Ruby

Chris Sass

Red Colored Rock Candy: Ruby is Sweet :)

What is Ruby?

"Ruby is... a dynamic, open source language with a focus on simplicity and productivity. It has a elegant syntax that is natural to read and easy to write."

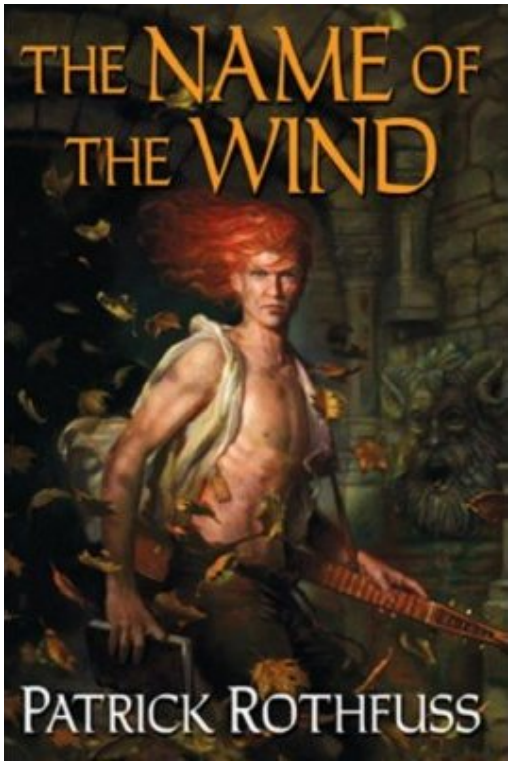
Every Class inherits from Object

```
class Foo
end
Foo.methods
 # =>  [:allocate, :new, :superclass...
Foo.methods.size # => 97
Foo.instance_methods
 # =>  [[:nil?, :===, :=~, :!~, :eql?...
Foo.instance_methods.size
 # => 56
Foo.superclass == 5 # => Object
```

Symbols are not just a percussion instrument

In Ruby **:symbols** are a great way to keep track of **names** of things

- Strings are not the best flags
- Efficient comparison
- Only one instance for each



Symbols are not just a percussion instrument

```
'flag_value' == 'flag_value'
 # => true (the same value)
'flag_value'.equal? 'flag_value'
 # => false (diffe instance of String)
:flag_value == :flag_value
 # => true (the same value)
:flag_value.equal? :flag_value'
 # => true (same instance of Symbol)
1337.send(:to_s)
 # => "1337" (argument is a method's name)
```

Loopy
Ranges & Enumerables

Performs like it reads :)

```
5.times do
  print "w3w7 "
end
w3w7 w3w7 w3w7 w3w7 w3w7 => 5
3.downto(1) { |i| print i.to_s + " " }
3 2 1 => 3
```

Ranges are nice as well too

```
(4..7).each { print i }
4567 => 4..7
```

Loopy
Ranges & Enumerables

Splat ranges to an array

```
[*0..5]
 # => [0, 1, 2, 3, 4, 5]
[*0...5]
 # => [0, 1, 2, 3, 4, 5] (... is off-by-one)
[*Date.new(2012,8,17)..Date.today]
 # => [#<Date: 2012-08-17(an array of Dates)
[*'a'..'e']
 # => ["a", "b", "c", "d", "e"]
```

Loopy
Ranges & Enumerables

Inject the facts

```
def factorial(n)
  1.upto(n).inject(:*)
end
```

Reject the oddballs

```
[*0...5].reject { |i| i if i % 2 != 0 }
 #=> [0, 2, 4]
```

Pythons

Can get sick too

True = False

(Python's internals still return the real `True`)

```
>>> True = False
>>> True == False
True
>> 1 == 1
True
(1 == 1) == True
False
```

2 = 3

```
>>> import ctypes
>>> new_value = 2
>>> ob_ival_offset = ctypes.sizeof(ctypes.c_size_t) + \
... ctypes.sizeof(ctypes.c_voidp)
>>> ob_ival = ctypes.c_int.from_address(id(new_value)
+ob_ival_offset)
>> ob_ival
c_int(2)
>>> ob_ival.value = 3
```

2 = 3

```
>>> print 2+2
6
>>> 2 == 3
True
```

(stolen from http://hforsten.com/redefining-the-number-2-in-python.html)

Custom Namespace

```
from collections import MutableMapping

class CaseInsensitiveNamespace(MutableMapping):
    def __init__(self):
       self.ns = {}
    def __getitem__(self, key):
       return self.ns[key.lower()]
    def  __delitem__(self, key):
        del self.ns[key.lower()]
    def __setitem__(self, key, value):
        self.ns[key.lower()] = value
    def __len__(self):
        return len(self.ns)
    def __iter__(self):
        return iter(self.ns)
```

