Coffee and Cake

Florian Salihovic visits the Flex User Group in Hamburg, 05/22/12

$ Whoami

- Florian Salihovic
- Flex developer, primarily
- Freelancer for about 4 years as a Flex dev
- http://icodeapps.net
- Not good with slides, presentations [...]

ls -la

- Get CoffeeScript
- Why CoffeeScript?
- What is CoffeeScript?
- A little piece of Cake
- Discussion?

Get some Coffee

- Install Node.js and NPM
- npm install coffee-script -g (as admin)
- Use package.json and add a dependency (@example)
- Install it without node works as well, but you don’t want that.

Why CoffeeScript

- it might be simpler
- it might be more productive
- it is cool

What is CoffeeScript?

- "CoffeeScript is a little language, that compiles into JavaScript." http://coffeescript.org/
- It is just JavaScript - in the end
- It offers a lot
- It avoids a bit

We know JavaScript

- Scope shifts
- No OOP (sort of)
- Operators lost their sanity
- We go global if everything fails
- Those facts are known as the bad parts

CoffeeScript for the rescue

- No more scope shifts
- Yes, you're right
- The operators work as expected
- It tries to hide the bad parts

Syntax

- concise
- pythonish
- no brackets involved for blocks
- some call it functional
- a lot of sugar, so sweet

FizzBuzz: let's do it!

- A function:

```
fizzbuzz = ->
  # some code
```

- Equals

```
var fizzbuzz = function() { // some
code }
```

Loops

- Getting the number from 1 to 100

```
for num in [1..100]
```

- This generates the whole for loop code for us
- Apply filter and invoke methods in the same line

```
add num for num in [1..100] when num
```

Everything is an expression

- so is for
- if else and the switch
- try catch finally
- this is so functional, isn't it?
- beware of side effects (console.log might break your code -> no, it will)

The Code

```
fizzbuzz = ->

  words = for num in [1..100]

    if num % 15 is 0
      'FizzBuzz'
    else if num % 5 is 0
      'Buzz'
    else if num % 3 is 0
      'Fizz'
    else
      num
```

I promised OOP

- We get classes
- No real namespaces
- We get inheritance
- But no polymorphism (no static typing)

A Person

- ```
  class Person

      constructor: (@name) ->

          sayHello: ->

              "#{@name} is my name"
  ```

- Btw: string interpolation only works with double quotes.
  ```

A loud person

- ```
  class LoudPerson extends
  Person
    sayHello: ->
      super().toUpperCase()
  ```
  ```

Building CoffeeScript

- Use the command line tool/*repl* `$ coffee`
- Compile it via `$ coffee -c fizzbuzz.coffee`
- Automate with *Cake* (which ships with the module)

Writing a Cakefile

- Name: Cakefile
- Objective: implement tasks
- Automate every day things

Baking Coffee with Cake

- ```
  {print} = require 'util'

  {spawn} = require 'child_process'

task 'compile', 'Compiling CoffeeScript to JavaScript.', (options) ->
    console.log "Executing task: #{@name}"
     spawnProcess 'coffee', ['-c', '-o', 'public/javascript', 'src/
coffee'], =>
        console.log "Finished task: #{@name}"

spawnProcess = (name, args, callback) ->
  spawnedProcess = spawn name, args
  spawnedProcess.stderr.on 'data', (data) ->
    print data.toString()
  spawnedProcess.stdout.on 'data', (data) ->
    print data.toString()
  spawnedProcess.on 'exit', (code) ->
    callback?() if code is 0
Just look on the actual code - this is a mess :(
  ```
  ```

Questions? Discussion!

- That's all regarding CoffeeScript ... well ... there are more things to learn.
- Questions!
- Visit http://coffeescript.org for more infos and examples!