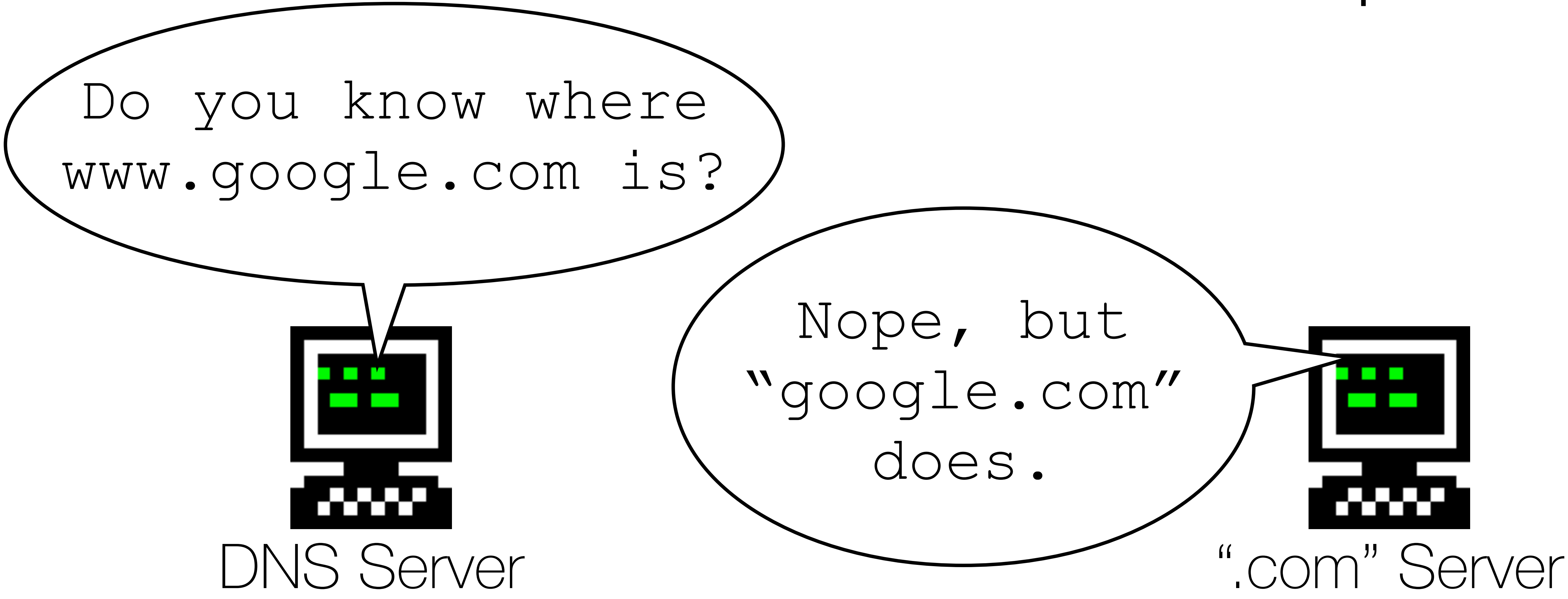
… or it asks a bunch more servers

# 1. IP Address Lookup



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

# 1. IP Address Lookup



Do you know where www.google.com is?

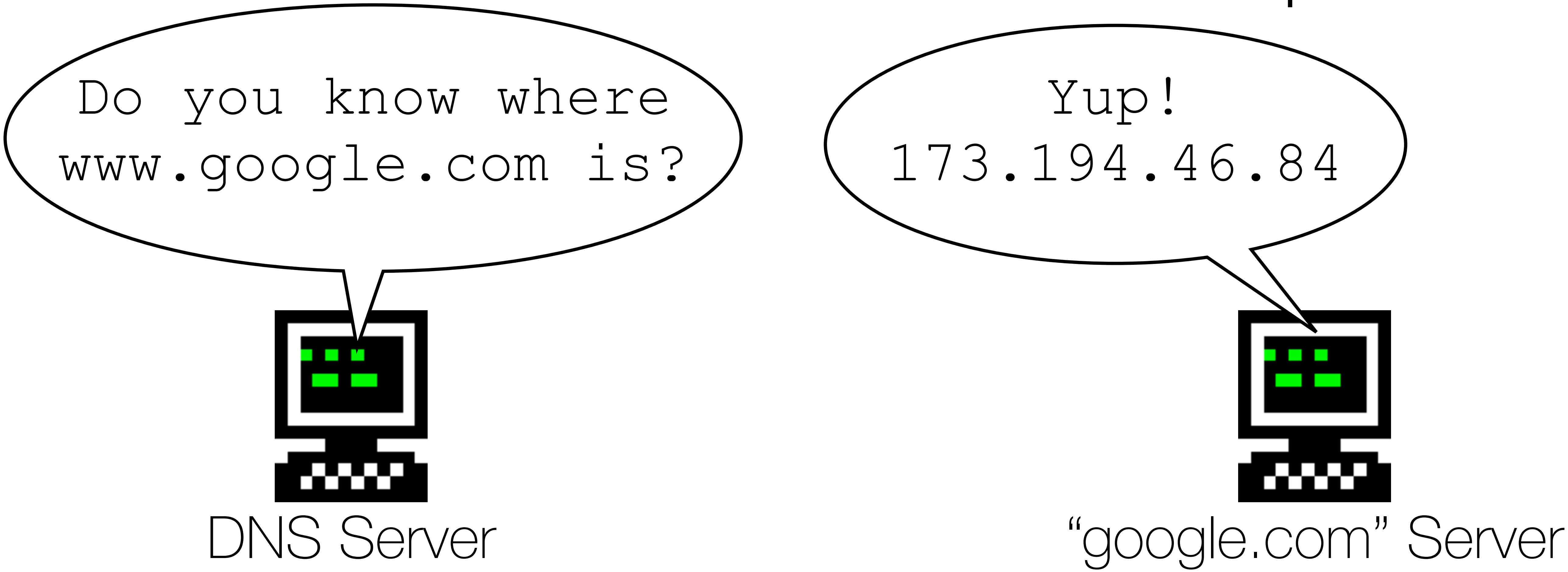
Nope, but “google.com” does.

DNS Server

“.com” Server

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

# 1. IP Address Lookup



Do you know where www.google.com is?

