



The Evil Tale Of Evil
Pair

Mathilde Mouw

[@MathildeMouw](#) [#sfdotrb](#)

Evil Pair



ave
hl
now
au yar

Code retreat

One day

One problem, language-agnostic

Test-drive pairing on the solution

Every 45 minutes, start over

Code retreats...

Allow you to learn about pairing, the language you choose to work in, the problem, OO design, git hell, and more

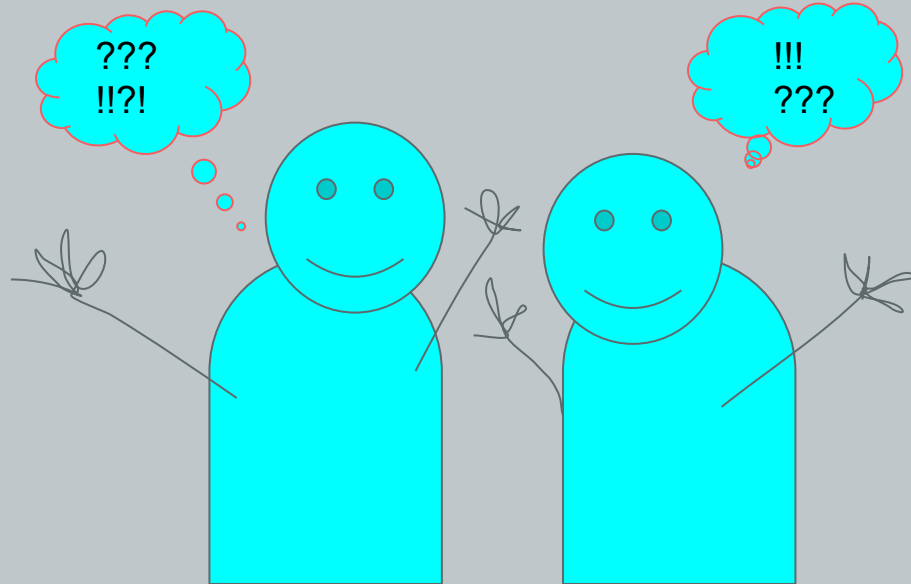
Are very tiring

Make you quickly better at solving one problem

Code retreat energy

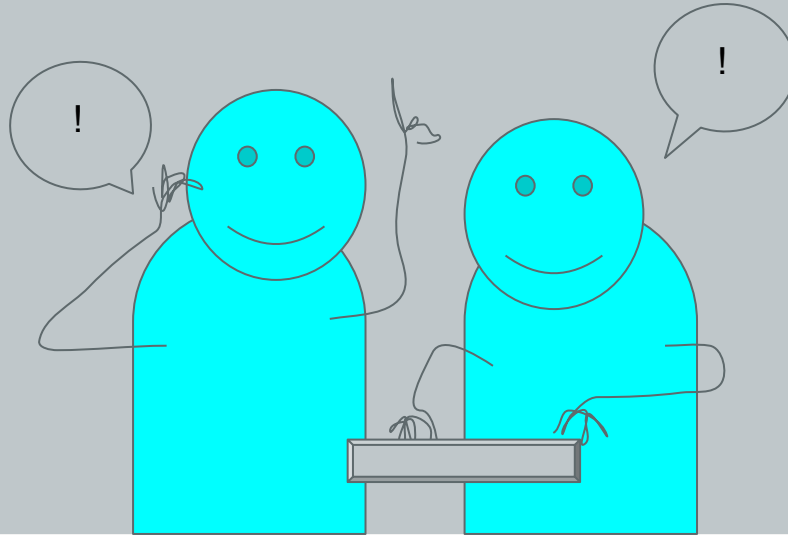
Pairs confused

But excited



Code retreat energy

Getting the hang of it

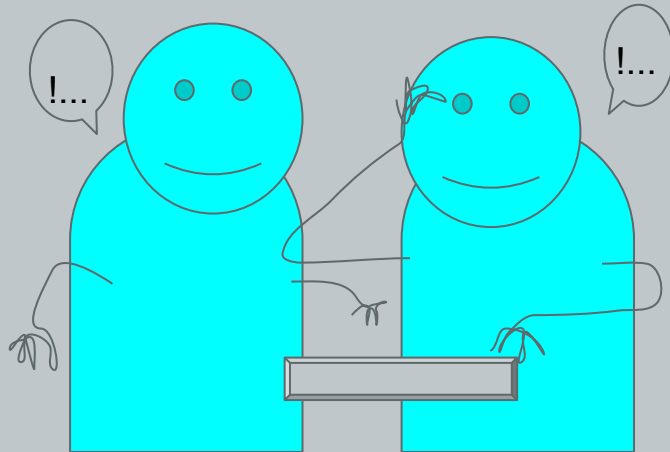


Code retreat energy

Pairs are getting sleepy,

Speeding through

is now less fun



Code retreat + Evil Pair

One 45 minutes round

Same problem

Test-drive pairing on the solution

Make each test pass in the least helpful way possible



Why at the end?

Shared experience/conventions





Why at the end?

Shared experience/conventions

Coffee has worn off = creativity hard





Why at the end?

