intro to jQuery

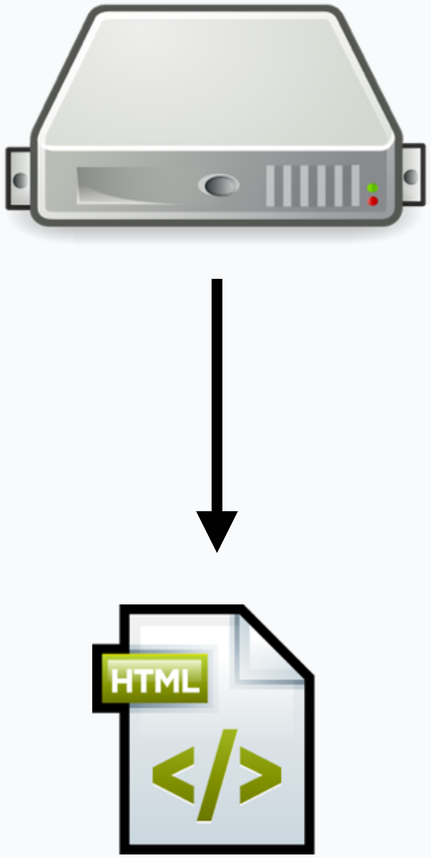
makin' web pages like it's 2006

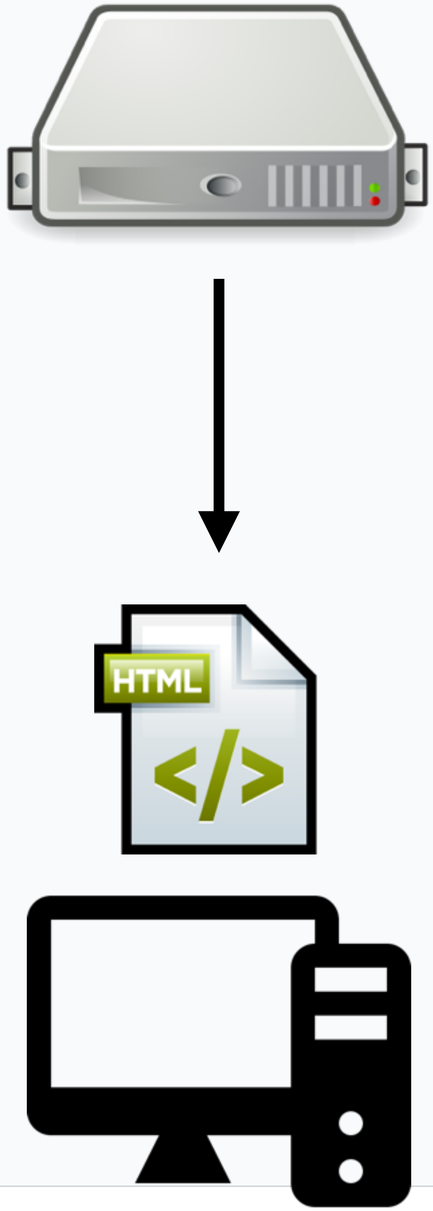
glossary of terms

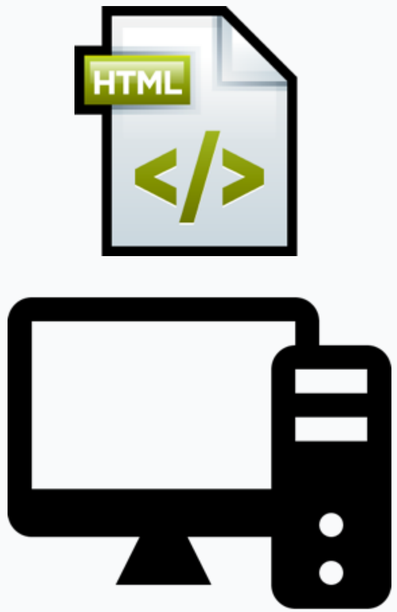
- DOM: Document Object Model
  - Browser's representation of HTML
  - All elements in the tree

