

# Advanced Topics in Ruby

# Aaron Patterson

- Email: [aaron.patterson@gmail.com](mailto:aaron.patterson@gmail.com)
- IM: [aaron.patterson@gmail.com](mailto:aaron.patterson@gmail.com)
- Phone: 206-228-1736
- Tuesday Nights, 6-9pm

# Ryan Davis

- Email: [ryand-uwruby09@zenspider.com](mailto:ryand-uwruby09@zenspider.com)
- Tuesday Nights, 6-9pm

**HALP!!!!**



**i'z stuk**

# Where To Get Help

- UW-Ruby mailing list  
<http://www.zenspider.com/mailman/listinfo/uw-ruby>
- IRC: [#seattle.rb](http://irc.freenode.org)
- ruby-talk mailing list  
<http://www.ruby-lang.org/en/community/mailling-lists/>
- Email, IM, Call ME!
- Seattle.rb meetings

# Format

- 3 hour class
  - ~90 minute lecture
  - ~30 minute meet and greet
  - ~60 minute exporation

Interrupt us *anytime*  
with questions!

# Homework

- Due 9 days (Friday) after assigned.
- Absolutely no late work accepted.

# Term Syllabus

- Metaprogramming
- Concurrency/Network Programming
- Patterns
- Alternate Ruby Implementations
- Packaging and Extending Ruby

# Metaprogramming

```
class Alphabet
  ('a'..'z').each do |letter|
    define_method(letter) do
      puts letter
    end
  end
end
```

```
abcs = Alphabet.new
abcs.a
```

# Threading

```
threads = (0..2).map {  
  Thread.new do  
    do_something_expensive  
  end  
}  
  
puts "hello!!!"  
threads.each { |thread| thread.join }
```

# Network Programming

```
require 'socket'

server = TCPServer.new(12321)
while session = server.accept
  session.puts Time.new
  session.close
end
```

# Patterns in Ruby

```
class Printer  
  include Singleton  
end
```

```
printer.instance
```

# Extending with C

```
require 'rubygems'
require 'inline'

class MyTest
  def factorial(n)
    f = 1
    n.downto(2) { |x| f *= x }
    f
  end

  inline do |builder|
    builder.c "
    long factorial_c(int max) {
      int i=max, result=1;
      while (i >= 2) { result *= i--; }
      return result;
    }"
  end
end

puts MyTest.new.factorial(10)
puts MyTest.new.factorial_c(10)
```

# Testing Frameworks and Strategies

# Debugging

```
(rdb:1) list
[82, 91] in /Library/Ruby/Gems/1.8/gems/mechanize-0.7.5/lib/www/
mechanize.rb
 82     def initialize
 83         # attr_accessors
 84         @cookie_jar      = CookieJar.new
 85         @log              = nil
 86         @open_timeout    = nil
=> 87         @read_timeout   = nil
 88         @user_agent      = AGENT_ALIASES['Mechanize']
 89         @watch_for_set   = nil
 90         @ca_file         = nil # OpenSSL server certificate file
 91
(rdb:1)
```

# Implementations

- Ruby 1.9 (YARV)
- JRuby
- Rubinius
- IronRuby

What would you like to  
learn this quarter?

# Tonight's Topics

- Open Classes
- **alias**
- singleton methods
- **send**

# Open Classes

- Define your class anywhere
- Define your class any time
- Extend a class anywhere
- Extend a class anytime

# Extending

```
class MyClass
  def my_first_method
    "the first method"
  end
end
```

```
# ... Some other file ...
class MyClass
  def my_second_method
    "the second method"
  end
end
```

# Extend Core Classes

```
class String
  def dotify
    split('').join('.')
  end
end

puts "Hello world".dotify
```

# Override Existing Methods

```
class String
  def +(other)
    "#{self} HI!! #{other}"
  end
end

puts "Hello" + "World"
```

