# CoffeeScript is For Closers

Brandon Satrom
@BrandonSatrom
UserInExperience.com

Slides: bit.ly/pLWIMf


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
github.com/bsatrom

# #html5tx

# Agenda

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


VS.

# A JavaScript Dialectic

# Don't Be A JavaScriptster

# Don't Be A CoffeeScriptster

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

**LIAR!**

Compiled Languages

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

# 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 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 | *no equivalent* |

# 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

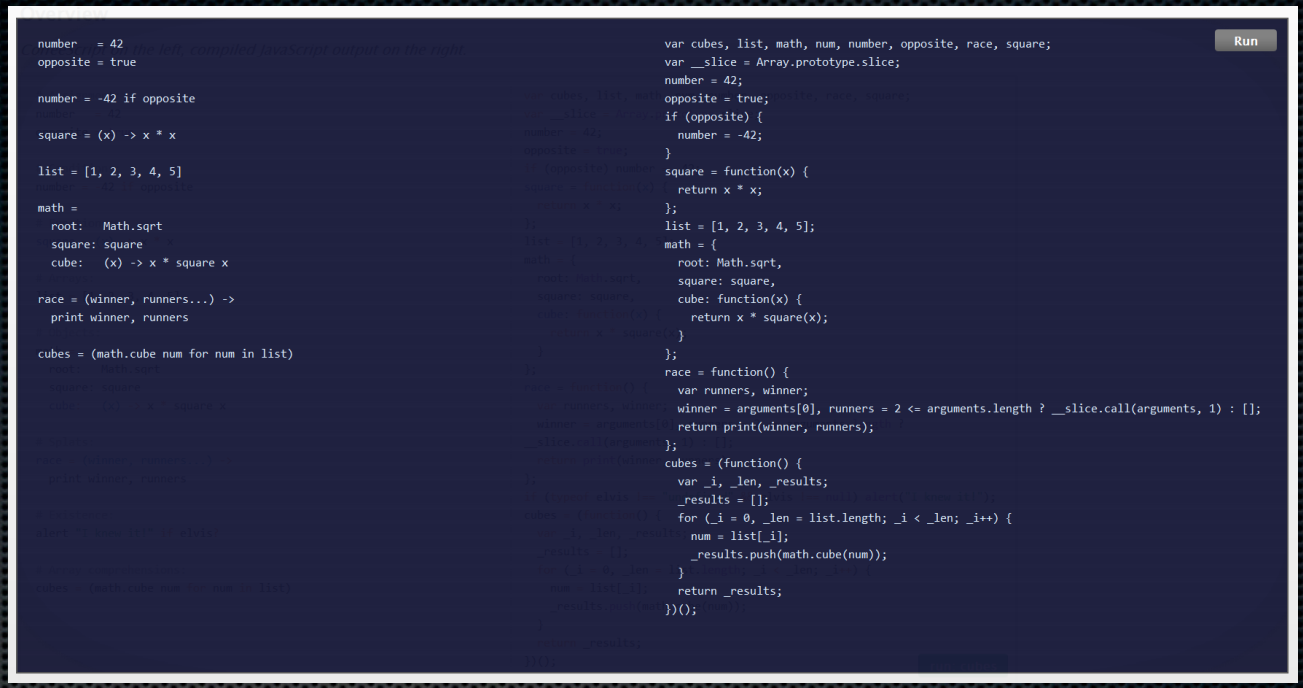
```
invoke 'build'
invoke 'tests'
```

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?



