Hot Cocoa Lisp: a JavaScript Lisp

Sam Auciello
Marlboro College

JavaScript in Scheme's clothing

JavaScript has been described as Scheme in C's clothing. The language natively provides many lisp-like tools such as annonamous functions with full closures. However, due to it's C-like structure, JavaScript isn't really ideal for functional programming. Hot Cocoa Lisp is an attempt to make a transcompiling language like CoffeeScript that looks and feels like Lisp and encourages a functional style.

Transcompiling

As with CoffeeScript, Hot Cocoa Lisp is compiled into, and shares many features with, JavaScript. The compiler is written in JavaScript and designed to be run at the command line using Node.js.

Basic Usage

```
;; hello.hcl
(console.log "Hello World!")

$ hcl hello.hcl
$ cat hello.js
// compiled from Hot Cocoa Lisp

// (console.log "Hello World!")

console.log("Hello World!");

$ node hello.js
Hello World!
```

Annotations

Hot Cocoal Lisp provides annotations in its compiled output to aid debugging. For example:

```
;; choose.hcl

;; choose function
(def choose
  (# (m n)
     (if (or (= 0 n) (= m n)) 1
       (+ (choose (--1 m) n)
          (choose (--1 m) (--1 n))))))

// choose.js

// ;; choose function
// (def choose
//   (# (m n)
//      (if (or (= 0 n) (= m n)) 1
//        (+ (choose (--1 m) n)
//           (choose (--1 m) (--1 n))))))

choose = (function(m, n) {  return ((((0
=== n)) || ((m === n))) ? 1 : (choose((m -
1), n) + choose((m - 1), (n - 1)))); });
```

This allows the code to follow a purely functional style while keeping the compiled Javascript relatively legible.

Examples

```
;; if, when, and cond work as they do
;; in Common Lisp:

(when (condition) (console.log "yes"))
(if (condition) "yes" "no")
(cond
  ((test1) result1)
  ((test2) result2)
  (true default))

;; Functions in HCL can be defined
;; using the '#' symbol which is
;; simply a shorter spelling of lambda

(def my-func (# (a b) (* a (+1 b))))

;; Iteration can be done using times:

(times (x 5) (console.log x))

;; the above outputs "0 1 2 3 4"

;; Python-style for loops are also
;; provided:

(for (word [ "one" "two" "three" ] )
  (console.log word))

;; Note that the list literal here
;; doesn't require commas
```