# Oh Noes!!!

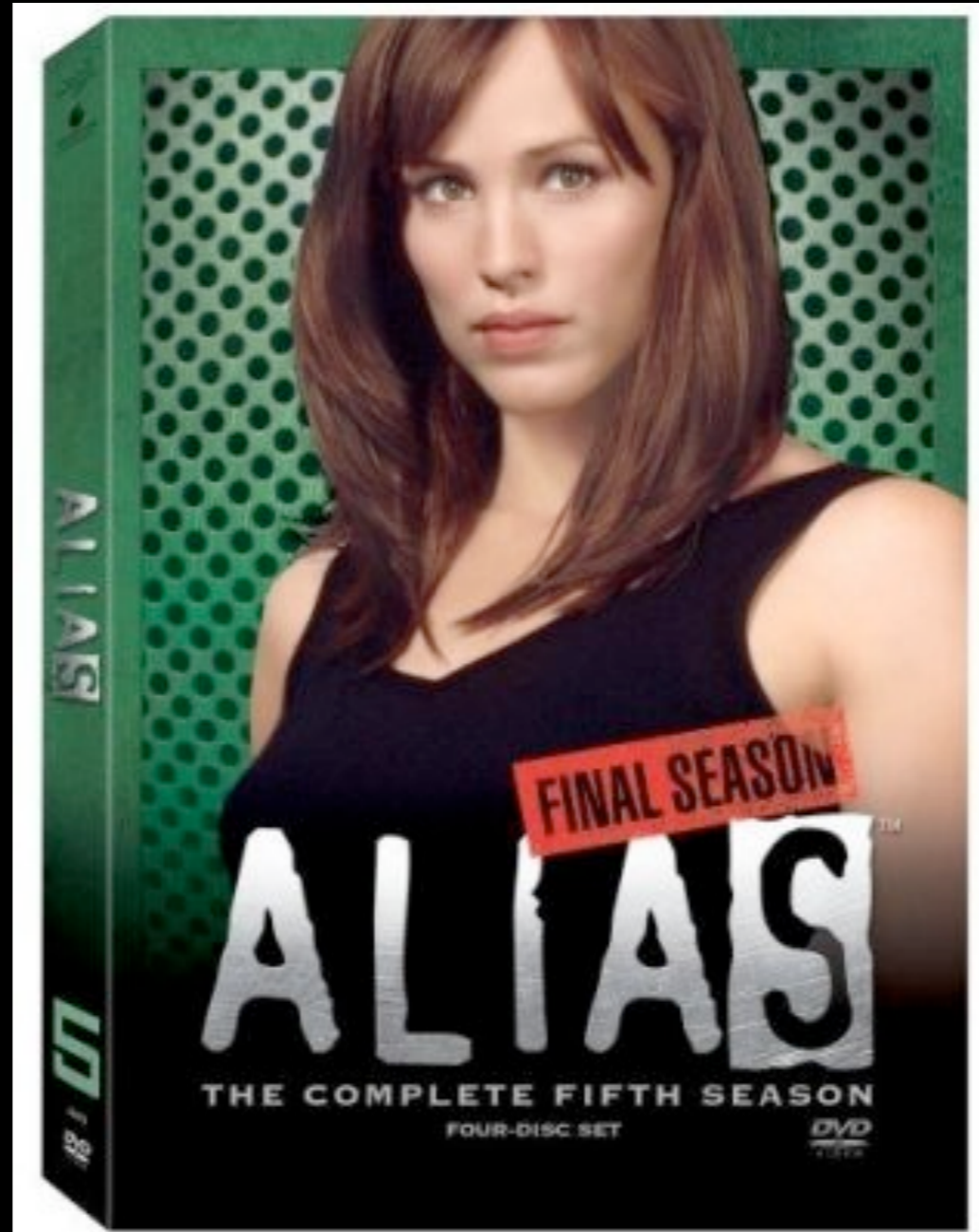
```
% ruby -w test.rb  
test.rb:2: warning: method redefined; discarding old +  
Hello HI!! World  
%
```

# USE alias

```
class String
  alias :old_plus :+
  def +(other)
    "#{self} HI!! #{other}"
  end
end

puts "Hello" + "World"
```

# What is “alias”?



# alias:

- Not a TV show
- Super secret **ruby** keyword
- **Adds a name**

# Alias in Action

```
class String
  alias :old_plus :+
  def +(other)
    puts "Plus got called"
    old_plus(other)
  end
end

puts "Hello" + "World"
```

# Stuff You can Alias

- methods
- globals

# Extending with Modules

# Object#extend

```
module DirtyHarry
  def six_shots?
    false
  end

  def only_five?
    true
  end
end

shooter.extend(DirtyHarry)
```

# Monkey Patching



# When To Monkey Patch

- Never
- Never
- Never
- Never
- Okay, sometimes.....

# Monkey Patching HOWTO:

1. Reopen a class
2. Redefine a method

# Dangers

- Debugging problems
- Confusing code
- Maintenance problems

# When to Monkey Patch

- Fixing a bug in code you don't own

# Singleton Methods

For when you only need one.

# Every object in Ruby:

- Has attributes
- Has methods
- Has constants
- Has a class

# Classes may **Contain** Methods

```
class IContainMethods
  def let_me_out!
    . . .
  end
end
```

# Classes may **Have** Methods

```
class IHaveMethods
  def self.im_singleton!
    ...
  end
end
```

```
IHaveMethods.im_singleton!
```

# We May Define Methods

- Classes
- Modules
- Any instance!

# No, Really. Any Instance

```
string1 = "I am special!"  
string2 = "Hello world"
```

```
def string1.callme!  
  puts "Thanks for calling"  
end
```

```
string1.callme!  
string2.callme!
```

# Even Your Objects

```
class Jukebox
  def initialize
    @songs = ['Kids', 'Electric Feel']
  end
end
```

```
j = Jukebox.new
def j.my_songs; @songs; end
```

```
p j.my_songs
```

# The Chevron

```
j = Jukebox.new
class << j
  def my_songs
    @songs
  end
end
```

# Many Static Methods

```
class SomeRailsModel
  class << self
    def find_something
      . . .
    end

    def find_another_thing
      . . .
    end
  end
end

SomeRailsMode.find_something
```

# Sending a Message

```
obj.send(symbol [, args...])
```

# The **send** method

- takes a **symbol**
- and a **list of arguments**
- and **sends** them to **obj** as a method call
- `obj.send(symbol [, args...])`

# Example

```
class Foo
  def bar(argument)
    puts "you called me with #{argument}"
  end
end
```

```
foo = Foo.new
foo.bar("baz")
foo.send(:bar, "baz")
```

```
x = "bar"
foo.send("#{x}", "baz")
```

# Hello Major Tom

Object#respond\_to?

# Object#respond\_to?

```
object = Struct.new(:phil_collins).new  
  
puts object.respond_to?(:phil_collins)  
puts object.respond_to?(:pat_benatar)
```

Any Questions?