Ruby and it's Precious Gems

OpenWest Conference 2013

Matthew Nielsen - @xunker

Overview

Overview

- Just what are “gems”?

Overview

- Just what are “gems”?
- How to find them...

Overview

- Just what are “gems”?
- How to find them...
- Use them...

Overview

- Just what are “gems”?
- How to find them...
- Use them...
- Package them...

Overview

- Just what are “gems”?
- How to find them...
- Use them...
- Package them...
- Extend them...

Overview

- Just what are “gems”?
- How to find them...
- Use them...
- Package them...
- Extend them...

Audience: new ruby developers or those interested in ruby.

Overview

- Just what are “gems”?
- How to find them...
- Use them...
- Package them...
- Extend them...

Audience: new ruby developers or those interested in ruby.

Assumption: Ruby already installed

Sponsor Page here

Obligatory Biography

Matthew Nielsen

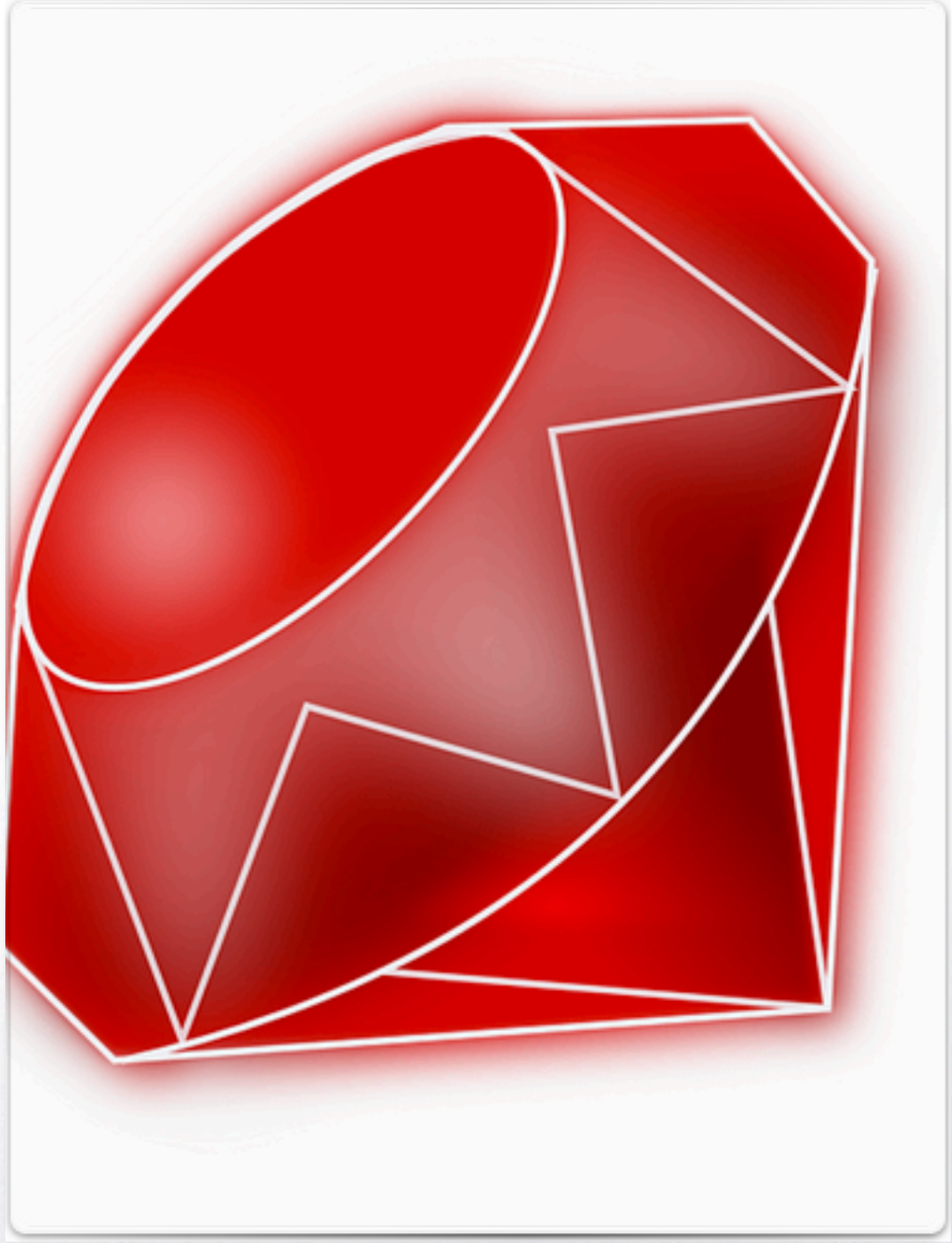
Lead Developer

Salt Lake City, Utah

Just what is a “gem”?

