Coffeescriptease

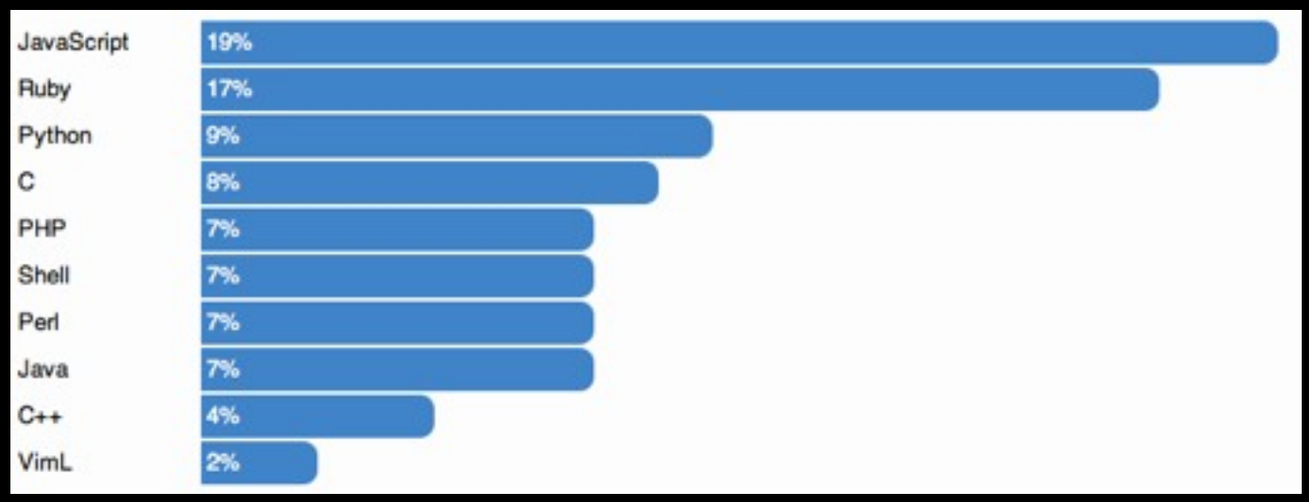
An intimate introduction

The problem with Javascript

- Hacked together in 10 days in 1995
- Bastard child of C, Scheme and Self
- Misnamed as a marketing ploy
- Lots of language level mistakes
- Impossible to get fix now.

Javascript renaissance

- Fastest mainstream dynamic language
- It's everywhere
- It's actually extremely powerful once you get used to the quirks
- How can we make it more fun?



Hello Coffeescript

- First released Christmas 2009
- 1.0 Christmas 2010
- Described as "Javascript's less ostentatious kid brother"
- Quickly became one of the most watched projects on github
- Coffeescript support built into rails 3.1
- 23rd most popular language on github
- Why?

- All the power of Javascript is still there. There is nothing you can do in Javascript you can't do Coffeescript
- Really easy to learn. Syntax you expect to work, *just works* (coming from Ruby, Perl, Python)
- Great documentation (generated by docco)
- Toolchain JFDI
- Stable
- Good source readability means fewer bugs
- Hiding bad language features means fewer bugs (global variables, comparison casting, etc)
- Great IDE support

Installing

- npm - github.com/isaacs/npm
- npm install coffee-script
- homebrew - github.com/mxcl/homebrew
- brew install coffee-script

What's in the box

```
$coffee --help

Usage: coffee [options] path/to/script.coffee

  -c, --compile      compile to JavaScript and save as .js files
  -i, --interactive  run an interactive CoffeeScript REPL
  -o, --output       set the directory for compiled JavaScript
  -j, --join         concatenate the scripts before compiling
  -w, --watch        watch scripts for changes, and recompile
  -p, --print        print the compiled JavaScript to stdout
  -l, --lint         pipe the compiled JavaScript through JavaScript Lint
  -s, --stdio        listen for and compile scripts over stdio
  -e, --eval         compile a string from the command line
  -r, --require      require a library before executing your script
  -b, --bare         compile without the top-level function wrapper
  -t, --tokens       print the tokens that the lexer produces
  -n, --nodes        print the parse tree that Jison produces
      --nodejs       pass options through to the "node" binary
  -v, --version      display CoffeeScript version
  -h, --help         display this help message
```

