

JRuby, Duby, Surinx, and invokedynamic

Fun with Languages

Intro

- Charles Oliver Nutter
- JRuby Architect at Engine Yard
 - “Bringing Ruby to the Java world”
- Amateur language auteur
- blog.headius.com
- headius@headius.com
- @headius

JRuby

- Ruby on JVM
- Fastest production Ruby impl
 - Much more to come!
- Many challenges on JVM

JRuby Dispatch

- Dynamic
- Mutable classes
- Complex, mutable hierarchy
- Call frames (backtrace, cross-call data)
- Non-local flow control (return in closure)
- Variable arity, boxing, IRubyObject,

JRuby + indy

- Eliminate generated handles
- Simplify call logic at call site
- Reduce call path complexity
- Inlinable dynamic calls
- Unlock the secret power of the JVM
- Using almost all MH adapters already

Duby

- Ruby syntax (mostly)
- Local type inference
 - Minimal type declarations
- No runtime library (provided by backend)
- AOT compiled, mostly
- Written in (J)Ruby
- As fast as Java

Why?

- Implementation lang for JRuby?
 - Make JRuby hacking more approachable
- Ruby syntax is clean, “beautiful”
- Incrementally better than Java alone
- Research optional/gradual typing for Ruby
- Yet another language

Ruby

```
def fib(a)
  if a < 2
    a
  else
    fib(a - 1) + fib(a - 2)
  end
end
```


Duby

```
def fib(a => :fixnum)
  if a < 2
    a
  else
    fib(a - 1) + fib(a - 2)
  end
end
```

Calling Java

```
import "System", "java.lang.System" |
time_start = System.currentTimeMillis
puts "fib(45):"
puts fib(45)
time_total = System.currentTimeMillis - time_start
puts "Total time:"
puts time_total
```

Define a Class

```
class Foo
  def initialize
    puts 'constructor'
    @hello = 'Hello, '
  end

  def hello(a => :string)
    puts @hello; puts a
  end
end

Foo.new.hello('Duby')
```


A Bit More

```
import javax.servlet.http.HttpServlet

class HelloServlet < HttpServlet
  def doGet(req, resp)
    resp.getWriter.println("Hello, Duby!")
  end
end
```

Status

- Basic type and method defs
- Importing types
- Java object construction, dispatch (static and instance)
- Primitives, math, most boolean tests
- String + String
- Basic class extension

To do

- Arrays
- Reopen class (extension methods)
- Mixin inheritance
- Closures (anon class sugar?)
- Runtime libraries in Duby?
- Java 5 features (annotations, enums, etc)
- Other backends (C? CLR? LLVM?)

Surinx

- Ruby syntax (exactly)
- Dynamic dispatch (invokedynamic)
- Minimal runtime library (indy dispatcher)
- Scriptable, but no interpreter
- Written in (J)Ruby
- As fast as Java*
 - Limited mostly by invokedynamic, numerics

Why?

- Same reasons as Duby
- Experiment with invokedynamic
 - Help indy implementers test
 - Try things outside of JRuby
 - Show off!
- Yet another language

Ruby

```
def fib(a)
  if a < 2
    a
  else
    fib(a - 1) + fib(a - 2)
  end
end
```


Surinx

```
def fib(a)
  if a < 2
    a
  else
    fib(a - 1) + fib(a - 2)
  end
end
```

Status

- Basic method definition
- Most primitive math (long and double)
- Importing classes
- Constructing, calling Java objects

To Do

- Class definition
- Better method dispatch (JLS++)
 - Attila's indy+MOP?
- Primitives as much as possible
- Arrays
- Java 5 features as appropriate

Back to JRuby

- Duby can advise optional static typing
 - JRuby parser support
 - New optimizing compiler
- Surinx demonstrates invokedynamic
 - invokedynamic in JRuby
 - Better, simpler call protocol

Workshop

- Deep dive on Duby, Surinx, JRuby+indy
- Walkthrough code
- Discuss indy, compilation, next steps
- Solicit help :)

Thank you!

- headius@headius.com
- blog.headius.com
- [@headius](https://twitter.com/headius)
- www.jruby.org
- kenai.com/projects/duby
- github.com/headius/surinx