CS50 Quiz 1 Review

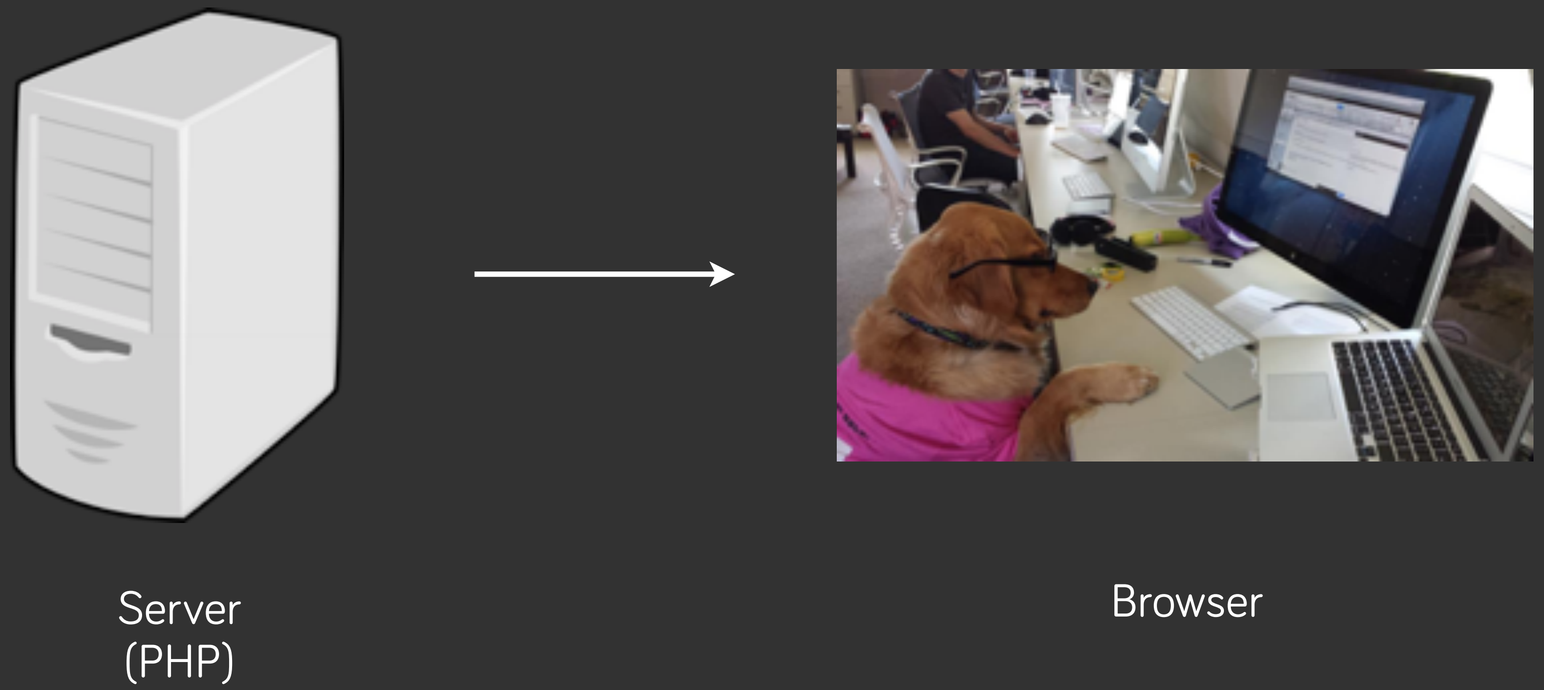
this is a cat

JavaScript

CS50 Quiz 1 Review

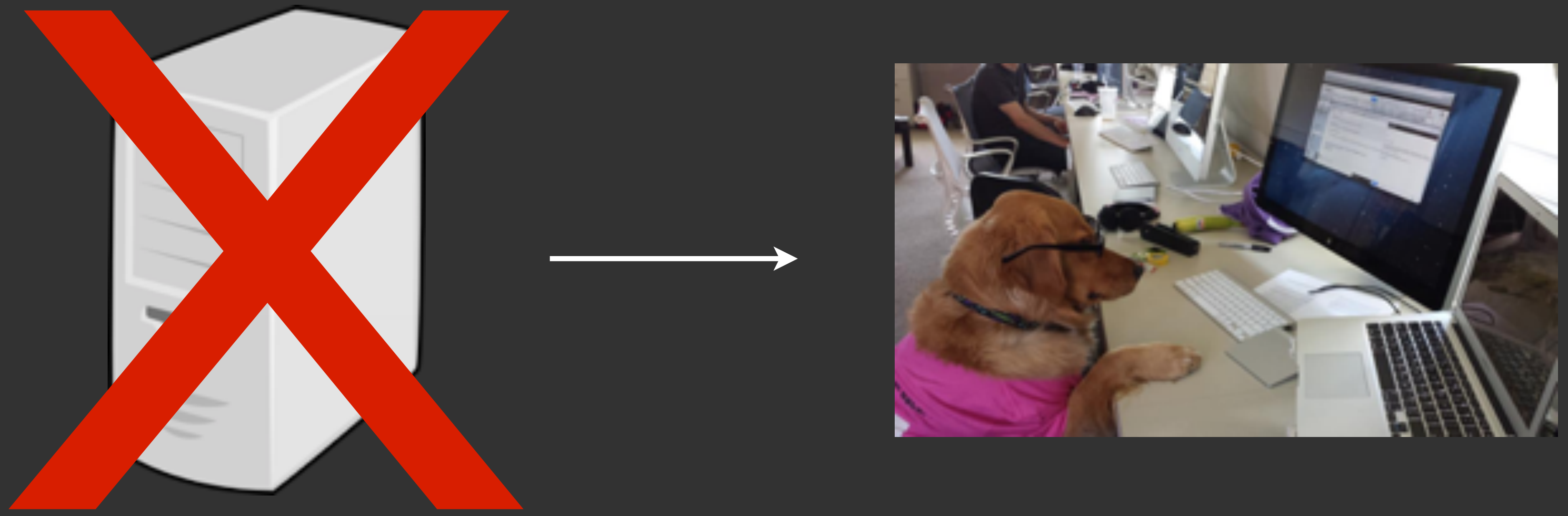
first, recall from zamyla

- Remember, PHP is run **server-side**. The HTML output of this PHP code is sent to the user.



