

CoffeeScript is For Closers

Brandon Satrom

@BrandonSatrom

UserInExperience.com

Slides: bit.ly/pLWIMf

A close-up photograph of a man with dark hair and a mustache, wearing a dark suit jacket, a white shirt, and a patterned tie. He is shouting into a large, light-colored megaphone. A speech bubble is positioned to the left of the megaphone, containing text.

Put the
Coffee(Script)
Down!
Coffee(Script) is for
Closers!

Or...

JavaScript, Linguistics and
Cockney Rhyming Slang

Or...

How CoffeeScript Made me a
Better JavaScript Developer

So Who Am I?

- ✓ Husband and Father
- ✓ Web Developer & Evangelist, Microsoft
- ✓ Austin, TX



www.userinexperience.com
[@BrandonSatrom](https://twitter.com/BrandonSatrom)
github.com/bsatrom

#html5tx



Agenda

- The JavaScript Dialectic
- What?
- Where?
- How?
- Why?

JS

VS.



A JavaScript Dialectic



Don't Be A JavaScriptster



I loved JavaScript
before it was cool

Of course, I also hated it back when
everyone else did too...

Don't Be A CoffeeScriptster



JavaScript is lame!
Significant
Whitespace FTW!



Let's hope they don't notice that I still
debug in JavaScript...

What is CoffeeScript?

WHAT?

“It’s Just JavaScript”

Or “Unfancy JavaScript”

“A polyglot language”

A Ruby, Haskell and Perl love child...

“A little language”

Meaning...it's not DART

Transpiler vs. Compiler

Transpiled Languages

- CoffeeScript
- Traceur
- EcmaScript 5 Parser
- Reflect.js

Compiled Languages

- DART
- GWT
- Objective-J
- ClojureScript
- JSIL
- Script#
- Emscripten
- Haxe

LIAR!

More A Dialect, Less A Language

Color === Colour

function(foo) {} === (foo) ->

Subtle Linguistics

Dave has two children. His youngest is in Reception, and his favourite colour is green. His oldest is Year 4, and loves to ride the lift outside his flat.

Dave has two children. His youngest is in Pre-K, and his favorite color is green. His oldest is in 3rd Grade, and he loves to ride the elevator outside his apartment.

Not so subtle linguistics...

Sentence	Extended Phrase	Rhymes With
I've got a sore billy.	Billy Goat	Throat
That's a nasty old boris you've got there son.	Boris Karloff	Cough
Ere, you've got your brass wrong!	Brass Tacks	Facts
'ow about a Brittney?	Brittney Spears	Beers
We're all in Barney!	Barney Rubble	Trouble
Think he's been smoking a bit of Bob	Bob Hope	(you get the idea)

CoffeeScript is not a replacement
for *Learning JavaScript*

CoffeeScript Axiom #1

Compiles 1:1 to JavaScript

1:1 being Concept to Concept

Conceptual Equivalence

CoffeeScript

```
numbers = [1..10]
```

```
printNum = (number) ->  
  console.log "This is  
number #{number}"
```

```
printNum for num in numbers
```

JavaScript

```
var num, numbers, printNum, _i,  
_len;  
numbers = [1, 2, 3, 4, 5, 6, 7, 8,  
9, 10];
```

```
printNum = function(number) {  
  return console.log("This is  
number " + number);  
};
```

```
for (_i = 0, _len =  
numbers.length; _i < _len; _i++) {  
  num = numbers[_i];  
  printNum;  
}
```

Conceptual Equivalence

CoffeeScript

```
numbers = [1..10]

printNum = (number) ->
  console.log "This is
  number #{number}"

printNum num for num in
  numbers
```

JavaScript

```
var num, numbers, printNum, _i,
  _len;
numbers = [1, 2, 3, 4, 5, 6, 7, 8,
  9, 10];

printNum = function(number) {
  return console.log("This is
  number " + number);
};

for (_i = 0, _len =
  numbers.length; _i < _len; _i++) {
  num = numbers[_i];
  printNum(num);
}
```

Conceptual Equivalence

CoffeeScript

```
numbers = [1..10]

printNum = (number) ->
  console.log "This is
  number #{number}"

printNum num for num in
  numbers
```

