Introduction to Ruby

(for programmers)

Glenn Vanderburg
glenn@thinkrelevance.com

Primitives

relevance

Counterintuitive

```
public void testIntOverflow() {
  assertEquals(4, 2+2);
  int TWO_BILLION = 2000000000;
  assertEquals(-294967296 , TWO_BILLION + TWO_BILLION);
}
```

relevance

Better

```
$ irb
irb(main):001:0> x = 10
=> 10
irb(main):002:0> x *= 1000
=> 10000
irb(main):003:0> x.class
=> Fixnum
irb(main):004:0> x *= 1000
=> 10000000
irb(main):005:0> x *= 1000
=> 10000000000
irb(main):006:0> x.class
=> Bignum
```

relevance

Primitives

- no value limits
- no overflows
- runtime chooses representation

relevance

Strings

relevance

Interpolation

```
irb(main):007:0> name = "jim"
=> "jim"
irb(main):008:0> "hello, #{name.upcase}"
=> "hello, JIM"
```

relevance

Quoting

```
%Q{"One", "Two", and "Three" are strings"}
=> "\"One\", \"Two\", and \"Three\" are strings\""
```

relevance

Multiline Strings

```
irb> puts <<MY_DELIMITER
irb" one
irb" two
irb" three
irb" MY_DELIMITER
one
two
three
```

```
irb> puts %Q{
irb" four
irb" five
irb" six
irb" }

four
five
six
```

relevance

Regexp Literals

```
'Are four letter words mean?'.gsub(/\b\w{4}\b/, "(bleep)")
=> "Are (bleep) letter words (bleep)?"
```

relevance

Friendly Strings

- multiple quoting styles
- multiline literals
- regular expression literals

relevance

Collections

relevance

High Ceremony

```
public class PrintArgs {
  public static void main(String[] args) {
    for (int n=0; n<args.length; n++) {
      System.out.println(args[n]);
    }
  }
}
```

relevance

Low Ceremony

```
ARGV.each {|x| puts x}
```

relevance

Hash#each

```
irb(main):032:0> ENV.each {|k,v| puts "#{k}: #{v}"}
TERM_PROGRAM: Apple_Terminal
TERM: xterm-color
SHELL: /bin/bash
ANT_HOME: /users/stuart/java/apache-ant-1.6.2
...lots more...
```

relevance

Hash#fetch, Hash#store

```
dict = {:do => "a deer", :re => "a drop of golden sun"}
=> {:do=>"a deer", :re=>"a drop of golden sun"}

dict.fetch(:do)
=> "a deer"

dict.store(:so, "a needle pulling thread")
=> "a needle pulling thread"
```

relevance

Hash#[]

```
dict[:so]
=> "a needle pulling thread"

dict[:dos] = "a beer"
=> "a beer"
```

relevance

Enumerable: Building on each

```
[1,2,3,4,5].map {|x| x**2}
=> [1, 4, 9, 16, 25]

[1,2,3,4,5].find_all {|x| x%2==0}
=> [2, 4]
```

relevance

Collections

- low ceremony
- common iteration style
- literal syntax
- Hash and Array do most of the work

relevance

Control Flow

relevance

if

```
if n>5
  puts "big"
else
  puts "little"
end

puts(if (n>5)
  "big"
else
  "little"
end)

puts n>5 ? "big" : "little"
```

relevance

nil is False, 0 is True!

```
o = nil
=> nil

o ? "true" : "false"
=> "false"

o = 0
=> 0

o ? "true" : "false"
=> "true"
```

relevance

nil is an object!

```
o = nil
=> nil

o.nil?
=> true

o.to_s
=> "nil"

o.to_i
=> 0

o.class
=> NilClass
```

relevance

Many ifs

```
if lines > 1
  puts "you have more than one line"
end

puts "you have more than one line" if lines > 1

if lines > 1; puts "you have more than one line"; end

puts "you've got lines" if lines != 0

puts "you've got lines" unless lines == 0
```