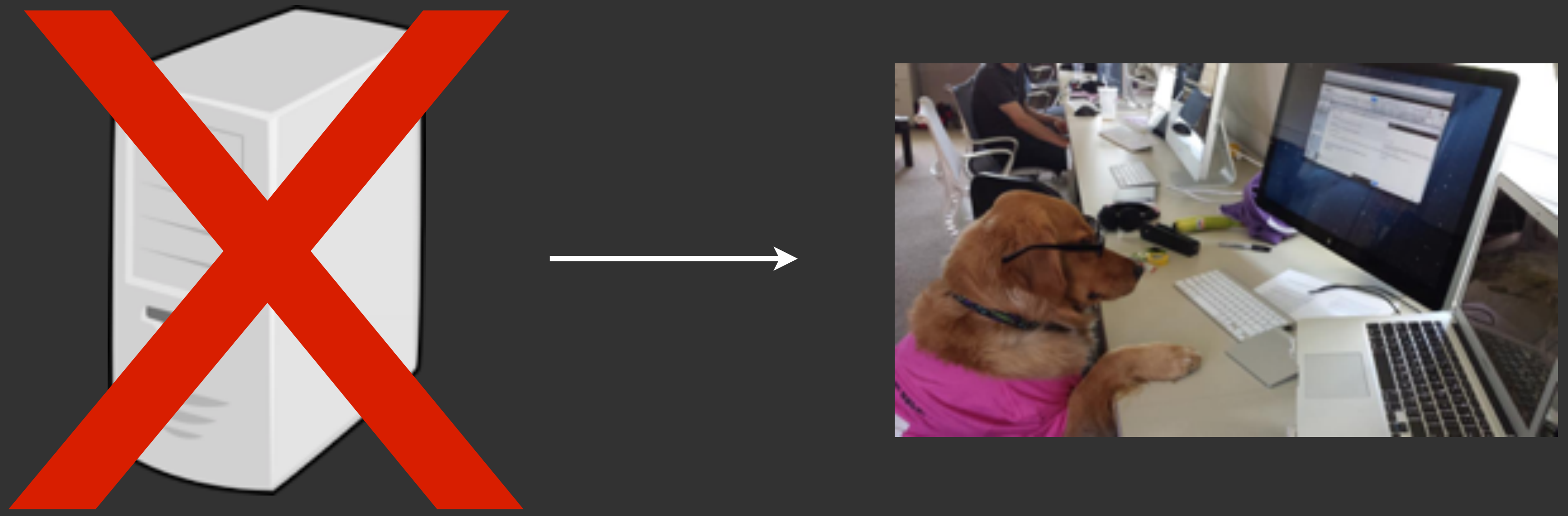
first, recall from zamyla

- Remember, PHP is run **server-side**. The HTML output of this PHP code is sent to the user.
- No more PHP code can be executed after this HTML output is sent, unless you reload!



why javascript?

- What if want something to happen when a user interacts with the page?
    - clicks on a button; presses a key; resizes the page?



- JavaScript fulfills this role. It's a **client-side** programming language,
    - used to control interactions and logic in a web browser.

JavaScript != PHP

some different **syntax**, very different **uses**

syntax differences

```
var i = 0;

var milo = {
    "name":  "Milo Banana",
    "house": "CS50 House",
    "role":  "Mascot"
};

milo.owner = "Lauren";
milo["owner"] = "Lauren";

console.log(milo);

console.log(milo.owner + " rocks!");
```

JavaScript

```
$i = 0;

$milo = [
    "name"  => "Milo Banana",
    "house" => "CS50 House",
    "role"  => "Mascot"
];

$milo["owner"] = "Lauren";


print_r($milo);

print($milo["owner"] . " rocks!")
```

PHP

control flow differences

```
var milo = {
    "name":  "Milo Banana",
    "house": "CS50 House",
    "role":  "Mascot"
};

for (var i in milo)
{
    console.log(i);
}
```

```
name
house
role
```

JavaScript, keys

```
$milo = [
    "name"  => "Milo Banana",
    "house" => "CS50 House",
    "role"  => "Mascot"
];

foreach ($milo as $i)
{
    print($i);
}
```

```
Milo Banana
CS50 House
Mascot
```

PHP, values

control flow differences

```
var milo = {
    "name":  "Milo Banana",
    "house": "CS50 House",
    "role":  "Mascot"
};

for (var i in milo)
{
    console.log(milo[i]);
}
```

```
Milo Banana
CS50 House
Mascot
```

JavaScript, values

```
$milo = [
    "name"  => "Milo Banana",
    "house" => "CS50 House",
    "role"  => "Mascot"
];

foreach ($milo as $key => $value)
{
    print($key);
}
```

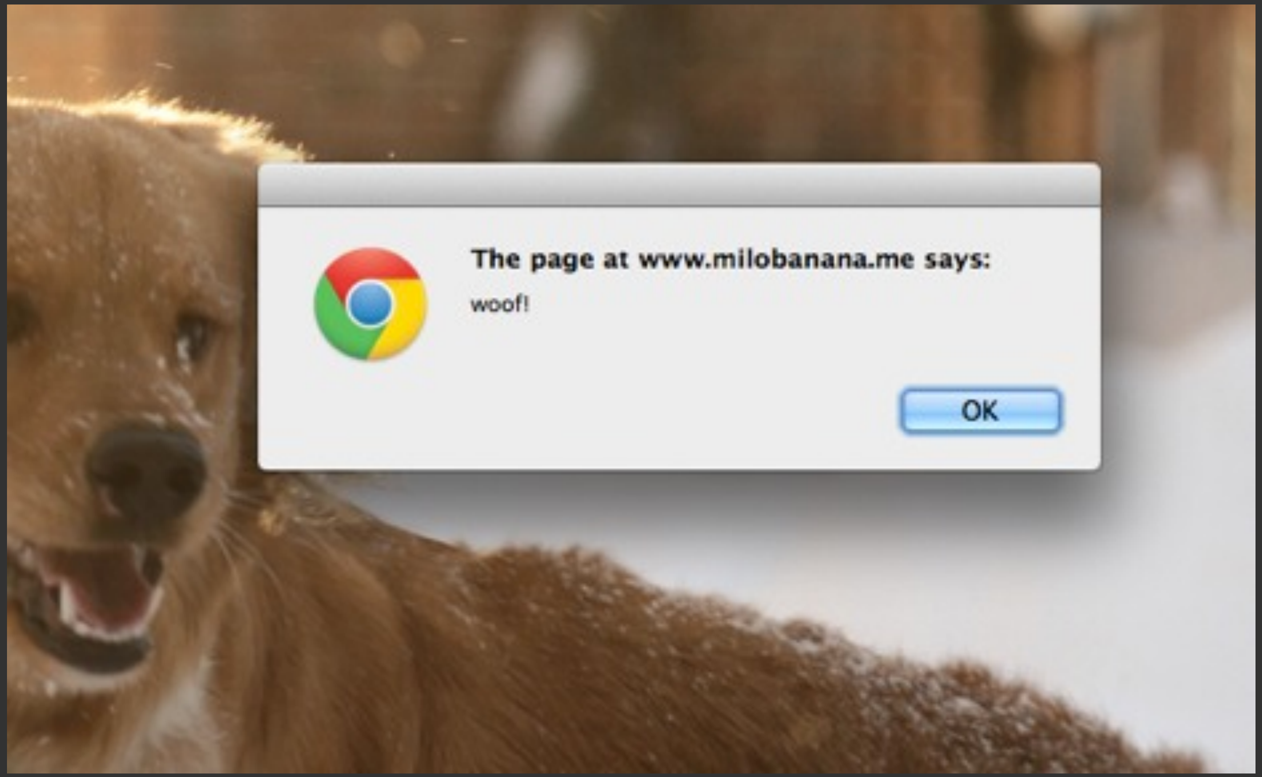
```
name
house
role
```

PHP, keys

JS objects

- You already know it's a mapping of keys to values. An associative array.
- But realize that values can be anything -- even a function!

```
var milo = {
    "name":   "Milo Banana",
    "house": "CS50 House",
    "role":   "Mascot",
    "bark":   function() {
        alert("woof!");
    }
};

milo.bark();
```


The page at www.milobanana.me says:

woof!

