CoffeeScript

Arne Martin Aurlien
Boost Communications

Hei og velkommen etc.

/me

- twitter.com/arnemart
- github.com/arnemart
- am.aurlien.net
- episkebryggeri.no

Hvem er jeg:
- Utvikler i Boost Communications
- Backend (PHP, dessverre) og frontend
- Premiert hjemmebrygger med noe over snittet interesse for øl (noe som muligens blir tydelig underveis i presentasjonen)
- Relativt god kontroll på JS og ca. en uke erfaring med CS
- Still for all del spørsmål underveis og arrester meg ikke hvis, men når, jeg driter meg ut.

WTF CoffeeScript

coffeescript.org
github.com/jashkenas/coffee-script

Først: Jeg har ganske lite erfaring med å faktisk CS, men det har stått på radaren min siden det var nytt.

- Hvor mange har brukt CS?
    - I produksjon?

Hva er CoffeeScript?
- CS er et språk som kompilerer til JS
- Laget av Jeremy Ashkenas, utviklingen startet i 2009
- Syntaks inspirert av f.eks Python og Ruby

Hvorfor CoffeeScript?
- JS er vakkert, men har mange alderdomstegn
- Mange enkle operasjoner krever mye overflødig “boilerplate”-kode
- Koden kan bli betraktelig kortere

Hvem burde bruke CS
- Erfaring med JS
- Foretrekker (ruby|python|?)

Hvem burde ikke bruke CS
- “Jeg har veldig lite erfaring med JS, men jeg har brukt jQuery i tre år”

Alternativer

- JavaScript
- Dart
- ClojureScript
- Objective-J
- GWT

Alternativer:
- Ren JS
- Dart: Hvis du liker å late som du skriver java. 17000 linjer hello world.
- ClojureScript: Hvis du liker lisp/clojure
- Objective-J: Hvis du bruker Cappucino, eller hvis du er ekstremt glad i objective-c
- GWT: Hvis du heller vil skrive java

"It's just JavaScript"

Hovedfilosofi: "Det er bare JavaScript!"
Målet med CS er å beholde semantikken i JS, men å forenkle syntakset og å lage snarveier for ofte brukte operasjoner.

Det er kort vei fra CS til JS, og CS-kompilatoren genererer ren og lesbar JS-kode, som passerer gjennom jslint/jshint uten problemer. Ytelsen er generelt svært bra, tilsvarende håndskrevet JS.

Debugging er for øyeblikket litt vanskelig, siden feilmeldinger i nettlesere vil henvise til linjenummer i JS-koden, men en teknologi kalt source maps er under utvikling som skal forbedre dette.

Hvordan

```
# Installer CoffeeScript
$ npm install -g coffee-script

# Kompiler script.coffee til script.js
$ coffee -c script.coffee

# Kompiler script.coffee hver gang filen endres
$ coffee -w script.coffee
```


Kommandolinjeversjonen av CS er implementert med node.js, og kan installeres med npm (node sin pakkebehandler)
Da blir “coffee” tilgjengelig på kommandolinjen og brukes f.eks slik som dette.
Det er også mulig å kompilere CS til JS direkte i nettleseren, dette er veldig kjekt under utvikling men bør ikke brukes i produksjon

Look ma, no var

```
foo = 'bar'

baz = "foo #{foo}"
```

```
log = (s = 'log') ->
    console.log s

even = (i for i in [1..10] when i % 2 == 0) # [ 2, 4, 6, 8, 10 ]

beers =
    IPA: 'ale'
    bitter: 'ale'
    bock: 'lager'
    pils: 'lager'
    lambic: 'spontaneous'

ales = (ale for ale, type of beers when type is 'ale') # [ 'IPA', 'bitter' ]
```


Kjapp intro til deler av syntakset i CS

1. Variabeldeklarasjon gjøres nesten som i JS — bare uten “var”. CS gjør det automatisk for deg, og passer på at variabler havner i riktig scope.
2. CS har string-interpolering, som php/ruby etc.
3. Funksjoner har mye enklere syntaks enn i JS, og argumenter kan ha standardverdier
CS bruker whitespace i stedet for krøllparenteser (ala python)
4. CS har “Comprehensions”, ala python (og lisp, f.eks loop-makroen i commonlisp) for å gjøre operasjoner på arrayer og objekter
(Legg også merke til [1..10] som lager et array med alle tallene fra 1 til 10)
5. Forenklinger i objektnotasjon: Whitespace er signifikant, krøllparenteser er valgfritt, linjeskift kan brukes i stedet for komma.
6. Comprehensions fungerer også på objekter
“is” er et alias for ==

Have you no `class`

```coffeescript
class Beer
    constructor: (@name) -> # == (name) -> @name = name
    ferment: (time) ->
        console.log "Fermenting #{@fermentationTemp} for #{time} days"

class Lager extends Beer
    fermentationTemp: 'cold'
    ferment: ->
        super 30

class Ale extends Beer
    fermentationTemp: 'warm'
    ferment: ->
        super 14

taddy = new Ale 'Taddy Porter'

taddy.ferment() # "Fermenting warm for 14 days"
```


Objektorientering er noe av det som skiller CS mest fra JS. CS har et objekt- og klassesyntaks som minner mer om tradisjonell OO og arv. Dette er i praksis bare et tynt skall oppå JS sin prototypebaserte arv, men kan endre måten du jobber med klasser i JS ganske mye.

jQuery

```
$ ->
    $('.loadMore').bind 'click', ->
        $.ajax
            url: '/loadmore',
            success: (data) ->
                $('.more').html(data).show()
```


JS-kode og CS-kode kan samarbeide uten problemer. Her er et eksempel på relativt idiomatisk jquery-kode (“spaghetti”), skrevet i coffeescript...

jQuery

```
$(function() {
  return $('.loadmore').bind('click', function() {
    return $.get({
      url: '/loadmore',
      success: function(data) {
        return $('.more').html(data).show();
      }
    });
  });
});
```


...som blir oversatt til denne javascript-koden.
Funksjoner i CS returnerer alltid verdien fra den siste operasjonen, noe som gjør at man ikke trenger å ta med “return”-statements i koden. Dette gjør at JS-koden blir full av returns, slik som her.

node.js

```
http = require 'http'

http.createServer (req, res) ->
    res.writeHead 200, 'Content-Type': 'text/plain'
    res.end 'Hello World\n'
.listen 1337, '127.0.0.1'
```


CS kan også brukes på serversiden, her er en node.js “hello world” i CS

node.js

```
var http;
```

```
http = require('http');

http.createServer(function(req, res) {
  res.writeHead(200, {
    'Content-Type': 'text/plain'
  });
  return res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
```

Let's hack

La oss skrive litt kode!

Takk for meg!

github.com/arnemart/Meetup25042012