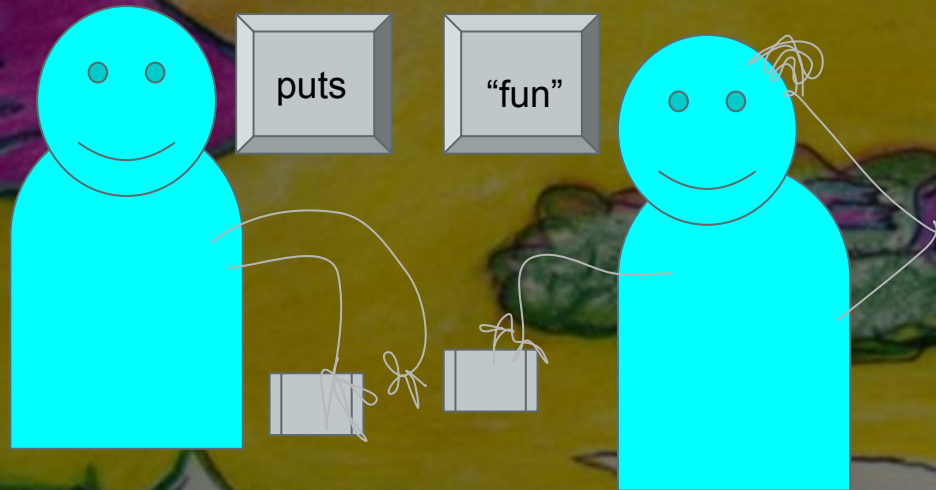
Shared experience/conventions

Coffee has worn off = creativity hard

Makes it all gel together



Pair

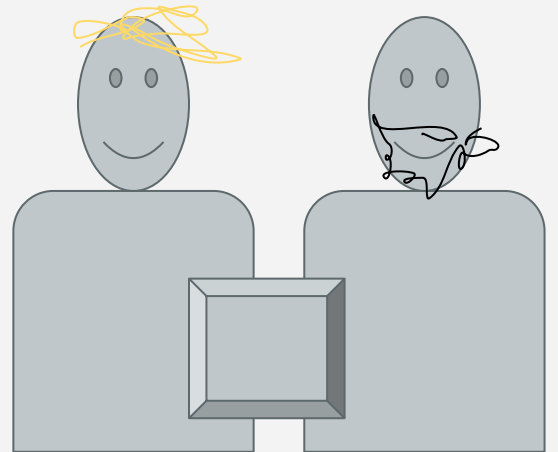


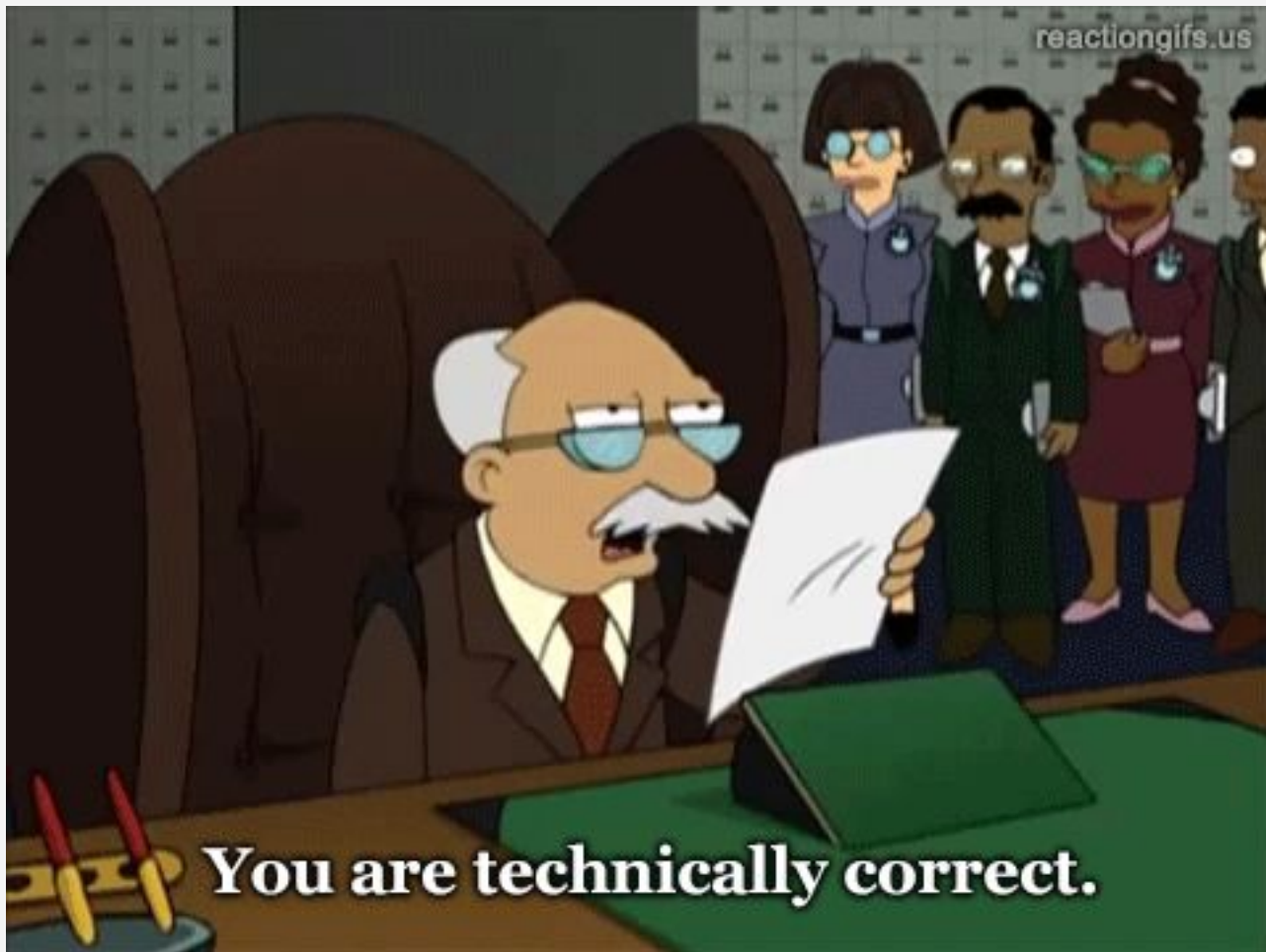
Evil Pair



Find minimum number of coins that make a given value

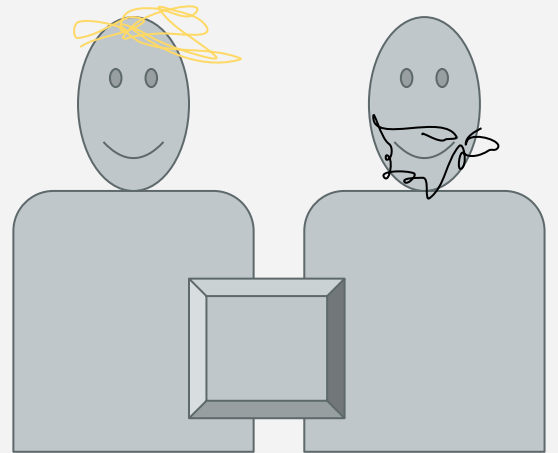
Given a value V , if we want to make change for V cents, and we have infinite supply of each of $C = \{ C_1, C_2, \dots, C_m \}$ valued coins, what is the minimum number of coins to make the change?