"loosely typed"

- JavaScript **does have types**. We just don't explicitly state them in declarations and the like.
- We can freely convert between types.

```
var i = 123;
i = "hello" + i;  // gives us the string "hello123" in i
```

- We can also freely compare between types using ==.

```
console.log("123" == 123);
true
```

- You can enforce comparison with types by using ===.

```
console.log("123" === 123);
false
```

global scope, nooooo

except in **functions**! phew!

```
  function foo(i)
  {
      i++;
      console.log(i);
  }

  var i = 999;
  for (var i = 0; i < 5; i++)
  {
→     console.log(i);
  }

  console.log(i);
  foo(i);
  console.log(i);
```

```
0
```

global scope, nooooo

except in **functions**! phew!

```
    function foo(i)
    {
        i++;
        console.log(i);
    }

    var i = 999;
    for (var i = 0; i < 5; i++)
    {
→       console.log(i);
    }

    console.log(i);
    foo(i);
    console.log(i);
```

```
0
1
```

global scope, nooooo

except in **functions**! phew!

```
function foo(i)
{
    i++;
    console.log(i);
}

var i = 999;
for (var i = 0; i < 5; i++)
{
    console.log(i);
}

console.log(i);
foo(i);
console.log(i);
```

```
0
1
2
```

global scope, nooooo

except in **functions**! phew!

```
function foo(i)
{
    i++;
    console.log(i);
}

var i = 999;
for (var i = 0; i < 5; i++)
{
→   console.log(i);
}

console.log(i);
foo(i);
console.log(i);
```

```
0
1
2
3
```

global scope, nooooo

except in **functions**! phew!

```
function foo(i)
{
    i++;
    console.log(i);
}

var i = 999;
for (var i = 0; i < 5; i++)
{
    console.log(i);
}

console.log(i);
foo(i);
console.log(i);
```

```
0
1
2
3
4
```

global scope, nooooo

except in **functions**! phew!

```
function foo(i)
{
    i++;
    console.log(i);
}

var i = 999;
for (var i = 0; i < 5; i++)
{
    console.log(i);
}

!!! ⟶ console.log(i);
foo(i);
console.log(i);
```

```
0
1
2
3
4
5
```

global scope, nooooo

except in **functions**! phew!

```
function foo(i)
{
    i++;
    console.log(i);
}

var i = 999;
for (var i = 0; i < 5; i++)
{
    console.log(i);
}

console.log(i);
foo(i);
console.log(i);
```

```
0
1
2
3
4
5
6
```

global scope, nooooo

except in **functions**! phew!

```
function foo(i)
{
    i++;
    console.log(i);
}

var i = 999;
for (var i = 0; i < 5; i++)
{
    console.log(i);
}

console.log(i);
foo(i);
console.log(i);
```

```
0
1
2
3
4
5
6
5
```

objects are passed by reference

(similar to arrays in C)

```
function cattify(object) {
    object.name = "cat";
}

var milo = {
    "name":  "Milo Banana",
    "house": "CS50 House",
    "role":  "Mascot"
};

console.log(milo.name);
cattify(milo);
console.log(milo.name);
```

```
Milo Banana
```

objects are passed by reference

(similar to arrays in C)

```
function cattify(object) {
    object.name = "cat";
}

var milo = {
    "name":  "Milo Banana",
    "house": "CS50 House",
    "role":  "Mascot"
};

console.log(milo.name);
cattify(milo);
console.log(milo.name);
```

```
Milo Banana
cat
```

objects are passed by reference

(similar to arrays in C)

```
function cattify(object) {
    object.name = "cat";
}

var milo = {
    "name":  "Milo Banana",
    "house": "CS50 House",
    "role":  "Mascot"
};

console.log(milo.name);
cattify(milo);
console.log(milo.name);
```

```
Milo Banana
cat
```

D:

using javascript in a webpage

```html
<!DOCTYPE html>

<html>
  <head>
    <title>Hello World!</title>
    <script>
      alert("I executed!");
    </script>
  </head>
  <body>
    <div>hello world</div>
  </body>
</html>
```

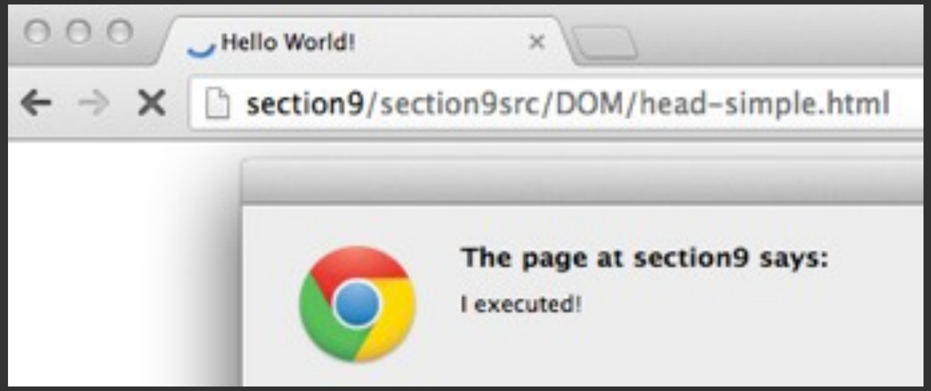
inline JS, works, but messy

```html
<!DOCTYPE html>

<html>
  <head>
    <title>Hello World!</title>
    <script src="path/to/file.js"></script>
  </head>
  <body>
    <div>hello world</div>
  </body>
</html>
```

JS in external file, ahhhh … nice.

placement matters

```
<!DOCTYPE html>

<html>
  <head>
    <title>Hello World!</title>
    <script>
      alert("I executed!");
    </script>
  </head>
  <body>
    <div>hello world</div>
  </body>
</html>
```



