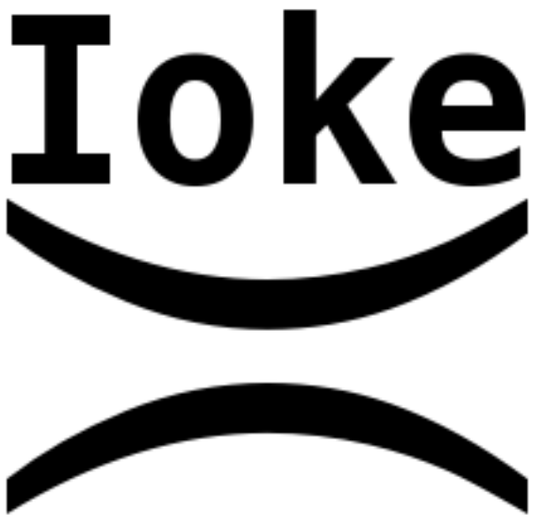
ThoughtWorks®



A Folding Language

Ola Bini
computational metalinguist
ola.bini@gmail.com
http://olabini.com/blog

Ioke

ThoughtWorks®

Your host

From Sweden

Language geek at ThoughtWorks

Experience with C/C++, C#, Java, Ruby, Lisp and many more

JRuby core developer

Creator of the Ioke language

Author of Practical JRuby on Rails (APress)

Creator of/contributor to numerous open source projects

Member of the JSR 292 EG

Ioke

ThoughtWorks®

Assumptions

You all know language implementations

You know the difference between syntax and semantics

You are all experts in at least one language

In summary: I will skip the niceties

Ioke

ThoughtWorks®

Ioke ≠ Rich Hickey

Ioke

ThoughtWorks®

Quick overview

An experiment

Dynamic typing

Strong typing

Prototype based OO

Hosted on the JVM and CLR

Developed using TDD

Ioke E (preceded by Ioke 0 and Ioke S)

Ioke

ThoughtWorks®

Expressiveness

Ioke

ThoughtWorks®

Principles/guidelines

Expressiveness

Communication

Core simplicity

Homoiconicity > Syntax > APIs

DSLness

Language flexibility

Ioke

ThoughtWorks®

Folding?

Ioke

ThoughtWorks®

Language inspirations

- Io
  - Core syntax
  - Basic AST model
  - Object model
  - Initial implementation
- Ruby
  - Library design (ie Enumerable)
- Common Lisp
  - Condition system
  - format
  - Several smaller pieces

Ioke

ThoughtWorks®

Naming

Object

Origin

Base, DefaultBehavior, Ground

Prototype, Delegate, Parent

Mimics

clone, dup

mimic

property, slot

cell

String

Text

Ioke

ThoughtWorks®

Syntax

```
chain:

  expression+ ("," expression+)*

expression:

  Identifier ("(" chain? ")")? |
  ("(" chain? ")") |
  ("[" chain? "]") |
  ("{" chain? "}") |
  ("#{" chain? "}")|
  literal |
  Terminator

literal:

  StringLiteral | RegexpLiteral  |
  NumberLiteral | DecimalLiteral | UnitLiteral

Terminator: "\n" | "."
```

Syntax continued

```
Identifier:

   '[]' |
   '{}' |
   '.' '.'+ |
   OperatorChar+ |
   Letter (Letter|IDDigit|':'|'!'|'?'|'$')* |
   ':'(Letter|IDDigit)(Letter|IDDigit|':'|'!'|'?'|'$')*

OperatorChar:

   '+' | '-' | '*' | '%' | '<' | '>' | '!' |
   '?' | '~' | '&' | '|' | '^' | '$' | '=' |
   '@' | '\''| '`' | ':'
```

Ioke

ThoughtWorks®

Syntax - alternative view

```
   program        ::=  messageChain?
messageChain   ::=  expression+
expression     ::=  message   |
                    brackets  |
                    literal   |
                    terminator
literal        ::=  Text      |
                    Regexp    |
                    Number    |
                    Decimal   |
                    Unit
message        ::=  Identifier ( "(" commated? ")" )?
commated       ::=  messageChain ( "," messageChain )*
brackets       ::=  ( "[" commated? "]" ) |
                    ( "{" commated? "}" )
terminator     ::=  "." | "\n"
comment        ::=  ";" .* "\n"
```

