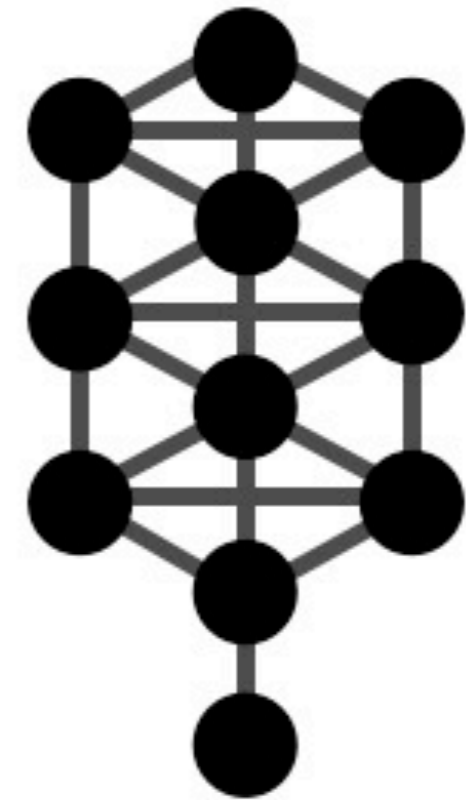


Seph



Ola Bini

computational metalinguist

[ola.bini@gmail.com](mailto:ola.bini@gmail.com)

<http://olabini.com/blog>

# Ola Bini

Swedish language geek

Works for ThoughtWorks in Chicago

JRuby core committer

Designer of Ioke and Sesh

Member of JSR292 Expert Group

# Language design

Communication

Expressiveness

Simplicity

Homoiconicity

Language flexibility

# Expressiveness

A precursor: loke

# The position of loke

# Some examples

0 fact = 1

Number fact = method(self \* (self - 1) fact)

10 fact println

"Command line arguments:" println  
System programArguments each(println)

```
fib = method(  
  fn(a, b, [b, a + b]) iterate(1, 1) mapped(first)  
)
```

```
(fib indexed(from: 1) takeWhile(second < 1000) last first + 1) println  
fib indexed(from: 1) droppedWhile(second < 1000) first first println
```

```
bottle = method(i,  
  case(i,  
    0, "no more bottles of beer",  
    1, "1 bottle of beer",  
    "#{i} bottles of beer"))
```

```
(99..1) each(i,  
  "#{bottle(i)} on the wall, " println  
  "take one down, pass it around," println  
  "#{bottle(i - 1)} on the wall.\n" println  
)
```

```
use("mandarin")
```

```
帐户 = 本 摹拟 做(
```

```
    转移 = 法(数量, 自: 自我, 至:,
```

```
        自 平衡 -= 数量
```

```
        至 平衡 += 数量
```

```
)
```

```
打印 = 法(
```

```
    "<帐户 名字: #{名字} 平衡: #{平衡}>" 打印行
```

```
)
```

```
)
```

```
箫 = 帐户 带有(名字: "箫", 平衡: 142.0)
```

```
俊 = 帐户 带有(名字: "俊", 平衡: 45.7)
```

```
帐户 转移(23.0, 自: 俊, 至: 箫)
```

```
帐户 转移(10.0, 至: 俊, 自: 箫)
```

```
俊 转移(57.4, 至: 箫)
```

```
箫 打印
```

```
俊 打印
```

```
words = method(text, #/[a-z]+/ allMatches(text lower))
```

```
train = method(features,  
  features fold({ withDefault(1), model, f, model[f] ++. model))
```

```
NWORDS = train(words(FileSystem readFully("small.txt")))
```

```
alphabet = "abcdefghijklmnopqrstuvwxyz" chars
```

```
edits1 = method(word,  
  s = for(i <- 0..(word length + 1), [word[0...i], word[i..-1]])  
  set(*for(ab <- s, ab[0] + ab[1][1..-1]), ;deletes  
    *for(ab <- s[0..-2], ab[0] + ab[1][1..1] + ab[1][0..0] + ab[1][2..-1]), ;transposes  
    *for(ab <- s, c <- alphabet, ab[0] + c + ab[1][1..-1]), ;replaces  
    *for(ab <- s, c <- alphabet, ab[0] + c + ab[1])) ;inserts
```

```
knownEdits2 = method(word, for:set(e1 <- edits1(word), e2 <- edits1(e1), NWORDS key?(e2), e2))
```

```
known = method(words, for:set(w <- words, NWORDS key?(w), w))
```

```
correct = method(word,  
  candidates = known([word]) ?| known(edits1(word)) ?| knownEdits2(word) ?| [word]  
  candidates max(x, NWORDS[x]))
```

```
import(:javax:swing, :JFrame, :JButton)
import java:awt:GridLayout
```

```
button = JButton new("Press me!") do(
  addActionListener(fn(e, button text = "Hello from loke"))
  addActionListener(fn(e, "button pressed" println)))
```

```
JFrame new("My Frame") do(
  layout = GridLayout new(2, 2, 3, 3)
  add(button)
  setSize(300, 80)
  visible = true)
```