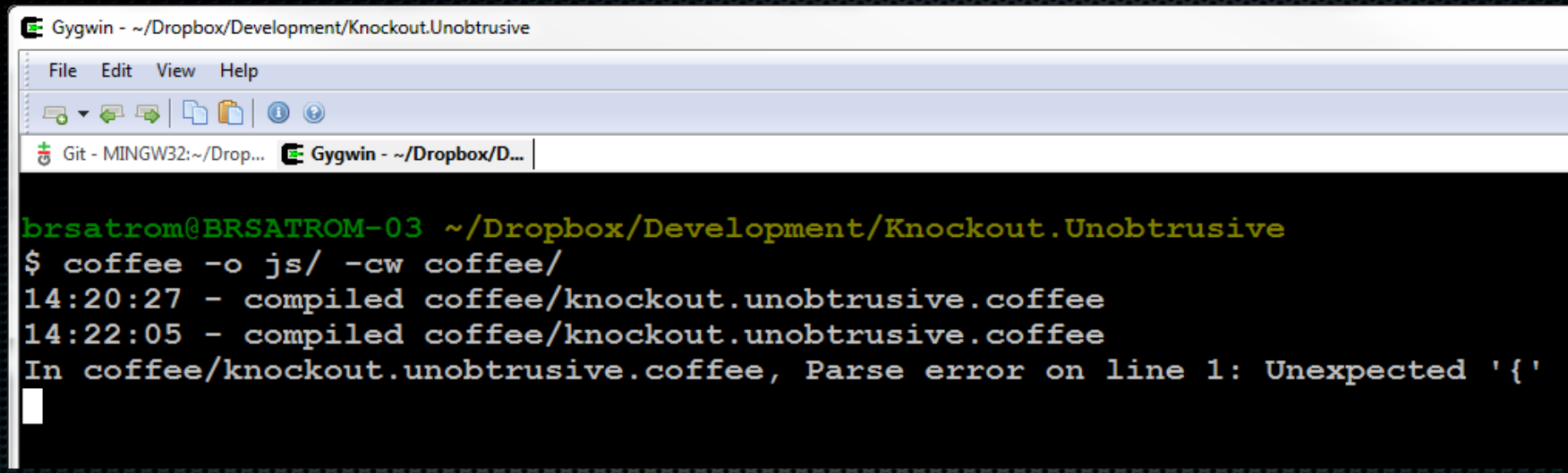
```
number   = 42
opposite = true

number = -42 if opposite

square = (x) -> x * x

list = [1, 2, 3, 4, 5]

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)
```

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

Run

# More succinct code

# PRO




Extra compile step between you and the “real” code

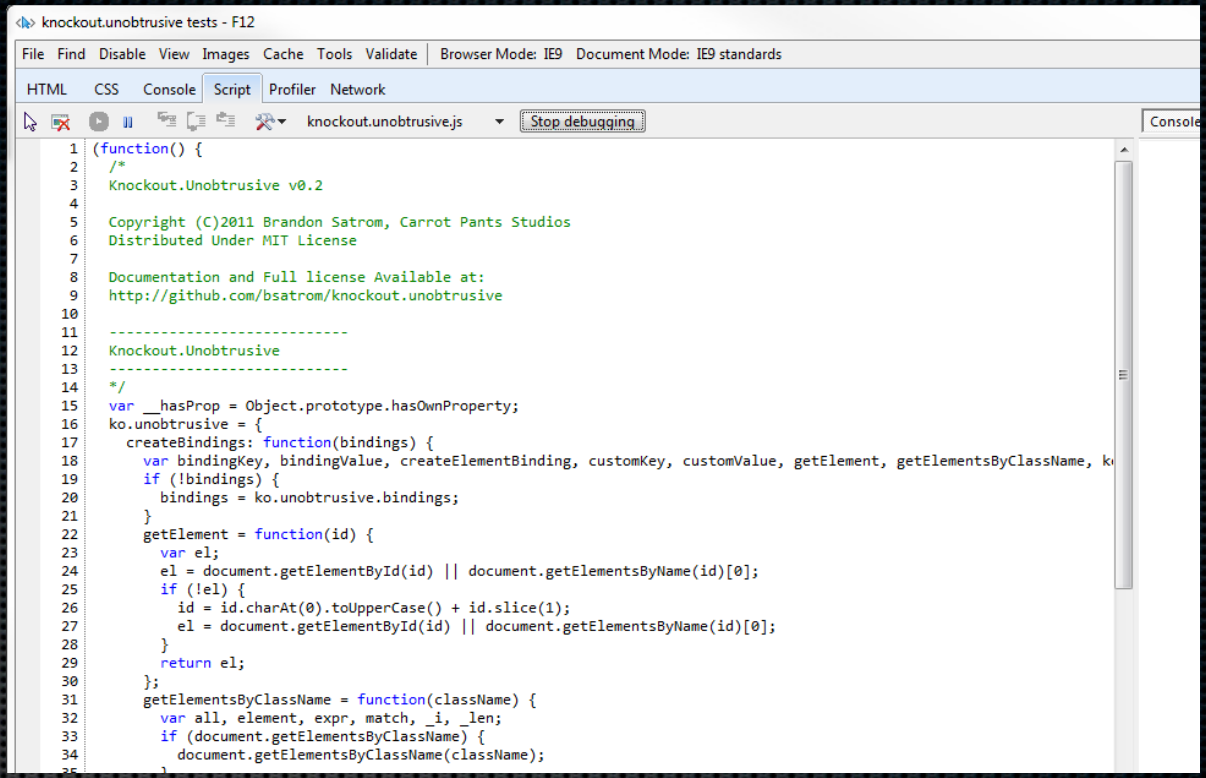
# CON

```
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
```

```
(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);
  }
})();
```