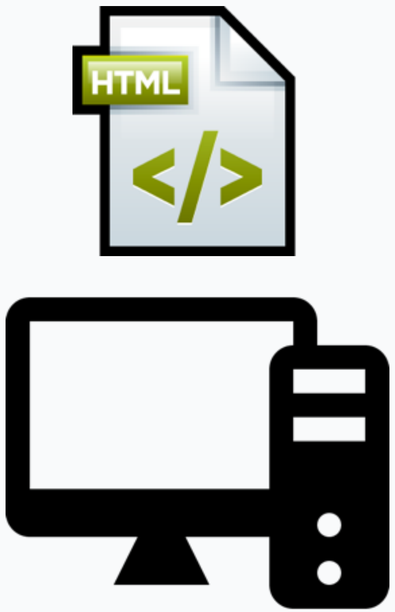
- Client-side
  - As opposed to server-side
  - Javascript executed in the browser

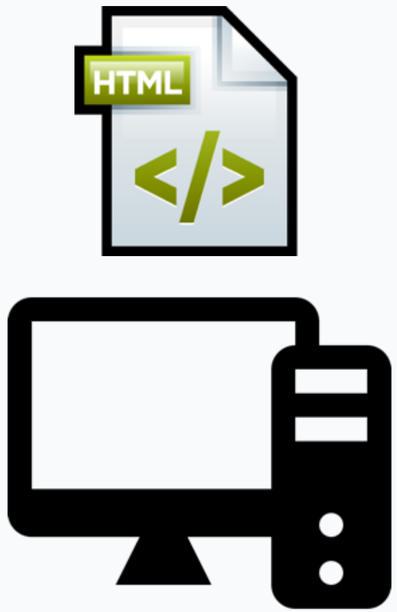
```
app.get('/', function (req, res) {
    res.sendFile(__dirname + '/index.html');
});
```