Gems are ruby code

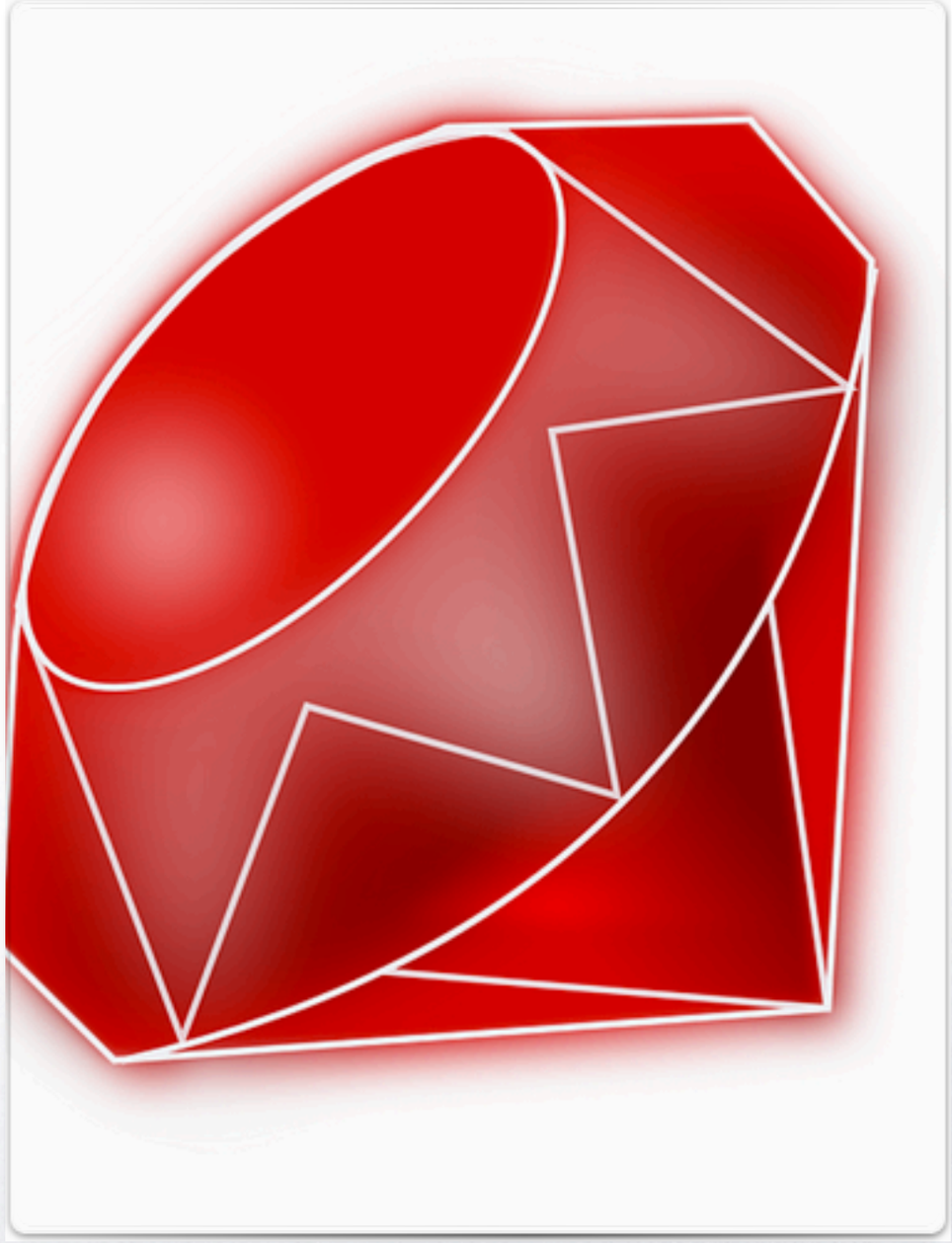
Gems are ruby code



http://pixabay.com/en/red-shiny-gem-jewels-ruby-gems-44532/

Gems are ruby code

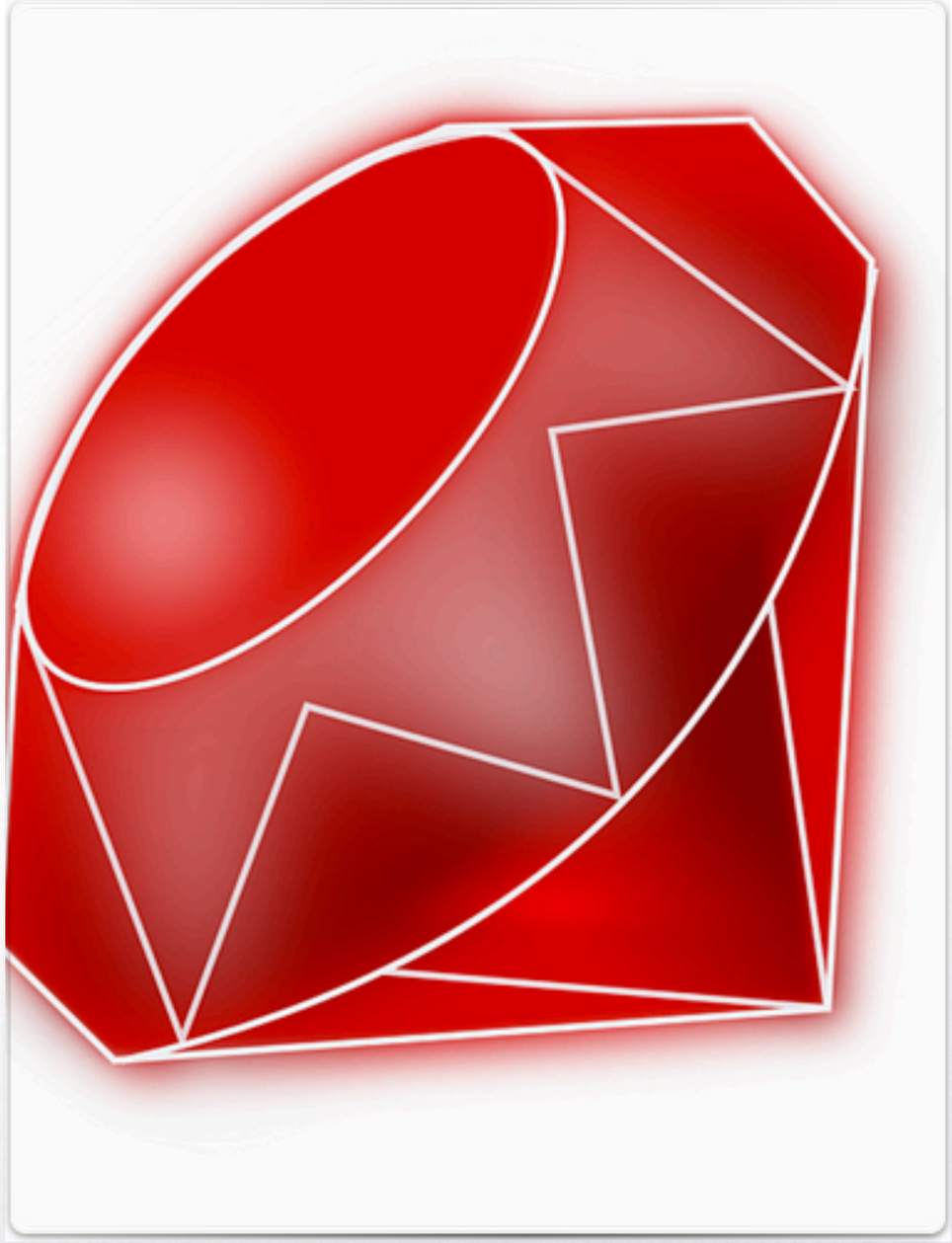
- Ruby gems are written in ruby



http://pixabay.com/en/red-shiny-gem-jewels-ruby-gems-44532/

Gems are ruby code

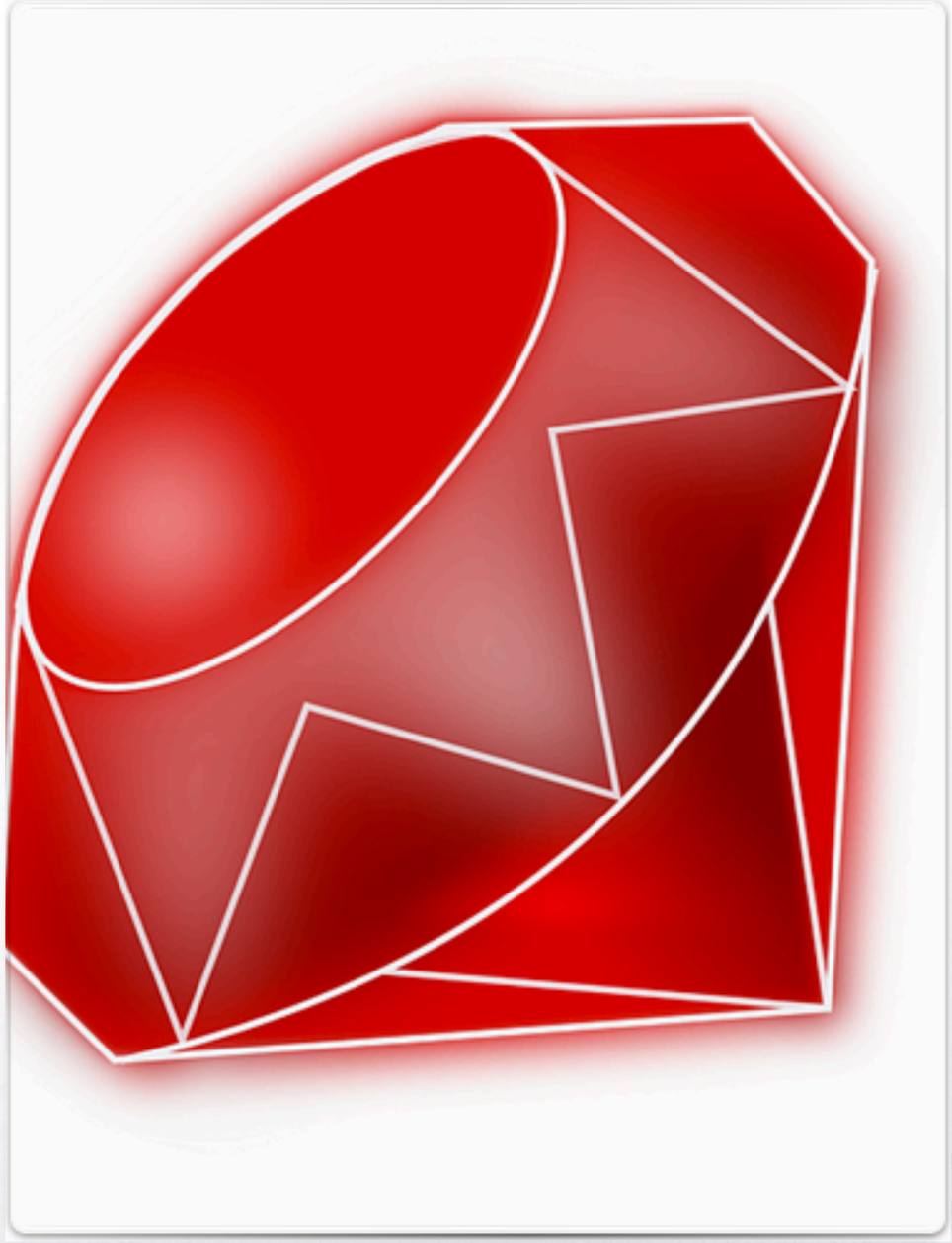
- Ruby gems are written in ruby
- They can talk to other ruby code



http://pixabay.com/en/red-shiny-gem-jewels-ruby-gems-44532/

Gems are ruby code

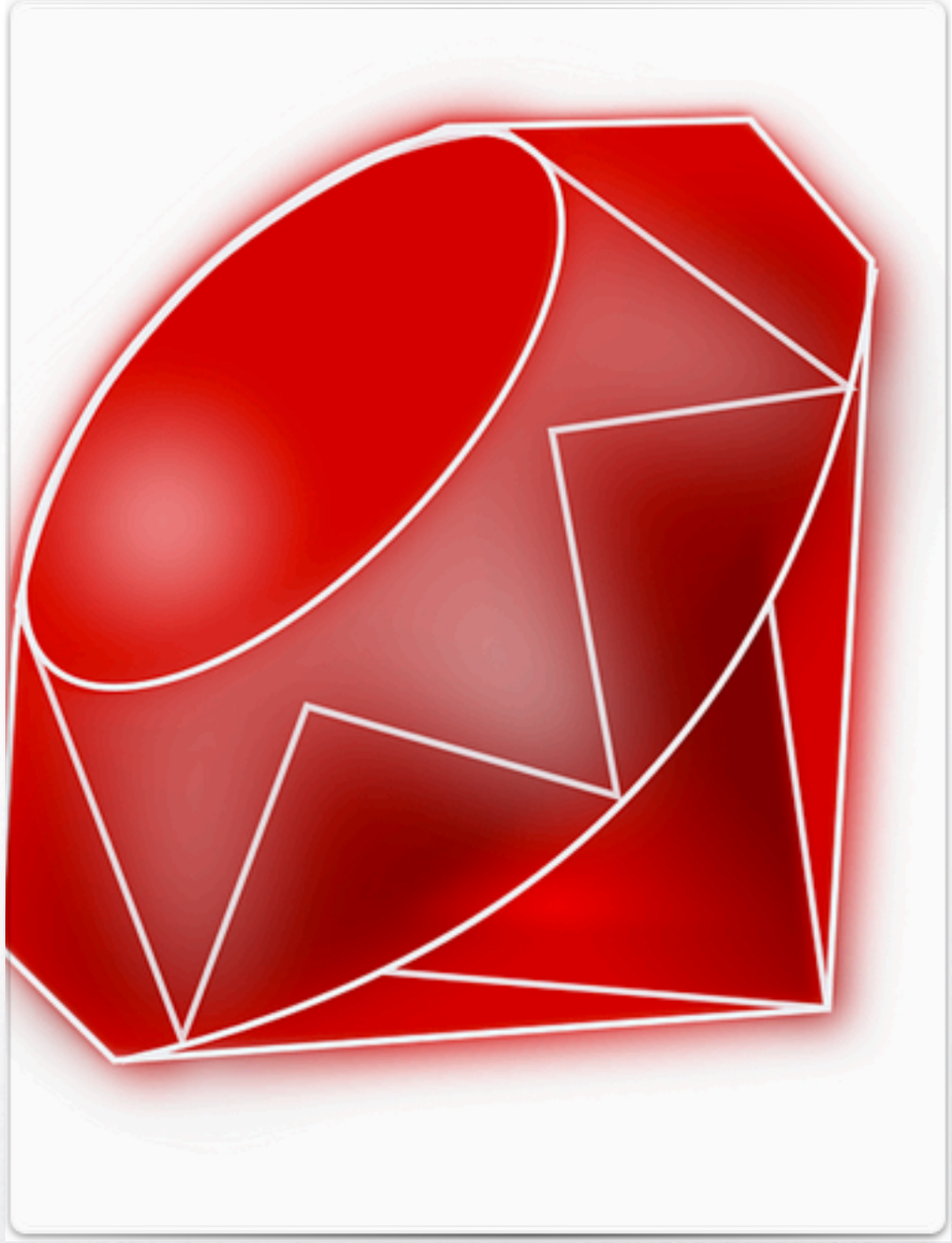
- Ruby gems are written in ruby
- They can talk to other ruby code
- They can talk to things that are not ruby



http://pixabay.com/en/red-shiny-gem-jewels-ruby-gems-44532/

Gems are ruby code

- Ruby gems are written in ruby
- They can talk to other ruby code
- They can talk to things that are not ruby
- But they are still just ruby