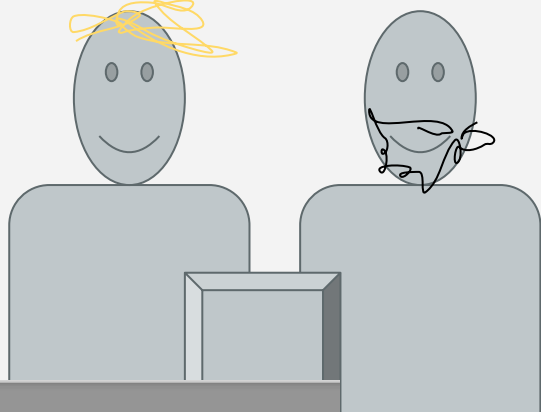




You are technically correct.

Find minimum number of coins
that make a given value



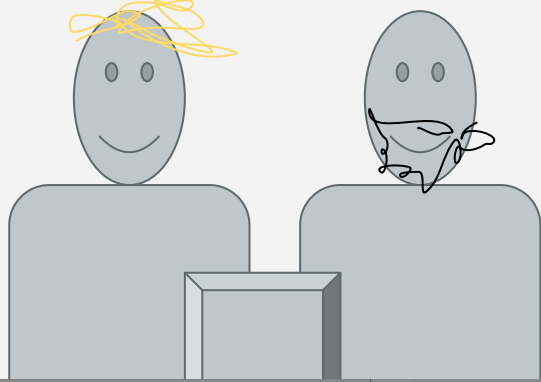


coins.rb x coins_spec.rb

```
1 require "./coins"
2
3 def assert(message)
4   if message
5     puts "*"
6   else
7     puts "something went wrong"
8   end
9 end
10
11 assert(value_to_least_coins(1) == "1 penny")
12
13 puts "tests passed"
```

coins.rb coins_spec.rb x

```
1 def value_to_least_coins _arg
2   "1 penny"
3 end
```

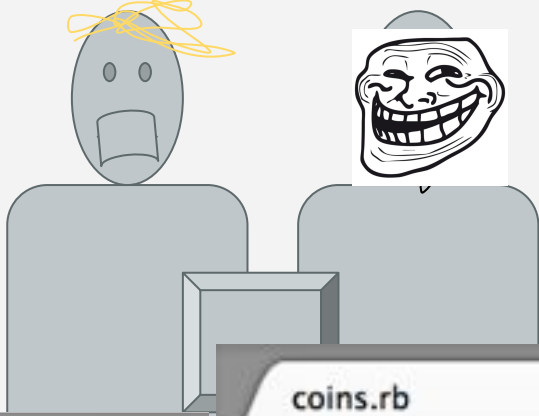


coins.rb × coins_spec.rb ●

```
1 require "./coins"
2
3 def assert(message)
4   if message
5     puts "*"
6   else
7     puts "something went wrong"
8   end
9 end
10
11 assert(value_to_least_coins(1) == "1 penny")
12 assert(value_to_least_coins(25) == "1 quarter")
13
```

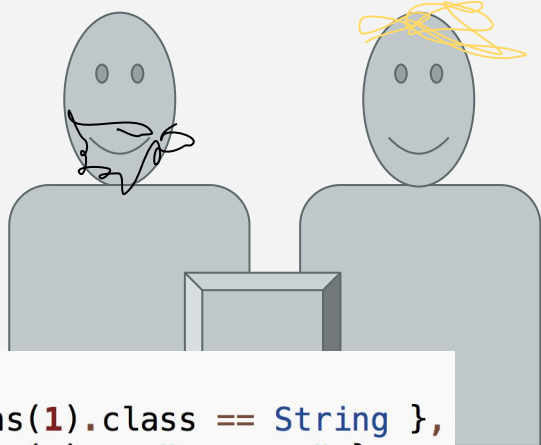
coins.rb × coins_spec.rb ×

```
1 @attempt = 1
2 def value_to_least_coins _arg
3   result = case @attempt
4     when 1 then "1 penny"
5     when 2 then "1 quarter"
6   end
7
8   @attempt += 1
9   return result
10 end
```