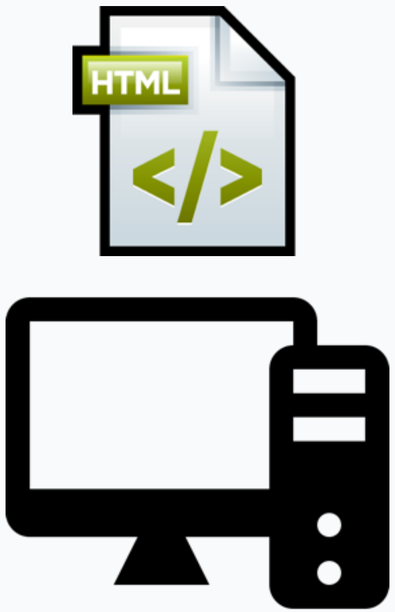

HTML


HTML

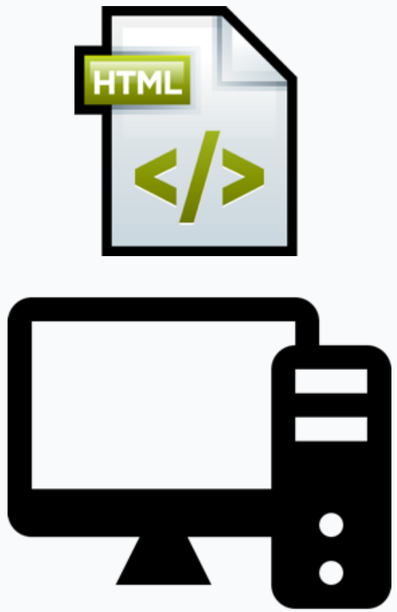

HTML

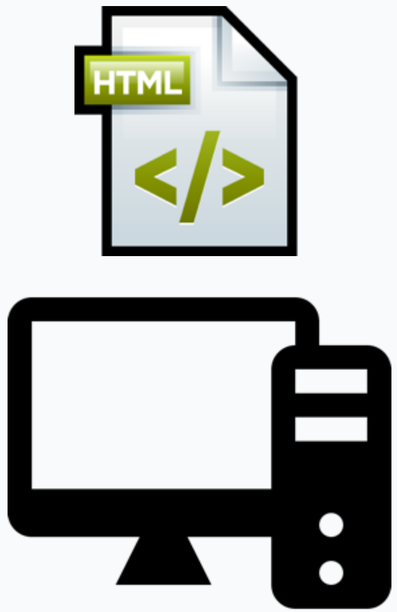

HTML


HTML

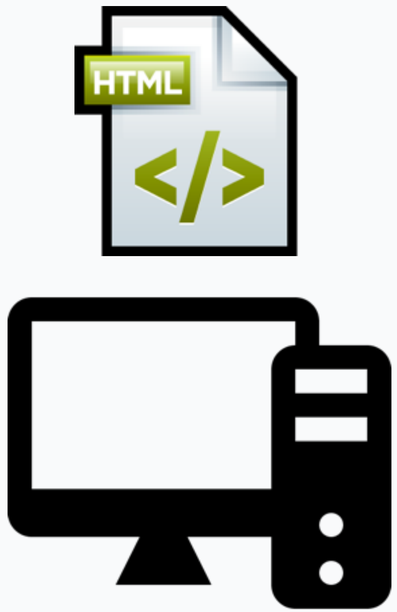

HTML

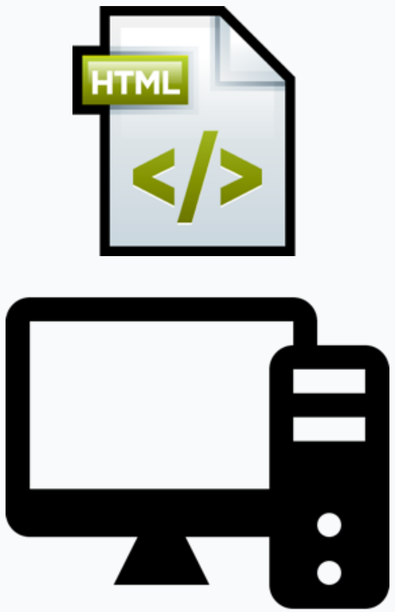
```
<link rel="stylesheet" href="style.css" />
```