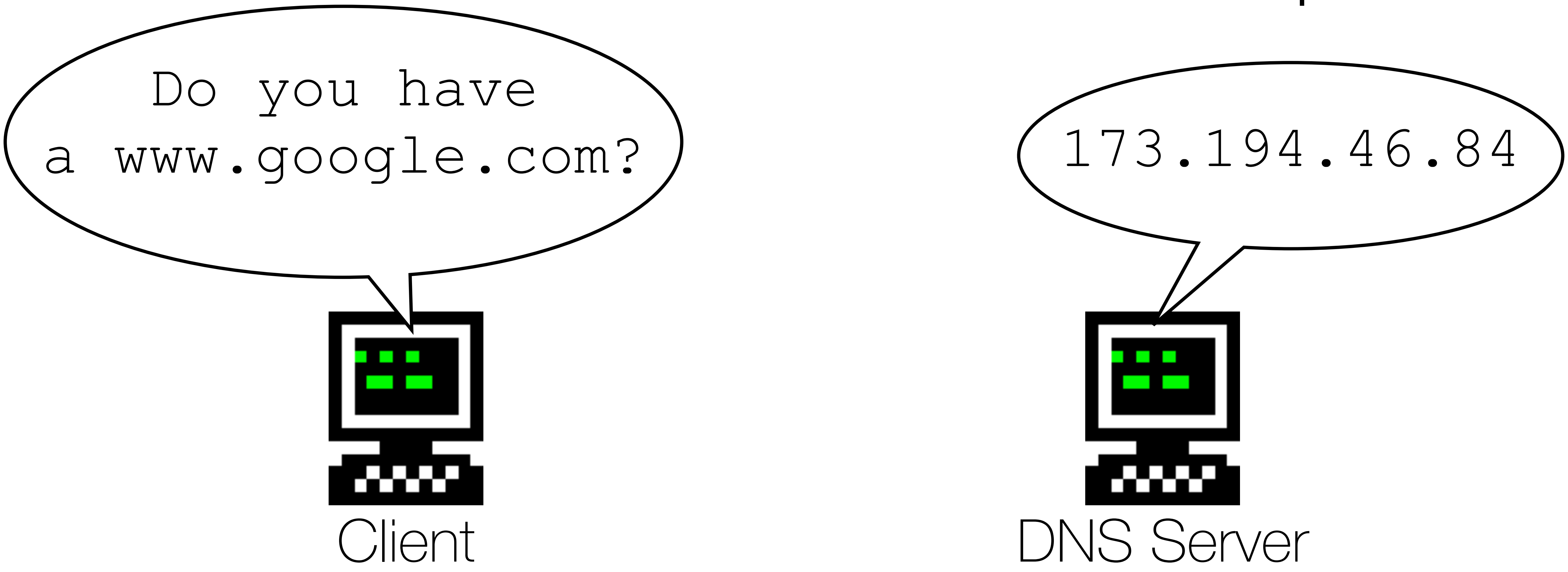
Yup! 173.194.46.84

DNS Server

“google.com” 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?

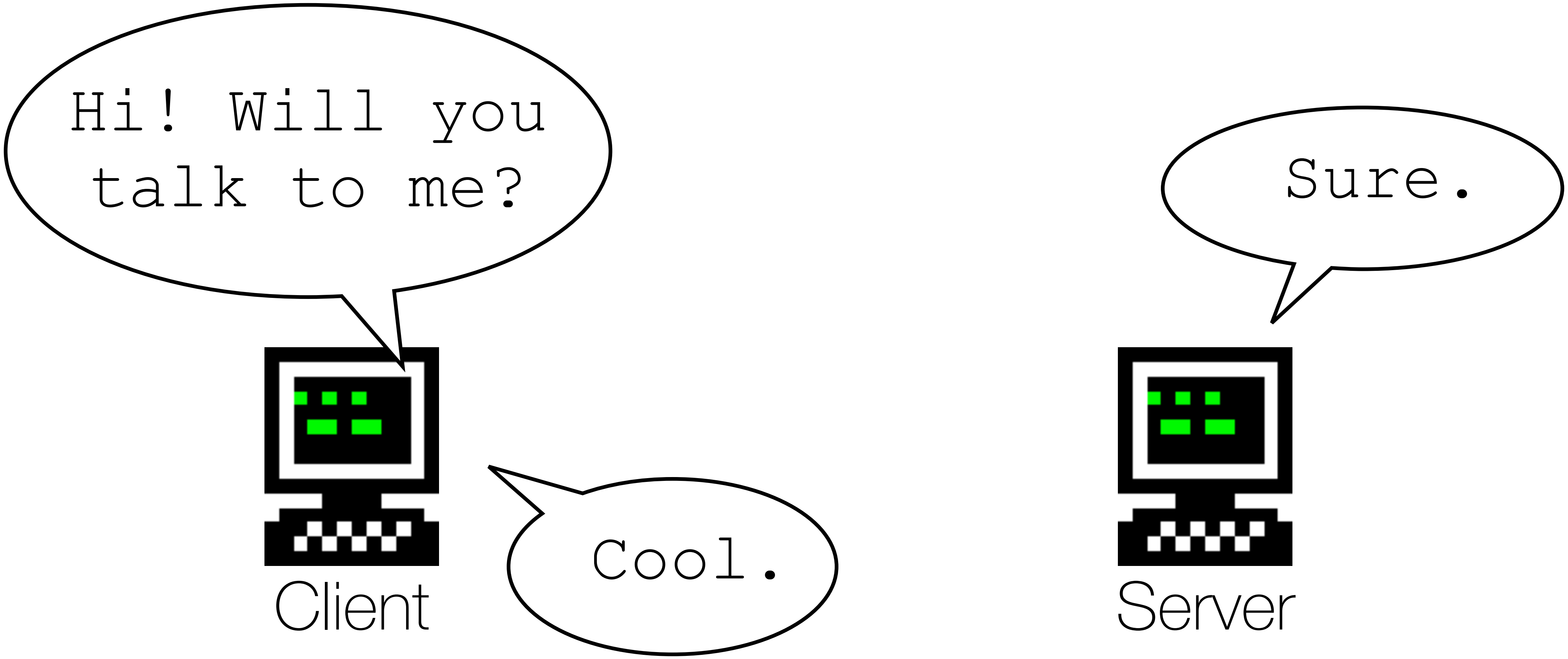
173.194.46.84

Client

DNS Server

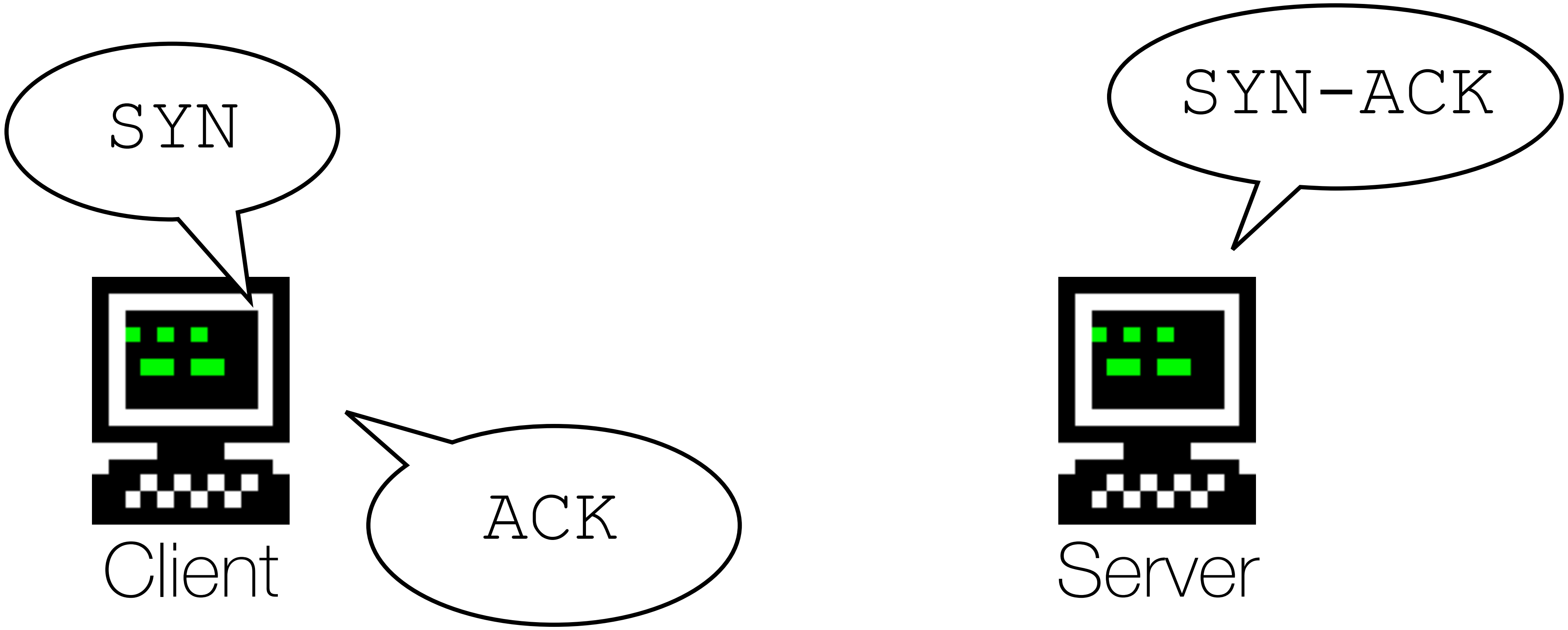
Browser hits DNS server which returns answer
(…after it asks a bunch more servers)