relevance

while/until

```
i=1
while (i<5)
  puts i*i
  i+=1  # no ++ in Ruby
end

i=1
until (i>5)
  puts i*i
  i+=1
end
```

relevance

Ranges

```
(1..5).each {|x| puts x*x}

1
4
9
16
25
```

relevance

Range#step

```
('A'..'I').step(2) {|x| print x}

ACEGI
```

relevance

case

```
def number_grade(letter)
  case letter
    when 'A': 100
    when 'B': 90
    when 'C': 80
    when 'D': 70
    else 0
  end
end
```

relevance

Case Equality

```
def letter_grade(x)
  case x
    when 90..100: 'A'
    when 80..90:  'B'
    when 70..80:  'C'
    when 60..70:  'D'
    when Integer:   'F'
    when /[A-F]/:   x
    else raise "not a grade: #{x}"
  end
end
```

relevance

Control Flow

- many options allow expressive code
- no traditional `for`
- range literal
- first-class support for case

relevance

Blocks: Low-Ceremony Functions

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
List al = new ArrayList();
String line = null;
while (null != (line = br.readLine())) {
  al.add(line);
}
Collections.sort(al, new Comparator() {
  public int compare(Object o, Object o1) {
    return ((String)o).length() - ((String)o1).length();
  }
});
System.out.println("sorted:");
for (Iterator it = al.iterator(); it.hasNext();) {
  System.out.println(it.next());
}
```

```
sorted = readlines.sort {|x,y| x.length-y.length}
puts "sorted:\n#{sorted.join}"
```

relevance

Yielding to a Block

```
def expect_exception(type)
  begin
    yield
  rescue type => e
    return
  end
  raise "Expected exception: #{type}"
end

expect_exception(ZeroDivisionError) {10/0}
```

relevance

Explicitly Calling a Block

```
def expect_exception(type, &blk)
  begin
    blk.call if block_given?
  rescue type => e
    return
  end
  raise "Expected exception: #{type}"
end

expect_exception(ZeroDivisionError) {10/0}
```

relevance

OO

High Ceremony

```
public class Person {
  private String firstName;
  private String lastName;

  public Person(String firstName, String lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }

  public String getFirstName() {
    return firstName;
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }

  public String getFullName() {
    return String.format("%s %s", firstName, lastName);
  }

  public void marry(Person other) {
    String newLastName = String.format("%s-%s", getLastName(), other.getLastName());
    setLastName(newLastName);
    other.setLastName(newLastName);
  }
}
```

relevance

Low Ceremony

```
class Person
  def initialize(first_name, last_name)
    @first_name = first_name
    @last_name = last_name
  end

  attr_accessor :first_name, :last_name

  def full_name
    "#{first_name} #{last_name}"
  end

  def marry(other)
    other.last_name = self.last_name = \
    "#{self.last_name}-#{other.last_name}"
  end

end
```

relevance

Lower Still

```
Person = Struct.new(:first_name,:last_name) do

  def full_name
    "#{first_name} #{last_name}"
  end

  def marry(other)
    other.last_name = self.last_name = \
    "#{self.last_name}-#{other.last_name}"
  end

end
```

relevance

Inheritance

```
class Programmer < Person
  def initialize(first_name, last_name, favorite_language)
    super(first_name, last_name)
    @favorite_language = favorite_language
  end
  attr_accessor :favorite_language
end

p = Programmer.new "James", "Hansson", "Ruby"
# do Person things
puts p.first_name
puts p.last_name
# do Programmer thing
puts p.favorite_language
```

relevance

Access Specifiers

```
class AccessMe
  def initialize(name)
    @name = name
    @stuff = []
  end
  attr_accessor :name

  protected

  attr_accessor :stuff

  private

  def clear
    @name = @stuff = nil
  end
end
```

relevance

Poorly Hung Method

```
public class StringUtils {
  public static boolean isBlank(String str) {
    int strLen;
    if (str == null || (strLen = str.length()) == 0) {
      return true;
    }
    for (int i = 0; i < strLen; i++) {
      if ((Character.isWhitespace(str.charAt(i)) == false)) {
        return false;
      }
    }
    return true;
  }
}
```

relevance

Well-Hung Method