Case Insensitive!

```
exec '''
foo = 42
Bar = 23
print (Foo, BAR)
''' in {}, CaseInsensitiveNamespace()
```

1 + 2 = -1 (thanks, AST)

```
>>> import ast
>>> x = ast.parse('1 + 2',
mode='eval')
>>> x.body.op = ast.Sub()
>>> eval(compile(x, '<string>',
'eval'))
-1
```

Bad Ideas

- Import code directly from Github
- Patch `__builtins__` to add new built-in functions
- Implicit Self
- Much much more!

Slides:
http://www.scribd.com/doc/58306088/Bad-Ideas
Video:
http://ep2011.europython.eu/conference/talks/5-years-of-bad-ideas

Sick Javascript

```
"foo".blink();
```

PHP

Will turn you Sick && Insane

P. P. P. PHP

1) PHP will treat undefined constants as a bare word (but throws a notice)

+

2) You can silence everything (notices included) with @

=

@print(hello);
@print(PHPはありえへん);

PHP: Never Say NO

NO: Variables that start with a number

```
$1 = "fish";
var_dump($1);

// PHP Parse error:  syntax error, unexpected
// T_LNUMBER, expecting T_VARIABLE
// or '$' in ...
```

PHP: Never Say NO

YES: Variables that start with a number

```
// since ${'a'} is same as $a,
${1} = 'wat';
var_dump(${1});

// string(3) "wat"
```

Neither P in PHP is Parser

here are some ~~functions~~:

list()

empty()

array()

isset()

Neither P in PHP is Parser

They are handled by the parser:

```
empty($var1 || $var2)
// Parse error: syntax error, unexpected
// T_BOOLEAN_OR, ...

$b = 'empty';
$b($a);
// Fatal error: Call to undefined function empty()
```

Neither P in PHP is Parser

So they can't be defined as class methods:

```
class Foo {
    function list(){}
    function empty(){}
    function array(){}
    function isset(){}
}
// all are parse errors
```

Neither P in PHP is Parser

But can be called as class methods:

```
class Foo {
    function __call($method, $args){ var_dump($method); }
}
$f = new Foo();
$f->list();
$f->empty();
$f->array();
$f->isset();
// all work
```

Plz Haz Protected?

Protected means... it's protected right?

```
class Secure{
    protected $secret = 'wat';
    protected function get_password() {
        return 'hunter2';
    }
}
$s = new Secure();
$s->secret;  // Cannot access protected property ...
$s->get_password();  // Call to protected method ...
```

Plz Haz Protected?

Protected means 4EVRYBDY

```
class NotSecure extends Secure {
    function wat($wat){
        var_dump($wat->secret);
        var_dump($wat->get_password());
    }
}
$s = new Secure();
$ns = new NotSecure();
$ns->wat($s); // trololol
```

Plz Haz Private?

OK, so Private is Private right?

```
class Secure{
    private $secret = 'wat';
}
$s = new Secure();
var_dump($s->secret); // NO
```

Plz Haz Private?

OK, so Private is Private right?

```
class Secure{
    private $secret = 'wat';
}
$s = new Secure();
$wat = (array) $s; // cast to array
var_dump($wat); // YES!
// array(1) { ["Securesecret"]=> string(3) "wat" }
```

Plz Haz Private?

OK, so Private methods are Private right?

```
class Secure{
    private function get_password(){ return 'hunter2'; }
}
$s = new Secure();
$ref = new ReflectionClass(get_class($s));
$haha = $ref->getMethod('get_password');
$haha->setAccessible(true);
var_dump($haha->invoke($s));
// string(7) "hunter2"
```

One P in PHP means Proud

Add any of the following to
a php powered server:

?=PHPE9568F34-D428-11d2-A769-00AA001ACF42

?=PHPE9568F35-D428-11d2-A769-00AA001ACF42

?=PHPE9568F36-D428-11d2-A769-00AA001ACF42

?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000

http://en.wikipedia.org/wiki/Main_Page?=PHPE9568F36-D428-11d2-A769-00AA001ACF42

http://www.colourlovers.com/?=PHPE9568F36-D428-11d2-A769-00AA001ACF42

http://digg.com/?=PHPE9568F36-D428-11d2-A769-00AA001ACF42

http://ikayzo.com/about/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000

http://www.tetrisonline.com/?=PHPE9568F35-D428-11d2-A769-00AA001ACF42