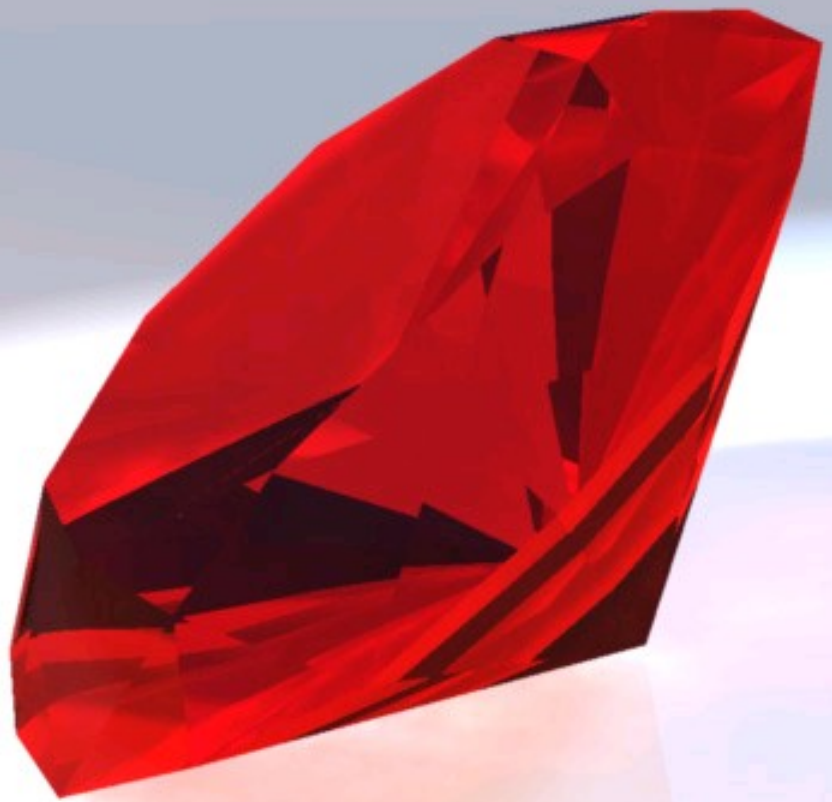


#!/usr/local/bin/ruby

Lucas Pierce

Hello World!

```
puts “Hello World!”
```

Who is this guy?

- CalPoly Graduate
- Team Lead for Ingestion group at CustomFlix
  - On Demand DVD/CD/Digital download service
  - Subsidiary of Amazon.com
- We use Ruby, Objective-C, C, and Java

Operating System Analogy

- You aren’t going to like this…
- Java : Perl : Ruby :: Windows : Gentoo : OS X

Java/Windows

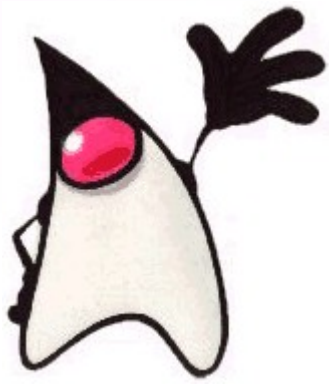
- Inelegant
- Loved by managers
- Major selling point is widespread use

```
public class SomeJavaProgram {

        public static void main(String[] args) {
                System.out.println("Hi");

         }

}
```



Perl/Gentoo

- More hax0r than user friendly
- Delight in the obscure
- At least twenty ways to do anything

```
@P=split//,".URRUU\c8R";@d=split//,"\nrekcah
xinU / lreP rehtona tsuJ";sub p{
@p{"r$p","u$p"}=(P,P);pipe"r$p","u$p";++$p;($q*=
2)+=$f=!fork;map{$P=$P[$f^ord
($p{$_})&6];$p{$_}=/
^$P/ix?$P:close$_}keys%p}p;p;p;p;p;map{$p{$_}=~/
^[P.]/&& close$_}%p;wait
until$?;map{/^r/&&<$_>}%p;$_=$d[$q];sleep
rand(2)if/\S/;print
```

http://perl.plover.com/obfuscated/

Ruby/OS X

- Elegant
- Simple
- Powerful

```
5.times { puts "Maybe I should have said Ruby
is like #{favorite_linux_distro}?" }
```

Ruby is…

- slow
- not useful in writing a C compiler
- just like any other language (because all languages are the same right?)

Ruby is…

- A language
- A community
- A philosophy

Ruby as a language

- True OO
- Closures
- RegExp
- Dynamic
- Meta-programming

True OO

- There is no such thing as a primitive

Ruby

```
-46.abs
```

Java

```
Math.abs(-46)
```

Closures

- Like an anonymous inner class, but cleaner

Ruby

```
array.inject(0) { |sum, e| sum + e }
```

Java

```
int sum = 0;
for (int i : array) sum += i;
return sum;
```

Closures II

Ruby

```
num = 2

array2 = array.select { |i| i.foo > num }
```

Java

```
List<FooBar> array2 = new ArrayList<FooBar>();

int num = 2;

for (FooBar i : array) {

    if (i.foo() > num)

        array2.add(i);

}
```

Closures III

Ruby

```
array.sort! { |i, j| i.foo <=> j.foo }
```

Java

```
array = Collections.sort(array, new
Comparator<FooBar>() {
    public int compare(FooBar i, FooBar j) {
        return i.foo().compare(j.foo());
    }
});
```

RegExp

- Now it really becomes unfair!

