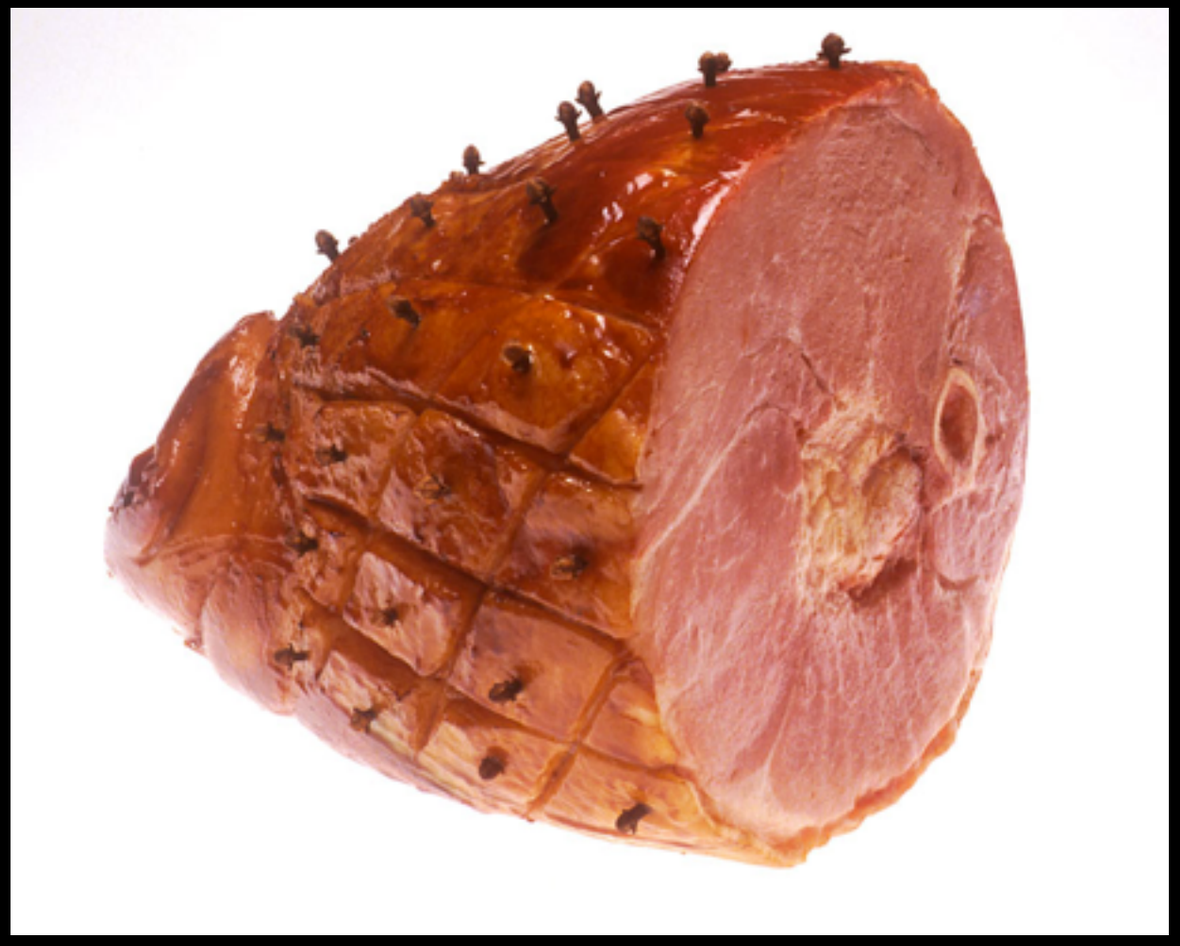
6.148

Frontend II: Javascript and DOM Programming

Let’s talk about Javascript :)

Why Javascript?

Designed in *ten days* in December 1995!



How are they similar?

“Javascript is to Java as hamster is to ham”

Marketing!

- Java began to become immensely popular in the ‘90s as a “powerful programming language”

Marketing!

- Java began to become immensely popular in the ‘90s as a “powerful programming language”
- Javascript -- influenced more by Scheme than Java!
    - Scheme, with syntax borrowed from C

wat

"... we have to appreciate the reasons for picking not the most powerful solution **but the least powerful**"

-- Tim Berners-Lee, 1998

"... we have to appreciate the reasons for picking not the most powerful solution **but the least powerful**"

-- Tim Berners-Lee, 1998

"Any application that can be written in Javascript, will eventually be written in Javascript"

-- "Atwood's Corollary", 2007

Javascript: The Good Parts

Javascript: The Definitive Guide

So why are we learning a language that

... was built in 10 days
... can't decide if it's functional or object-oriented
... has become the butt of computer-science jokes
everywhere???