JavaScript

```
var num, numbers, printNum, _i,
  _len;
numbers = [1, 2, 3, 4, 5, 6, 7, 8,
  9, 10];

printNum = function(number) {
  return console.log("This is
  number " + number);
};

for (_i = 0, _len =
  numbers.length; _i < _len; _i++) {
  num = numbers[_i];
  printNum(num);
}
```

Conceptual Equivalence

CoffeeScript

```
numbers = [1..10]

printNum = (number) ->
  console.log "This is
  number #{number}"

printNum num for num in
  numbers
```

JavaScript

```
var num, numbers, printNum, _i,
  _len;
numbers = [1, 2, 3, 4, 5, 6, 7, 8,
  9, 10];

printNum = function(number) {
  return console.log("This is
  number " + number);
};

for (_i = 0, _len =
  numbers.length; _i < _len; _i++) {
  num = numbers[_i];
  printNum(num);
}
```

Where do I get it?

WHERE?

Ways to Get CoffeeScript

- node.js & npm

```
$ npm install -g coffee-script
```

- Script Tags
 - <script type="text/coffeescript"></script>
 - <script src="coffee-script.js"></script>
- Rails 3.1
- SassAndCoffee (.NET)
- CoffeeScript Packages

Tools for CoffeeScript



How can I use, learn and automate it?

HOW?

```
$ coffee  
coffee> [1..10]
```

REPL

```
$ coffee -o js/ -cw lib/
```

COMPILE



CoffeeScript is a little language that compiles into JavaScript. Underneath all of those embarrassing braces and semicolons, JavaScript has always had a gorgeous object model at its heart. CoffeeScript is an attempt to expose the good parts of JavaScript in a simple way.

The golden rule of CoffeeScript is: "*It's just JavaScript*". The code compiles one-to-one into the equivalent JS, and there is no interpretation at runtime. You can use any existing JavaScript library seamlessly (and vice-versa). The compiled output is readable and pretty-printed, passes through [JavaScript Lint](#) without warnings, will work in every JavaScript implementation, and tends to run as fast or faster than the equivalent handwritten JavaScript.

Latest Version: [1.1.2](#)

Overview

CoffeeScript on the left, compiled JavaScript output on the right.

```
# Assignment:  
number = 42  
opposite = true  
  
# Conditions:  
number = -42 if opposite  
  
# Functions:
```

```
var cubes, list, math, num, number, oppo,  
var __slice = Array.prototype.slice;  
number = 42;  
opposite = true;  
if (opposite) number = -42;  
square = function(x) {  
    return x * x;  
};
```

COFFEESCRIPT BASICS

Assignment

```
name = "Brandon"
```

```
married = yes
```

```
children = 2
```

```
var children, married, name;
```

```
name = "Brandon";
```

```
married = true;
```

```
children = 2;
```



Functions

```
square = (x) -> x * x
```

```
printInfo = (name, numChildren) ->
```

```
  "Hi #{name}, I see you have #{numChildren}  
children."
```

```
var printInfo, square;  
square = function(x) {  
  return x * x;
```

```
};  
printInfo = function(name, numChildren) {  
  return "Hi " + name + ", I see you have " + numChildren + "  
children";  
};
```



Strings

```
num = 99
```

```
“I’ve got #{num} problems, and JavaScript ain’t  
one.”
```

```
city = "Brooklyn"
```

```
"No. Sleep. 'Till. #{city}"
```

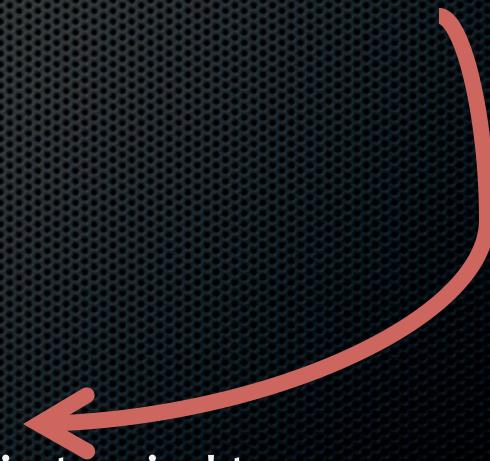
```
var city, num;
```

```
num = 99;
```

```
"I've got " + num + " problems, but JavaScript ain't  
one";
```

```
city = "Brooklyn";
```

```
"No. Sleep. 'Till. " + city;
```