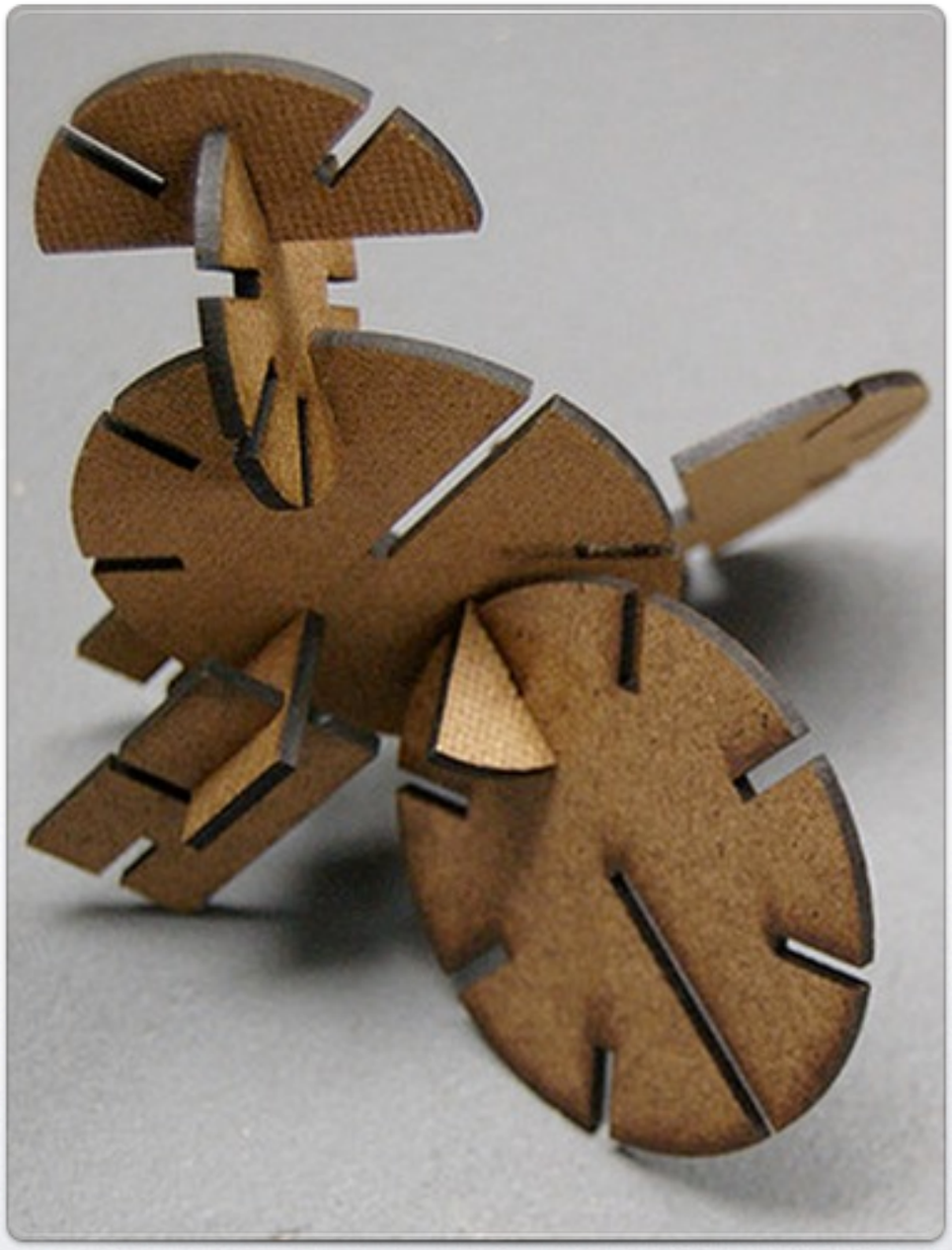


http://pixabay.com/en/red-shiny-gem-jewels-ruby-gems-44532/

Gems are
modular
ruby code

Gems are modular ruby code

Usually...*

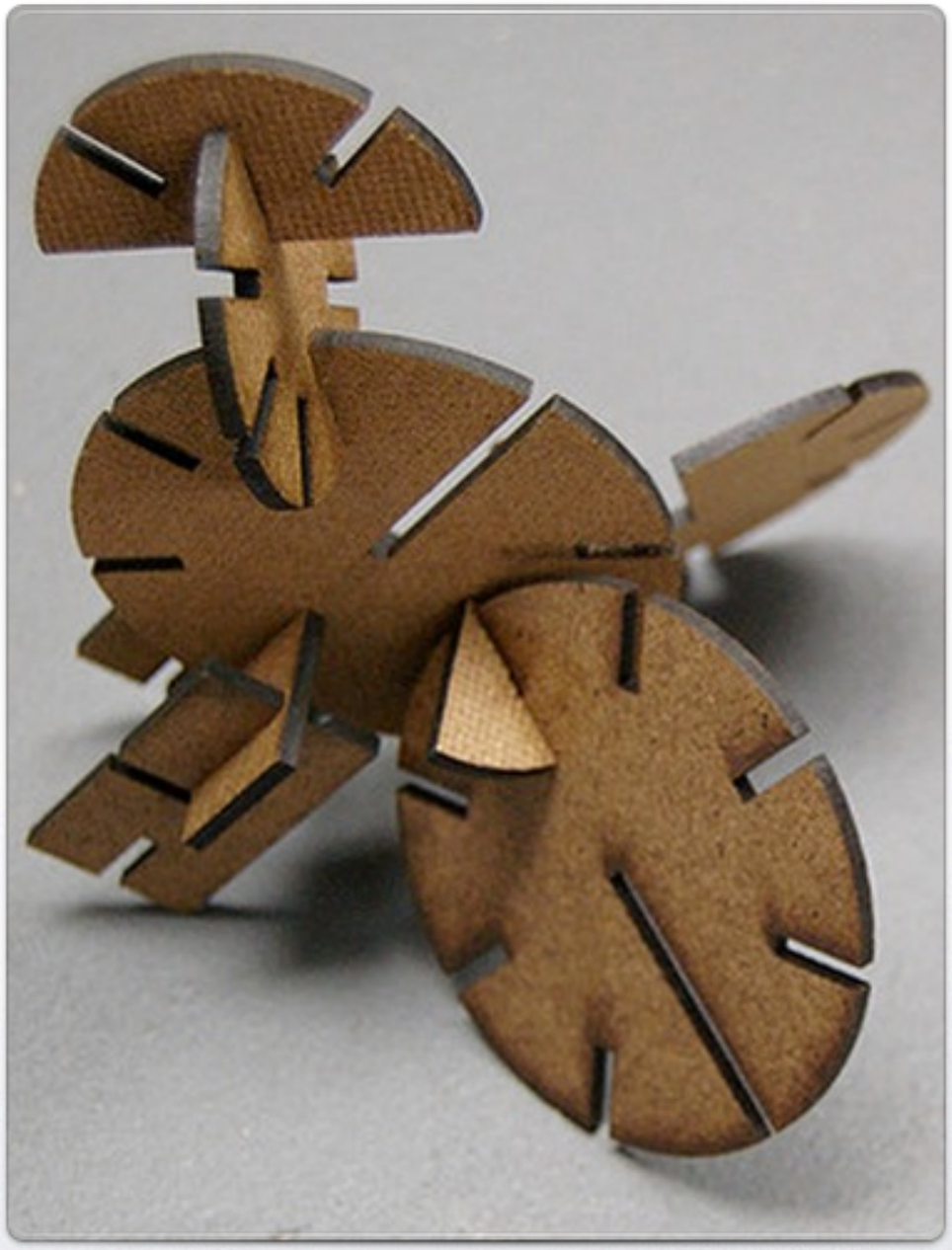


http://www.flickr.com/photos/blmurch/1155304378/

Gems are modular ruby code

Usually...*

- Self-contained

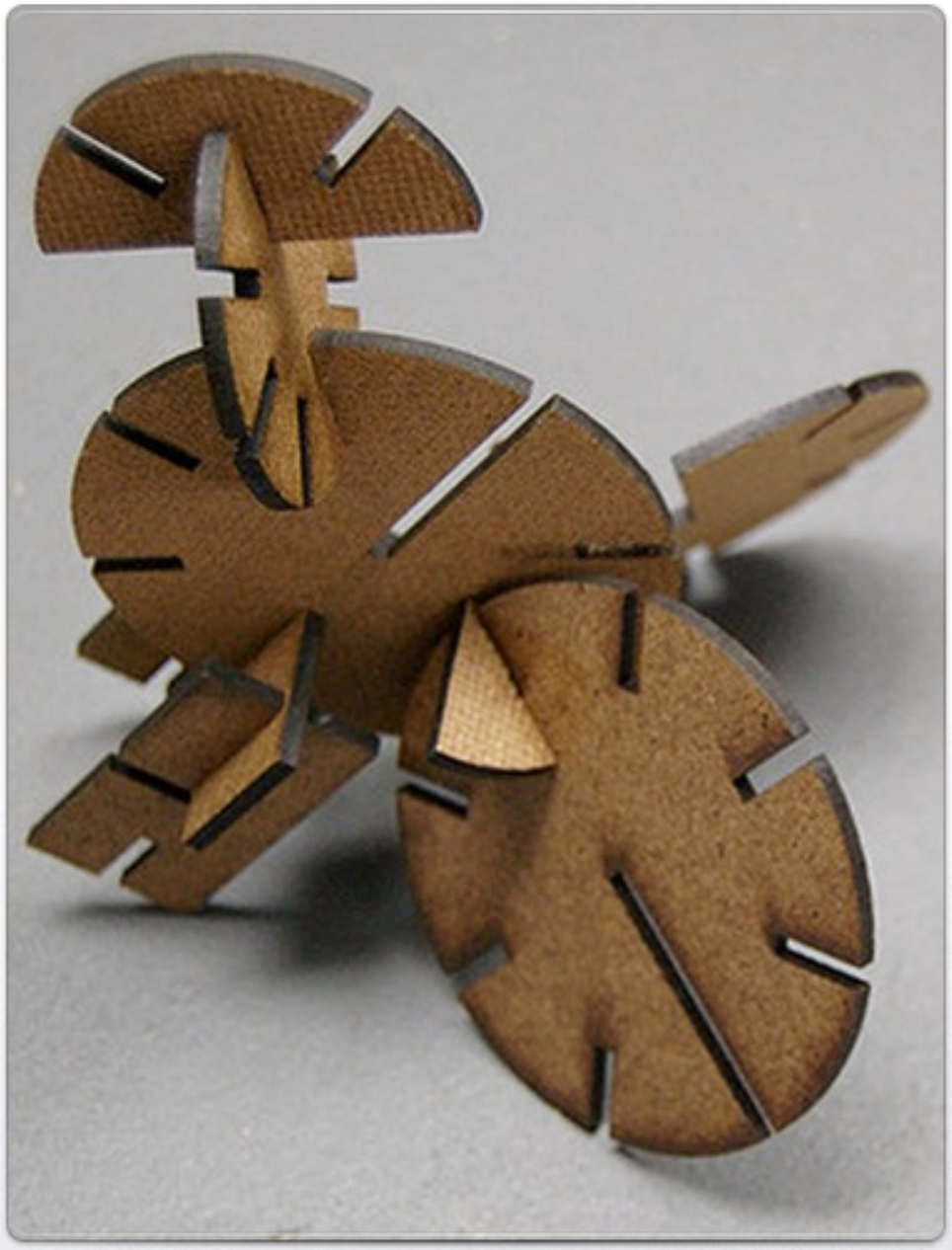


http://www.flickr.com/photos/blmurch/1155304378/

Gems are modular ruby code

Usually...*

- Self-contained
- Meant to be used to fulfill a need in a larger project



http://www.flickr.com/photos/blmurch/1155304378/

Gems are
portable and **modular**
ruby code

Gems are portable and modular ruby code

http://www.flickr.com/photos/lefotodiricky/4464819208/

Gems are portable and modular ruby code

- Packaged in a single installable package

http://www.flickr.com/photos/lefotodiricky/4464819208/

Gems are portable and modular ruby code

- Packaged in a single installable package
- Often platform independent (but not always...)

http://www.flickr.com/photos/lefotodiricky/4464819208/

Gems are portable and modular ruby code

- Packaged in a single installable package
- Often platform independent (but not always...)
- Versioned

http://www.flickr.com/photos/lefotodiricky/4464819208/

Gems are portable and modular ruby code

- Packaged in a single installable package
- Often platform independent (but not always...)
- Versioned
- Contain lists of dependencies

http://www.flickr.com/photos/lefotodiricky/4464819208/

Gems are portable and modular ruby code

- Packaged in a single installable package
- Often platform independent (but not always...)
- Versioned
- Contain lists of dependencies
- May contains test suite, if applicable

http://www.flickr.com/photos/lefotodiricky/4464819208/

