Coffee

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

Coffee Interpreter

- Starting coffee without an input file…

  $ coffee

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

- Starting coffee with an input file…

  ```
  $ coffee myprogram.coffee
  ```

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

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

- 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, EXPLISTI, EXPI, EXPB, …`

Coffee – Syntax

- START -> INPUT
- INPUT -> EXPI | EXPLISTI

Coffee – Syntax

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

Coffee – Syntax

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

Coffee – Syntax

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

Coffee – Syntax

- Functions:
  - Definition:
    - `EXPI -> (deffun Id IDLIST EXPLISTI)`
  - Call:
    - `EXPI -> (Id EXPLISTI)`
  - Parameter passing by value
  - Returning the value of the last expression
  - *Note that function definition always returns 0*

Coffee – Syntax

- Control Statements:
  - `EXPI -> (if EXPB then EXPLISTI else EXPLISTI)`
  - `EXPI -> (if EXPB EXPLISTI)`
  - `EXPI -> (if EXPB EXPLISTI EXPLISTI)`
  - `EXPI -> (while (EXPB)  EXPLISTI)`
  - `EXPI -> (for (Id EXPI EXPI) EXPLISTI)`

Coffee – Variables

- `EXPI -> (defvar Id EXPI)` `// defining a variable`
- `EXPI -> (set Id EXPI)` `// 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)
```