Objects

```
kids =  
  bigBrother:  
    name: "Benjamin"  
    age: 2  
  
  littleBrother:  
    name: "Jack"  
    age: 1
```



```
var kids;  
kids = {  
  bigBrother: {  
    name: "Benjamin",  
    age: 2  
  },  
  littleBrother: {  
    name: "Jack",  
    age: 1  
  }  
};
```

Lexical Scoping

outer = 42

findMeaning = ->

 inner = 43

 outer

inner = findMeaning()



```
var findMeaning, inner,  
outer;  
outer = 42;  
findMeaning = function() {  
    var inner;  
    inner = 43;  
    return outer;  
};  
inner = findMeaning();
```

Lexical Scoping 2

```
notGlobal = "Hello"
```

```
global = "Goodbye"
```

```
window.global = global
```

```
(function() {  
  var global, notGlobal;  
  notGlobal = "Hello";  
  global = "Goodbye";  
  window.global = global;  
}).call(this);
```

Everything, an Expression

```
grade = (student) ->
  if student.excellentWork
    "A+"
  else if student.okayStuff
    if student.triedHard then "B"
  else "B-"
  else
    "C"

eldest = if 24 > 21 then "Liz"
else "Ike"
```



```
var eldest, grade;
grade = function(student) {
  if (student.excellentWork) {
    return "A+";
  } else if (student.okayStuff)
  {
    if (student.triedHard) {
      return "B";
    } else {
      return "B-";
    }
  } else {
    return "C";
  };
}
eldest = 24 > 21 ? "Liz" :
"Ike";
```

Conditionals

```
mood = greatlyImproved if singing
```

```
if happy and knowsIt  
    clapsHands()  
    chaChaCha()  
else  
    showIt()
```

```
lunch = if friday then water else  
redBull
```

```
options or= defaults
```



```
var lunch, mood;  
if (singing) {  
    mood = greatlyImproved;  
}  
if (happy && knowsIt) {  
    clapsHands();  
    chaChaCha();  
} else {  
    showIt();  
}  
lunch = friday ? water :  
redBull;  
Options || (options =  
defaults);
```

Operators and Aliases

CoffeeScript	JavaScript
is, ==	==
isnt, !=	!=
not	!
and	&&
or	
true, yes, on	true
false, no, off	false
@, this	this
of	in
in	<i>no equivalent</i>

How I Use CoffeeScript

Let's pretend for a moment that how
I use it, matters.

Coffee, Cake and Growl

1. Cake (think Make or Rake)
2. 'watch' Task

```
invoke 'build'  
invoke 'tests'
```

3. wingrr (use node-growl for *nix)

```
msg = 'All test pass'  
sys.puts msg.green  
wingrr.notify msg
```

Why should I bother using it?

WHY?