Gems are
portable and **modular**
things written in ruby

Gems are portable and modular things written in ruby

http://commons.wikimedia.org/wiki/File:1328101914_Config-Tools.png

Gems are portable and modular things written in ruby

- libraries to be called by you (set, majority of gems)

http://commons.wikimedia.org/wiki/File:1328101914_Config-Tools.png

Gems are portable and modular things written in ruby

- libraries to be called by you (set, majority of gems)
- libraries that call other gems (multi_xml)

http://commons.wikimedia.org/wiki/File:1328101914_Config-Tools.png

Gems are portable and modular things written in ruby

- libraries to be called by you (set, majority of gems)
- libraries that call other gems (multi_xml)
- executables, run-able programs (rails, rake)

http://commons.wikimedia.org/wiki/File:1328101914_Config-Tools.png

Gems are portable and modular things written in ruby

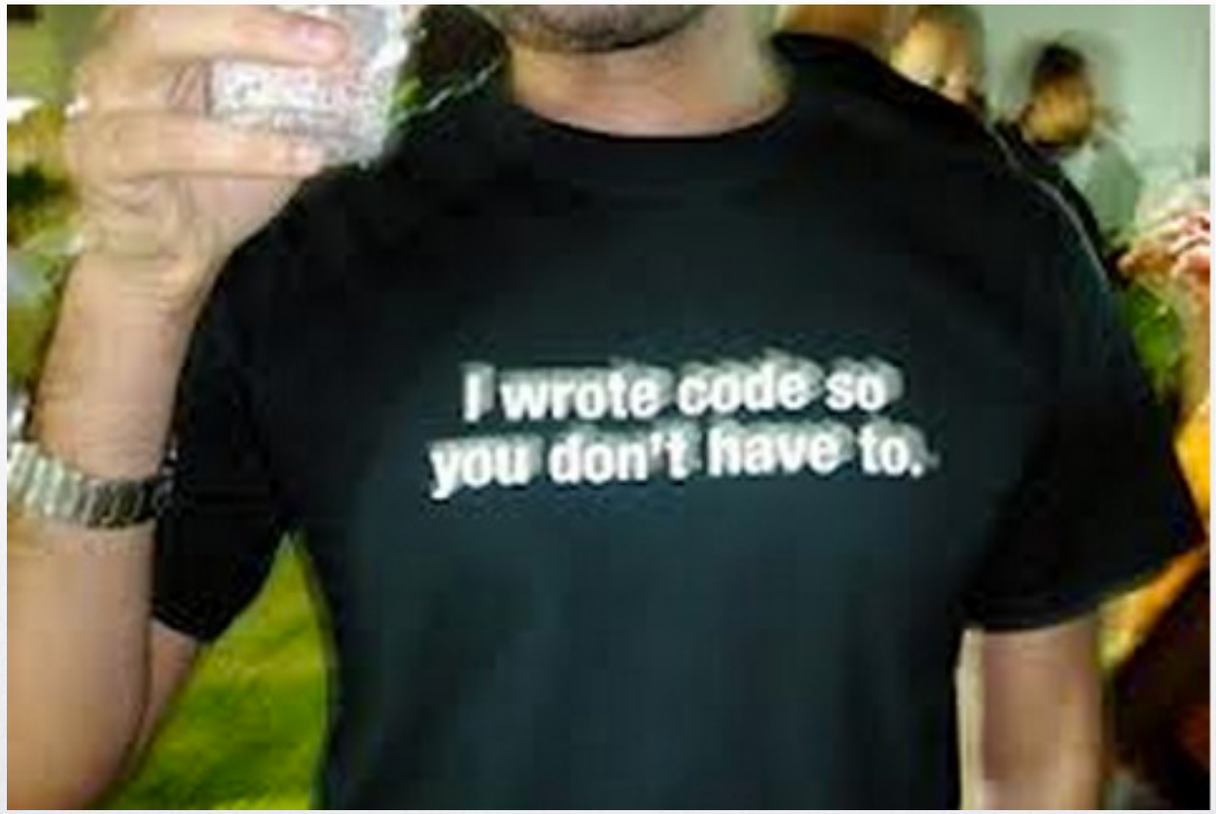
- libraries to be called by you (set, majority of gems)
- libraries that call other gems (multi_xml)
- executables, run-able programs (rails, rake)
- modifications to other libraries (rails_parse_fixes)

http://commons.wikimedia.org/wiki/File:1328101914_Config-Tools.png

Ultimate Truth

Above all:
Gems are work done
by other people
so you don't have to.

Above all:
Gems are work done
by other people
so you don't have to.



So let’s do some work

Your new title:

Your new title:

Intern

Your new title:

Intern

Your new task:

Your new title:

Intern

Your new task:

http://www.flickr.com/photos/defwheezer/1336793469/

BUILDING DEATH ROBOTS YEHAW!

Lots to do!

Lots to do!

- Guns!

Lots to do!

- Guns!
- Treds!

Lots to do!

- Guns!
- Treds!
- Robot brain!

Lots to do!