# 2. TCP Handshake



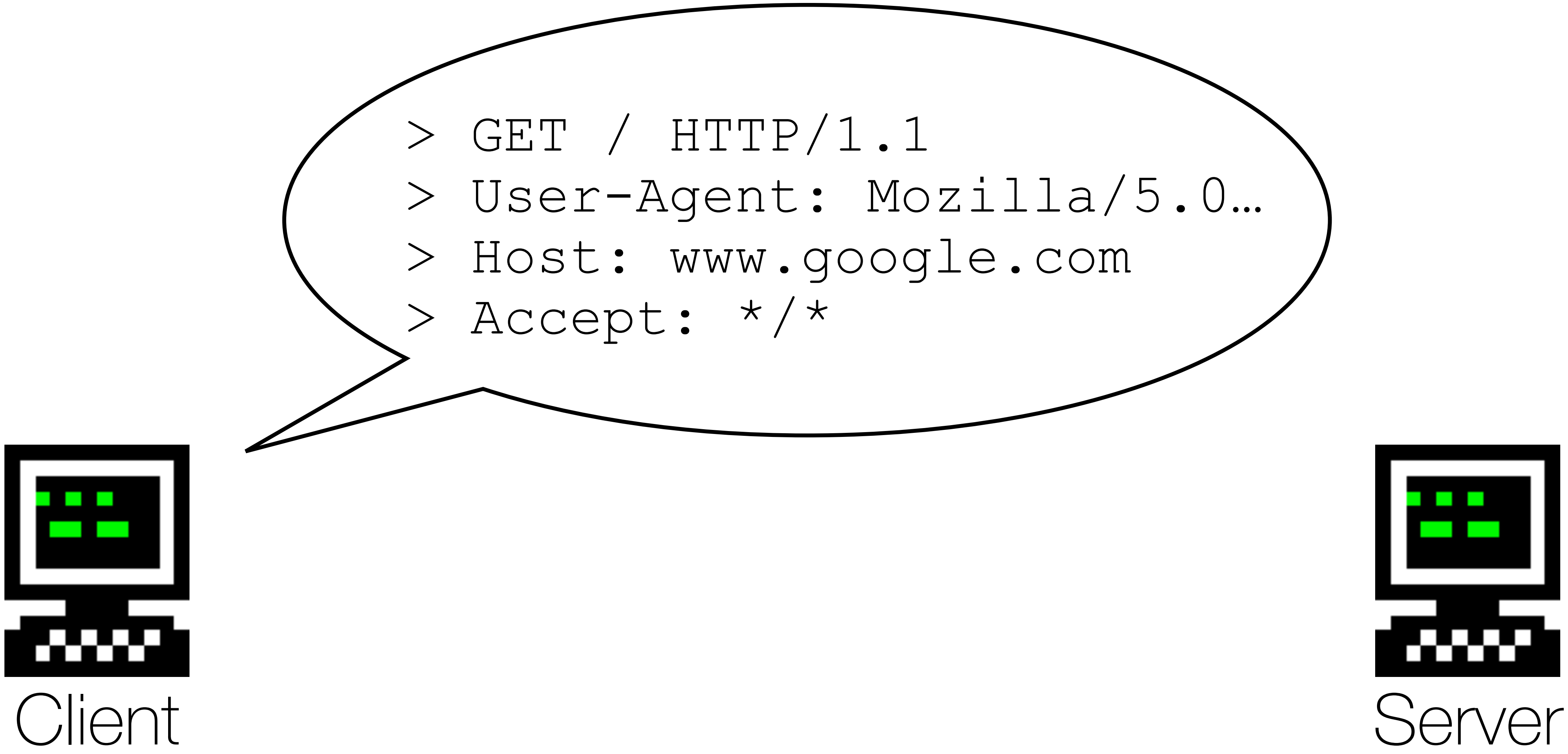
Client establishes TCP connection(s) with the server