More succinct code PRO

```
number = 42      // the left, compiled JavaScript output on the right.
opposite = true

number = -42 if opposite
number = 42
square = (x) -> x * x

list = [1, 2, 3, 4, 5]
number = 42 if opposite
math =
  root: Math.sqrt
  square: square
  cube: (x) -> x * square x

race = (winner, runners...) ->
  print winner, runners

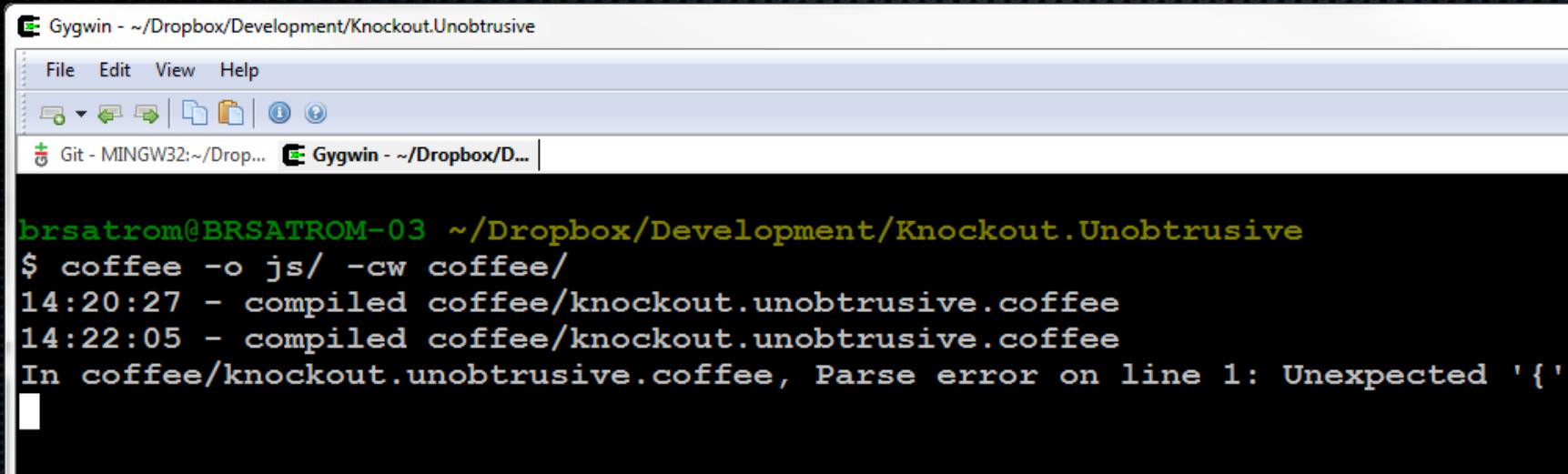
cubes = (math.cube num for num in list)
  root: Math.sqrt
  square: square
  cube: (x) -> x * square x

# elvis
race = (elvis, runners...) ->
  print winner, runners

# narrowing
alert("I know who it is elvis")

# anonymous functions
cubes = (math.cube num for num in list)

var cubes, list, math, num, number, opposite, race, square;
var __slice = Array.prototype.slice;
number = 42;
cubes, list, math, num, number, opposite, race, square;
var __slice = Array.prototype.slice;
number = 42;
if (opposite) {
  number = 42;           number = -42;
  opposite = false;     }
if (opposite) number square = function(x) {
  number = function(x) { return x * x;
    return x * x; };   };
list = [1, 2, 3, 4, 5];
list = [1, 2, 3, 4, 5]; math = {
math: {                   root: Math.sqrt,
  root: Math.sqrt,       square: square,
  square: square,       cube: function(x) {
  cube: function(x) {   return x * square(x);
    return x * square(x); }
  } };                 };
race = function() {
  race = function() {
    var runners, winner;
    var runners, winner, winner = arguments[0], runners = 2 <= arguments.length ? __slice.call(arguments, 1) : [];
    winner = arguments[0]; return print(winner, runners);
    __slice.call(arguments);
    return print(winner, cubes = (function() {
      var _i, _len, _results;
      if (!proto.elvis || !proto.elvis[_i]) _results = [];
      cubes = (function() {
        for (_i = 0, _len = list.length; _i < _len; _i++) {
          var _i, _len, _results;
          num = list[_i];
          _results = [];
          _results.push(math.cube(num));
          for (_i = 0, _len = 1) {
            num = list[_i];
            _results.push(num);
            _results.push(num));
          }
        }
        return _results;
      })();
      return _results;
    }));
  }
}
```



A screenshot of a terminal window titled "Gygwin - ~/Dropbox/Development/Knockout.Unobtrusive". The window has a menu bar with "File", "Edit", "View", and "Help". Below the menu is a toolbar with icons for file operations like copy, paste, and search. The terminal window shows a command-line session:

```
brsatrom@BRSATROM-03 ~/Dropbox/Development/Knockout.Unobtrusive
$ coffee -o js/ -cw coffee/
14:20:27 - compiled coffee/knockout.unobtrusive.coffee
14:22:05 - compiled coffee/knockout.unobtrusive.coffee
In coffee/knockout.unobtrusive.coffee, Parse error on line 1: Unexpected '{'
```

Extra compile step between you and the “real” code

CON

```
hi == bye
global = "everywhere?"

switch day
  when "Fri", "Sat"
    if day is bingoDay
      go bingo
      go dancing
  when "Sun" then go church
  else go work
```