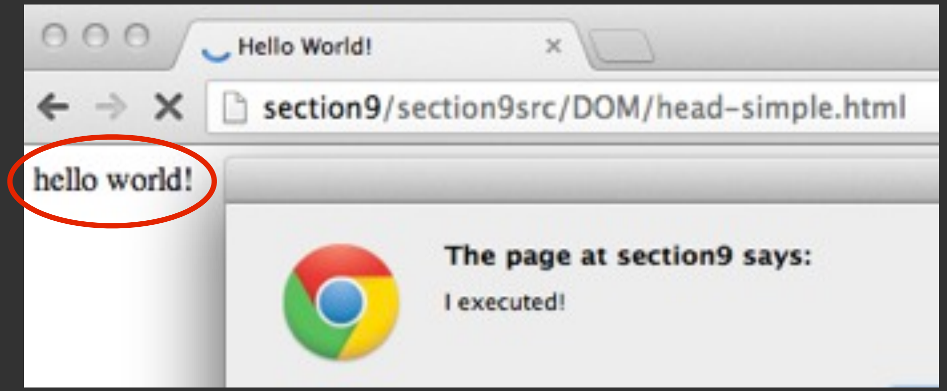
Hello World!

section9/section9src/DOM/head-simple.html

The page at section9 says:

I executed!

```
<!DOCTYPE html>

<html>
  <head>
    <title>Hello World!</title>
  </head>
  <body>
    <div>hello world</div>
    <script>
      alert("I executed!");
    </script>
  </body>
</html>
```



Hello World!

section9/section9src/DOM/head-simple.html

hello world!

The page at section9 says:

I executed!

how to wait until page loaded?

event handlers, hold the thought, we'll look at those when we get to jQuery

DOM

CS50 Quiz 1 Review

motivation

- So, if you look at HTML source code, it's just a bunch of text.

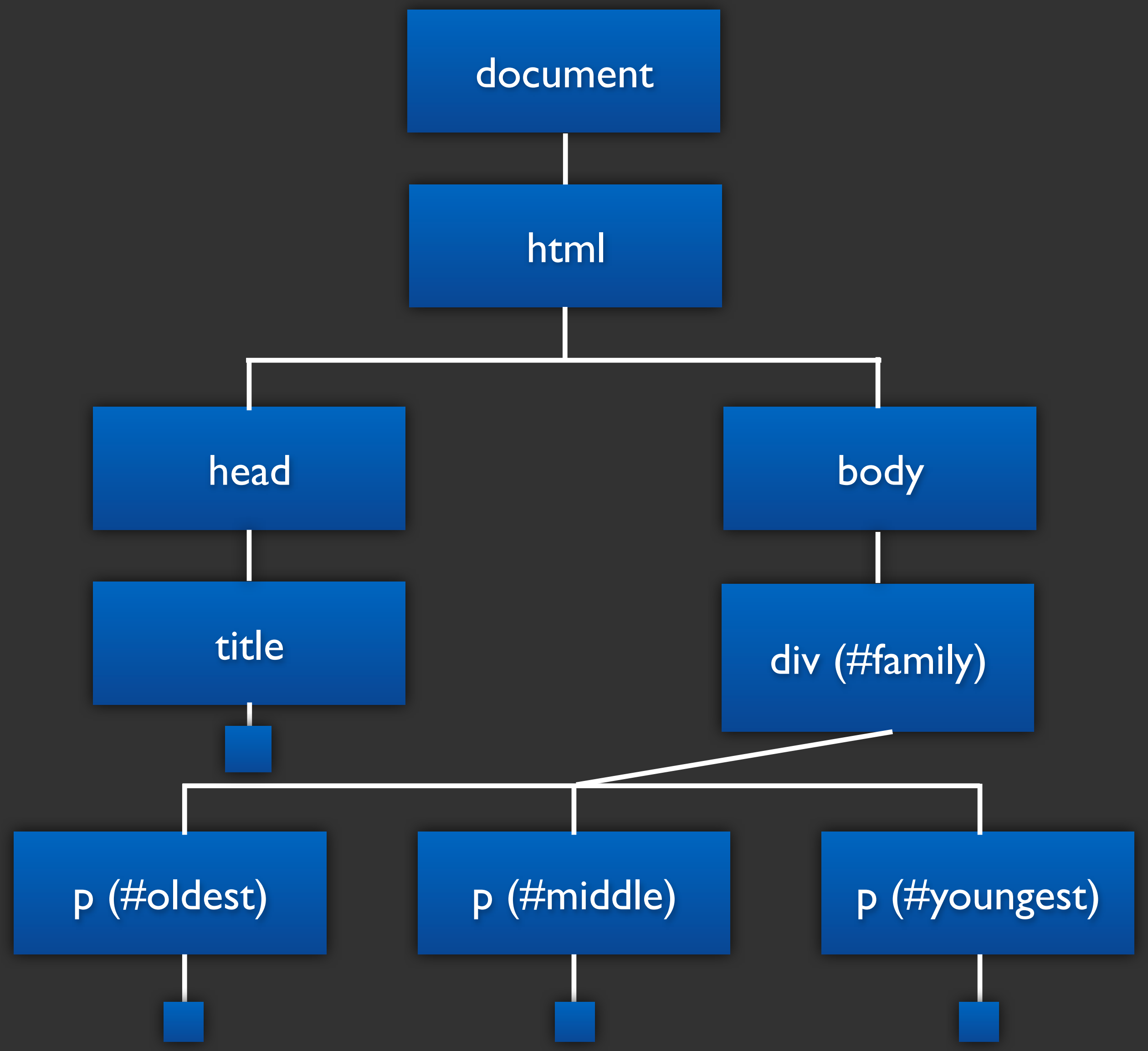
```
<!DOCTYPE html>

<html>
  <head>
    <title>Hello World!</title>
  </head>
  <body>
    <div>hello world</div>
  </body>
</html>
```

- However, JavaScript has to be able to select and modify HTML elements.
  - We need an in-memory representation of these elements.

