

Rails from the Ground Up!

with Jared Richardson
RoleModel Software

Start of Day 3

Ruby Part Two Commonly Used Classes



attr_accessor

<i>Shortcut</i>	<i>Effect</i>
<code>attr_reader :v</code>	<code>def v; @v; end</code>
<code>attr_writer :v</code>	<code>def v=(value); @v=value; end</code>
<code>attr_accessor :v</code>	<code>attr_reader :v; attr_writer :v</code>
<code>attr_accessor :v, :w</code>	<code>attr_accessor :v; attr_accessor :w</code>

<http://www.rubyist.net/~slagell/ruby/accessors.html>

The Day One Review

Command line

brew

git

rvm

ruby 1.9.3

basic language features

Day 2 Review

Control flow

Loops

Ranges

Iterators

Fizz Buzz! (and modulo)

Class of class

Classes

Methods

Classes vs instances

Monkey patching

Inheritance

Control Flow

if/end

if/else/end

if/elsif/elsif/end

<code> if <condition>

Loops

loop{..}

loop{ .. break}

while <condition> { .. }

while <condition> do ... end

until <condition> ... end

until <condition> do ... end

for i in (range) ... end

5.times do ... end

Ranges

1..10

"a".."z"

"A".."Z"

Iterators

```
(1..10).each{|x| ...}
```

Classes

```
class ShoppingCart ... end
```

Methods

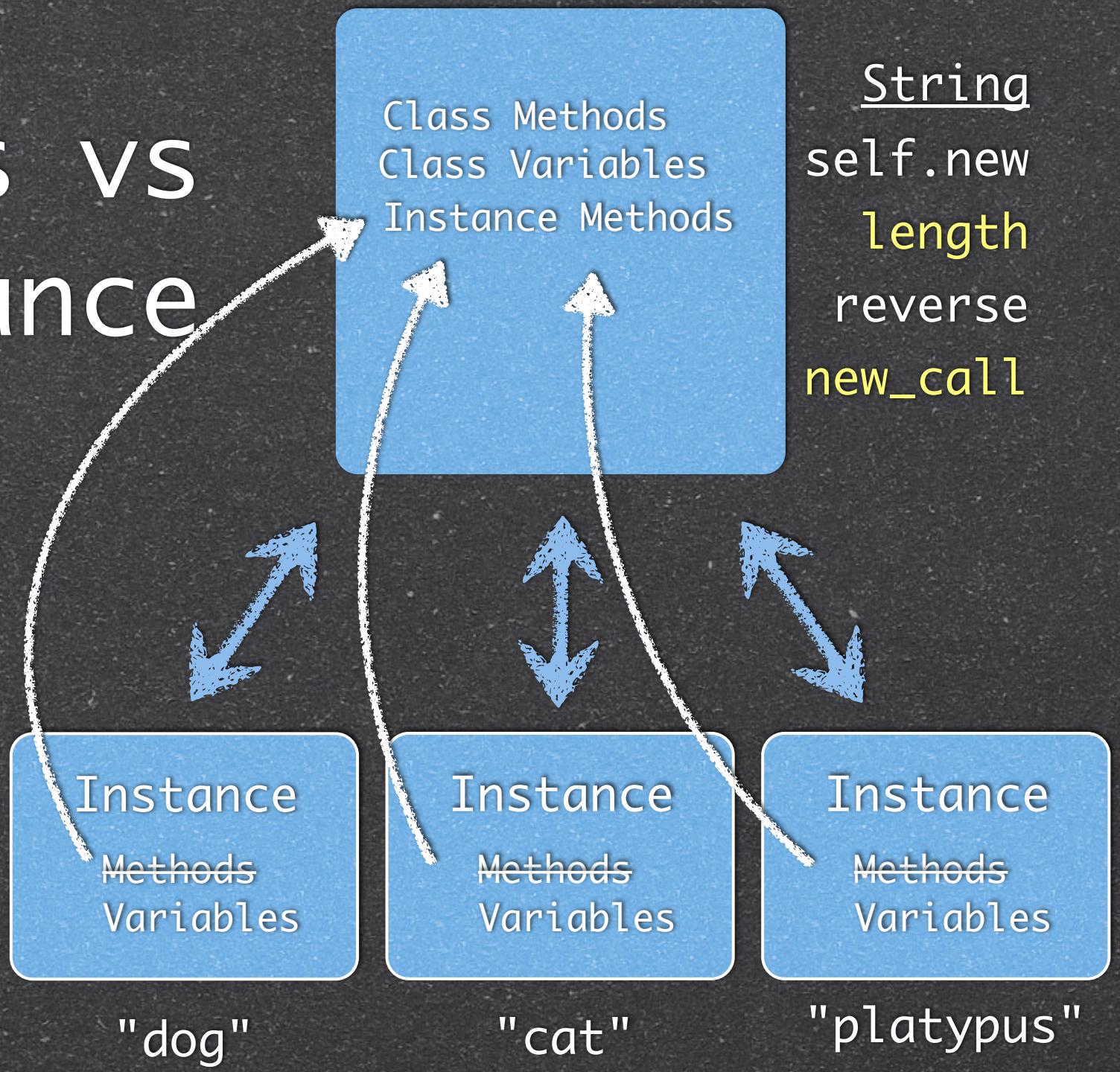
```
def checkout( items ) ... end
```

```
def initialize ... end # for class.new
```

```
def checkout ... return total end
```

```
... return items, total, time
```