Avoid Snake Pits & Get JS Patterns for Free

PRO

```
(function() {
  var global;
  hi === bye;
  global = "everywhere?";
  switch (day) {
    case "Fri":
    case "Sat":
      if (day === bingoDay) {
        go(bingo);
        go(dancing);
      }
      break;
    case "Sun":
      go(church);
      break;
    default:
      go(work);
  }
})();
```

The screenshot shows the Microsoft Internet Explorer F12 developer tools interface. The title bar reads "knockout.unobtrusive tests - F12". The menu bar includes File, Find, Disable, View, Images, Cache, Tools, Validate, Browser Mode: IE9, Document Mode: IE9 standards. The tabs at the top are HTML, CSS, Console, Script (which is selected), Profiler, and Network. Below the tabs is a toolbar with icons for Stop, Run, Break, and Stop debugging. The main area displays the source code of the "knockout.unobtrusive.js" file. The code is a JavaScript file for Knockout.js version 0.2, containing functions for creating bindings, getting elements by ID or class name, and handling custom values.

```
1 (function() {
2  /*
3   Knockout.Unobtrusive v0.2
4
5   Copyright (C)2011 Brandon Satrom, Carrot Pants Studios
6   Distributed Under MIT License
7
8   Documentation and Full license Available at:
9   http://github.com/bsatrom/knockout.unobtrusive
10
11 -----
12 Knockout.Unobtrusive
13 -----
14 */
15 var __hasProp = Object.prototype.hasOwnProperty;
16 ko.unobtrusive = {
17   createBindings: function(bindings) {
18     var bindingKey, bindingValue, createElementBinding, customKey, customValue, getElement, getElementsByClassName, k
19     if (!bindings) {
20       bindings = ko.unobtrusive.bindings;
21     }
22     getElement = function(id) {
23       var el;
24       el = document.getElementById(id) || document.getElementsByName(id)[0];
25       if (!el) {
26         id = id.charAt(0).toUpperCase() + id.slice(1);
27         el = document.getElementById(id) || document.getElementsByName(id)[0];
28       }
29       return el;
30     };
31     getElementsByClassName = function(className) {
32       var all, element, expr, match, _i, _len;
33       if (document.getElementsByClassName) {
34         document.getElementsByClassName(className);
35       }
36     };
37   }
38 };
39
40 // Exports
41 window.KnockoutUnobtrusive = ko.unobtrusive;
42
43 // Exports for AMD
44 define("knockout/unobtrusive", ["ko"], function(ko) {
45   return ko.unobtrusive;
46 });
47
48 // Exports for CommonJS
49 if (typeof module !== "undefined" && module.exports) {
50   module.exports = ko.unobtrusive;
51 }
```

Errors and Debugging: it's still JavaScript in there

CON

Conjecture!

As fast or faster JavaScript

PRO

Code Generation(?)!

Dare I say it? Code Generation!

CON

Shirts are for
n00bs...



Like all software is...

OK, COFFEESCRIPT IS OPINIONATED

WITH
APOLOGIES TO
THE NINJAS

JS NINJA, READY TO PWN THE INTERNET



TROLLS ON REDDIT FOR
NEXT 4 HOURS

The Five Stages of JavaScript

1. Avoidance
2. Concession
3. Abstraction
4. Failure
5. Enlightenment

```
arr = new Array(4, 8, 15, 16);
arr[4] = 23;
arr[5] = 42;
for (i = 0; i < arr.length; i++)
{
    enterLottoNum(arr[i]);
}
```

```
var arr = [4, 8, 15, 16, 23, 42];
for (var i = 0; i < arr.length;
      i++) {
    enterLottoNum(arr[i]);
}
```

```
var arr, num, _i, _len;
arr = [4, 8, 15, 16, 23, 42];
for (_i = 0, _len = arr.length;
     _i < _len; _i++) {
    num = arr[_i];
    enterLottoNum(num);
}
```

CoffeeScript made me want to be
a better ~~man~~

JavaScript
Programmer

*CoffeeScript can help you become a
better JavaScript programmer*

CoffeeScript Axiom #2



But there's
nothing wrong
with JavaScript!



O RLY?