# 2. TCP Handshake



Client establishes TCP connection(s) with the server

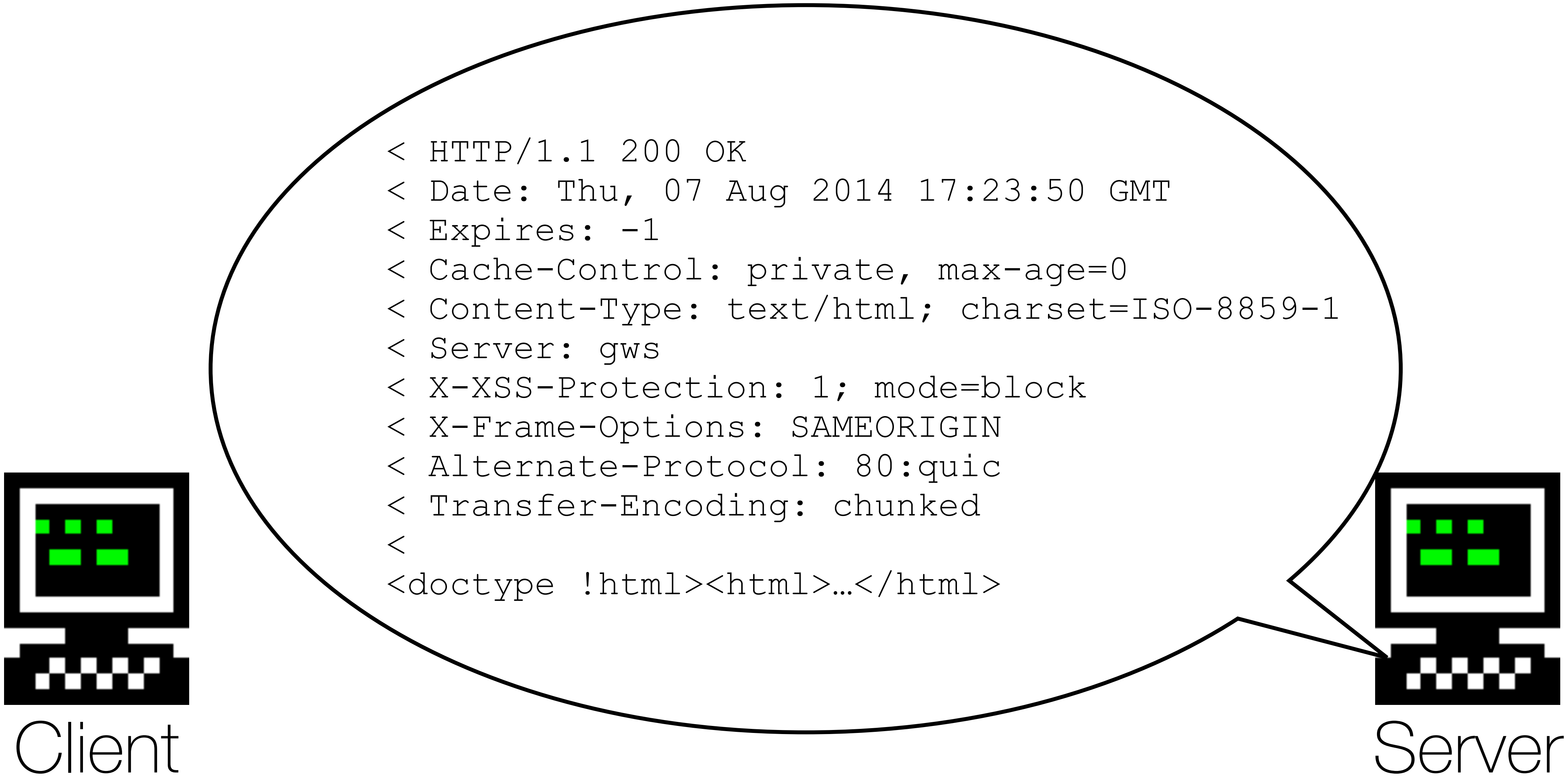
# 3. HTTP request



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

Client sends HTTP request to server

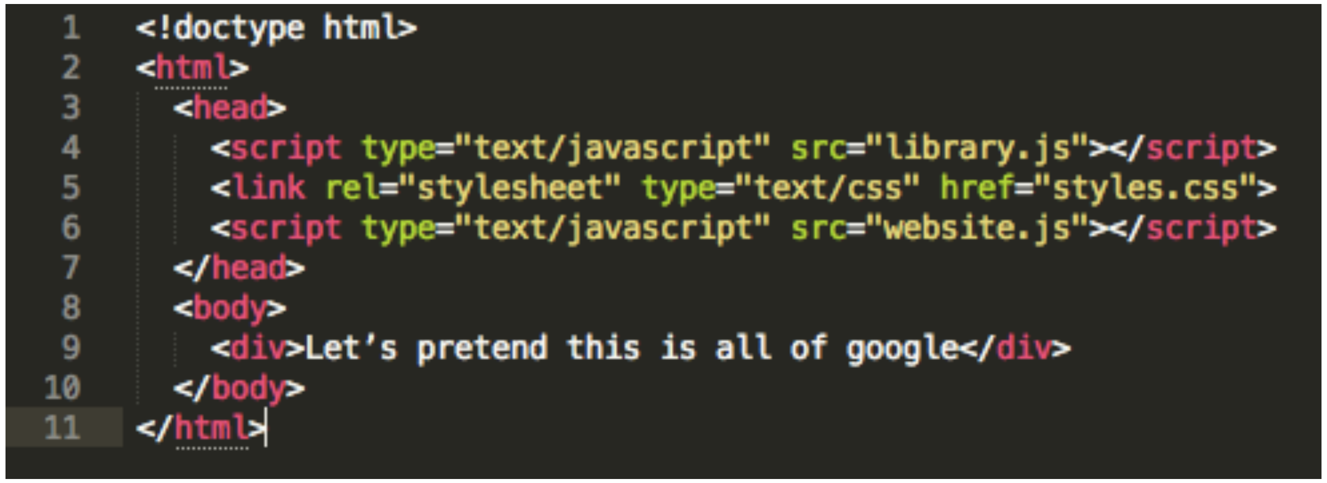
# 4. HTTP response



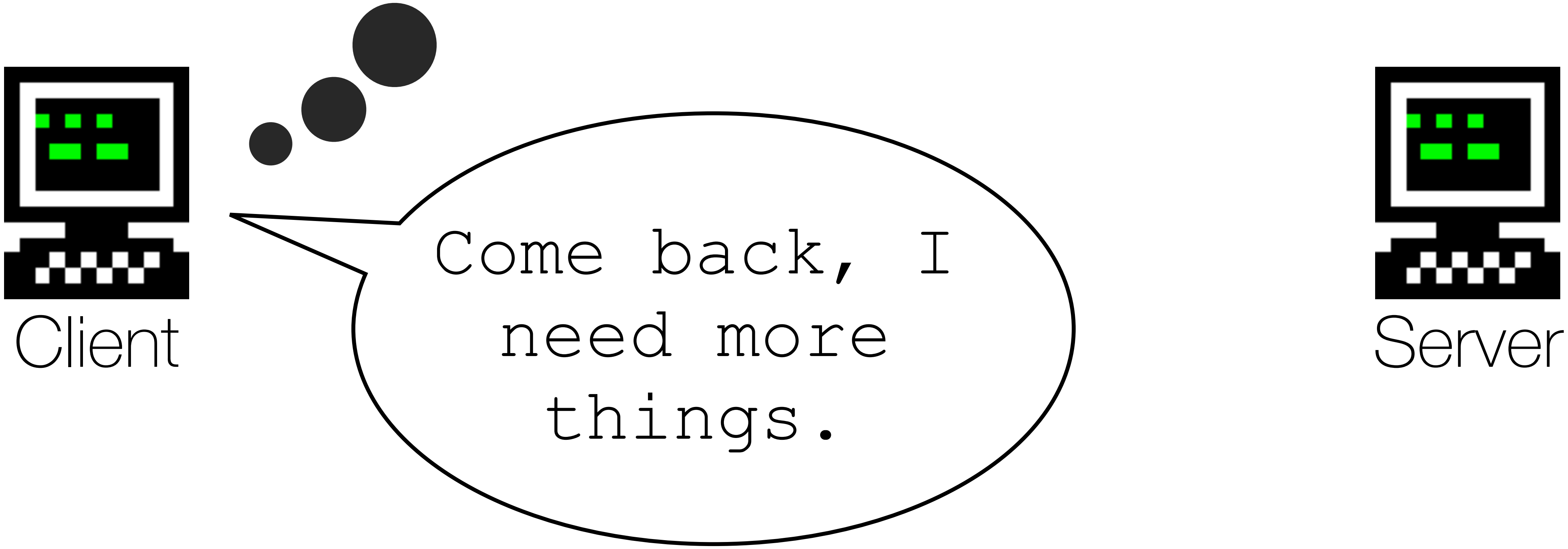
```
< HTTP/1.1 200 OK
< Date: Thu, 07 Aug 2014 17:23:50 GMT
< Expires: -1
< Cache-Control: private, max-age=0
< Content-Type: text/html; charset=ISO-8859-1
< Server: gws
< X-XSS-Protection: 1; mode=block
< X-Frame-Options: SAMEORIGIN
< Alternate-Protocol: 80:quic
< Transfer-Encoding: chunked
<
<doctype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

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

# 5. HTML Parsing

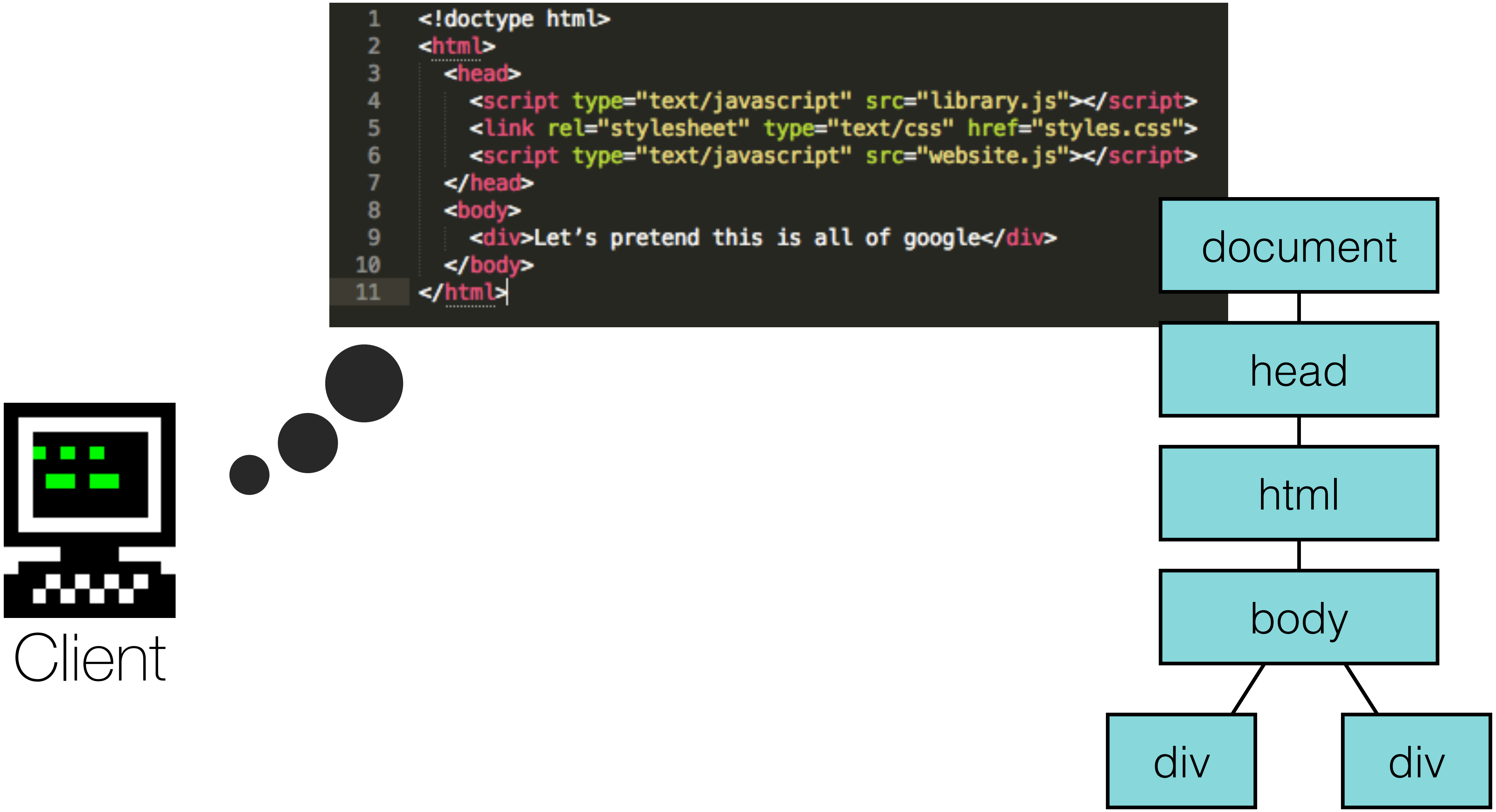