- Guns!
- Treds!
- Robot brain!
- Armor!

Lots to do!

- Guns!
- Treds!
- Robot brain!
- Armor!
- More guns!

Lots to do!

- Guns!
- Treds!
- Robot brain!
- Armor!
- More guns!
- Scary Things!

Lots to do!

- Guns!
- Treds!
- Robot brain!
- Armor!
- More guns!
- Scary Things!
- Pointy bits!

Lots to do!

- Guns!
- Treds!
- Robot brain!
- Armor!
- More guns!
- Scary Things!
- Pointy bits!
- Big crushing things!

Lots to do!

- Guns!
- Treds!
- Robot brain!
- Armor!
- More guns!
- Scary Things!
- Pointy bits!
- Big crushing things!
- Intimidating paint!

Lots to do!

- Guns!
- Treds!
- Robot brain!
- Armor!
- More guns!
- Scary Things!
- Pointy bits!
- Big crushing things!
- Intimidating paint!
- Still more guns!

Lots to do!

- Guns!
- Treds!
- Robot brain!
- Armor!
- More guns!
- Scary Things!
- Pointy bits!
- Big crushing things!
- Intimidating paint!
- Still more guns!
- Frickin' lasers!

Lots to do!

- Guns!
- Treds!
- Robot brain!
- Armor!
- More guns!
- Scary Things!
- Pointy bits!
- Big crushing things!
- Intimidating paint!
- Still more guns!
- Frickin' lasers!
- System configuration

That's right, you get to do the System Configuration.

That's right, you get to do the System Configuration.

Exciting, huh?

Enterprise-y configuration file:

```
1  <configuration>
2    <mode>kill</mode>
3    <targets>anyone</targets>
4    <excludes>
5      <exclude>Mad Scientist</exclude>
6      <exclude>Minions</exclude>
7      <exclude>Interns</exclude>
8    </excludes>
9  <configuration>
```

Parse it by hand?

Parse it by hand?

Yes! It's easy!

Just say no:

Just say no:

4431

You can't parse [X]HTML with regex. Because HTML can't be parsed by regex. Regex is not a tool that can be used to correctly parse HTML. As I have answered in HTML-and-regex questions here so many times before, the use of regex will not allow you to consume HTML. Regular expressions are a tool that is insufficiently sophisticated to understand the constructs employed by HTML. HTML is not a regular language and hence cannot be parsed by regular expressions. Regex queries are not equipped to break down HTML into its meaningful parts. so many times but it is not getting to me. Even enhanced irregular regular expressions as used by Perl are not up to the task of parsing HTML. You will never make me crack. HTML is a language of sufficient complexity that it cannot be parsed by regular expressions. Even Jon Skeet cannot parse HTML using regular expressions. Every time you attempt to parse HTML with regular expressions, the unholy child weeps the blood of virgins, and Russian hackers pwn your webapp. Parsing HTML with regex summons tainted souls into the realm of the living. HTML and regex go together like love, marriage, and ritual infanticide. The cannot hold it is too late. The force of regex and HTML together in the same conceptual space will destroy your mind like so much watery putty. If you parse HTML with regex you are giving in to Them and their blasphemous ways which doom us all to inhuman toil for the One whose Name cannot be expressed in the Basic Multilingual Plane, he comes. HTML-plus-regexp will liquify the nerves of the sentient whilst you observe, your psyche withering in the onslaught of horror. Regex-based HTML parsers are the cancer that is killing StackOverflow *it is too late it is too late we cannot be saved* the trangession of a child ensures regex will consume all living tissue (except for HTML which it cannot, as previously prophesied) *dear lord help us how can anyone survive this scourge* using regex to parse HTML has doomed humanity to an eternity of dread torture and security holes *using regex* as a tool to process HTML establishes a breach *between this world* and the dread realm of corrupt entities (like SGML entities, but *more corrupt) a mere glimpse* of the world of reg**ex parsers for HTML will ins**tantly transport a *programmer's consciousness* into a world of ceaseless screaming, he comes, the pestilent slithy regex-infection will **devour your HT**ML parser, application and existence for all time like Visual Basic only worse *he comes he comes do not fi*ght h**e comes, his** unholy radiance de*stroying all enlightenment, HTML tags* ***leaking from your eyes' like liq***uid pain, the song of regular expression parsing will exti*nguish the voices of mor**tal man from the sp**here I can see it can you see it it is beautiful t* he f`inal snuf`fing of *the lie***s of Man ALL IS LOST ALL IS** LOST the *pony he comes* he comes he comes **the ich**or permeates *all* MY FACE *MY FACE oh god n***o NO NOOOO N**O stop *the an*gr es a**not real ZALGO IS TONY THE PONY HE COMES**

Have you tried using an XML parser instead?

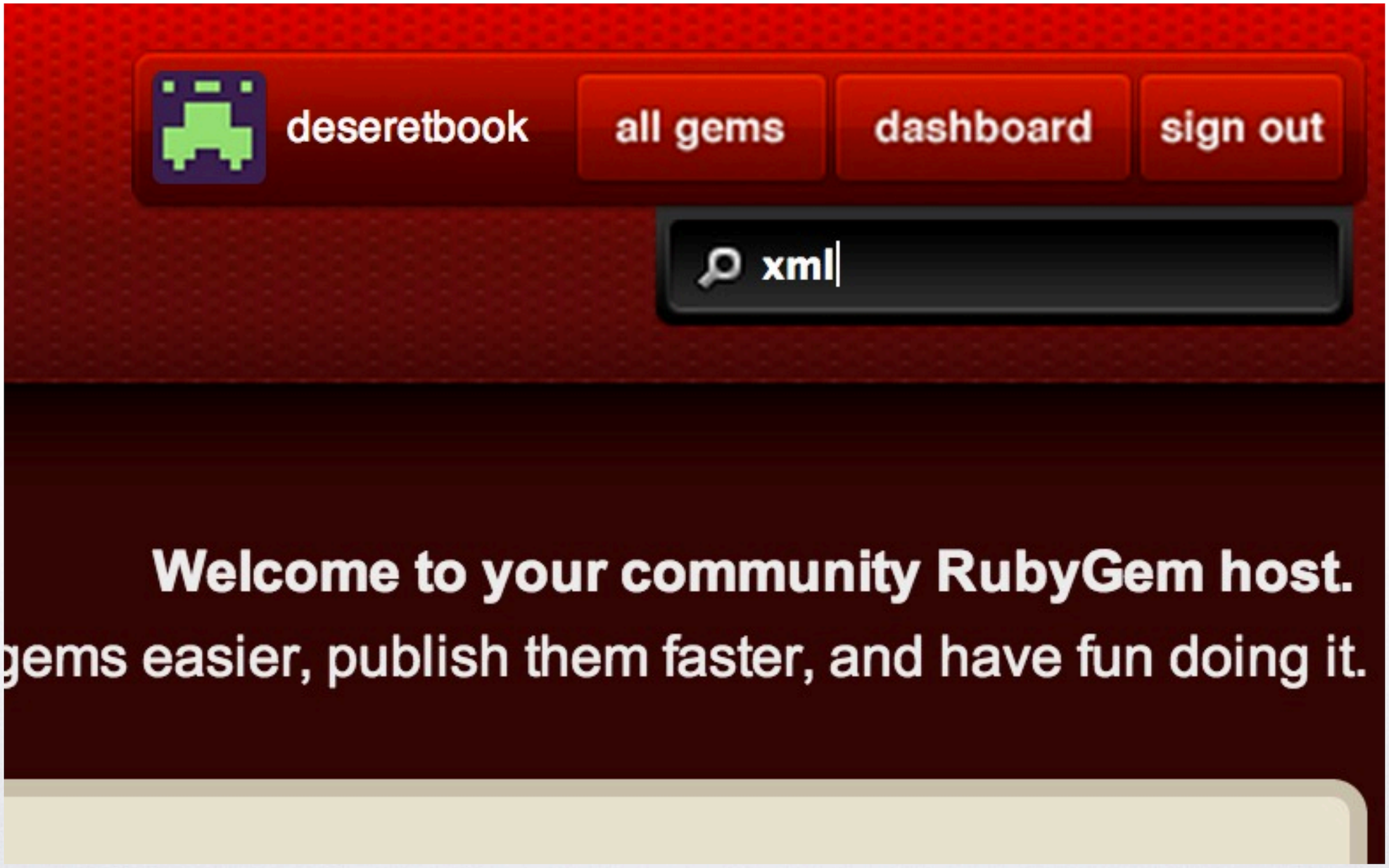
Instead, let's have someone else do the work.

Let's find a gem.

