CoffeeScript

„It‘s just JavaScript“

Agenda

1. Was ist CoffeeScript?
2. Installation
3. Integration in die IDE
4. Vorzüge von CoffeeScript
5. Basics
6. Advanced
7. Debugging
8. Beispiel
9. Webframeworks
10. Fazit

Was ist CoffeeScript?

- Seit 2009 von Jeremy Ashkenas entwickelt
- Programmiersprache, wird zu Javascript transkompiliert
  - Syntaktischer Zucker aus Python, Ruby und Haskell
  - Übersetzung des Codes ist vorhersehbar und kann daher nachvollzogen werden
  - Der CoffeeScript-Compiler ist selbst in CoffeeScript geschrieben und nutzt Node.js
- Offizieller Javascript-Präprozessor von Ruby on Rails

Installation

- Node.js wird vorausgesetzt
- npm install -g coffee-script
- coffee -c -w *.coffee
  - c Kompiliert .coffee-Dateien zu .js
  - w setzt einen Watcher auf das Filesystem und kompiliert bei jeder Änderung

Integration in die IDE

- Netbeans:
  - http://plugins.netbeans.org/plugin/39007
  - Syntaxhighlighting
  - CoffeeScript-Linter
  - Nutzt Rhino-Engine zum Kompilieren
- Eclipse:
  - https://github.com/Nodeclipse/coffeescript-eclipse
  - Noch keine Erfahrung gesammelt
- CoffeeScript-Compiler in PHP:
  - https://github.com/alxlit/coffeescript-php

Vorzüge von CoffeeScript

- Einrückung statt Klammern (analog zu Python) bei Funktionen,
- Bedingungen, Switch- und Try/Catch-Anweisungen
- Verzicht auf Parameterklammern bei Funktionsaufrufen
- Verzicht auf Semikolons
- Trailing Commas sind erlaubt
- Kompilat ist formatiert und lesbar
- Sehr gute Dokumentation mit Beispielen auf http://coffeescript.org

Basics

- funktion/object/array: http://coffeescript.org/#literals
- conditions: http://coffeescript.org/#conditionals
- loops: http://coffeescript.org/#loops
- classes: http://coffeescript.org/#classes
- embed plain old JavaScript: http://coffeescript.org/#embedded

Advanced

- array slicing: http://coffeescript.org/#slices
- operators: http://coffeescript.org/#operators
- chained comparison: http://coffeescript.org/#comparisons
- cake files: http://coffeescript.org/#cake

Debugging

- Kompilat ist vorhersehbar und kann daher leicht zurückübersetzt werden
- Durch den Einsatz von SourceMaps kann die Debug-Konsole direkt ableiten, in welcher Zeile der *.coffee-Datei der Fehler existiert:
  - http://coffeescript.org/#source-maps
  - Vorgehensweise:
    - Aufruf: coffee –m –c *.coffee
    - Aufruf im Browser und der Debug-Konsole, die HTML5-SourceMaps unterstützt (Chrome, Firebug), siehe
    - http://www.html5rocks.com/en/tutorials/developertools/sourcemaps/?ModPagespeed=noscript

Beispiel

- Beispielcode zum Experimentieren siehe Anhang
- Livecompiler zum Experimentieren: http://js2coffee.org/#coffee2js

Webframeworks

- Clientside Frameworks:
  - TowerJS: http://tower.github.io/
  - DerbyJS: http://derbyjs.com/
  - Mozart: http://www.mozart.io/
  - batman.js: http://batmanjs.org/
- Diese Frameworks sind alle auch auf Single-Page-Applikationen ausgelegt.

Fazit

- schnelle Entwicklung durch intuitive Syntax
- „lohnt“ sich vor allem bei größeren Projekten, u.a. durch „lesbare“ OOP.
- Generierter Code durch Nachvollziehbarkeit und SourceMaps gut zu debuggen
- Fokus auf dem Problem, nicht auf dem Code
- Zusammenspiel mit normalem Javascript einwandfrei

Quellen

- http://de.wikipedia.org/wiki/CoffeeScript
- http://coffeescript.org/

Vielen Dank!