What should be in the box

```
$jitter

Jitter takes a directory of *.coffee files and recursively compiles
them to *.js files, preserving the original directory structure.

Jitter also watches for changes and automatically recompiles as
needed. It even detects new files, unlike the coffee utility.

If passed a test directory, it will run each test through node on
each change.

Usage:
  jitter coffee-path js-path [test-path]

Available options:
  -b, --bare          compile without the top-level function wrapper
```

github.com/TrevorBurnham/Jitter

Candy

- Language
- Variables
- Objects, Arrays
- Strings
- Functions
- Classes
- See docs for many more delicious treats

- Can omit parentheses
- Whitespace is significant
- No semi-colons (unless you want multiple statements on one line)
- Implicit parentheses don't close until the end of the expression
- I use parentheses on everything but outermost function to avoid confusion
- Curly braces only used for declaring JSON style objects (even then optionally)
- Comment is #
- Everything is an expression

Variables

- Coffeescript always ensures variables are properly declared within lexical scope. Identical to Ruby, Perl etc
- Declared at top of scope so you can do cool things like

```
six = (one = 1) + (two = 2) + (three = 3)

becomes

var one,six,three,two;
six = (one = 1) + (two = 2) + (three = 3);
```

- self = @

Object / Array

- YAML style creation
- Slicing / splicing (.., ...)

```
object =
   a: 1
   b: 2
   more:
     c: 3
     d: 4

array = [
  1,2,3
  4,5,6
  7,8,9
]

or traditional

object = { a: 1, b: 2 }
array = [‘a’,’b’,’c’]

numbers = [0 .. 10]

numbers[1..3] = [‘a’,’b’,’c’]

lots more examples in docs.
```

Iterating objects / arrays

```
object =
  a: 1
  b: 2

for key, value of object
  console.log key

hasOwnProperty? no problem...

for own key, value of object

array = [1 .. 10]
for value in array
   console.log value

alternate style

console.log key for key of object
console.log value for value in array
```

Strings

```
alert(“My name is “ + name + “ nice to meet you”);

vs

alert ”My name is #{name} nice to meet you”

alert ”My name is #{name.toUpperCase()} nice to meet you”
```

Heredocs

```
’’’
any
  indented
    formatting but no interpolation
’’’

”””
with
   interpolation #{name}
”””
```

Functions

```
var fn = function (x,y,z) {
  return x * y * z;
}
```

```
fn = (x,y,z) -> { x * y * z }

or

fn = (x,y,z) ->
  x * y * z
```

```
var fn = function (x,y,z) {
  if( x == null) {
    x = 5;
  }
  return x * y * z;
 }
```

```
fn = (x=5,y,z) -> { x * y * z }
```

Even provide defaults or add to context (this)

```
fn = (x = 50, @y) ->
```

```
var fn = function(x,y) {
  if( x == null) x = 50;
  this.y = y
  ...
}
```

Classes

- No libraries required
- Easy to superclass,
- Easy to add functions

```
class Point
  x:50
  y:50
  clear: ->
    @x = @y = null
class Point3D extends Point
  z:50
  clear: ->
    @z = null
    super
```

Contextual functions

- => is awesome
- no more that = this
- functions created with it will always run in the context (this) they were created
- Great for jQuery

```
Point = (@x,@y) ->
   $('document').click, (ev) =>
      @x = ...
```

Don’t worry about it...

hasOwnProperty	for own key, value of object
function context	=>
global namespace pollution	outputs in safety wrapper
global variables	not allowed (must be namespaced)
un-optimised defaults	optimizations compiled in eg list.length etc are factored out
===	== or ‘is’
‘arguments’	provides easy way to handle variable amount of arguments with ... (splat), even with named arguments. Avoids having to know it’s not a real array

Debugging

- Coffeescript output is very readable
- Some solutions in pipeline
- I've not really had to do it very often.
- Jitter + growl is a lifesaver

Discussion at github.com/jashkenas/coffee-script/issues/558

Lets make something

- Browser project
- Coffeescript
- With jQuery, Backbone events
- This is probably not going to work...

Further Reading

- jashkenas.github.com/coffee-script/
- pragprog.com/titles/tbcoffee/coffeescript
- rosettacode.org/wiki/Category:CoffeeScript