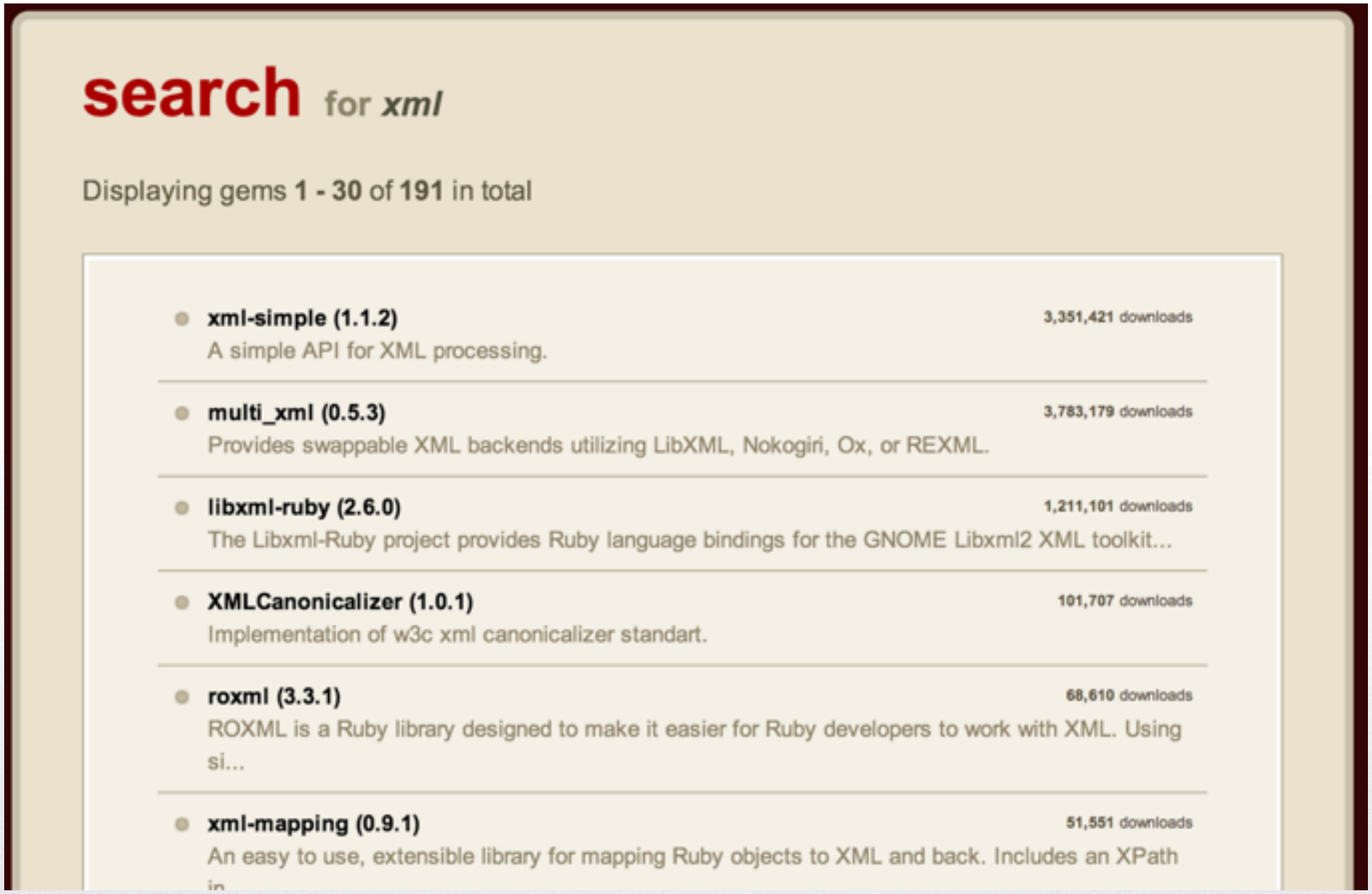
Where do we go first?

Where do we go first?

Search:


deseretbook
all gems
dashboard
sign out
xml
Welcome to your community RubyGem host.
gems easier, publish them faster, and have fun doing it.

Whoa! 191 total!



“XML rhymes with h*ll.”

- Nokigiri
- Hpricot
- libxml
- REXML
- Ox

Other resources:

Ruby Toolbox

Stack Overflow

Other Rubyists*

Other resources:

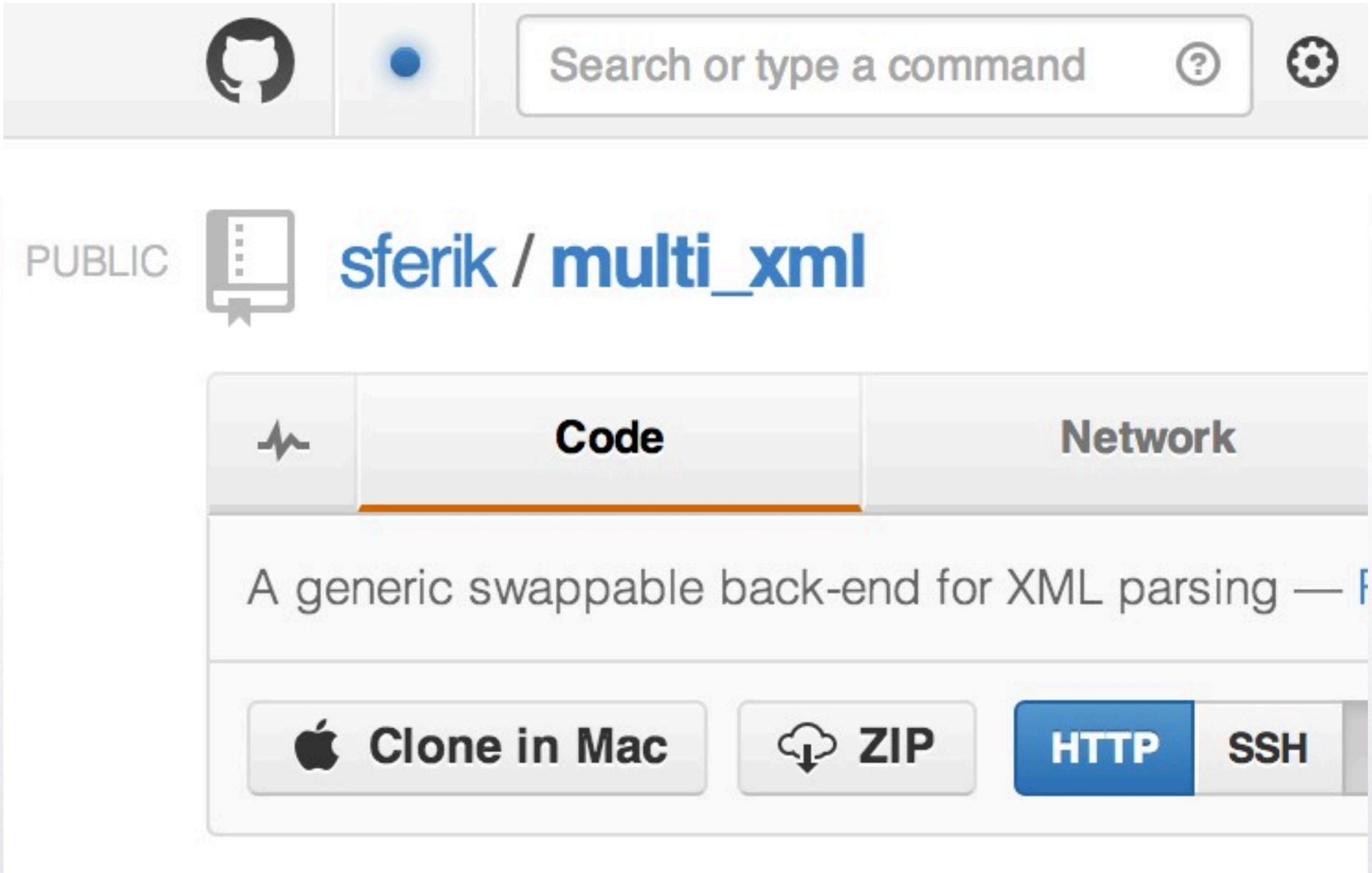
Ruby Toolbox

Stack Overflow

Other Rubyists*

* most awesome way

Ah-ha!

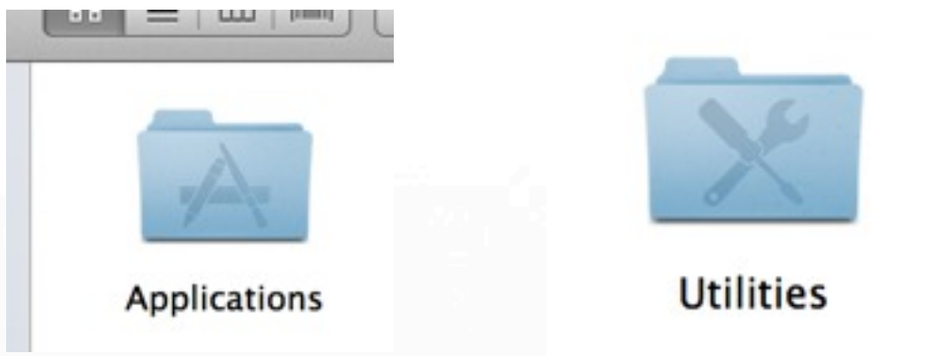

Search or type a command
PUBLIC
sferik / multi_xml
Code
Network
A generic swappable back-end for XML parsing —
Clone in Mac
ZIP
HTTP
SSH

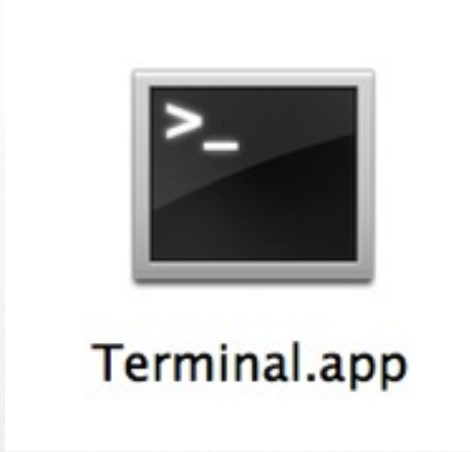
So, we decided.

Now what?