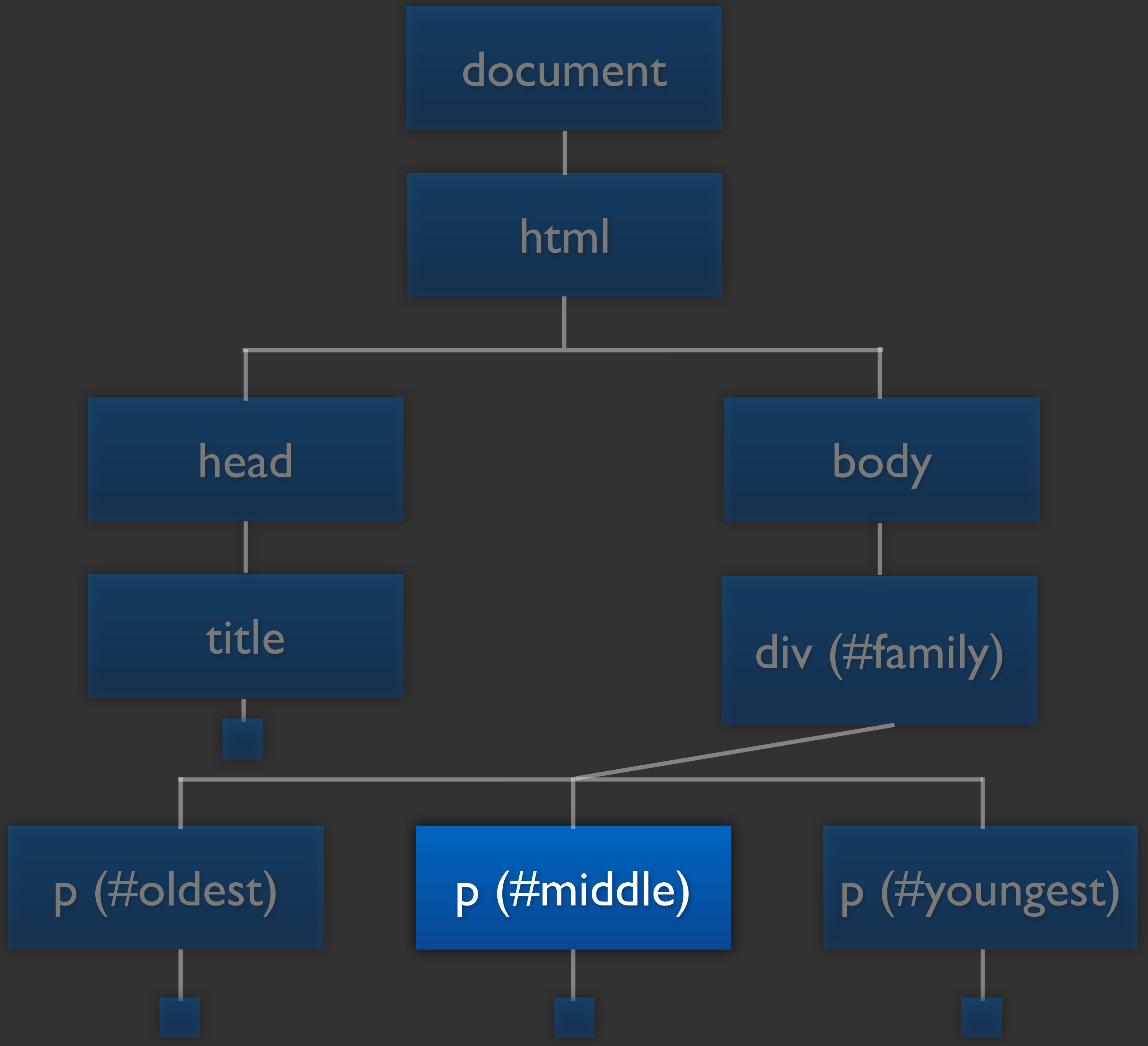
DOM

- This in-memory representation of HTML elements is the DOM. It's a DOM tree!

```html
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
    <div id="family">
      <p id="oldest">Alice</p>
      <p id="middle">Bob</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```



working with the DOM



Select DOM element

Do stuff to element

- Change html attributes of element
- Change css of element
- Change inner html of element
- Attach an event handler to an element

working with the DOM

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
    <div id="family">
      <p id="oldest">Alice</p>
      <p id="middle">Milo</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Select DOM element

$\downarrow$

Do stuff to element

- Change html attributes of element
- Change css of element
- Change inner html of element
- Attach an event handler to an element

what does this look like in JavaScript code?

Select DOM element → Do stuff to element

- Change html attributes of element
- Change css of element
- Change inner html of element
- Attach an event handler to an element

jQuery

what is jQuery?

- A JavaScript **library** that makes JavaScript easier to write.

- What are some of the things it does?
  - Makes selecting elements easier.
  - Make changing HTML, adding classes to elements, creating elements, easier.
  - Makes writing Ajax requests easier (we'll cover this soon).

- Analogous to how `<string.h>` is a C library.
  - Makes dealing with strings easier (gives you `strlen`, `strcpy`, etc.)

jQuery != JavaScript

jQuery is a **library** written in JavaScript

jQuery != JavaScript

jQuery is **not** a programming language, JavaScript **is**

$

how you invoke jQuery.

it is not PHP's $.

Selecting DOM Elements

Select DOM element → Do stuff to element

- How do we select HTML elements, once we have the DOM?

- Nasty regular JavaScript way, using the `document` variable.

→ `document.getElementById("family");`

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
    <p>Look at all the children!</p>
 →  <div id="family">
      <p id="oldest">Alice</p>
      <p id="middle">Bob</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Selecting DOM Elements

Select DOM element → Do stuff to element

- How do we select HTML elements, once we have the DOM?
- Nasty regular JavaScript way, using the `document` variable.

```
   document.getElementById("family");
→  document.getElementsByTagName("p");
```

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
 →  <p>Look at all the children!</p>
    <div id="family">
   →  <p id="oldest">Alice</p>
   →  <p id="middle">Bob</p>
   →  <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Selecting DOM Elements

Select DOM element → Do stuff to element

- How do we select HTML elements, once we have the DOM?
- Nasty regular JavaScript way, using the `document` variable.

```
→  document.getElementById("family");
   document.getElementsByTagName("p");
```

- Nice, simplified way, using a library we call jQuery.

```
→  $("#family");
```

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
    <p>Look at all the children!</p>
 →  <div id="family">
      <p id="oldest">Alice</p>
      <p id="middle">Bob</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Selecting DOM Elements

Select DOM element → Do stuff to element

- How do we select HTML elements, once we have the DOM?

- Nasty regular JavaScript way, using the `document` variable.

```
   document.getElementById("family");
→  document.getElementsByTagName("p");
```

- Nice, simplified way, using a library we call jQuery.

```
   $("#family");
→  $("p");
```

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
 →  <p>Look at all the children!</p>
    <div id="family">
   →  <p id="oldest">Alice</p>
   →  <p id="middle">Bob</p>
   →  <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Selecting DOM Elements

Select DOM element → Do stuff to element

- How do we select HTML elements, once we have the DOM?
- Nasty regular JavaScript way, using the `document` variable.

```
document.getElementById("family");
document.getElementsByTagName("p");
```

- Nice, simplified way, using a library we call jQuery.

```
  $("#family");
  $("p");
→ $("#family p");
```

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
    <p>Look at all the children!</p>
    <div id="family">
   → <p id="oldest">Alice</p>
   → <p id="middle">Bob</p>
   → <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

look familiar?

jQuery uses CSS Selectors

```
$("#family");
$("p");
$("#family p");
```

Using DOM Elements

Select DOM element → Do stuff to element

- jQuery also augments DOM elements with extra **methods**, or functions, that make using them easier.
- Let's change an DOM element's HTML with a jQuery method.