```
# from Rails blank.rb
class String
  def blank?
    empty? || strip.empty?
  end
end
```

relevance

Solving a Common Problem

```ruby
# from Rails blank.rb
class NilClass
  def blank?
    true
  end
end

# with that plus String#blank?, we can change
# a common web programming idiom from this:
if params[:foo] != nil && params[:foo].strip != ""

# to:
if params[:foo].blank?
```

relevance

Bang Implies Mutation

```
s = "bang"
=> "bang"

s.upcase
=> "BANG"

s
=> "bang"

s.upcase!
=> "BANG"

s
=> "BANG"
```

relevance

High-Ceremony Delegation

```
public class Manager {
  private Programmer programmer;
  private Tester tester;

  public void debug(int hours) {
    programmer.debug(hours);
  }

  public void code(int hours) {
    programmer.code(hours);
  }

  public void writeTestPlan(int hours) {
    tester.writeTestPlan(hours);
  }

  public void runTests(int hours) {
    tester.runTests(hours);
  }
```

relevance

Low-Ceremony Delegation

```
class Manager
  attr_accessor :programmer, :tester
  delegate :code, :debug, :to=>:programmer
  delegate :write_test_plans, :run_tests, :to=>:tester
end
```

relevance

Mixins

```
module Employer
  def employees
    @employees ||= []
  end
  def add_employee(employee)
    if employee.employer
      employee.employer.remove_employee(employee)
    end
    self.employees << employee
    employee.employer = self
  end
  def remove_employee(employee)
    self.employees.delete employee
    employee.employer = nil
  end
end

class BusinessPerson < Person
  # employee not shown
  include Employer, Employee
end
```

relevance

OO

- **class definitions are code**
  - write metaprograms
  - create your own idioms
- **class definitions are open**
  - put methods where they belong

relevance

Miscellany

relevance

Exceptions

```
begin
  File.read 'nonexistent'
rescue SystemCallError => e
  puts 'system call failed'
rescue Exception => e
  puts 'generic failure of some kind'
else
  puts 'nothing failed'
ensure
  puts 'this always executes'
end

begin
  1/0
rescue Exception => e
  puts "Message " + e.message
  puts "Backtrace " + e.backtrace.join("\n")
end
```

relevance

Namespaces

```
module Relevance
  class User
    def initialize(name); @name=name; end
    attr_accessor :name
  end
end
module Codecite
  class User
    def initialize(name); @name=name; end
    attr_accessor :name
  end
end
u1 = Relevance::User.new("Justin")
u2 = Codecite::User.new("Stu")
include Relevance
u3 = User.new("Jared")
```

relevance

Load Path

```
irb(main):001:0> $:
=> ["/opt/local/lib/ruby/site_ruby/1.8",\
    "/opt/local/lib/ruby/site_ruby/1.8/powerpc-darwin8.2.0",\
    "/opt/local/lib/ruby/site_ruby", "/opt/local/lib/ruby/vendor_ruby/1.8",\
    "/opt/local/lib/ruby/vendor_ruby/1.8/powerpc-darwin8.2.0",\
    "/opt/local/lib/ruby/vendor_ruby", "/opt/local/lib/ruby/1.8",\
    "/opt/local/lib/ruby/1.8/powerpc-darwin8.2.0", "."]
```

```
irb(main):002:0> $LOAD_PATH
=> ["/opt/local/lib/ruby/site_ruby/1.8",\
    "/opt/local/lib/ruby/site_ruby/1.8/powerpc-darwin8.2.0",\
    "/opt/local/lib/ruby/site_ruby", "/opt/local/lib/ruby/vendor_ruby/1.8",\
    "/opt/local/lib/ruby/vendor_ruby/1.8/powerpc-darwin8.2.0",\
    "/opt/local/lib/ruby/vendor_ruby", "/opt/local/lib/ruby/1.8",\
    "/opt/local/lib/ruby/1.8/powerpc-darwin8.2.0", "."]
```

relevance