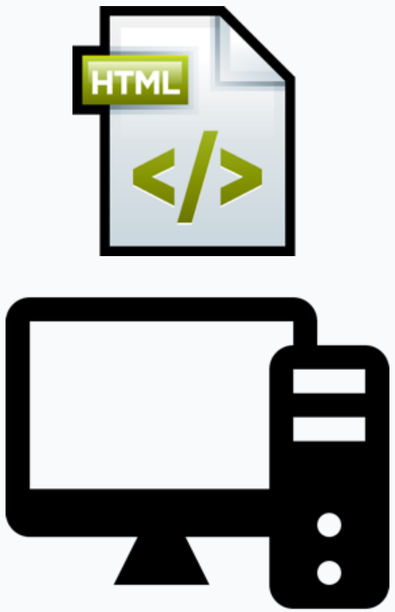


GET /style.css


HTML

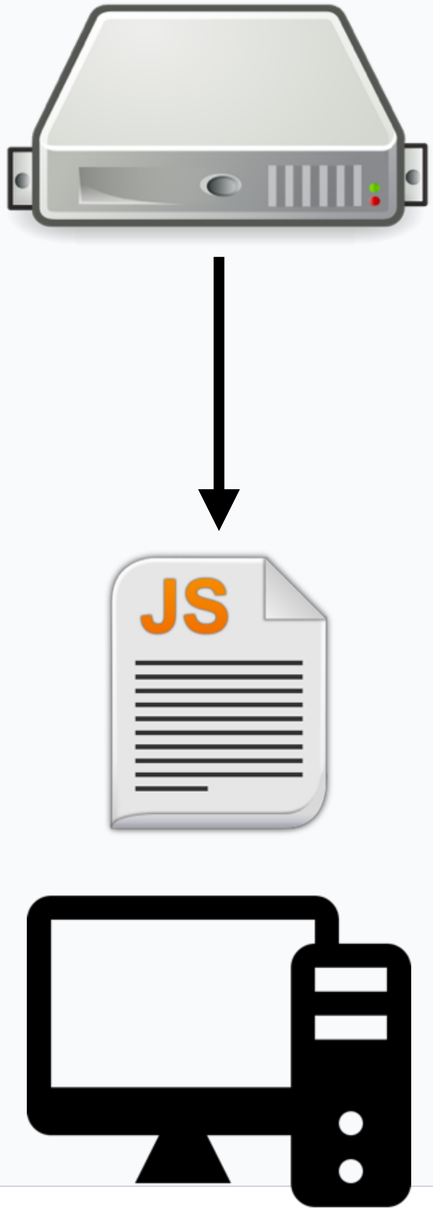


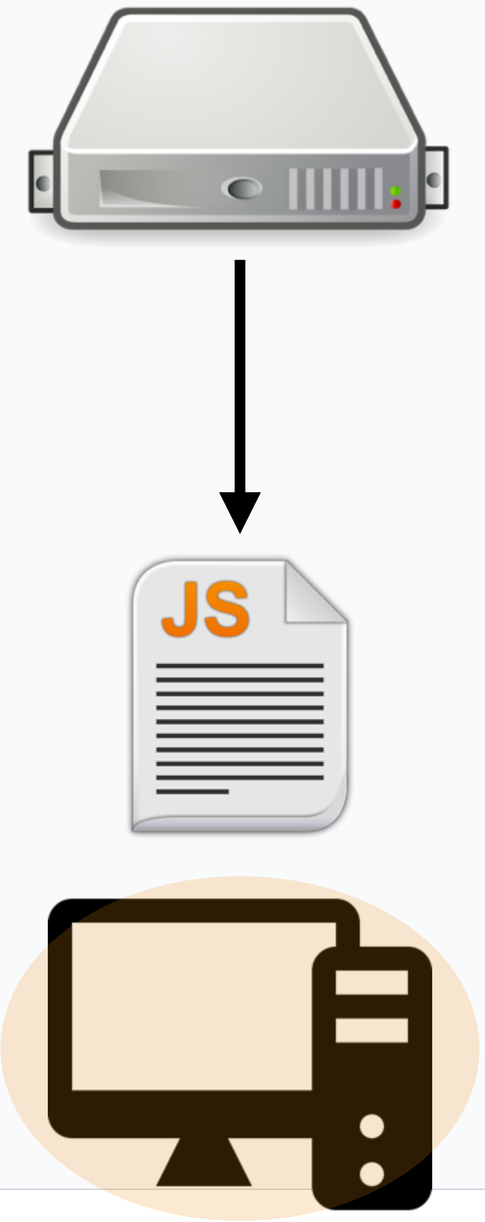

```
<script src="scripts/app.js"></script>
```