```
→ var element = $("#middle");
```

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
    <p>Look at all the children!</p>
    <div id="family">
      <p id="oldest">Alice</p>
  →   <p id="middle">Bob</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Using DOM Elements

Select DOM element → Do stuff to element

- jQuery also augments DOM elements with extra **methods**, or functions, that make using them easier.
- Let's change an DOM element's HTML with a jQuery method.

```
   var element = $("#middle");
→  element.html("Milo");
                ↑
            "method"
```

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
    <p>Look at all the children!</p>
    <div id="family">
      <p id="oldest">Alice</p>
   →  <p id="middle">Milo</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Using DOM Elements

Select DOM element → Do stuff to element

- jQuery also augments DOM elements with extra **methods**, or functions, that make using them easier.

- Let's add a class to a DOM element.

```
   var element = $("#family");
→  element.addClass("shadow");
```

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
    <p>Look at all the children!</p>
 →  <div id="family" class="shadow">
      <p id="oldest">Alice</p>
      <p id="middle">Bob</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Using DOM Elements

Select DOM element → Do stuff to element

- jQuery also augments DOM elements with extra **methods**, or functions, that make using them easier.
- Let's remove an element from the DOM.

```
  var element = $("#youngest");
→ element.remove();
```

- You can also shorten this to just:

```
$("#youngest").remove();
     ↑            ↑
Select DOM element   Do stuff to element
```

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
  </head>
  <body>
    <p>Look at all the children!</p>
    <div id="family">
      <p id="oldest">Alice</p>
      <p id="middle">Bob</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Using DOM Elements

Select DOM element → Do stuff to element

- jQuery also augments DOM elements with extra **methods**, or functions, that make using them easier.
- Hmmmm, but is it as easy as doing something like this?
  - Remember, execution order, why does it matter?

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
    <script>
      $("#youngest").remove();
    </script>
  </head>
  <body>
    <p>Look at all the children!</p>
    <div id="family">
      <p id="oldest">Alice</p>
      <p id="middle">Bob</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Using DOM Elements

Select DOM element → Do stuff to element

- JavaScript is **event-driven**. We can go about using events to help solve the problem on the previous slide.

- Let's attach a **ready event handler** to the document, so JS code executes only once the document is ready (HTML has loaded).

```
$(document).ready(foo);
```

select document

when what we selected is ready

run this function

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
    <script>
      $(document).ready(function(event) {
        $("#youngest").remove();
      });
    </script>
  </head>
  <body>
    <p>Look at all the children!</p>
    <div id="family">
      <p id="oldest">Alice</p>
      <p id="middle">Bob</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Using DOM Elements

Select DOM element → Do stuff to element

- JavaScript is **event-driven**. We can go about using events to **add interactivity to our website**.

- More compellingly, let's attach a **click event handler** this time to do something when the user clicks on Charlie.

```
$("#youngest").click(foo);
```

select document

when what we selected is clicked

run this function

```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
    <script>
        $(document).ready(function(event) {
           $("#youngest").click(function(event) {
               alert("I'm Charlie!");
           });
        });
    </script>
  </head>
  <body>
    <p>Look at all the children!</p>
    <div id="family">
      <p id="oldest">Alice</p>
      <p id="middle">Bob</p>
      <p id="youngest">Charlie</p>
    </div>
  </body>
</html>
```

Combining Everything

let's do some **client-side validation** on a form

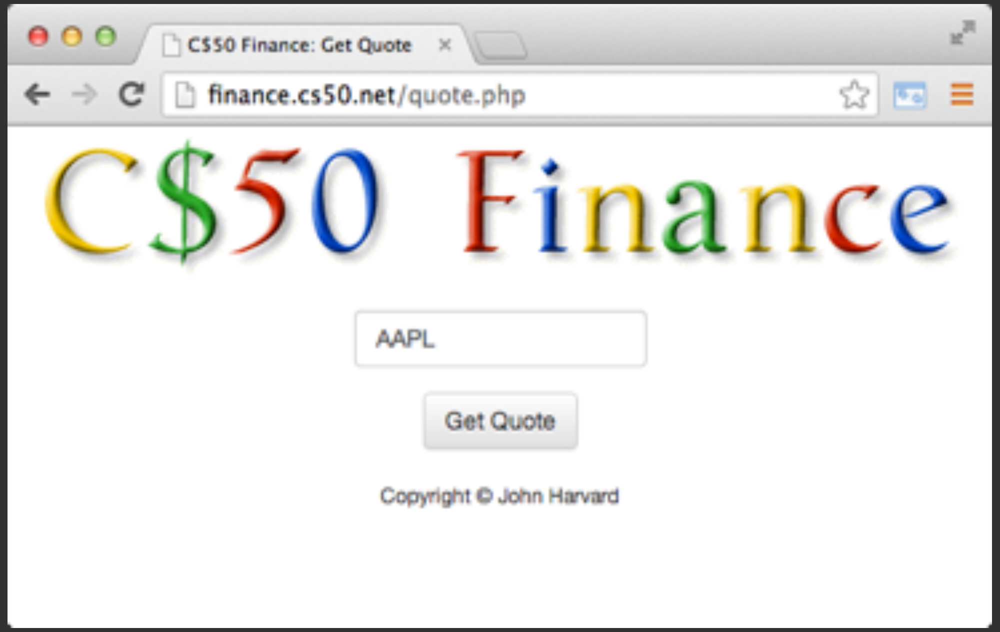
```
<!DOCTYPE html>

<html>
  <head>
    <title>DOM!</title>
    <script>
       $(document).ready(function(event) {
           $("#registration").submit(function(event) {
               // validate inputted username
               if ($('#registration input[name=username]').val() == '')
               {
                   return false;
               }