# ICheck

forall(int x, int y,  
where: x < y,  
[x, y] sort should == [x, y])

forAll(int x, int y,  
 where:  $y < x$ ,  
 classify(trivial)  $x == y$ ,  
 classifyAs(close)  $(x - y) \text{ abs} < 2$ ,

$[x, y]$  sort should  $==$   $[y, x]$  sort)

forall(list(int) xs,  
xs reverse reverse should == xs)

forEvery(integer x, integer y,  
[x, y] max should  $\geq$  [x, y] min)

forall(int x, int y,  
[x,y] sort should == [x,y])

# Conditions

willFail = method( 10 / 0 )

```
bind(  
  rescue(  
    Condition Error Arithmetic DivisionByZero,  
    fn(c,  
      "Division by zero! Failing..." println  
      nil)),  
  result = willFail  
  "got result: #{result}" println  
) println
```

```
bind(  
  handle(  
    Condition Error Arithmetic DivisionByZero,  
    fn(c,  
      "Division by zero! Restarting..." println  
      invokeRestart(:useValue, 3))),  
  result = willFail  
  "got result: #{result}" println  
) println
```

arbitrarily sized numbers

operator shuffling

comprehensions

macros

reflection/introspection

decimal numbers

symbols

homoiconicity

messages

ranges

dictionaries

methods

lists

mixins

booleans

pairs

literals

pervasive documentation

conditions

dynamic typing

sets

multi-vm

hooks

regular expressions

strong typing

aspects

syntax

dynamic places

lexical blocks

closures

generalized assignment

tuples

ratios

destructuring

enumerables

icheck

java integration

sequences

prototype based OO

blank slate

# Some drawbacks

**Slow!**

VERY mutable

# No concurrency

# Library support

# Experiment

No real life use!

JSR292

# MethodHandle

# Combinator library

# CallSite

# ClassValue

# InvokeDynamic

# Bootstrap Methods

Seph

# Features

# Dynamic typing

# Polymorphic dispatch

# Prototype based OO AKA delegation

# Immutable objects

TCO

# Mutable lexical scopes

# Light weight threads

# Clojure STM

# Module system

# Some examples

```
fact = #(n,  
  if(n == 0,  
    1,  
    n * fact(n - 1)))
```

```
fact(10) println
```

```
fact = #(n,  
  acc = #(acc, n,  
    if(n == 0,  
      acc,  
      acc(n * acc, n - 1)))  
  acc(1, n))
```

```
fact(10) println
```