GET /scripts/app.js

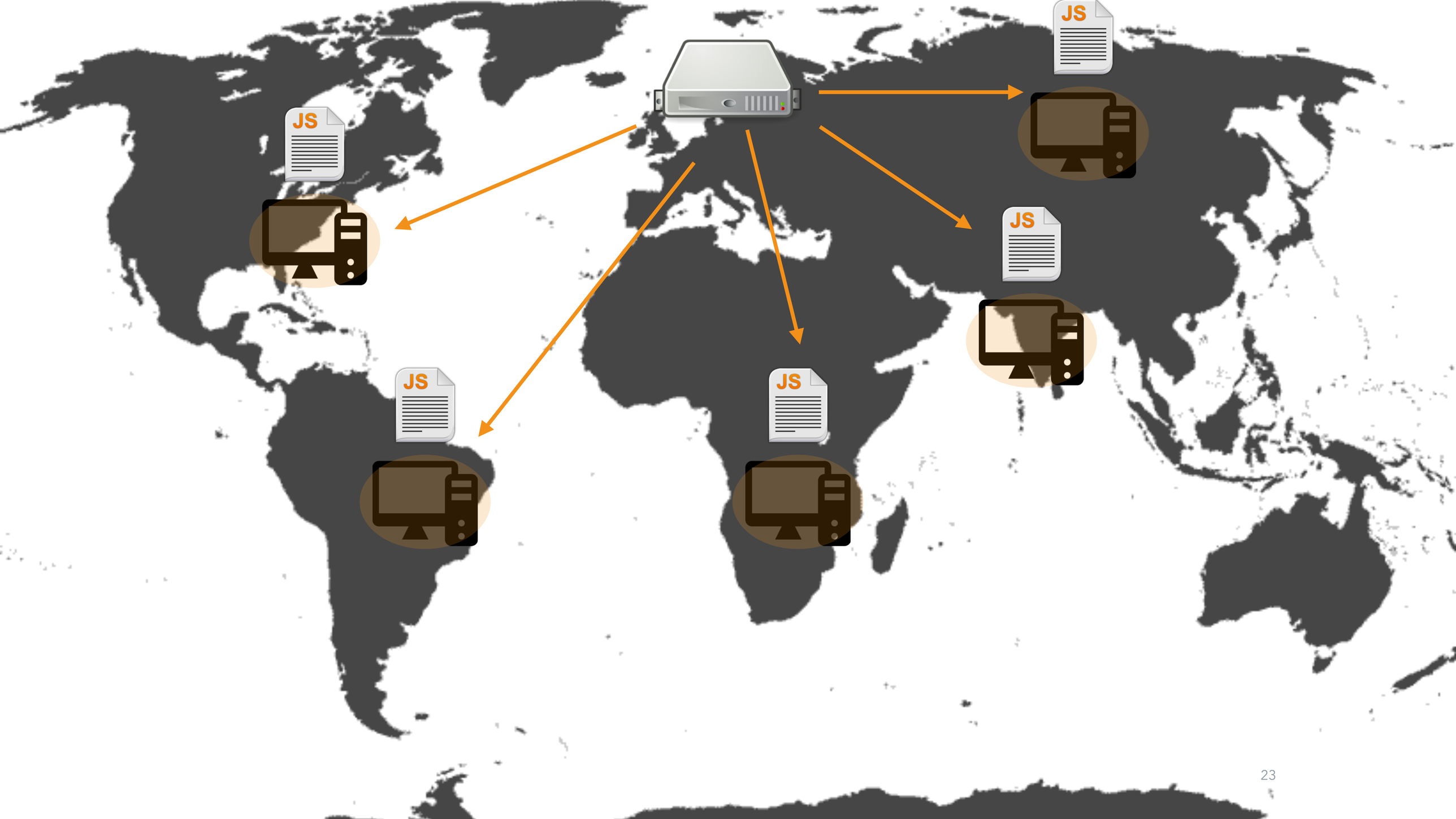
```
app.use(express.static(__dirname + '/public'));
```


JS


JS

front-end javascript is a file served to a browser

it does not share any code or functionality with the server itself


JS
JS
JS
JS
JS

cross-browser concerns

does what the competitors do, just slightly different.