               // validate inputted password
               var password = $('#registration input[name=password]').val();
               if (password == '' || password.length < 8)
               {
                   return false;
               }
           });
       });
    </script>
  </head>
  <body>
    <form id="registration" action="register.php" method="post">
      Username: <input name="username" type="text"/><br/>
      Password: <input name="password" type="password"/>
    </form>
  </body>
</html>
```

pros, cons

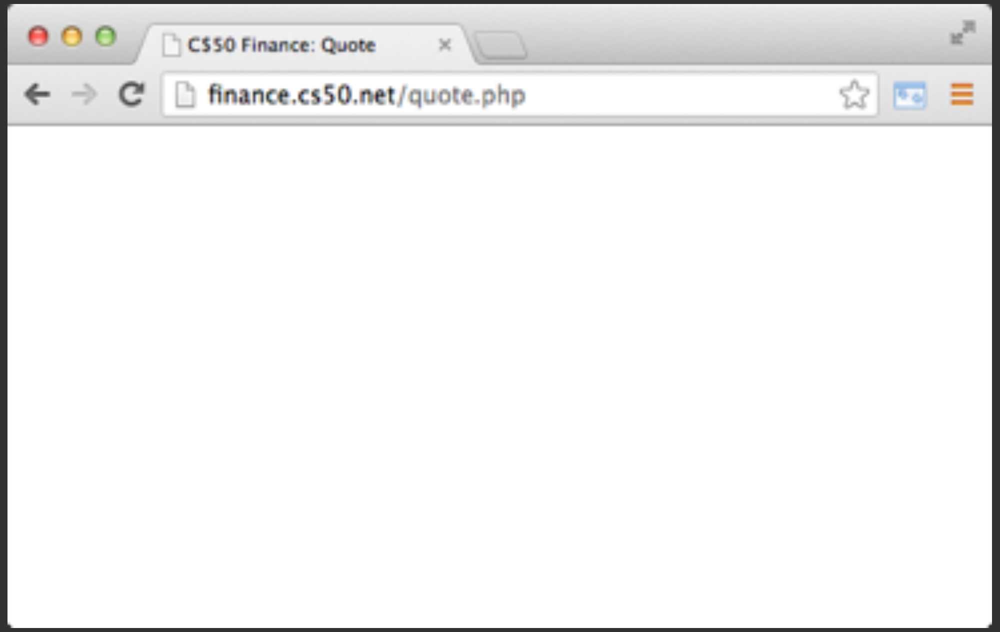
makes JavaScript easier to write, but slower than pure JavaScript

Ajax

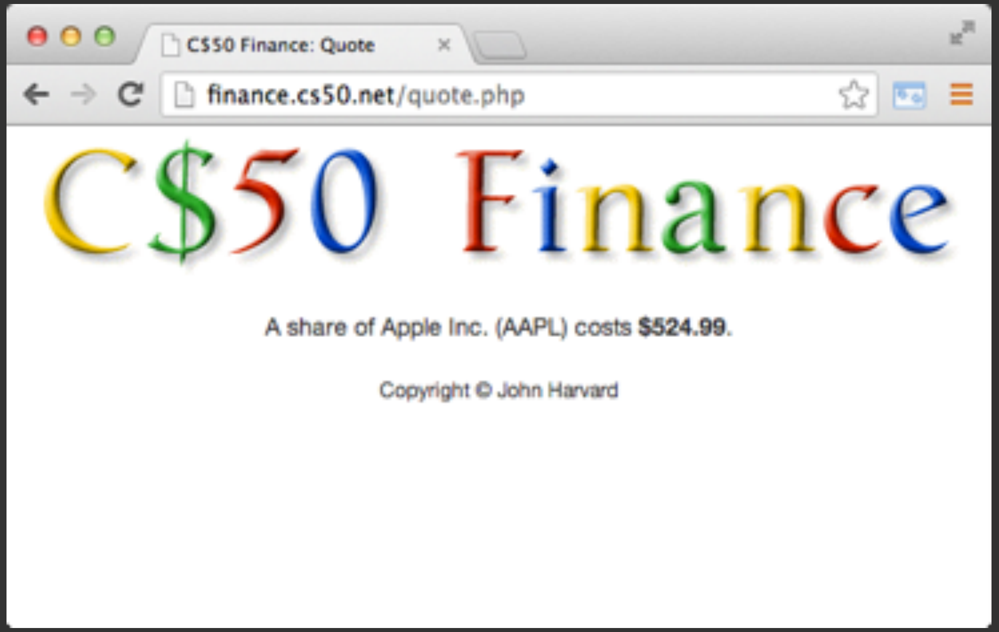
so far...



→



→



A share of Apple Inc. (AAPL) costs **$524.99**.

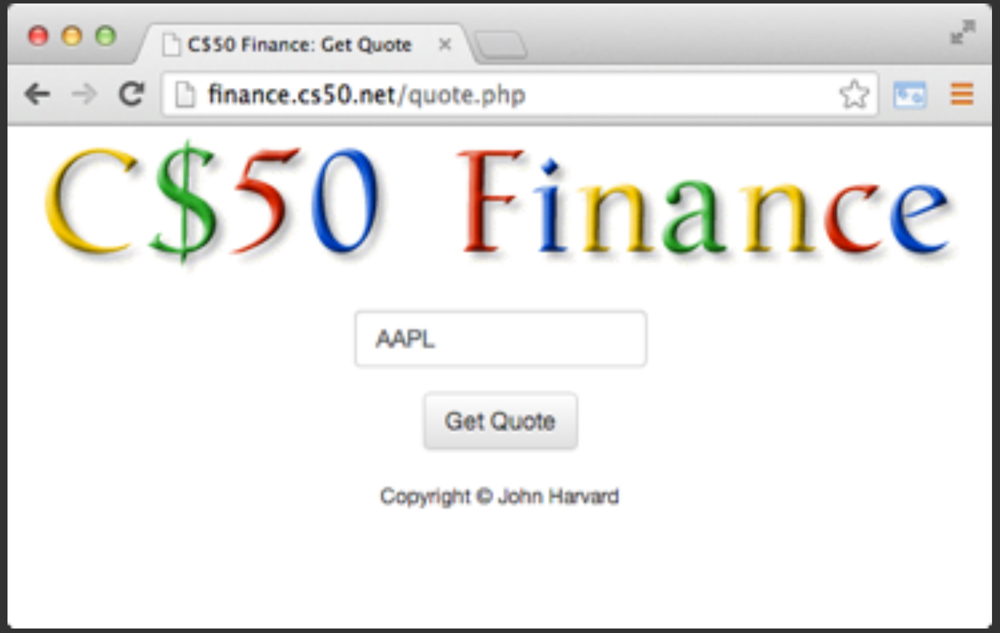
flash when loading new page

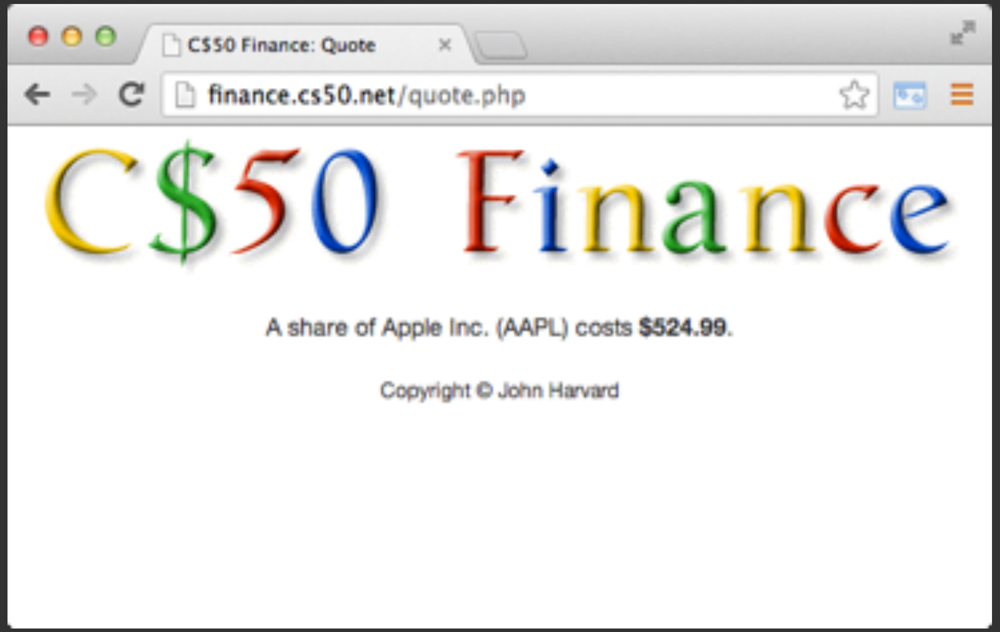
:(

something like this would be nice