```
<!doctype html>
<html>
  <head>
    <script type="text/javascript" src="library.js"></script>
    <link rel="stylesheet" type="text/css" href="styles.css">
    <script type="text/javascript" src="website.js"></script>
  </head>
  <body>
    <div>Let’s pretend this is all of google</div>
  </body>
</html>
```



1. Speculative parser looks ahead for assets it can fetch

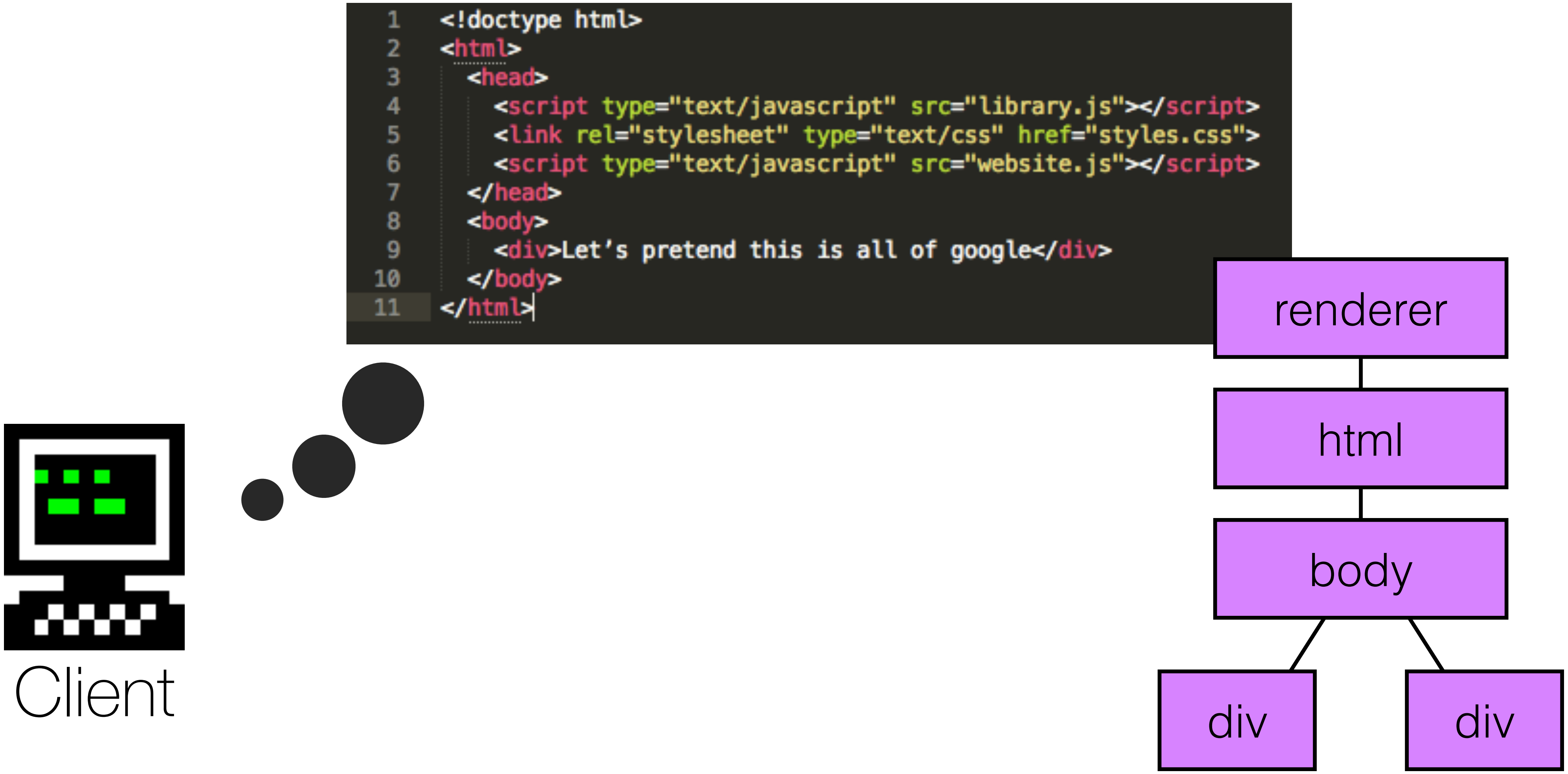
# 5. HTML Parsing