Class vs Instance



Monkey Patching

Classes are open

Methods & classes can be redefined`

Inheritance

Dog < Animal

Ford < Automobile

Lady Gaga < ???

Ready?

Commonly Used Classes

Array

Hash

File

Regexp

Symbol

Time

Array

```
ary = Array.new
```

```
ary = Array.new(16)
```

```
ary = Array.new(16, "a")
```

```
ary = []
```

```
ary = [ "a", 7, nil ]
```

Array Usage

ary << “a”

ary << “b”

ary << [“a”, “sub-Array”]

ary.length

ary.sort

ary.each

*

aka splat

It can perform ...

Array creations

```
def hello( first_person, *the_rest)
```

```
h = hello("Jay", "Matthew", "Dave")
```

```
the_rest = ["Matthew", "Dave"]
```

*

aka splat

It can perform ...

Array to method arg conversion

```
the_rest = ["Matthew", "Dave"]
```

```
h = say( "hello", *the_rest )
```

*

aka splat

It can perform ...

Array breakdowns

first, *list = [1, 2, 3, 4]

*list, last = [1, 2, 3, 4]

first, *list, last = [1, 2, 3, 4]

*

aka splat

It can perform ...

Array coercion

```
str_ary = *"Some string"
```

```
str_ary = *"First string ", "other"
```

```
num_ary = *1, 2, 3, 4
```

Array has the Each Iterator

```
ary.each{ | item |  
  puts item  
}
```

```
[2,4,6,8,10].each { |value|  
  print "#{value} "  
}
```

each_index

```
ary = ["7", "abc", 72, [1, 4, 1] ]  
ary.each_index{ | idx |  
  puts "#{idx}) #{ary[idx]}"  
}
```

Print and Puts

puts "Hello World!"

print "Hello again."

n = 7

puts "The number is #{n}"

puts 'The number is #{n}'

puts 'The number is '+ n

puts 'The number is '+ n.to_s

Array as Stack

```
ary.push("a")
```

```
ary.push "b"
```

```
ary.pop
```

```
ary.pop
```

```
ary.pop
```

Array as Queue

ary.first

ary.last

Breaking in Line

```
ary.insert(<position>, <value>)
```

```
ary.insert(2, "breaker")
```

Already Broke In?

ary.include? "17"

ary.index 17

Self Learning

```
ary.methods.sort.each{ |m|  
  puts m  
}
```

<http://apidock.com/ruby/browse>

<http://www.ruby-doc.org/core-1.9.3/Array.html>

Your Turn

create an array with all the odd numbers from 1 to 100

sort and print

reverse sort and print

create a new array from the sorted and unsorted arrays

sort and print

Hashes

Collection of key => value pairs

Creation

```
h = Hash.new
```

```
h = {}
```

```
h = { "a" => 3, "b" => 17 }
```

Access

`h.keys`

`h.values`

`h[“a”]`

`h[:a] = 17`

`h.key?(“a”)`

`h.value?(“3”)`

Other Hash Methods

h.store("c", 42)

h.flatten

h.include?("c")

h.invert

h.delete("c")