Ruby

```
a = IO.readlines("foobar.txt").grep(/\d{3}-\d{4}/)
```

RegExp II

Java

```
try {
   BufferedReader in = new BufferedReader(new
FileReader("foobar.txt"));
   String line;
   List<String> a = new ArrayList<String>();
   while ((line = in.readLine()) != null) {
      if (Pattern.matches("\\d{3}-\\d{4}", line))
         a.add(line);
   }
} catch(Exception e) {  }
```

RegExp III

Ruby

```
ruby -pe "gsub /(Luke|Lucos) Pierce/, 'Lucas Pierce'"
old > new
```

Java

```
nah!
```

Dynamic

- “Ruby isn’t good for real work, it has no types”
- “Ruby is weakly typed”
- These are evil lies!

Ruby

```
x = 5
y = “37”
x + y
TypeError: String can’t be coerced into Fixnum
```

Dynamic II

Ruby

```
class Array
  def every_two
    raise "Invalid array length, length must be even."
unless self.length % 2 == 0

    for i in 0...self.length/2
      yield self[i*2], self[i*2+1]
    end
  end
end

[2, “a”, 5, “b”].every_two {|i, j| puts “#{i}: #{j}” }
```

Dynamic III

Ruby

```
module Summable
   def sum
      inject {|v, n| v + n }
   end
end

class Array
   include Summable
end

[1, 2, 3, 4, 5].sum
```

Meta-programming

- *In Lisp, you don't just write your program down toward the language, you also build the language up toward your program.*
  Paul Graham

Meta-programming II

Java

```
public class FooBar {
    private String foo;
    private int bar;
    private float beef;

    public FooBar(String foo, int bar, float beef) {
        this.foo = foo;
        this.bar = bar;
        this.beef = beef;
    }
    public String getFoo() { return foo; }
    public int getBar() { return bar; }
    public float getBeef() { return beef; }
    public void setBeef(float val) { beef = val; }
}
```

Meta-programming III

Ruby

```
class FooBar
    attr_reader :foo, :bar
    attr_accessor :beef

    def initialize(foo, bar, beef)
        @foo, @bar, @beef = foo, bar, beef
    end
end
```

Meta-programming IV

Ruby

```
class Module
   def attr_reader (*syms)
      syms.each do |sym|
         class_eval %{def #{sym}
                        @#{sym}
                      end}
      end
   end
end
```

Ruby as a community

- RubyGems
- Ruby on Rails
- PickAxe
- Why

RubyGems

- Ruby's CPAN
- Common Lisp
- http://rubyforge.org/projects/rubygems/

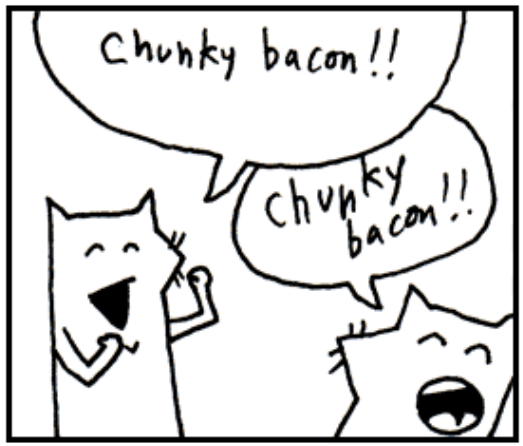
Ruby on Rails

- Ruby's web framework
- The anti-J2EE
- http://www.rubyonrails.org/

PickAxe

- Written by the Pragmatic Programmers
- Excellent!

why the lucky stiff



- http://www.poignantguide.net/ruby/

Blocks

Any code surrounded by **curly braces** is a block.

```
2.times { print "Yes, I've used chunky bacon in my examples, but never again!" }
```
is an example.

With blocks, you can group a set of instructions together so that they can be passed around your program. The curly braces give the appearance of crab pincers that have snatched the code and are holding it together. When you see these two pincers, remember that the code inside has been pressed into a single unit.

It's like one of those little Hello Kitty boxes they sell at the mall that's stuffed with tiny pencils and microscopic paper, all crammed into a glittery transparent case that can be concealed in your palm for covert stationary operations. Except that blocks don't require so much squinting.

Ruby as a Philosophy

- Test-first development
- Domain Specific Languages
- Simplicity is a Virtue

Test-first development

- Heavy emphasis on writing unit tests
- Often said that well tested code is much better than statically type checked code
- Easy to write tests

Domain Specific Languages

Rake

```
task :codeGen do
  # do the code generation
end
task :compile => :codeGen do
  #do the compilation
end
task :test => :compile do
  # run the tests
end
```

Domain Specific Languages II

Ruby

```
event_queue "airspace" do
  generate_event_every 5.minutes + 26.seconds
  collision_radius 197.feet + 6.inches
end
```

From Jim Weirich RubyConf.new(2005)

Domain Specific Languages III

Ruby

```
class Numeric
  def seconds
    self
  end
  def minutes
    60 * self.inches
  end
  def hours
    60 * self.minutes
  end
  def inches
    self
  end
  def feet
    12 * self.inches
  end
  def miles
    5280 * self.feet
  end
end
```

Simplicity is a Virtue

- Little Configuration
- No rewriting Design Patterns
- Minimal Architectures
- DRY

Resources

- http://www.ruby-lang.org/
- http://redhanded.hobix.com/
- PickAxe “Programming Ruby”