```
<!doctype html>
<html>
  <head>
    <script type="text/javascript" src="library.js"></script>
    <link rel="stylesheet" type="text/css" href="styles.css">
    <script type="text/javascript" src="website.js"></script>
  </head>
  <body>
    <div>Let’s pretend this is all of google</div>
  </body>
</html>
```

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

# 5. HTML Parsing



```
<!doctype html>
<html>
  <head>
    <script type="text/javascript" src="library.js"></script>
    <link rel="stylesheet" type="text/css" href="styles.css">
    <script type="text/javascript" src="website.js"></script>
  </head>
  <body>
    <div>Let's pretend this is all of google</div>
  </body>
</html>
```

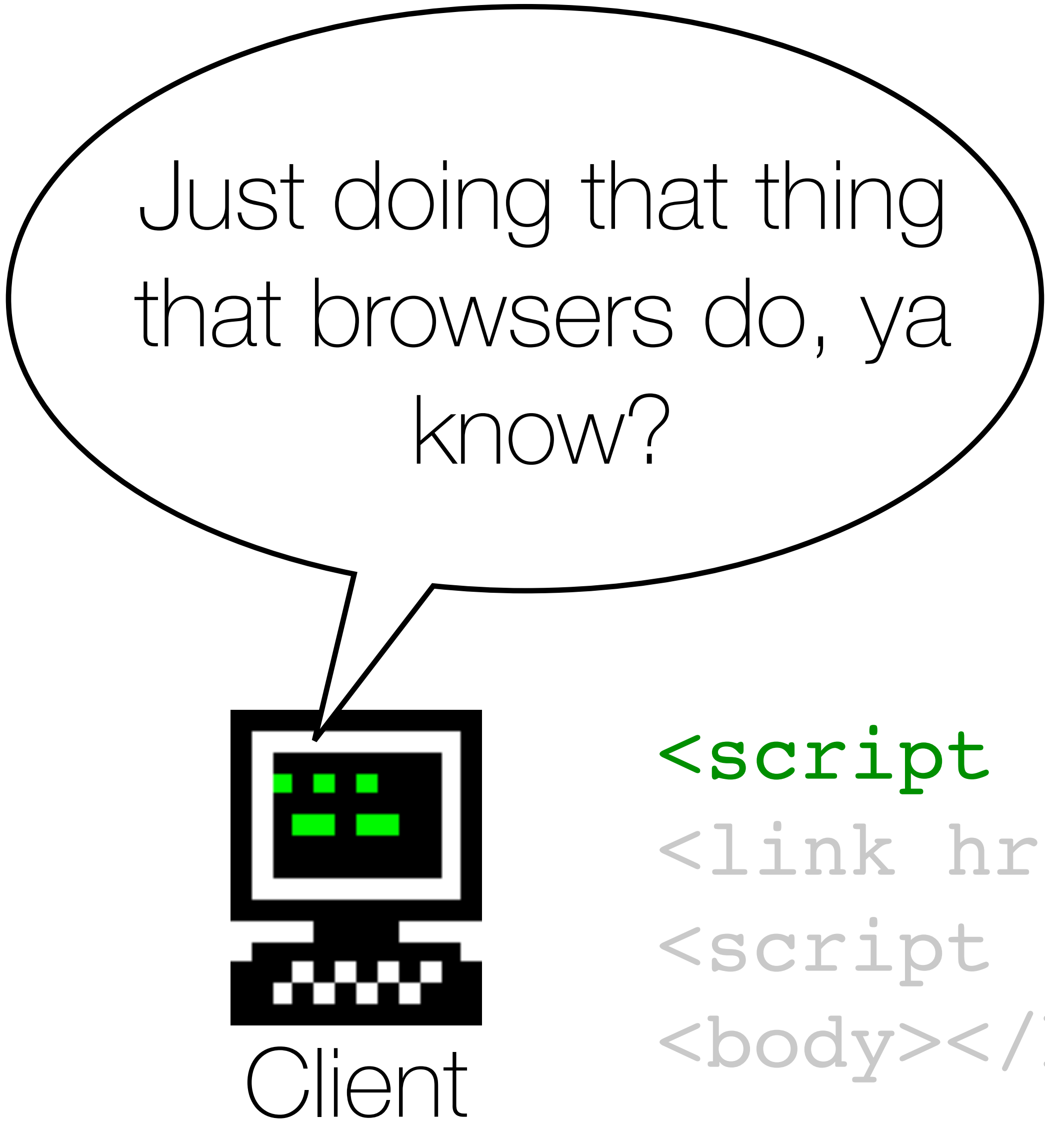
3. Render tree is also made based on the DOM, containing only things that get rendered (so, not `<head>` or `display: none`)

# 5. HTML Parsing

```
<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



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

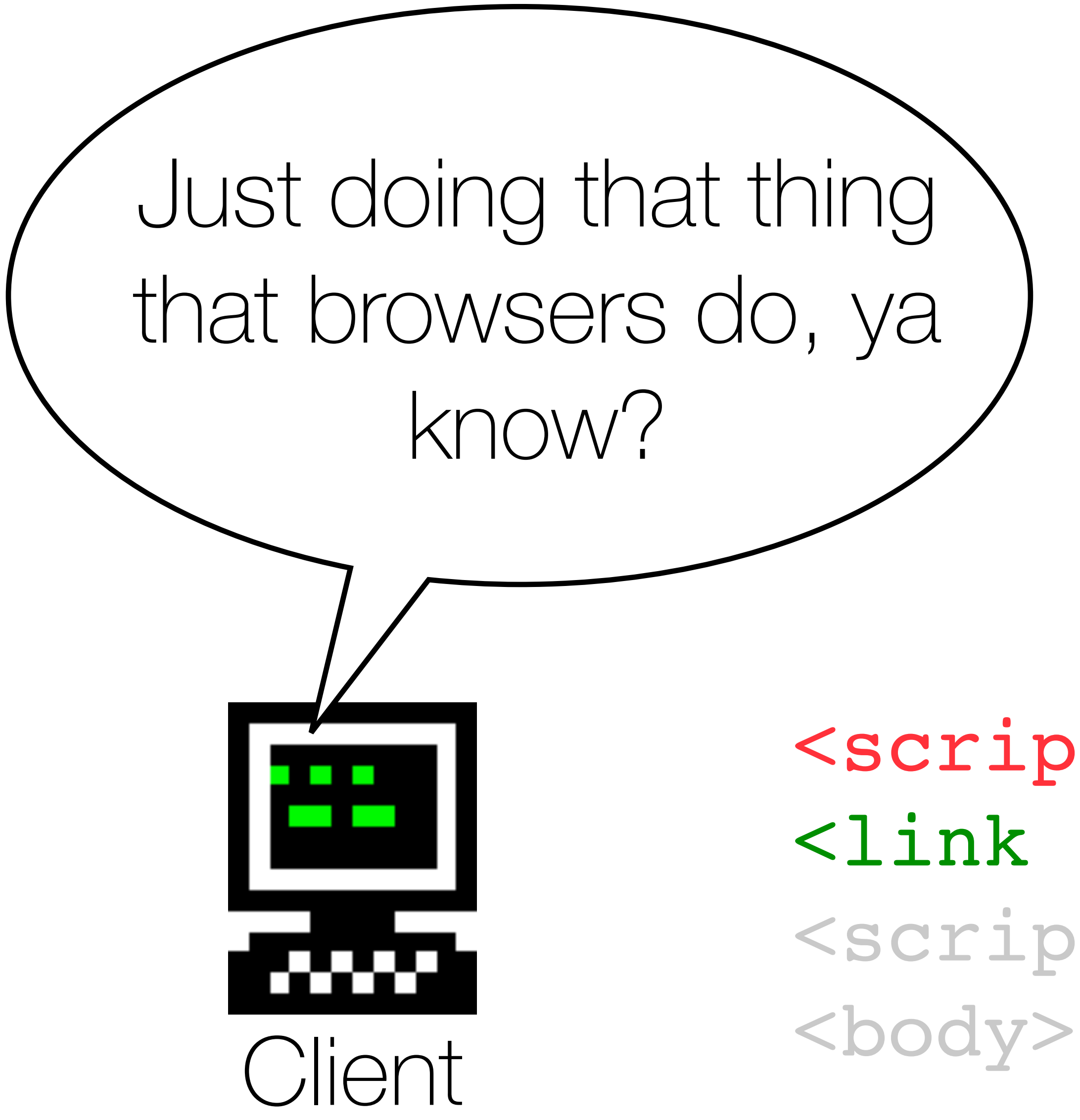
Client requests, parses, and executes all inlined assets (images, scripts, stylesheets).