Your Turn

Create a new hash with names and favorite colors

```
{ :Jay => "blue", :Jared => "blue", :Matt => "Orange", :Dave => "black"}
```

print the keys

print the values

print the keys : values

File

```
f = File.new("nails.txt", "r+")
File.exist?("file name.txt")
File.const_get(:SEPARATOR)
```

Access Modes

r Read Only

r+ Read-Write

w Write Only

w+ Read-Write, erasing existing file

a Write Only

a+ Read-Write (creates)

Usage

```
counter = 1

File.open("list.txt", "r") do |file|
  while (line = file.gets)
    puts "#{counter}: #{line}"
    counter += 1
  end
end
```

What's This Do?

```
result = File.foreach("list.txt").collect do |line|
  *search, replace = line.strip.split("|", -1)
  [Regexp.new(search.join("|")), replace]
end
```

Bonus Points

```
result = File.foreach("list.txt").collect do |line|
  *search, replace = line.strip.split("|", -1)
  [Regexp.new(search.join("|")), replace]
end
```

list.txt ->

```
<p[^>]*>|<p>
<\\/?(<font|<span>)[^>]*>|<
<\\u>\\s*<u>|<
<\\u>\\s*<i>\\s*<u>|<i>
```

Your Turn!

back to RubyFromTheGroundUp

cd exercise_two

Read input_file.txt

Add up the numbers in the file

And Again . . .

Print the sorted numbers

Print the square root of each number

Print each odd number

Third Time's a Charm

Print the average

Max

Min

RegExp

Regular Expressions

Pattern match against strings

Creating a Regexp

/.../

/hay/.class

%r{...}

%r{hay}

Regexp.new("hay")

Use

=~

/hay/ =~ “needle in a haystack”

“in the haystack” =~ /hay/

re = /hay/

re =~ “haystack”

re =~ “no hit”

More Information

<http://www.ruby-doc.org/core-1.9.3/Regexp.html>

Your Turn!

cd exercise_three

Read input_file.txt

Count the number of times "Hello" occurs

Save each matching line in an Array after
changing "Hello" to "Howdy"

Sort and print

Symbol

:Hello vs “Hello”

:”Hello World”

Immutable

Runtime wide constant

Usage

```
hash = { :a => 7, :b => 12 }
```

Time

Date/Time abstraction

Usage

```
t = Time.now
```

```
t.year
```

```
t.month
```

```
t.day
```

```
t.tuesday?
```

```
t.isdst
```

```
require 'date'

Date.new(2001,2,3)          #=> #<Date: 2001-02-03 ...>
Date.jd(2451944)           #=> #<Date: 2001-02-03 ...>
Date.ordinal(2001,34)       #=> #<Date: 2001-02-03 ...>
Date.commercial(2001,5,6)   #=> #<Date: 2001-02-03 ...>
Date.parse('2001-02-03')    #=> #<Date: 2001-02-03 ...>
Date.strptime('03-02-2001',
              '%d-%m-%Y')  #=> #<Date: 2001-02-03 ...>
Time.new(2001,2,3).to_date #=> #<Date: 2001-02-03 ...>
```

Your Turn

What day will it be in 374 days?

What day was it 253 days ago?

What year will it be in 13,453 days?

What month?

eval

Code running code

```
eval("2 + 7")
```

Build Up Code

```
in irb:
```

```
str = "while true \n"
```

```
str += " puts 'loopy' \n"
```

```
str += "end"
```

```
eval str
```

Your Turn

Build up a string that:

- contains a method that accepts a string
- loops 10 times and ...
- prints "Hello " and the string

Still Your Turn

Read and run from file this string:

```
for i in (1..10)
```

```
acc *= i
```

```
puts "#{i} => #{i * i} => #{acc}
```

```
end
```

What Went Wrong?

```
acc = 0
```

```
for i in (1..10)
```

```
    acc *= i
```

```
    puts "#{i} => #{i * i} => #{acc}
```

```
end
```

Still Broken?

```
acc = 1
for i in (1..10)
  acc *= i
  puts "#{i} => #{i * i} => #{acc}"
end
```

One More Exercise!

Go to <http://ruby-lang.org>

On the right hand side

Try Ruby! (in your browser)

Get Started, it's easy!

[Try Ruby! \(in your browser\)](#)

[Ruby in Twenty Minutes](#)

[Ruby from Other Languages](#)

Ruby Koans

<http://rubykoans.com/>

https://github.com/edgecase/ruby_koans

Next...

Thinking in objects...