RLisp
Lisp naturally embedded in Ruby

Tomasz Węgrzanowski
Tomasz.Wegrzanowski@gmail.com
http://t-a-w.blogspot.com/

What is RLisp ?

- It's Lisp
  - with full access to Ruby runtime
- It's Ruby
  - with macros
- It's both at the same time
- It's a fun toy

What is Lisp ?

```
(+ 1 2)

(lambda (x) (* x 2))

(defmacro  defun (name . body)
    `(let ,name (fn ,@body))
)
```

- Dialects of Lisp have simple core
  - and usually a lot on top of it
- Scheme, Common Lisp, Arc, Emacs Lisp, RLisp

Why Lisp is different

- Data and code have the same representation
- Code can be handled just like data
  - Generated
  - Analyzed
  - Transformed
- “Code as string” does not scale

Lisp sucks

- Ancient and weird (Common Lisp)
- Few libraries (Scheme)
- Vapourware (Arc)
- Nonstandard OO (CLOS)
- Little integration with Unix (all of them)

Ruby sucks

- Limited metaprogramming
  - Difficult
  - Fragile
  - No macros
- Complex non-orthogonal core
  - How many ways to define a function ?
    - All slightly different
  - Too many different node types in AST
    - Cannot define your own

Functions in Ruby

- `x="foo"; Proc.new{|a,*b| a}.call(x)`
  - => `"foo"`
- `x=[1,2]; Proc.new{|a,*b| a}.call(x)`
  - => `1`
- `Proc.new` vs `lambda`
- `def` vs `define_method`
- Ruby is not just small core + syntactic sugar

Ruby + Lisp

- RLisp tries to take best of both worlds
  - Ruby standard library
  - Ruby object model
  - Lisp code as data
  - Lisp macros

Compromises

- Semantics of Lisp and Ruby differ
- RLisp `'(foo bar)` is Ruby `Array [:foo, :bar]`, not linked list
- `[Foo bar 1 2 3]` - syntactic sugar for `(send Foo 'bar 1 2 3)`
- `[5 times & (fn (i) (print i))]`
- Ruby-style local variables

Example

```
(defun adder (n)
  (fn (x) (+ x n))
)
(let add5 (adder 5))
(print (add5 10))
```

```
def adder(n)
  Proc.new{|x| x+n}
end
add5 = adder(5)
puts add5.call(10)
```

Lisp with Ruby libraries

```
(ruby-eval "require 'webrick'")
(let http-server [WEBrick const_get 'HTTPServer])
(let config [Hash new])
[config set 'Port 1337]
(let server [http-server new config])
[server mount_proc "/hello" &
    (fn (req res)
        [res body= "<html><body><h3>Hello,
  world!</h3></body></html>"]
        [res field_set "Content-Type" "text/html"]
    )
]
[server start]
```

Lisp macros

```
(let Foo [Class new])
(class Foo
  (attr_accessor 'x 'y)
  (method initialize (x y)
    [self x= x]
    [self y= y]
  )
  (method to_s ()
    (+ "<" [[self x] to_s] "," [[self y] to_s] ">")
  )
)
[[Foo new 1 2] to_s] ; "<1,2>"
```

Rlisp OO sugar

```
(defmacro class (name . body)
    `[,name instance_eval & (fn () ,@body)]
)

(defmacro method (name args . body)
    `[self define_method ',name & (fn ,args ,@body)]
)

(defmacro attr_accessor args `[self attr_accessor ,@args])
```

Sugar-free

```
(send Foo 'instance_eval & (fn ()
  (send self 'attr_accessor 'x 'y)
  (send self 'define_method 'initialize & (fn (x y)
    (send self 'x= x)
    (send self 'y= y)
  ))
  (send self 'define_method 'to_s & (fn ()
    (+ "<" (send (send self 'x) 'to_s) ","
           (send (send self 'y) 'to_s) ">")
  ))
))
```

XSS Protection

```
(print-html
  (html
    (head (title "<<< Ruby & Lisp united >>>"))
    (body
      (p "<>& are safe by default."
        (html-raw
          "Unsafe only when explicitly requested &excl;"
        )
      )
    )
  )
)
```

My blog

http://t-a-w.blogspot.com/