Function Declaration!

```
let name = "block scoped!"  
const REALLY = "srsly"  
#add(a, b) { a + b }  
  
module Mod {  
    export const K = "foo"  
    export #add(x, y) { x + y }  
}
```

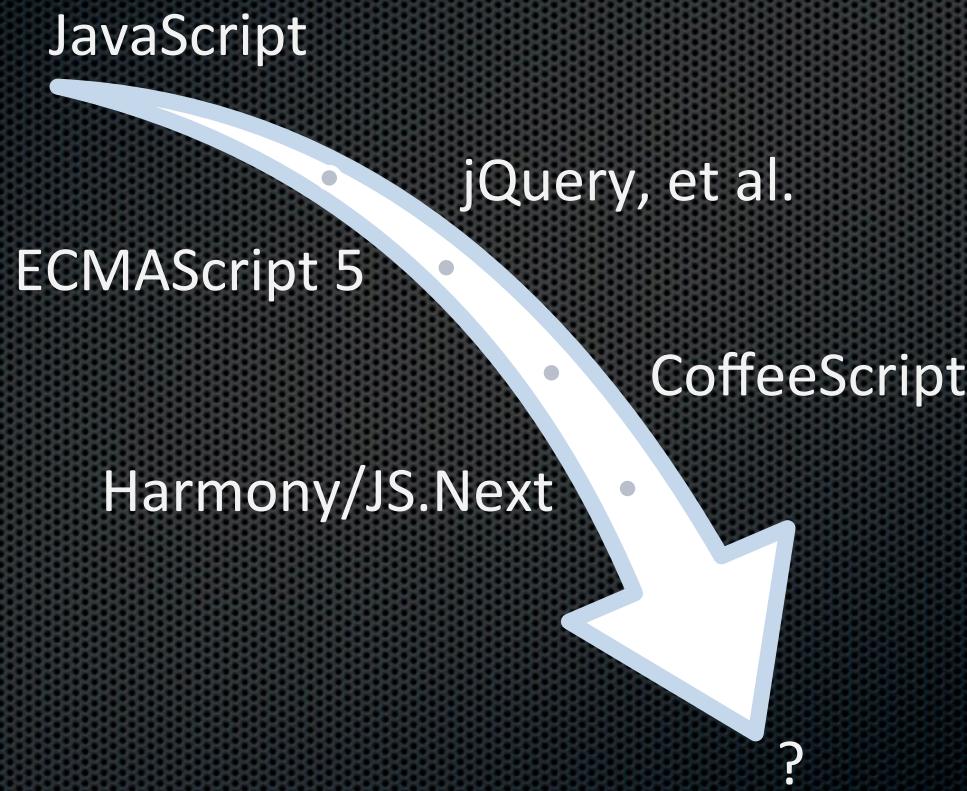
INFLUENCING JS.NEXT

“The only thing that happens when new languages pop up is that JavaScript gets better.”

– Chris Williams

WHITHER JAVASCRIPT?

JavaScript Gets Better



"We are not just advocating a language on top of JS that you use to *avoid* JS (GWT, Haxe, Objective-J, etc.). We are advocating that you all help *build a better JS on JS*, which then becomes standardized as JS.next."

- Brendan Eich



CoffeeScript Axiom #3

IT'S NOT A REPLACEMENT
FOR JAVASCRIPT

IV, V, VI, I, II, III.
It's just good
parenting.



CodeMash. Just for geeks.

CoffeeScript is a JavaScript *Polyfill*

CoffeeScript Axiom #4, Special
CodeMash Edition!



What Can I/We/Everyone Learn from CoffeeScript?

THE JAVASCRIPT DIALECTIC, REVISITED



- 1) It's not a replacement for *learning* JavaScript
- 2) It *can* help you become a better *JavaScript* programmer
- 3) It's not a replacement *for* JavaScript
- 4) It can be a *Polyfill* for JavaScript

THOSE AXIOMS

Resources - [http://bit.ly/...](http://bit.ly/)

✓ Slides - [pLWIMf](#)

✓ CoffeeScript Website - [nfWYwn](#)

✓ Annotated Source - [qdAXfn](#)

✓ CoffeeScript Koans - [o1K6bL](#)

✓ Smooth CoffeeScript Book - [nI7GEU](#)

✓ Jeremy & Brendan @ JSConf 2011 - [o9KieZ](#)

✓ “Harmony of my Dreams” - [mQkdtp](#)

@BrandonSatrom
brsatrom@microsoft.com
UserInExperience.com

QUESTIONS?



ps10g (ps injugr)