CoffeeScript in Rails

Devpoint Labs - Jake Sorce / Dave Jungst

What is CoffeeScript?

- CoffeeScript is a language that compiles into JavaScript

- The golden rule of CoffeeScript is: "It's just JavaScript"

Why do I want to use CoffeeScript?

- CoffeeScript is an attempt to expose the good parts of JavaScript in a simple way.
- Compiles one-to-one into the equivalent JS
- You can use any existing JavaScript library seamlessly from CoffeeScript (and vice-versa)
- Compiled output is readable and pretty-printed, will work in every JavaScript runtime
- Tends to run as fast or faster than the equivalent handwritten JavaScript.
- CoffeeScript is more readable and developer friendly than its JavaScript counterpart.
- WATCH YOUR INDENTATION!

How Do I Use CoffeeScript?

- The CoffeeScript Gem is installed by default in a new rails project.
- Any controller that you generate with rails will have a CoffeeScript file generated automatically for that controller.
- There is no setup, installation or configuration, just start writing!

Biggest Differences Between CoffeeScript and JavaScript

- no more declaring var
- no more semicolons
- no more squiggly brackets
- no more using the function keyword
- a lot less parenthesis
- a lot less syntax in general
- a lot less code to write to accomplish the same thing

Basic Functions

JavaScript:

- function doSometing() {
  console.log(“Doing something”);
  }

CoffeeScript:

- doSomething = ->
  console.log “Doing Something”

Functions With Params

JavaScript:

- function doMoreThings(word){
  console.log(word);
  }

CoffeeScript:

- doMoreThings = (word) ->
  console.log word

Defaulting Function Params

JavaScript:

- function defaultedFunction(firstArg, defaultedArg){
  if(typeof defaultedArg == “undefined”){
  defaultedArg = ‘the default string’;
  }
  console.log(defaultedArg);
  }

CoffeeScript:

- defaultedFunction = (firstArg, defaultedArg = ‘the default string’) ->
  console.log defaultedArg

Arrays

JavaScript:

- var myArray = [1, 2, 3, 4, 5];

CoffeeScript:

- myArray = [1, 2, 3, 4, 5]

Objects

JavaScript:

- var myObject = {
  vehicles: {
  truck: {
  make: ‘Chevy’,
  model: ‘Silverado’
  },
  atv: {
  make: ‘yahama’,
  model: ‘YFZ450’
  }
  };

Objects Cont...

CoffeeScript:

- myObject =
  vehicles:
  truck:
  make: ‘chevy’
  model: ‘silverado’
  atv:
  make: ‘yahama’
  model: ‘YFZ450’

If, Else, Unless, and Conditionals

Basic If Else:

- if morning and tired
    drinkCoffee()
    wakeUp()
  else
    suckItUp()

If, Else, Unless, and Conditionals

Reverse If:

- drinkCoffee() if morning

Unless statement:

- makeMoreCoffee unless coffeeFull()

If, Else, Unless, and Conditionals

Ternary Operator:

- if car.started then "go to school" else "go to bed"

Loops and Comprehensions

- Most of the loops you'll write in CoffeeScript will be comprehensions

- Comprehensions replace (and compile into) for loops

Loops and Comprehensions

```
eat = (food) ->
  console.log(“Ate #{food}”)

foods = ['broccoli', 'spinach', 'chocolate']

eat(food) for food in foods when food isnt 'chocolate'

countdown = (num for num in [10..1])
```

Loops and Comprehensions

```
list = [‘first.jpg’, ‘song.mp3’, ‘jt.mp3’]

for filename in list
  console.log(filename)
```

More CoffeeScript VS. JavaScript

CoffeeScript	JavaScript
is	===
isnt	!==
not	!
and	&&
or	||
true, yes, on	true
false, no, off	false
@, this	this
of	in
in	no JS equivalent

MOAR Examples

launch() if ignition is on

volume = 10 if band isnt SpinalTap

if car.speed < limit then accelerate()

winner = yes if pick in [47, 92, 13]

Classes

```
class Animal
  constructor: (@name) ->

  move: (meters) ->
    alert @name + " moved #{meters}m."

class Snake extends Animal
  move: ->
    alert "Slithering..."
    super 5

sam = new Snake "Sammy the Python"
sam.move()
```

CoffeeScript Project

Basic Goals:

1. Create a new rails project
2. Uncomment the welcome#index root in the routes file
3. Generate the welcome controller with the index action
4. Build out a tic tac toe game in the welcome.coffee file and the index.html.erb file

Bonus Goals:

1. Use a class or 2 in your CoffeeScript to make the tic tac toe game more organized and OOP like
2. Style your tic tac toe game all pretty like
3. At the end of your game make an ajax call to the server to save the number of times the computer won or the user won and print that out on your index page when it loads
4. Use HAML instead of ERB and don't just use the HTML to HAML converter. *(I will know)*

Resources

- http://coffeescript.org/
- http://js2.coffee/
- http://blog.teamtreehouse.com/the-absolute-beginners-guide-to-coffeescript