Installing a gem

Mac:
Open a terminal window





Windows:
Open a command prompt

+ R

“cmd” + enter

mnielsen — bash — 81×25

```
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ sudo gem install multi_xml
```

```
sudo gem install multi_xml
```

sudo: unix command, means "run this as root"

gem: the 'gem' program

install: what we want the 'gem' program to do

multi_xml: what we want installed

Other useful ‘gem’ commands:

list - get a list of installed gems

upgrade - upgrades the specified gem

uninstall - removes an installed gem

mnielsen — bash — 81×25

```
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ sudo gem install multi_xml
Password:
Sorry, try again.
Password:
Successfully installed multi_xml-0.5.3
1 gem installed
Matthew-Nielsens-MacBook-Pro:~ mnielsen$
```

Using a gem

mnielsen — bash — 80×25

```
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ subl parse_xml.rb
```

Ruby 1.8.x only:

```
1   require 'rubygems'
```

```
1  require 'rubygems'
2  require 'multi_xml'
```

“require” means load this code into the current code.

Read the instructions: It's OK!

Usage Examples

Lots of Ruby libraries utilize XML parsing in some form, and everyone has their favorite XML library. In order to best support multiple XML parsers and libraries, `multi_xml` is a general-purpose swappable XML backend library. You use it like so:

```
require 'multi_xml'

MultiXml.parser = :ox
MultiXml.parser = MultiXml::Parsers::Ox # Same as above
MultiXml.parse('<tag>This is the contents</tag>') # Parsed using Ox

MultiXml.parser = :libxml
MultiXml.parser = MultiXml::Parsers::Libxml # Same as above
MultiXml.parse('<tag>This is the contents</tag>') # Parsed using LibXML

MultiXml.parser = :nokogiri
MultiXml.parser = MultiXml::Parsers::Nokogiri # Same as above
MultiXml.parse('<tag>This is the contents</tag>') # Parsed using Nokogiri

MultiXml.parser = :rexml
MultiXml.parser = MultiXml::Parsers::Rexml # Same as above
MultiXml.parse('<tag>This is the contents</tag>') # Parsed using REXML
```

```ruby
require 'rubygems'
require 'multi_xml'

the_xml = File.open('configuration.xml').read
```

```ruby
require 'rubygems'
require 'multi_xml'

the_xml = File.open('configuration.xml').read

parsed_xml = MultiXml.parse(the_xml)
```

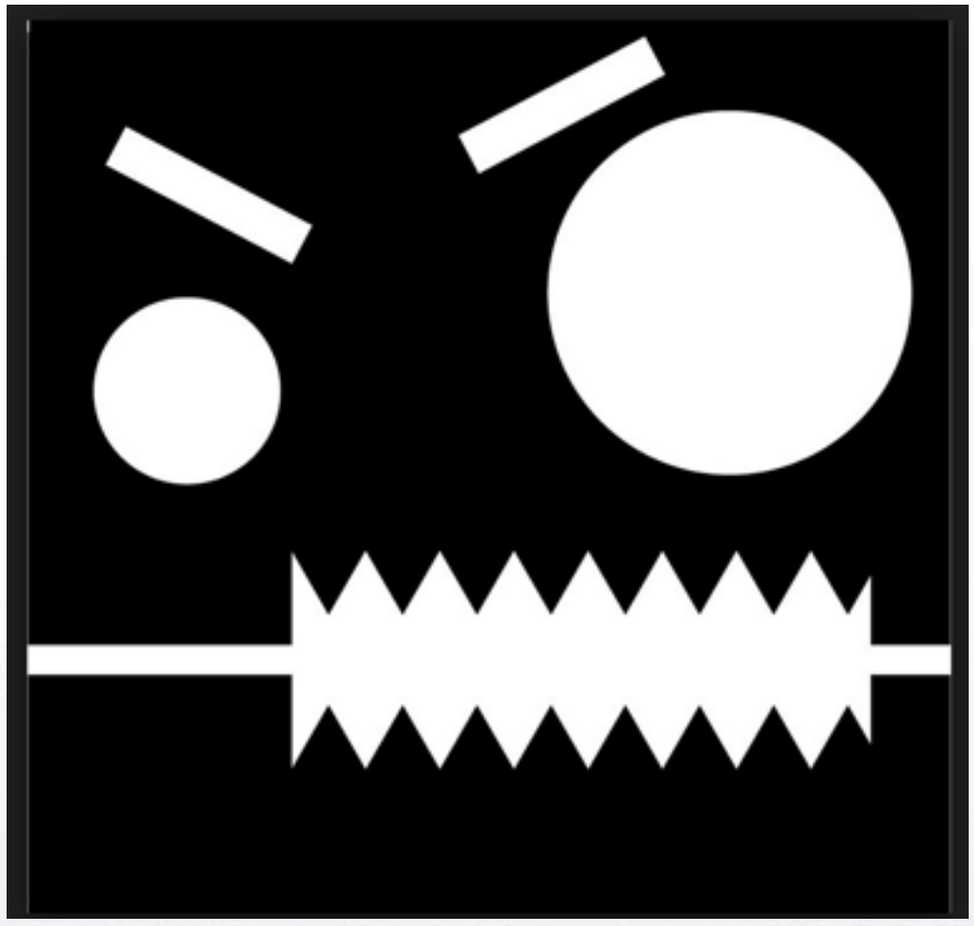
```ruby
require 'rubygems'
require 'multi_xml'

require 'death_robot_controller'

the_xml = File.open('configuration.xml').read

parsed_xml = MultiXml.parse(the_xml)

conf = parsed_xml["configuration"]

DeathRobot.mode(conf["mode"])
DeathRobot.targets(conf["targets"])
conf["exludes"]["exclude"].each do |exclude|
  DeathRobot.exclude_target(exclude)
end
```

Great! Done!

Angry Robots are go!

http://zerooso.deviantart.com/art/Angry-Robot-178716158

Great! Done!



Angry Robots are go!

http://zerooso.deviantart.com/art/Angry-Robot-178716158

Next step:

Get the code to the robots

Getting our code to the robots is the easy part.

Getting our code to the robots is the easy part.

How do we get the gems we need installed?

We could do it by hand?

We could do it by hand?

We have an ARMY of these things.
Do YOU want to do it by hand?

There's a gem for that!

The best way to manage your
application's dependencies
Bundler

“Bundler maintains a consistent environment for ruby applications.

It tracks an application's code and the rubygems it needs to run, so that an application will always have the exact gems (and versions) that it needs to run.”

How do we use it?

Ironically, it starts with a gem. How meta.

```
sudo gem install bundler
```

The Bundler “Gemfile”

Example Gemfile

```
source 'https://rubygems.org'

gem 'rails', '~> 3.2.13'
gem 'bootstrap-sass', '~> 2.1.0.0'
gem "backbone-on-rails", '~> 1.0.0.0'
gem 'jquery-rails', '2.2.1'
gem 'rails3-jquery-autocomplete',
  :git => "git://github.com/edsimpson/rails3-jquery-autocomplete.git"

group :production do
  gem 'activerecord-postgresql-adapter'
end

group :development, :test do
  gem 'ZenTest', '~>4.8.3'
  gem 'rspec-rails', '2.11.0'
  gem 'rspec-given', '2.4.0'
end
```

The real magic:

How Bundler handles multiple versions of the same gem.

What version will run?

```
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ gem list | grep rake
rake (10.0.2, 0.9.2.2)
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ rake -V
```

The newest one, of course!

```
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ gem list | grep rake
rake (10.0.2, 0.9.2.2)
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ rake -V
rake, version 10.0.2
Matthew-Nielsens-MacBook-Pro:~ mnielsen$
```

Run the old version with Bundler

```
Matthew-Nielsens-MacBook-Pro:bundler-test mnielsen$ gem list | grep rake
rake (10.0.2, 0.9.2.2)
Matthew-Nielsens-MacBook-Pro:bundler-test mnielsen$
```

```
1  source 'https://rubygems.org'
2
3  gem 'rake', '0.9.2.2'
```

```
Matthew-Nielsens-MacBook-Pro:bundler-test mnielsen$ bundle exec rake --version
rake, version 0.9.2.2
Matthew-Nielsens-MacBook-Pro:bundler-test mnielsen$
```

