Method Missing

method_missing

When you send a message to a Ruby object, Ruby looks for a method to invoke with the same name as the message you sent.

You can override method_missing anywhere along that method lookup path, and tell Ruby what to do when it can't find a method.

Developers often use this to create domain-specific languages (DSLs) in Ruby.

Ruby's Method Lookup Path

1. The current self object's own instance methods.
2. In the list of instance methods that all objects of that class share
3. In each of the included modules of that class, in reverse order of inclusion
4. The superclass
5. Then in the superclass's included modules, all the way up until it reaches the class Object.
6. If it still can't find a method, the very last place it looks is in the Kernel module, included in the class Object.
7. And there, if it comes up short, it calls method_missing

http://www.thirdbit.net/articles/2007/08/01/10-things-you-should-know-about-method_missing/

```
class Thing
  def method_missing(m, *args, &block)
    puts "There's no method called #{m} here -- please try again."
    puts "parameters = #{args.inspect}"
  end
end
>> t = Thing.new
>> t.anything("ddd",3)
There's no method called anything here -- please try again.
parameters = ["ddd", 3]
=> nil
```

Variable Arguments

```
def test(*params)
    params.each_with_index do |arg, i|
        puts "#{i} #{arg}"
     end
end
```

Variable Arguments

```
def test(a, b, *params)
    params.each_with_index do |arg, i|
        puts "#{i} #{arg}"
     end
end
```