# 5. HTML Parsing

```
<script defer 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).

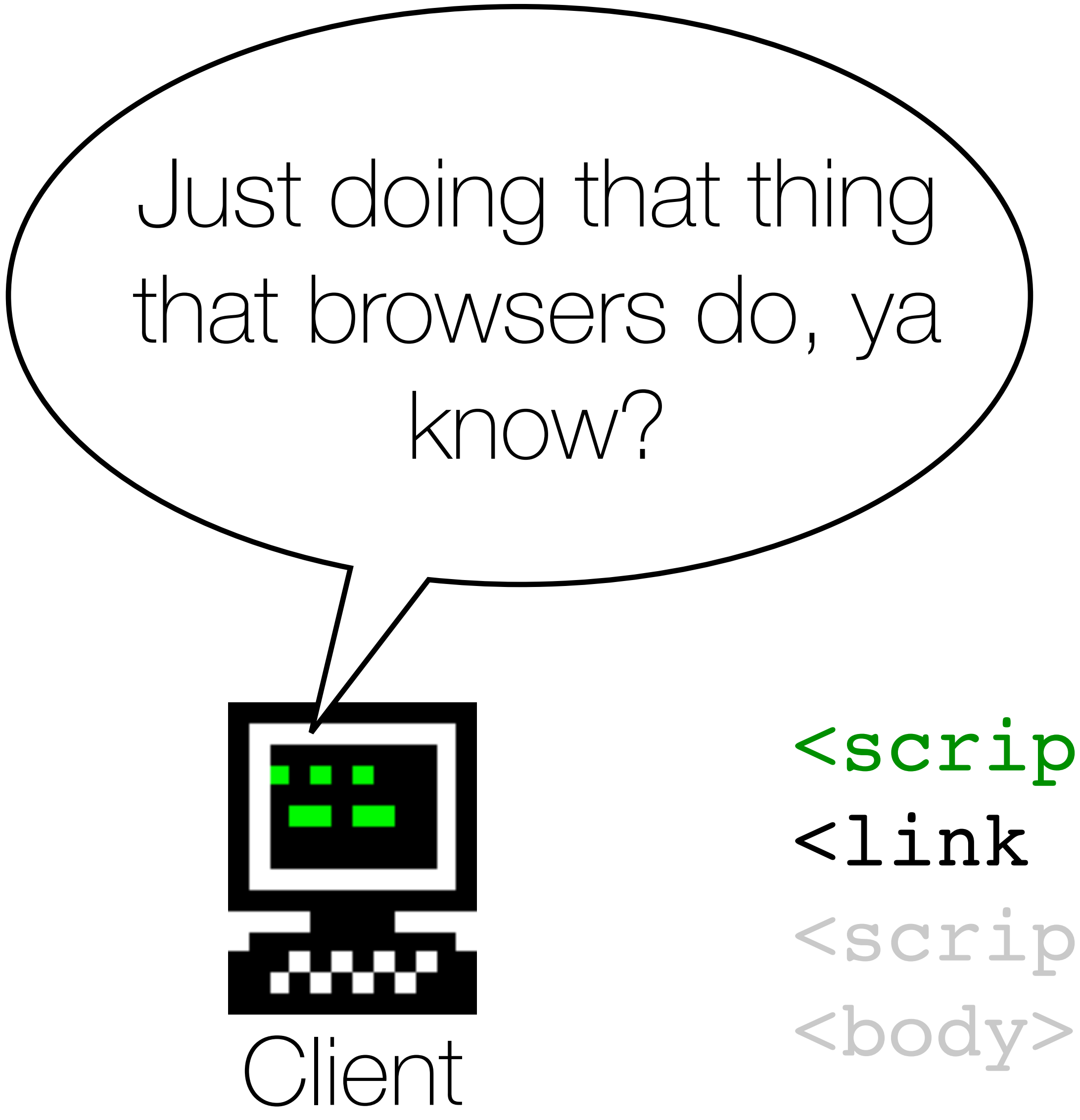
# 5. HTML Parsing



```
<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



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

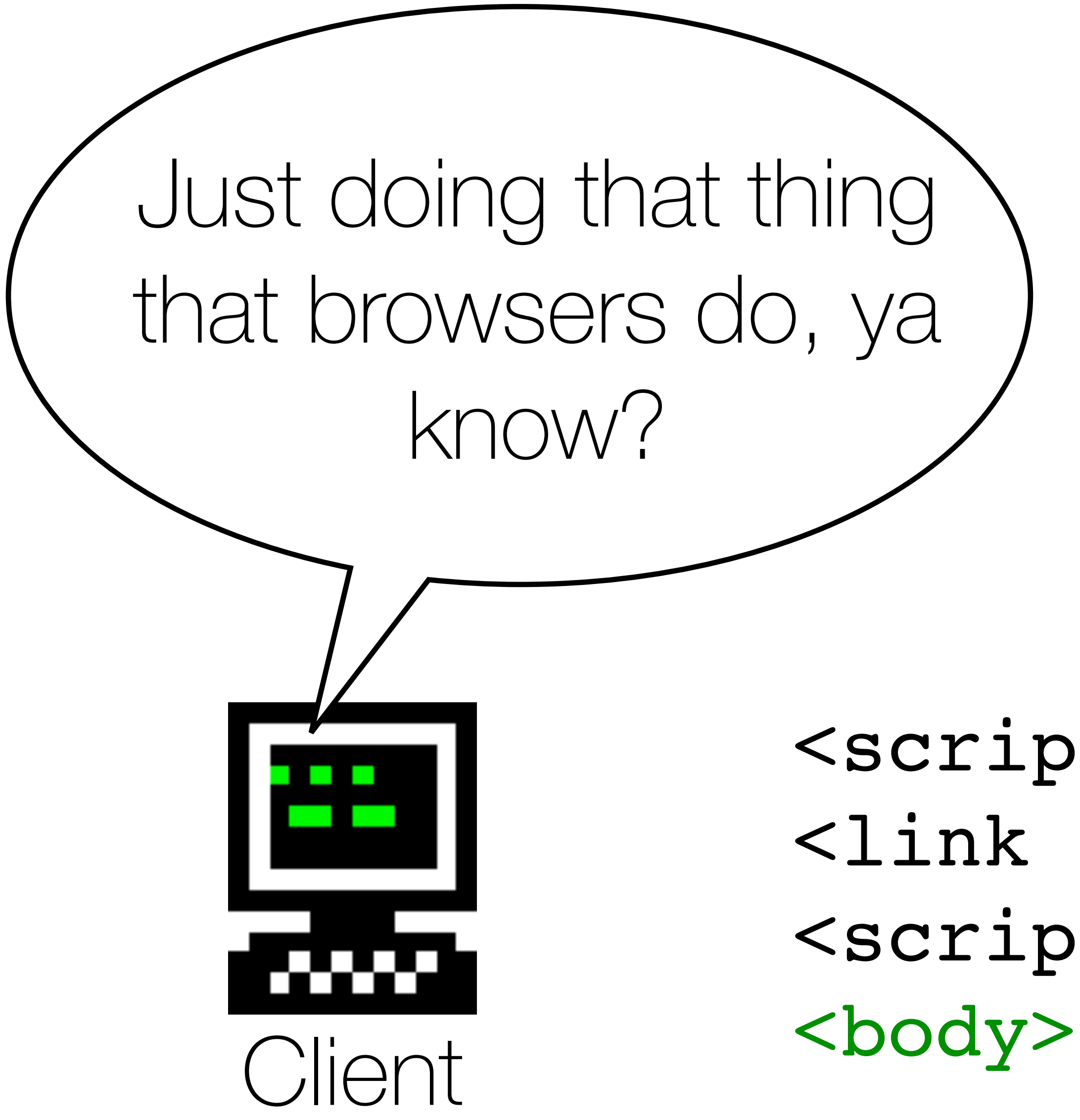
Client requests, parses, and executes all inlined assets (images, scripts, stylesheets).

