

Advanced Topics in Ruby

Aaron Patterson

- Email: aaron.patterson@gmail.com
- IM: aaron.patterson@gmail.com
- Phone: 206-228-1736
- 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 me *anytime*
with questions!

Term Syllabus

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

Tonight's Topics

Tonight's Topics

- Open Classes

Tonight's Topics

- Open Classes
- **alias**

Tonight's Topics

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

Tonight's Topics

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

Open Classes

Open Classes

- Define your class anywhere

Open Classes

- Define your class anywhere
- Define your class any time

Open Classes

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

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!!!

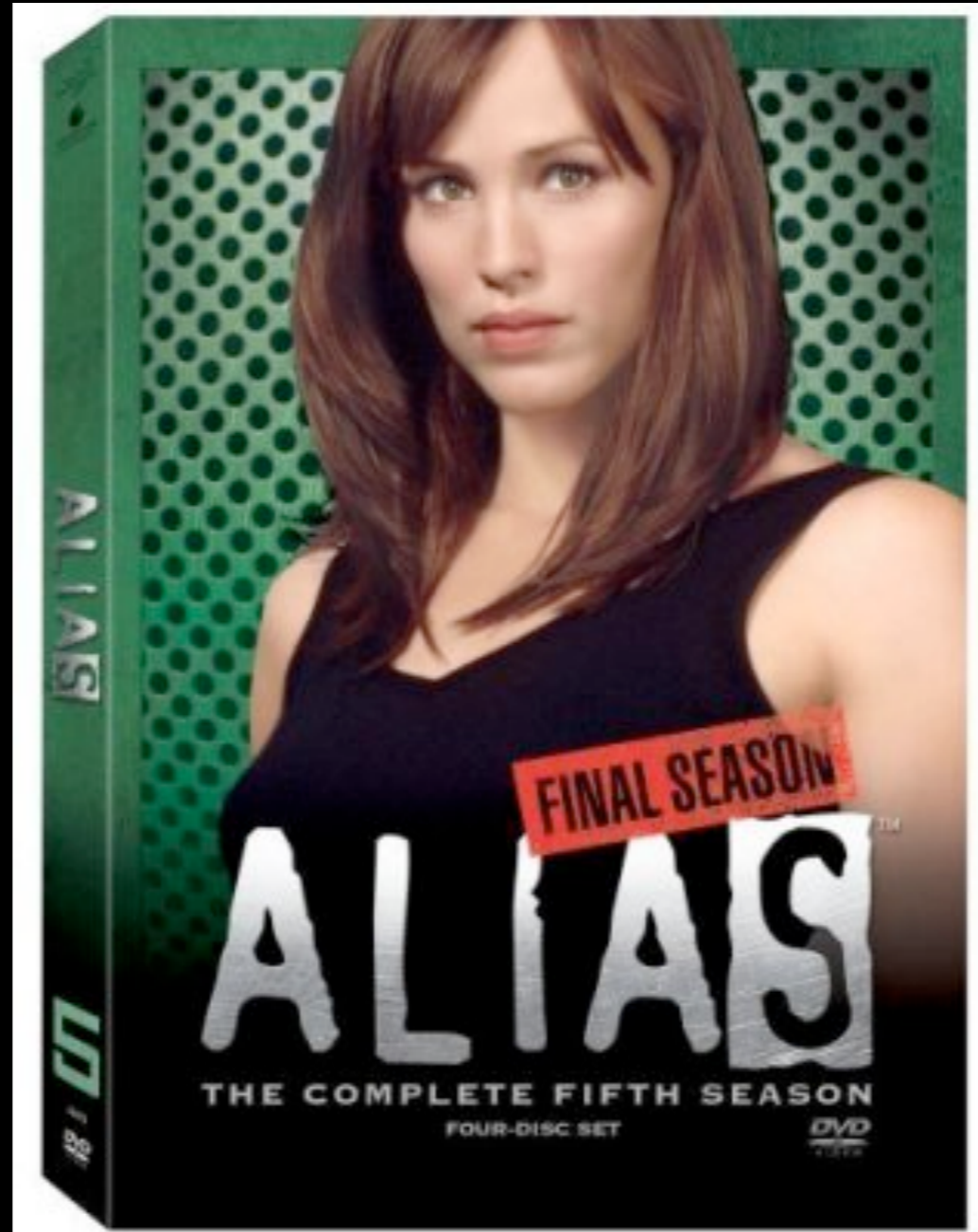
```
% ruby -w test.rb  
test.rb:2: warning: method redefined; discarding old +  
Hello HI!! World  
%
```

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

You'll shoot
your eye out
kid.



When To Monkey Patch

When To Monkey Patch

- Never

When To Monkey Patch

- Never
- Never

When To Monkey Patch

- Never
- Never
- Never

When To Monkey Patch

- Never
- Never
- Never
- Never

When To Monkey Patch

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

Monkey Patching HOWTO:

Monkey Patching HOWTO:

- I. Reopen a class

Monkey Patching

HOWTO:

1. Reopen a class
2. Redefine a method

Dangers

Dangers

- Debugging problems

Dangers

- Debugging problems
- Confusing code

Dangers

- Debugging problems
- Confusing code
- Maintenance problems

When to Monkey Patch

- Fixing a bug code you don't own

Singleton Methods

For when you only need one.

Every object in Ruby:

- Has attributes
- Has methods
- Has constants

Classes may **Contain** Methods

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

Classes may **Have** Methods

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

```
IHaveMethods.im_static!
```

We May Define
Methods

We May Define Methods

- **Classes**

We May Define Methods

- Classes
- Modules

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

The `send` method

- takes a `symbol`

The **send** method

- takes a **symbol**
- and a **list of arguments**

The **send** method

- takes a **symbol**
- and a **list of arguments**
- and **sends** them to **obj** as a method call

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#send

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

Any Questions?