Ioke

ThoughtWorks®

Numbers

- Integers
  - Arbitrary sized, exact math
  - Decimal and hexadecimal syntax
- Decimal numbers
  - No floats
  - Standard floating point syntax
  - Both arbitrarily sized, exact when possible
- Ratios
- Units

Ioke

ThoughtWorks®

Strings

Two syntaxes

"this is a string"

#[and so is this]

Interpolation with #{}

Can be recursive

Newlines OK

Standard escapes

Ioke

ThoughtWorks®

Regexps

Syntax

#/this is a regexp/x

#r[and so is this]x

Interpolation is fine

Can be used to compose regexps or insert literal text

Ioke

ThoughtWorks®

Operator shuffling

Used to implement operator precedence

Binary operators

Trinary operators

Inverted operators

Ioke

ThoughtWorks®

An Ioke object

Zero or more mimics

Zero or more cells

    Symbol to Ioke value

Can be frozen

Can have documentation

Opaque data

Ioke

ThoughtWorks®

Opaque data

Not used by regular objects

Used to handle what happens when an object is cloned

nil, true, false

Lists, Strings, Numbers, Decimals, Strings, Regexps

Methods, Macros, Lexical blocks

All of this could be implemented in pure Ioke

- But it isn't right now

Ioke

ThoughtWorks®

AST

- Message
  - First class citizen
  - `String name`
  - `Message next`
  - `Message previous`
  - `List<Message> arguments`
  - Positioning information
  - Cached value
- The AST is mutable
- Always available

Ioke

ThoughtWorks®

Execution model

```
loop with(receiver, ground, message, current = ground,
last = nil) while message != null

  name = message.name
  if name == "."
    current = ground
  eif symbol?(name)
    current = getSymbolFor(name)
    last = current
  else
    tmp = message.sendTo(current)
    if tmp != null: current = last = tmp
  message = message.next

return last
```

Ioke

ThoughtWorks®

Execution model, continued

```
message.sendTo(o):
  if message.cached: return message.cached
  return o.activate

obj.activate(message):
  cell = this.findCell(message.name)
  while(cell == nul &&
        ((cell = p = this.findCell("pass")) == nul ||
         !isApplicable(p)):
    signal condition

  return cell.getOrActivate(message)

obj.getOrActivate(message):
  if isActivatable(): return this.activate(message)
  return this

obj.activate(message):
  by default looks for "activate" cell
  if method-like object, dispatch polymorphically
```

Ioke

ThoughtWorks®

Code

Raw message chains

Methods (generally instances of DefaultMethod)

- Created with **`method`**

Lexical blocks (instances of LexicalBlock)

- Created with **`fn`** or **`fnx`**

Macros (DefaultMacro)

- Created with **`macro`**

Syntax (DefaultSyntax)

- Created with **`syntax`**

Lecros (LexicalMacro)

- Created with **`lecro`** or **`lecrox`**

Ioke

ThoughtWorks®

Destructuring variations

dmacro

dsyntax

dlecro / dlecrox

Ioke

ThoughtWorks®

DefaultMethod

Self

Documentation string

Positional arguments

Keyword arguments

Rest arguments

Keyword rest arguments

Splatting

Lexical blocks support the same

Ioke

ThoughtWorks®

Java Integration

Create objects

Pass around them

Implement interfaces

Extend classes

Implicit conversion from blocks to interfaces

Ioke

ThoughtWorks®

Demo

Ioke

ThoughtWorks®

Some interesting points

No globals

Not even nil, true or false

Literals are just message sends too

Methods can be implemented in terms of macros

Ioke

ThoughtWorks®

Testing

TDD

~ 4200 specs

ISpec

DokGen

Ioke

ThoughtWorks®

Future plans

Tree rewriting

IKIL

~~Hooks~~

Concurrency

Units

Important libraries

Cane

SQL DSL? JRuby DSL?

Parrot and V8 runtime

Static types

Ioke

ThoughtWorks®

Not covered

Project info, github

Conditions

Aspects

Comprehensions

Loads of other stuff

Ioke

ThoughtWorks®

Questions?