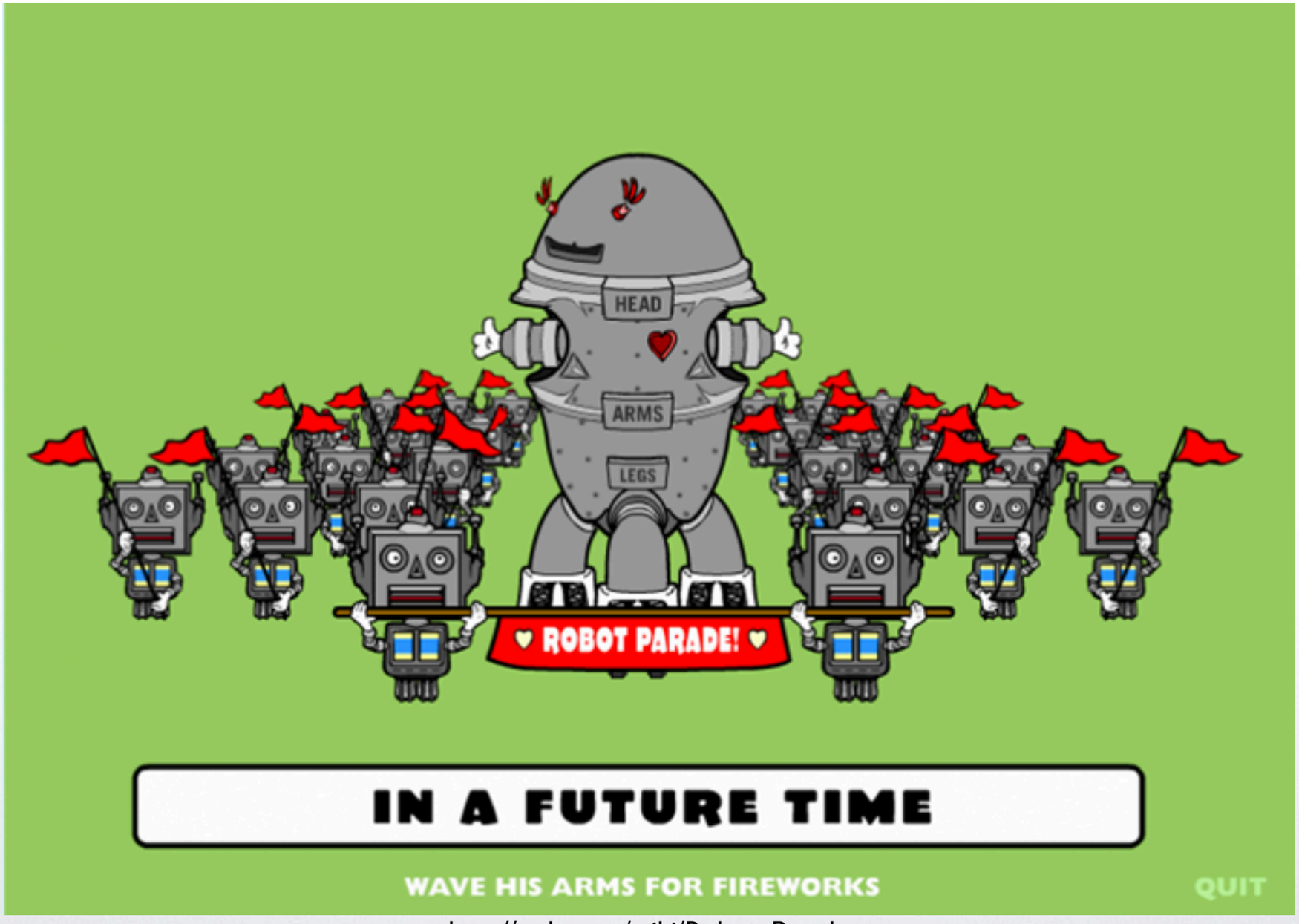
Our Gemfile

```
1   # Killer Death Robots Gemfile
2
3   source 'https://rubygems.org'
4
5   gem 'multi_xml', '0.5.3'
```

Installing the gems from our Gemfile

```
Matthew-Nielsens-MacBook-Pro:bundler-test mnielsen$ gem list | grep multi_xml
Matthew-Nielsens-MacBook-Pro:bundler-test mnielsen$ sudo bundle install
Fetching gem metadata from https://rubygems.org/..
Installing multi_xml (0.5.3)
Using bundler (1.2.2)
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled gem is installed.
Matthew-Nielsens-MacBook-Pro:bundler-test mnielsen$ gem list | grep multi_xml
multi_xml (0.5.3)
Matthew-Nielsens-MacBook-Pro:bundler-test mnielsen$
```

Awesome. Behold your robot army!



http://tmbw.net/wiki/Robot_Parade

Later

4PM, before a long weekend

Bug Report

Bug Report

"Sometimes, when the cannon is raised, the cannon control freeze up and the humans are able to destroy the death robot."

Problem:

```
class Cannon
  class Control
    include Math
    def get_inverse_cosine(x)
      acos(x)
    end
  end
end

# 1.8.7 :010 > x = Cannon::Control.new
# => #<Cannon::Control:0x10c4e2570>
# 1.8.7 :012 > x.get_inverse_cosine(0.1)
# => 1.47062890563334
```

Solutions

Change the “get_inverse_cosine()” method to return a rounded floating point value?

Change the “get_inverse_cosine()” method to return a rounded floating point value?

Can’t.

That code is used in another project that need the precision.

Change all the code in the cannon angle control to round the number returned from “get_inverse_cosine()”?

Change all the code in the cannon angle control to round the number returned from “get_inverse_cosine()”?

It occurs 314 times, do YOU want to stay late before a long weekend to change each of those?

Change all the code in the cannon angle control to round the number returned from “get_inverse_cosine()”?

It occurs 314 times, do YOU want to stay late before a long weekend to change each of those?

(Yes, it needs to be refactored. But not today)

Terribly Awesome Solution #1

“Monkey Patching”

http://en.wikipedia.org/wiki/Monkey_patch

“A **monkey patch** is a way to extend or modify the run-time code of dynamic languages without altering the original source code. This process has also been termed **duck punching**.[1]”

http://en.wikipedia.org/wiki/Monkey_patch

Re-opening classes and modules in Ruby

```ruby
Class Foo
  def bar
    puts "baz"
  end
end

puts Foo.new.bar
```

```ruby
Class Foo
  def bar
    puts "baz"
  end
end

puts Foo.new.bar
```

```
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ ruby mp_test_1.rb
baz
nil
Matthew-Nielsens-MacBook-Pro:~ mnielsen$
```

```ruby
require 'mp_test_1'

class Foo
  def bar
    'qux'
  end
end

puts Foo.new.bar
```

```ruby
require 'mp_test_1'

class Foo
  def bar
    'qux'
  end
end

puts Foo.new.bar
```

```
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ ruby mp_test_2.rb
baz
nil
qux
Matthew-Nielsens-MacBook-Pro:~ mnielsen$
```

The original source has not changed!

Every instance of Foo created after our “patch” will be effected, but only in our code base.

Fixing our bug

```
include Math
def get_inverse_cosine(x)
  acos(x)
end
```

The “Math” module is not a gem, but it’s still a library.

The “Math” module is not a gem, but it’s still a library.

Ruby gems are also (often) libraries.

The “Math” module is not a gem, but it’s still a library.

Ruby gems are also (often) libraries.

This section applies to gems and built-in libraries equally.

The Patch

```ruby
# math_patch.rb
module Math

  alias :original_acos :acos

  def acos(x)
    '%.2f' % original_acos(x)
  end

end
```

The Results

```
1   require 'math_patch'
2   class Cannon
3     class Control
4       include Math
5       def get_inverse_cosine(x)
6         acos(x)
7       end
8     end
9   end
10
11  # 1.8.7 :010 > x = Cannon::Control.new
12  # => #<Cannon::Control:0x10c4e2570>
13  # 1.8.7 :012 > x.get_inverse_cosine(0.1)
14  # => 1.47
15  #    ^^^^ THIS!
```

Why do this?

Why do this?

Makes the changes local only to this code base, doesn't require us to change the code base of the other project.

Why not?

Why not?

The code is assumed to be changed every
in the current running code.

Seriously: Monkey-patching is a last-resort option.
This example is hella-contrived.

Less Terrible, More Awesome Solution #2

Refinements
Ruby 2.0 feature

Yehuda Katz - http://yehudakatz.com/2010/11/30/ruby-2-0-refinements-in-practice/

“The first thing you need to understand is that the purpose of refinements in Ruby 2.0 is to **make monkey-patching safer**. Specifically, the goal is to make it possible to extend core classes, but to **limit the effect of those extensions** to a particular area of code.”

Yehuda Katz - http://yehudakatz.com/2010/11/30/ruby-2-0-refinements-in-practice/

Refinements are still officially “experimental”

But here is how they may work

unless they change :)

```ruby
Class Foo
  def bar
    puts "baz"
  end
end

puts Foo.new.bar
```

```
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ ruby mp_test_1.rb
baz
nil
Matthew-Nielsens-MacBook-Pro:~ mnielsen$
```

```ruby
# refinement_test.rb
module Qux
  refine Foo do
    def bar
      'qux'
    end
  end
end
```

```ruby
require 'mp_test_1'
require 'refinement_test'

class Foo1 < Foo; end

class Foo2 < Foo
  using Qux
end


puts Foo1.new.bar
puts Foo2.new.bar
```

```ruby
# refinement_test.rb
module Qux
  refine Foo do
    def bar
      'qux'
    end
  end
end
```

```ruby
require 'mp_test_1'
require 'refinement_test'

class Foo1 < Foo; end

class Foo2 < Foo
  using Qux
end


puts Foo1.new.bar
puts Foo2.new.bar
```

```
# This SHOULD print:
# baz
# qux
```

What currently happens:

What currently happens:

```
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ ruby refinements.rb
refinements.rb:2: warning: Refinements are experimental, and the behavior may ch
ange in future versions of Ruby!
refinements.rb:10:in `<class:Foo>': undefined method `using' for Foo:Class (NoMe
thodError)
        from refinements.rb:9:in `<main>'
Matthew-Nielsens-MacBook-Pro:~ mnielsen$
```

What currently happens:

```
Matthew-Nielsens-MacBook-Pro:~ mnielsen$ ruby refinements.rb
refinements.rb:2: warning: Refinements are experimental, and the behavior may ch
ange in future versions of Ruby!
refinements.rb:10:in `<class:Foo>': undefined method `using' for Foo:Class (NoMe
thodError)
        from refinements.rb:9:in `<main>'
Matthew-Nielsens-MacBook-Pro:~ mnielsen$
```

No Bueno, but will probably work by the end of 2013.

Will certainly work by Ruby 2.1.

Unless they remove it :)

Summary

We covered

We covered

- Ruby gems are just normal ruby code

We covered

- Ruby gems are just normal ruby code
- How to select and install a gem

We covered

- Ruby gems are just normal ruby code
- How to select and install a gem
- How to use a gem

We covered

- Ruby gems are just normal ruby code
- How to select and install a gem
- How to use a gem
- How to distribute gemset with Bundler

We covered

- Ruby gems are just normal ruby code
- How to select and install a gem
- How to use a gem
- How to distribute gemset with Bundler
- How to patch libraries at runtime with "Monkey Patches"

Questions?

Find me in person, or on Twitter: @xunker

Thank You.

Slide deck available at:
github.com/xunker/openwest2013