# 5. HTML Parsing

```
<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).

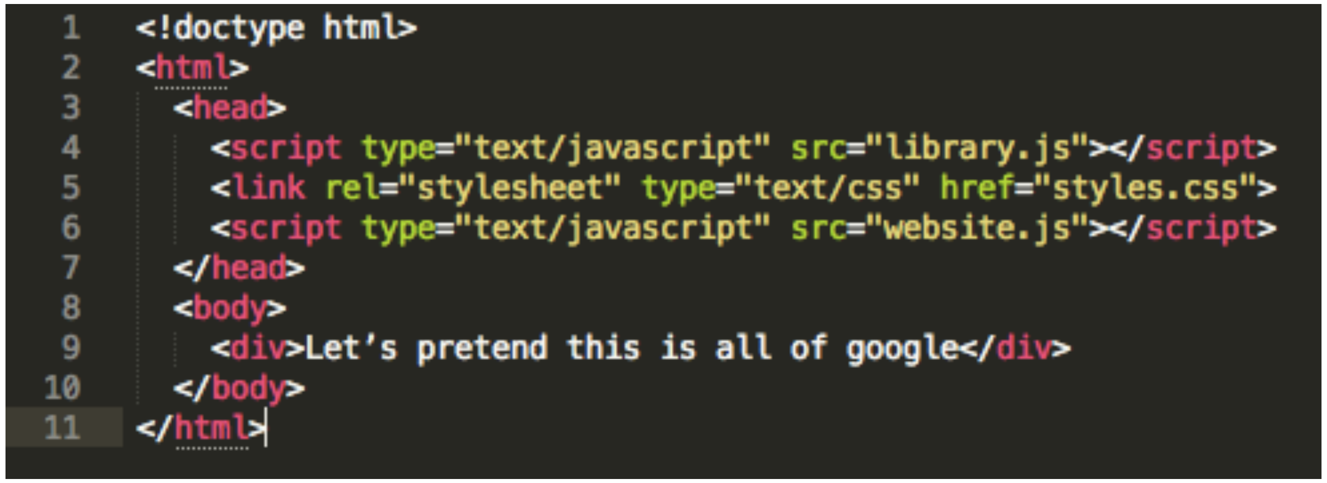
# 5. HTML Parsing



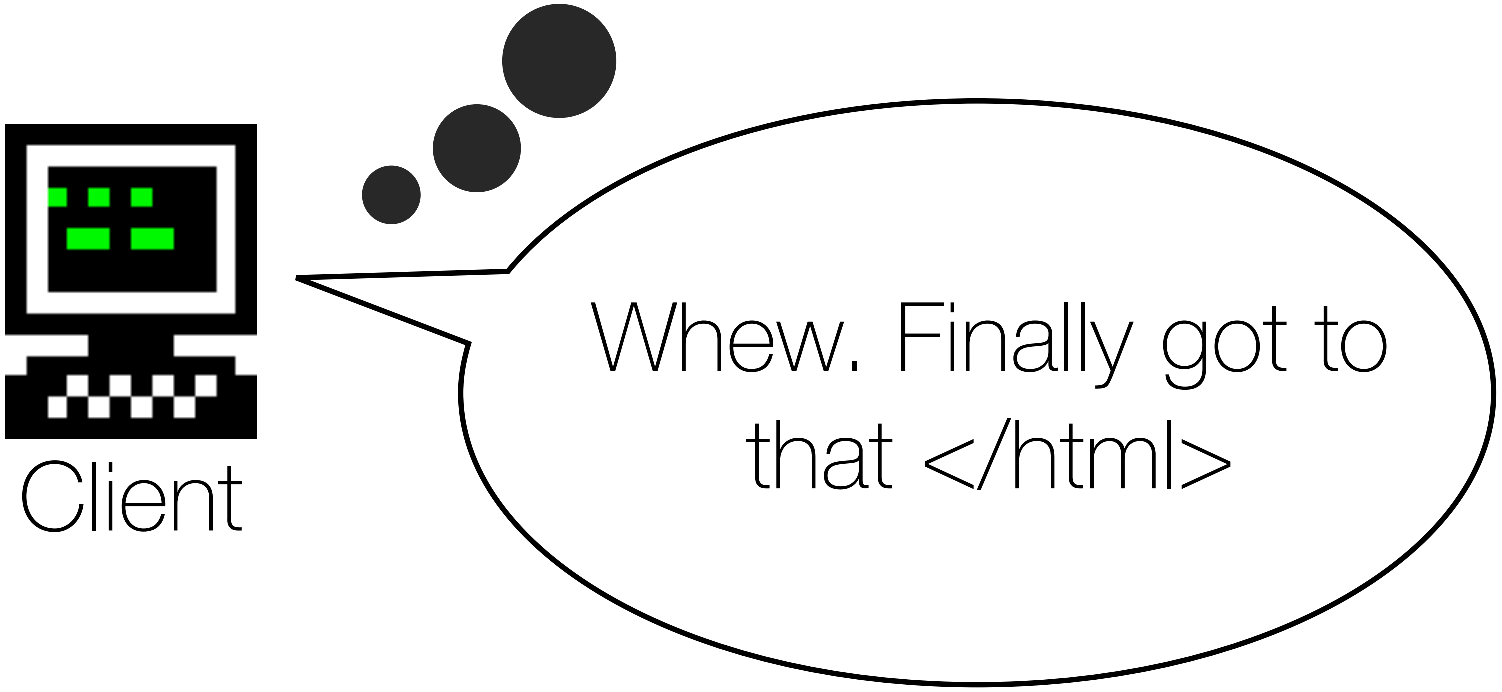
```
<script src="library.js"></script> ✓
<link href="styles.css"> ✓
<script src="website.js"></script> ✓
<body></body>
```

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

# 5. HTML Parsing



```
<!doctype html>
<html>
  <head>
    <script type="text/javascript" src="library.js"></script>
    <link rel="stylesheet" type="text/css" href="styles.css">
    <script type="text/javascript" src="website.js"></script>
  </head>
  <body>
    <div>Let’s pretend this is all of google</div>
  </body>
</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!!!!!