Avoid Snake Pits & Get JS Patterns for Free

# PRO




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


Like all software is...

# OK, COFFEESCRIPT *IS* OPINIONATED

WITH APOLOGIES TO THE NINJAS

# 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!
JS


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 }
}
```

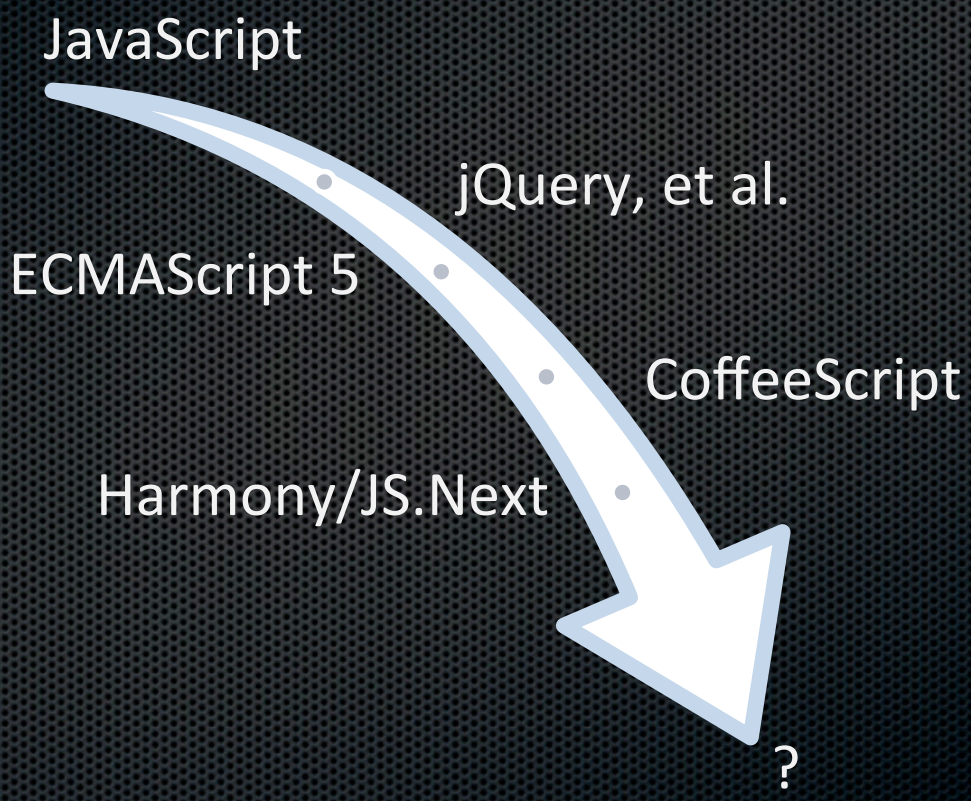
# INFLUENCING JS.NEXT

"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


# 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/...

- ✓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?