RUBY

BY EXAMPLE

Ruby is...

interpreted
object oriented
dynamic

Ruby is...

productive
intuitive
popular

VARIABLES & TYPES

Before we begin...

```
a.+(b)
a.is_valid?
a.decrement!
```

$counter	Global
@@counter	Class
@counter	Instance
counter	Local

‘hello’	String
‘’‘hello’’’	Herestring
%q{hello}	Perl-inspired

```
a = ‘James’
b = “Hello #{a}”
puts b

=> “Hello James”
```

`:hello` Symbol

```
puts <<DOC
  This is a "heredoc"
  Multi-line String
DOC
```

`[1,2,3]`	Array
`{:a=>1, :b=>2}`	Hash (1.8)
`{a: 1, b: 2}`	Hash (1.9)
`0..10`	Ranges

METHODS

```
def greet(name)
  “Hello, #{name}”
end

greet “James”
```

```
def greet(name=“World”)
 “Hello, #{name}”
end

greet “James”
greet
```

CONTROL FLOW

```
if rating >= 4
  puts "great"
elsif rating == 3
  puts "alright"
else
  puts "sucks"
end
```

```
unless rating == 5
  puts “Try harder”
end
```

```
puts “Try harder” unless rating == 5

puts “Seriously” if rating == 1
```

```
case rating
  when 4..5
    puts "great"
  when 3
    puts "alright"
  else
    puts "sucks"
end
```

```
while file.has_more_lines?
  puts file.next_line
end

puts file.next_line while
  file.has_more_lines?
```

```
until file.end_of_file?
  puts file.next_line
end

puts file.next_line until
  file.end_of_file?
```

```ruby
for i in 0...file.line_count
  puts file.lines[i]
end
```

BLOCKS

```
[1,2,3,4].each do |i|
  puts i
end
```

```
array.each(&block)
```

```ruby
printer = Proc.new do |i|
  puts i
end

[1,2,3,4].each &printer
```

```ruby
printer = lambda { |i| puts i }
```

```
def loggerWrapper
  puts "Executing Method"
  yield
  puts "Done Executing"
end

loggerWrapper { puts "Weeee!" }
```

COLLECTIONS

109 Array/Enumerable Operations

50 Hash Operations

```
[1,2,3,4,5].map { |i| i +1}
[1,2,3,4,5].reduce { |i, j| i + j }

({a: '1'}).merge({b: '2'})
```

CLASSES

```ruby
class Person
  def initialize(name, age)
    @name,@age = name,age
  end
end

person = Person.new("James", 32)
```

```ruby
class Person
  attr_accessor :name
  attr_accessor :age
  ...
end

p = Person.new("James", 32)
puts p.name, p.age
```

```
class Person ... end

class Hero < Person
  attr_accessor :powers
end

p = Hero.new("James", 32)
p.powers = [“Flying”]
```

MODULES & MIXINS

```
module StrUtils
  def self.salt(pass, slt)
    “#{pass}#{salt}”
  end
  def salt(slt)
    StrUtils::salt(@value, slt)
  end
end
```

```
StrUtils::salt("123", "456")
```

```ruby
class Password
  include Utils
  def initialize(value)
    @value = value
  end
end

Password.new("test").salt
```

TOOLS OF THE TRADE

irb	Interactive shell
ruby	Ruby interpreter
gem	Library manager
rake	Build tool
rails	Web framework

RUBY

BY EXAMPLE