"Command line arguments:" println  
System programArguments each(println)

```
fib = #(  
  #(a, b, [b, a + b]) iterate(1, 1) mapped(first)  
)
```

```
(fib indexed(from: 1) takeWhile(second < 1000) last first + 1) println  
fib indexed(from: 1) droppedWhile(second < 1000) first first println
```

```
bottle = #(i,  
  case(i,  
    0, "no more bottles of beer",  
    1, "1 bottle of beer",  
    "#{i} bottles of beer"))
```

```
(99..1) each(i,  
  "#{bottle(i)} on the wall, " println  
  "take one down, pass it around," println  
  "#{bottle(i - 1)} on the wall.\n" println  
)
```

foo: #(n, #(i, n += i))

```
foop = #(n,  
  receive(  
    (p, i), val = n + i. p <- val. foop(val))),
```

```
foo: #(n,  
  p = ->(foop(n))  
  #(i,  
    p <- (currentProcess, i)  
    receive(  
      v, v)))
```

```

IntSet: Something with(
  empty?: false,
  adjoin: #(x, Adjoin with(s: self, obj: x)),
  u: #(x, Union with(left: self, right: x))
),

Adjoin: IntSet with(
  contains?: #(y, obj == y || s contains?(y))
),

Union: IntSet with(
  empty?: #(left empty? && right empty?),
  contains?: #(y, left contains?(y) || right contains?(y))
),

Empty: IntSet with(
  empty?: true,
  contains?: #(_, false)
),

IntegersMod: IntSet with(
  contains?: #(y, y % n == 0)
),

(s, k, n) = (Empty, 2, 0)
while(n < 1_000_000,
  if(prime?(k),
    s = s adjoin(k)
    n++
  )
  k++
)
s contains?(13) println

```

# Implementation

# Simple dispatch

receiver.

activationFor(3, false).

invokeExact(receiver, arg0, arg1, arg2);

```
if(savedIdentity != receiver.identity()) {  
    savedIdentity = receiver.identity();  
    savedMH = receiver.activationFor(2, false);  
}  
savedMH.invokeExact(receiver, arg0, arg1);
```

# Arguments

```
private SephObject argument_1_42(boolean eval) {  
    if(eval) {  
        // compiled Seph code to eval the argument  
    } else {  
        return AST_ARGUMENT_1_42;  
    }  
}
```

```
private final static MethodHandle ARGUMENT_1_42_MH =  
    findVirtual(lookup().lookupClass(), "argument_1_42",  
        methodType(SephObject.class, boolean.class));
```

# Intrinsics

TCO

```
public class SThread {  
    public MethodHandle tail;  
} // SThread
```

```
MethodHandle saved =  
    insertArguments(receiver.  
        activationFor(3, false),  
        0,  
        arg0, arg1, arg2);  
thread.tail = saved;  
return SThread.TAIL_MARKER;
```

```
SephObject result = // real operation
while(current == SThread.TAIL_MARKER) {
    current = thread.tail.invokeExact();
}
return result;
```

Activation slow path  
(useful JSR292 trick)

MethodType#methodType(Class **ret**, Class... params)

CallSite#type()

MethodHandles#exactInvoker(MethodType **type**)

MethodHandles#insertArguments(MethodHandle, int **pos**, Object... args)

MethodHandles#filterArguments(MethodHandle, int **pos**, MethodHandle... filters)

```
interface SephObject {  
    // ...  
    MethodHandle activationFor(int arity);  
    // ...  
}
```

```
class SephCallSite extends CallSite {  
    public MethodHandle computeSlowPath() {  
        MethodHandle invoker = exactInvoker(type());  
        MethodHandle activation = findVirtual(SephObject.class, "activationFor",  
            methodType(MethodHandle.class, int.class));  
        MethodHandle boundActivation = insertArguments(activation, 0, arityOfThisCallSite());  
        return filterArguments(invoker, 0, boundActivation);  
    }  
}
```

# Current/Future

# Questions?

**OLA BINI**

**ThoughtWorks®**

<http://olabini.com>  
[obini@thoughtworks.com](mailto:obini@thoughtworks.com)

@olabini