

# JRuby: Understanding the Fuss

**Charles Oliver Nutter**  
**Thomas Enebo**  
**Tor Norbye**  
**Martin Krauskopf**



Except where otherwise noted, the content of this presentation is licensed under

the Creative Commons Attribution-Share Alike 3.0 United States License (<http://creativecommons.org/licenses/by-sa/3.0/us/>).

# Agenda

- What is Ruby
- Ruby Quick Tour
- What is JRuby
- IRB Demo
- What is Ruby on Rails
- NetBeans Ruby Demos

# What Is Ruby

- Dynamic-typed, pure OO language
  - > Interpreted
  - > Open source, written in C
  - > Good: easy to write, easy to read, powerful, “fun”
  - > Bad: green threads, unicode support, libraries, “slow”
- Created 1993 by Yukihiro “Matz” Matsumoto
  - > “More powerful than Perl and more OO than Python”
- Very active community, wide range of apps
- Ruby 1.8.x is current, 1.9 is in development to become 2.0

# Ruby Quick Tour: Pure OO

- Everything is an Object
  - > `Circle.new(4)` => instance of Circle
  - > `"abc".length` => 3
  - > `1.to_s` => "1"
- All Objects are instances of Classes
  - > `1.class` => Fixnum
- Single-Inheritance
- Object is base class of all classes

# Ruby Quick Tour: Basics

- Literals

- > Fixnum: 1

- > Float: 1.0

- > Bignum: 12345678987654321

- > String: "one" 'one' %Q[one] %q[one] ...

- > Multiline string ("here doc"):

- x = <<EOS

- extend across two  
lines

- EOS

- > Symbol: :one, %s[one]

- > Regexp: /^foo\w+.\*bar\$/, %r[^foo\$]

- > Array: [1, "ein", :ichi]

- > Hash: {:one => 1, :two => 2}

# Ruby Quick Tour: Basics

- String substitution

```
> a = "foo"
> b = "bar#{a}baz" => "barfoobaz"
```
- Operator overloading

```
> def +(arg); ...
```
- Attributes

```
> class Foo
  attr_accessor :a
end
x = Foo.new
x.a = "hello"
puts x.a => "hello"
```

# Ruby Quick Tour: Dynamically-Typed

- Variables do not declare type
  - > `a = 1`
  - > `a = "one"`
  - > `a = [1, "ein", a]`
- Strongly typed: objects stay the same type
- ...but type is less important than supported ops
  - > If you're defining behavior based on incoming types, you've just introduced a bug
  - > Don't rely on incoming types
  - > Rely on expected operations
- Often called "Duck Typing" today

# Ruby Quick Tour: Duck Typing

- “If it waddles like a duck and it quacks like a duck...”

```
def make_it_waddle(waddler)
  waddler.waddle
end
```

```
make_it_waddle(Duck.new)
make_it_waddle(Penguin.new)
make_it_waddle(Octopus.new)
```

- Runtime errors seem scary, but rarely happen
- Good unit testing helps prevent those rare cases

# Ruby Quick Tour: A Simple Class

- Anatomy of a simple class

```
class Circle < Object
  def initialize(radius)
    @radius = radius
  end
  ...
  def area()
    return Math::PI * @radius ** 2
  end
end
```

```
Circle.new(4)
```

# Ruby Quick Tour: A Simple Class

- '<' is 'extends' of Ruby

```
class Circle < Object
  def initialize(radius)
    @radius = radius
  end
  ...
  def area()
    return Math::PI * @radius ** 2
  end
end
```

```
Circle.new(4)
```

Object is optional  
like in Java



# Ruby Quick Tour: A Simple Class

- Constructors in Ruby are named 'initialize'

```
class Circle < Object
  def initialize(radius)
    @radius = radius
  end
```

```
  ...
```

```
  def area()
```

```
    return Math::PI * @radius ** 2
```

```
  end
```

```
end
```

new() Allocates Circle instance

and initialize() initializes that instance



```
Circle.new(4)
```

# Ruby Quick Tour: A Simple Class

- Instance variables begin with an '@'

```
class Circle < Object
  def initialize(radius)
    @radius = radius
  end
  ...
  def area()
    return Math::PI * @radius ** 2
  end
end
```

```
Circle.new(4)
```

# Ruby Quick Tour: Modules

- Modules Mix-in Behavior

```
module ShapeStuff
  def diameter
    return 2 * @radius
  end
end
```

```
class Circle
  include ShapeStuff
  ...
end
```

```
Circle.new(4).diameter ==> 8
```

# Ruby Quick Tour: Blocks

- Nameless functions you pass around and invoke

```
# Two syntaxes: Curly brackets {} or do ... end  
[1, 2, 3].each {|number| puts "I see #{number}" }
```

```
[1, 2, 3].each do |number|  
  puts "I see #{number}"  
end
```

```
def foo  
  yield "hello"  
end  
def foo2(&block)  
  block.call("hello")  
end
```

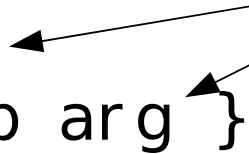
# Ruby Quick Tour: Blocks Cont'd

- Removes repetitive code
  - > No need for the same old external iterator code
    - **list.each { |element| ... do something ... }**
  - > Transactional logic can be internalized
    - **File.open(fname) { |file| line = file.gets ... }**
- Blocks capture surrounding scope

```
def capture(arg)
  return proc { p arg }
end
```

```
capture("captured").call
```

Arg is captured



# What Is JRuby

- Started in 2002
- Java implementation of Ruby language
- Open source, many active contributors
- Aiming for compatibility with current Ruby version
- Easy integration with Java libraries, infrastructure
  - > Call to Ruby from Java via JSR223, BSF
  - > Use Java classes from Ruby (e.g. Script Java)
- Growing set of external projects based on JRuby
  - > JRuby-extras (ActiveRecord-JDBC, rails-integration, ...)

# Demo: Interactive JRuby

# What is Rails

- Rapid MVC web development framework
- Convention over configuration
- Agile development; running app from day one
- Utilizes Ruby's best features extensively
- Rails 1.0 released in 2005, 1.1 this past spring
- RailsConf 2006 (the first) hosted 500 people
- Changed the way many look at web development

# Why Java Users Would Want JRuby

- Ruby is easy to learn, fun to use, very powerful
- Much less code required (claims of 1/10)
- Features missing from Java
  - > Closures
  - > Open classes, metaprogramming
  - > Literals for array, hash, regex
  - > Duck-typing
- Better language for tying many libraries together
- Creating domain-specific languages to wrap libs
- Excellent tool support!

# JRuby on Rails for Java Developers

- Greatly simplified web development
  - > “Less Rails code than Java app configuration”
  - > Instant applications: working code in minutes
- Active, excited community of users, developers
- Makes small apps trivial to create
- Ruby is an excellent language
- No need to leave Java servers, libraries, reliability
- All the cool kids are doing it

# **Demo: NetBeans Ruby Support**

# **JRuby: Understanding the Fuss**

**Q/A**

**Thank You!**

**charles.nutter@sun.com**

**tom.enebo@sun.com**