TM
JS


JS


JS

using javascript to get the style of an element

- **97.38%** of browsers: getComputedStyle()
- IE 6: *Element*.currentStyle
- **jQuery: for when you need to support IE 6**


JS

browser compatibility, 2006

- CSS: different browsers support different rules that do the same thing
- document.querySelectorAll: has not yet been born
- Events: There are five different ways to figure out where the mouse is
- WebGL: Maybe you'd like to use a Java applet?
- Microsoft is trying to make ActiveX in the browser happen

jquery is a javascript milestone

- Released in 2006.
- Smoothed out the differences between browsers
- Still used today.
- So popular that some people know jQuery without any Javascript fundamentals.


jQuery
super useful in 2006

```
$('h1').addClass('my-class');
```

```
$('h1')

$('#my-heading')

$('.control-buttons')

$('h1 span')
```

```
$('.btn').on('click', function () {
    // All that my heart desires.
});
```

```
$('.btn').on('click', function () {
    // All that my heart desires.
});
```

```
$('.btn').on('click', function () {
    // All that my heart desires.
});
```

```
<div class="author-profile">
    <h1 class="author-name">William Gibson</h1>
    <h2 class="genre">Science Fiction</h2>
    <ul class="titles">
        <li class="top-pick">Neuromancer</li>
        <li>Pattern Recognition</li>
        <li>Zero History</li>
    </ul>
</div>
```

```
$('.author-profile').children();
```

```
<div class="author-profile">
    <h1 class="author-name">William Gibson</h1>
    <h2 class="genre">Science Fiction</h2>
    <ul class="titles">
        <li class="top-pick">Neuromancer</li>
        <li>Pattern Recognition</li>
        <li>Zero History</li>
    </ul>
</div>
```

```
$('.author-profile').children();
```

```
<div class="author-profile">
    <h1 class="author-name">William Gibson</h1>
    <h2 class="genre">Science Fiction</h2>
    <ul class="titles">
        <li class="top-pick">Neuromancer</li>
        <li>Pattern Recognition</li>
        <li>Zero History</li>
    </ul>
</div>
```

```
$('.author-profile').children('.author-name');
```

```
<div class="author-profile">
    <h1 class="author-name">William Gibson</h1>
    <h2 class="genre">Science Fiction</h2>
    <ul class="titles">
        <li class="top-pick">Neuromancer</li>
        <li>Pattern Recognition</li>
        <li>Zero History</li>
    </ul>
</div>
```

```
$('.genre').parent();
```

```
<div class="author-profile">
      <h1 class="author-name">William Gibson</h1>
      <h2 class="genre">Science Fiction</h2>
      <ul class="titles">
            <li class="top-pick">Neuromancer</li>
            <li>Pattern Recognition</li>
            <li>Zero History</li>
      </ul>
</div>
```

```
$('.genre').parent();
```

```
<div class="author-profile">
      <h1 class="author-name">William Gibson</h1>
      <h2 class="genre">Science Fiction</h2>
      <ul class="titles">
            <li class="top-pick">Neuromancer</li>
            <li>Pattern Recognition</li>
            <li>Zero History</li>
      </ul>
</div>
```

```
$('.genre').siblings();
```

```
<div class="author-profile">
      <h1 class="author-name">William Gibson</h1>
      <h2 class="genre">Science Fiction</h2>
      <ul class="titles">
            <li class="top-pick">Neuromancer</li>
            <li>Pattern Recognition</li>
            <li>Zero History</li>
      </ul>
</div>
```

```
$('.genre').siblings();
```

```
<div class="author-profile">
      <h1 class="author-name">William Gibson</h1>
      <h2 class="genre">Science Fiction</h2>
      <ul class="titles">
          <li class="top-pick">Neuromancer</li>
          <li>Pattern Recognition</li>
          <li>Zero History</li>
      </ul>
</div>
```

```
$('.titles li');
```

```
<div class="author-profile">
    <h1 class="author-name">William Gibson</h1>
    <h2 class="genre">Science Fiction</h2>
    <ul class="titles">
        <li class="top-pick">Neuromancer</li>
        <li>Pattern Recognition</li>
        <li>Zero History</li>
    </ul>
</div>
```

```
$('.titles li').css({ color: 'red' });
```

```
<div class="author-profile">
    <h1 class="author-name">William Gibson</h1>
    <h2 class="genre">Science Fiction</h2>
    <ul class="titles">
        <li class="top-pick">Neuromancer</li>
        <li>Pattern Recognition</li>
        <li>Zero History</li>
    </ul>
</div>
```

implicit looping

```
$(input: String) -> jQuery.Selection
```

- $ takes a selector (or HTML code) and returns a **selection**
- A selection is a lot like an **array**.
  - ... which **broadcasts** anything you do to it to every element inside of it.
- This is jQuery's coolest feature by far.

```
var $titles = $('.titles li');

$titles.css({ width: 200 });
```

```
var $titles = $('.titles li');

$titles.text();
```

```
var $titles = $('.titles li');

$titles.children();
```