Let's get real

- Objectively, Javascript isn't *that* bad
  - In fact, many of us who have gotten to know it enjoy it
- A language with a *good, small core*, and lots of unnecessary fluff
  - Learn to use the good parts

WIDESCREEN

STAR WARS IV

A NEW HOPE

DVD VIDEO

DIGITALLY MASTERED
FOR SUPERIOR SOUND THX AND PICTURE QUALITY

A new hope...for JS

- Better standards: ECMAScript
- Google V8
- Javascript on the client and server!
- Tons of libraries -- Javascript is (re)gaining popularity!

A lightning overview

```
var n = 3;
var str = “hello!”

if (n == 3) {
    console.log(“three”);
} else {
    console.log(“not three”);
}

for (var i = 0; i < 10; i++) {
    console.log(3 * i);
}
```

Arrays

```
var a = [1, 2, 3];
var b = [];

for (var i = 0; i < a.length; i++) {
    b.push(2*a);
}

console.log(b)    // [2, 4, 6]
```

Objects

```
var obj = {
    1: “hi”,
    2: 3,
    “abc”: “def”,
    “array”: [1,2,3,4,5]
};

console.log(obj[1]);       // “hi”
console.log(obj.abc);      // “def”
console.log(obj.hello);    // undefined
```

Functions!

```
function f(a, b) {
    return a + b;
}

var f = function(a, b) {
    return a + b;
}
```

Functions!

```
function f(a, b) {
    return a + b;
}

var f = function(a, b) {
    return a + b;
}
```

A function can be treated just like any other variable!

Candy

```
var func = function(f) {
    return f(3);
}
```

What is `func(function(x) { return x; });`

Candy

```
var func = function(f) {
    return f(3);
}

var a = function(x) {
    return 2 * x;
}
```

What is `func(a);`

Candy

```
var func = function(f) {
    return f;
}
```

What is `func(function(x) { return x; });`

Candy

```
var func = function(f) {
    return f;
}
```

What is `func(function(x) { return x; })(3);`

Candy

```
var func = function() {
    return
        [1,2,3]
};
```

What is `func();`

Use your semicolons

More candy!

What is ...

```
"20" == 20
```

More candy!

What is ...

```
“20” == 20          // TRUE

false == 0
```

More candy!

What is ...

```
"20" == 20          // TRUE

false == 0          // TRUE

[] == []
```

More candy!

What is ...

```
“20” == 20          // TRUE
false == 0          // TRUE
[] == []            // FALSE
‘’ == 0
```

More candy!

What is ...

```
“20” == 20          // TRUE
false == 0          // TRUE
[] == []            // FALSE
‘’ == 0             // TRUE
‘\n\n\n’ == 0
```

More candy!

What is ...

```
“20” == 20          // TRUE
false == 0          // TRUE
[] == []            // FALSE
‘’ == 0             // TRUE
‘\n\n\n’ == 0       // TRUE
NaN == NaN
```

More candy!

What is ...

```
“20” == 20          // TRUE
false == 0          // TRUE
[] == []            // FALSE
‘’ == 0             // TRUE
‘\n\n\n’ == 0       // TRUE
NaN == NaN          // FALSE
```

Use === and !==

Javascript for the Browser

- HTML: gives us a “nested tree” structure of elements
- Manipulate these elements with Javascript!
- “The DOM”
- DEMO

jQuery

- Writing Javascript for the browser is cumbersome!
- jQuery allows us to write less by using CSS selectors and providing helper functions
  - `$('div')`
  - `$('.classname')`
  - `$('#element-id')`
- DEMO

What can we do with jQuery?

- Find an element -- `$('#element-id')`
- DOM "tree traversal"
- Element styles: show, hide, add/remove classes, change CSS
- Add and remove DOM elements and HTML!

Events!

```
$(‘#element-id’).on(
    ‘click’,
    function(event) { ... }
);
```

Events: hover, mousedown, mouseup, keypress, etc...

Shortcuts: `.click(...)`, `.hover(...)`, etc.

Events!

```
$(‘#element-id’).on(
    ‘click’,
    function(event) { ... }
);
```

Events: hover, mousedown, mouseup, keypress, etc...

Shortcuts: `.click(...)`, `.hover(...)`, etc.

REMEMBER: **functions are just like any other variable!**

Callbacks?!?

- Javascript is **asynchronous**
- For now, just think of it as -- not everything executes in the order written

Callbacks?!?

- Javascript is **asynchronous**
- For now, just think of it as -- not everything executes in the order written

```
$(‘#element-id’).on(
    ‘click’,
    function(event) { ... }
);
```

Callbacks?!?

```
$(‘#element-id’).on(
    ‘click’,
    function(event) { ... }
);
```

- “Hey browser, call this function whenever someone clicks on #element-id”
- The callback function is executed **only when the event triggers it**

DEMO

Making a list