Three Different Languages

ThreeLanguages

- Coffee
  - Imperative, lisp like syntax, …
- …
  - OO (smalltalk), lisp like syntax
- …
  - Functional

Coffee

- Lisp like syntax
- Imperative, non-object oriented
- Static scope, static binding, strongly typed, …

Coffee Interpreter

- Starting coffee without an input file…

```
$ coffee
coffee> _                    \\READ-EVAL-PRINT loop starts here…
```

- Starting coffee with an input file…

```
$ coffee myprogram.coffee

                             \\READ-EVAL-PRINT everything in the file…

coffee> _                    \\READ-EVAL-PRINT loop starts here…
```

- A statement does not return anything (prints "-s-")
- An expression returns either a binary, integer or integer list (prints the corresponding value, e.g. "true", "123", "(12,13,14)")

Coffee – Syntax

- Keywords: *and*, *or*, *not*, *equal*, *append*, *concat*, *set*, *deffun*, *for*, *while*, *if*, *then*, *else*, *true*, *false*
- Operators: *+, -, /, *, (,)*
- Terminals:
  - Keywords, operators, 0-9
  - BinaryValue -> true | false
  - IntegerValue -> [-]*[1-9]*[0-9]+
  - Id – [a-zA-z]+

Coffee – Syntax

- Non-terminals:
  - START, INPUT, STATEMENT, STATEMENTLIST, EXP, EXPLIST, EXPI, EXPB, …

Coffee – Syntax

- START -> INPUT
- INPUT -> STATEMENT | EXP | STATEMENTLIST | EXPLIST

Coffee – Syntax

- Lists
  - LISTVALUE -> ‘(VALUES) | ‘() | null
- VALUES -> VALUES IntegerValue | IntegerValue

Coffee – Syntax

- Expressions:
  - EXP -> EXPI | EXPB
  - EXPI -> (+ EXPI EXPI) | (- EXPI EXPI) | (* EXPI EXPI) | (/ EXPI EXPI) | Id | IntegerValue | (Id EXPLIST)
  - EXPB -> (and EXPB EXPB) | (or EXPB EXPB) | (not EXPB) | (equal EXPB EXPB) | (equal EXPI EXPI) | Id | BinaryValue
  - EXPLIST -> (concat EXPLIST EXPLIST) | (append EXPI EXPLIST) | LISTVALUE | null

Coffee – Syntax

- EXPLIST -> (EXPLISTELEMENTS)
- EXPLISTELEMENTS -> EXPI | EXPLISTELEMENTS EXPI | null

Coffee – Syntax

- IDLIST -> (IDLISTELEMENTS)
- IDLISTELEMENTS -> null | Id | IDLISTELEMENTS Id

Coffee – Syntax

- Assignment:
  - STATEMENT -> (set Id EXP)
  - Imperative, therefore EXP will be evaluated first…

Coffee – Syntax

- Functions:
  - Definition:
    - STATEMENT -> (deffun Id IDLIST EXPLIST)
  - Call:
    - EXP -> (Id EXPLIST)
  - Parameter passing by value
  - Returning the value of the last expression

Coffee – Syntax

- Control Statements:
  - STATEMENT -> (if EXPB then EXPLIST else EXPLIST)
  - STATEMENT -> (if EXPB EXPLIST)
  - EXP -> (if EXPB EXPLIST EXPLIST)
  - STATEMENT -> (while (EXPB) EXPLIST)
  - STATEMENT -> (for (Id EXPI EXPI) EXPLIST)

Note: A statement does not return any value…

Coffee – Variables

- STATEMENT -> (defvar Id EXP) // defining a variable
- STATEMENT -> (set Id EXP) // setting a variable
  - Scope:
    - Static, lexical scope (shadowing)
  - Binding:
    - Static binding
  - Typing:
    - Strong typing…

Programming in Coffee

```
(deffun sumup (x)
        (if (equal x 0)
                then 1
                else (+ x (sumup (- x 1)))
        ))

(sumup 8)
```