HOW STANDARDS PROLIFERATE:

(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)

SITUATION:
THERE ARE
14 COMPETING
STANDARDS.

14?! RIDICULOUS!
WE NEED TO DEVELOP
ONE UNIVERSAL STANDARD
THAT COVERS EVERYONE'S
USE CASES.

SOON:

SITUATION:
THERE ARE
15 COMPETING
STANDARDS.

things you cannot do

- jQuery Selections are **a lot like** arrays
- But of course they are not **actually** arrays, because we can't have nice things.

```
$('p').forE                                    ppercase()))
```

```
VM223:1 Uncaught TypeError: $(...).forEach
is not a function
```

things you cannot do

- jQuery Selections are **a lot like** arrays
- But of course they are not **actually** arrays, because we can't have nice things.

```
for (
VM154:1 Uncaught TypeError: $(...)
[Symbol.iterator] is not a function
```

explicit looping

- They do have .length and [0], [1], [2], etc.

```
const paras = $('p')
for (let i = 0; i != paras.length; ++i) {
  const p = paras[i]
  p.textContent = p.textContent.toUpperCase()
}
```

explicit looping

- They have map, **but the order of arguments to the iterator is different**

```
[].map(iterator: (element, index) -> any)
$(...).map(iterator: (index, element) -> any)

$('p').map((i, p) => p.textContent.toUpperCase())
```

explicit looping

- They have each rather than forEach, with the same weird (index, element) ordering

```
[].forEach(iterator: (element, index) -> any)
$(...).each(iterator: (index, element) -> any)

$(‘p’).each((i, p) =>
  p.textContent = p.textContent.toUpperCase())
```

your callbacks aren’t called with selections

- That too would be too nice.
- **You can’t do this:**

```
$(‘p’).each((i, p) => p.text(
```

```
VM1028:1 Uncaught TypeError: p.text is
not a function(…)
```

```
ase())
```

your callbacks aren’t called with selections

- **You must select each element if you want jQuery functions on it.**

```
$(‘p’).each((i, p) => $(p).text(p.text().toUpperCase())
```

$

$

"select"

select elements matching a selector

```
$('.titles li');
```

this is the most common use

(parse and) select html

```
$('<h1 id="new-heading">Hi everyone!</h1>');
```

select DOM nodes

```
$(this); // this being a DOM element.
```

```
$(document.getElementById('jukebox'));
```

```
$($('#jukebox'));
```

browser compatibility, 2016

- 97.05% of browsers support flexbox
- 97.39% support <canvas>
- 97.39% support the same UI events
- 88.58% support web GL (!!)
- 74.97% support web audio
- Microsoft trying to make .NET in the browser happen
- jQuery is a lot less useful

manipulating the dom sensibly

- **DOM Tree** – Stores the *state* of the app, including currently attached *listeners*
- **Listeners** – Describe how the state *changes* in response to *events*.
- **Events** – Triggered by user gestures on DOM nodes

recap

- Client-side Javascript library.
- Good for DOM selection, event handling, managing element properties.
- Great for ad-hoc scripts that slowly gain complexity until they fully emerge as lumbering horrors whose non-euclidian code geometries inexorably drain the sanity of any poor soul who wanders into their cursed realms
- jQuery still has a large slot in the industry
- So does PHP
  - **86% of the web!**
    - But if you really want longevity, learn COBOL