```
// get quote name from input element
var name = $("#quote_input").val();

// fetch stock price from Yahoo!
var price = stockInfoFromYahoo(name);

// insert message into the DOM, replacing form
var msg = "A share of " + name + " costs " + price;
$("#form_wrapper").html(msg);
```

ahhhh … nice! no refreshing!





A share of Apple Inc. (AAPL) costs **$524.99**.

Copyright © John Harvard

problem

"synchronous execution"

a line in javascript only executes after the previous line is done executing

```
var name  = $("#quote_input").val();
var price = stockPriceFromYahoo(name);
var msg   = "A share of " + name + " costs " + price;
$("#form_wrapper").html(msg);
```

What if Yahoo! is having a extremely slow day and this function takes **seconds**?

Your website will completely freeze for those seconds. This is Bad™.

solution

"**asynchronous** javascript and xml (**A**jax)"

keep going for now, but make a promise to execute a function later on when data comes back

```
var name = $("#quote_input").val();
var url = "http://yahoo.com/stock?s=" + name;

// how a GET Ajax request is done with jQuery
$.get(url, function(price) {
    var msg = "A share of " + name + " costs " + price;
    $("#form_wrapper").html(msg);
});
```

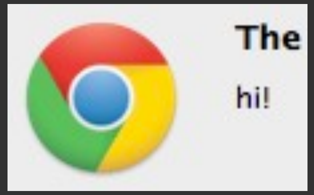
Instead of waiting **synchronously**, we immediately move on.

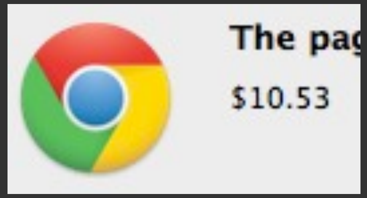
We'll get notified later when the data is ready for processing.

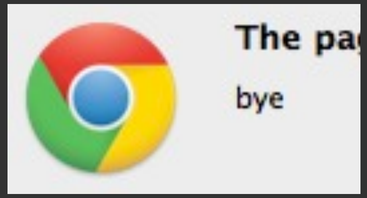
hopefully, this helps!

```
1  alert("hi!");
2  var price = stockInfoFromYahoo(name);
3  alert(price);
4  alert("bye!");
```

@ 0s 3s 3.001s


The
hi!

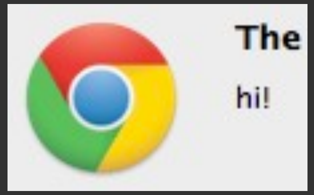

The pag
$10.53

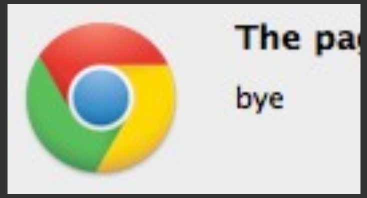

The pa
bye

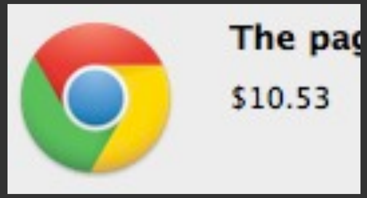
synchronous

```
1  alert("hi!");
2  $.get(url, function(price) {
   4  alert(price);
   });
3  alert("bye!");
```

@ 0s 0.001s 3s


The
hi!


The pa
bye


The pag
$10.53

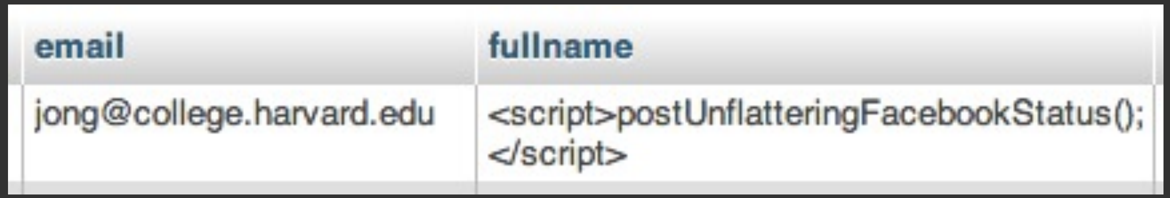
asynchronous

Ajax tl;dr

let's us fetch data, without refreshing the current page.

let's us do this in an asynchronous way that doesn't freeze our page.

XSS Attacks



email	fullname
jong@college.harvard.edu	<script>postUnflatteringFacebookStatus(); </script>

```
<?php foreach ($friends as $friend): ?>
    <div>
      <?= $friend["fullname"] ?>
    </div>
<?php endforeach; ?>
```

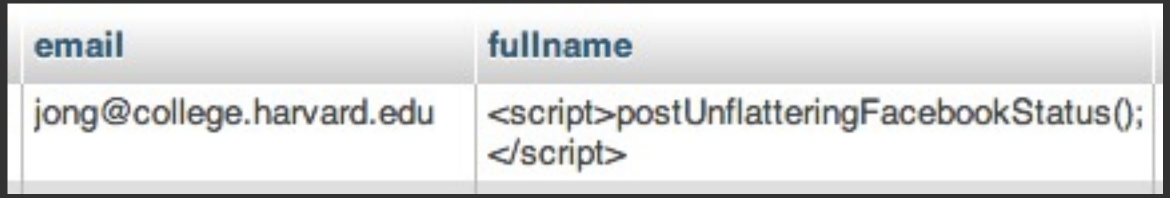
or

```
$("#name_wrapper").html(friend);
```

```
<!DOCTYPE html>

<html>
  <head>
    <title>Facebook!</title>
  </head>
  <body>
    <div>
      Lauren Carvalho
    </div>
    <div>
      Milo Banana
    </div>
    <div>
      <script>postUnflatteringFacebookStatus();</script>
    </div>
  </body>
</html>
```

??????????

simple solution



email	fullname
jong@college.harvard.edu	<script>postUnflatteringFacebookStatus(); </script>

```
<?php foreach ($friends as $friend): ?>
    <div>
      <?= htmlspecialchars($friend["fullname"]) ?>
    </div>
<?php endforeach; ?>
```