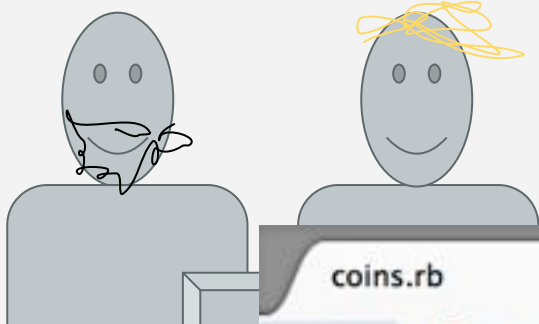
```
coins.rb x coins_spec.rb
8   end
9   end
10
11  def assert_random(blocks)
12    blocks.shuffle.map{ |block| assert(block.call) }
13  end
14
15  assert_random([
16    Proc.new { value_to_least_coins(1) == "1 penny" },
17    Proc.new { value_to_least_coins(25) == "1 quarter" }
18  ])
19
20
21  pennies = (2..4).to_a.map{|n| Proc.new {
22    value_to_least_coins(n) == "#{n} pennies"
23  }}
24
25  assert_random(pennies)
26
```

```
coins.rb x
1  class Rao
2    def == something
3      true
4    end
5  end
6
7  def value_to_least_coins arg
8    Rao.new
9  end
```



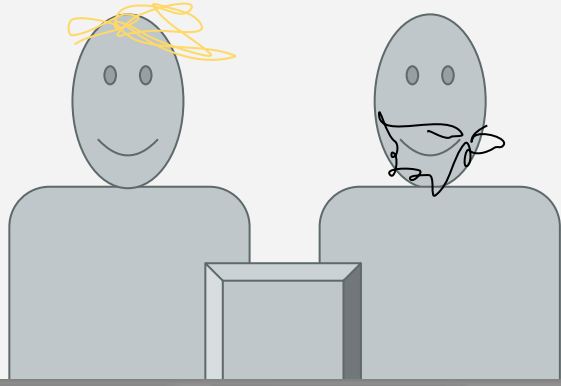
```
assert_random([  
  Proc.new { value_to_least_coins(1).class == String },
```

```
coins.rb  coins_spec.rb  x  
1 class String  
2   def == something  
3     true  
4   end  
5 end  
6  
7 def value_to_least_coins arg  
8   String.new  
9 end
```



```
coins.rb × coins_spec.rb ●  
1 String.freeze  
2
```

```
coins.rb × coins_spec.rb ×  
1 class Rao < String  
2   def class  
3     String  
4   end  
5  
6   def == (arg)  
7     true  
8   end  
9 end  
10  
11 def value_to_least_coins arg  
12   Rao.new  
13 end
```



coins.rb × coins_spec.rb ×

```
1
2 class String
3   def self.inherited arg
4     raise "nah homie"
5   end
6 end
7
8 String.freeze
9
```

coins.rb × coins_spec.rb ×

```
1 #this doesn't do anything
2 def unless(arg)
3   puts "I GOT HERE"
4   true
5 end
6
7
8 def value_to_least_coins number_of_cents
9   case
10    when number_of_cents == 1
11      "1 penny"
12    when number_of_cents < 5
13      "#{number_of_cents} pennies"
14    when number_of_cents == 5
15      "1 nickel"
16    when number_of_cents == 6
17      "1 nickel, 1 penny"
18    when number_of_cents < 10
19      "1 nickel, #{number_of_cents - 5} pennies"
20    else
21      "1 quarter"
22    end
23 end
24
```


on
of

ave

hl

now

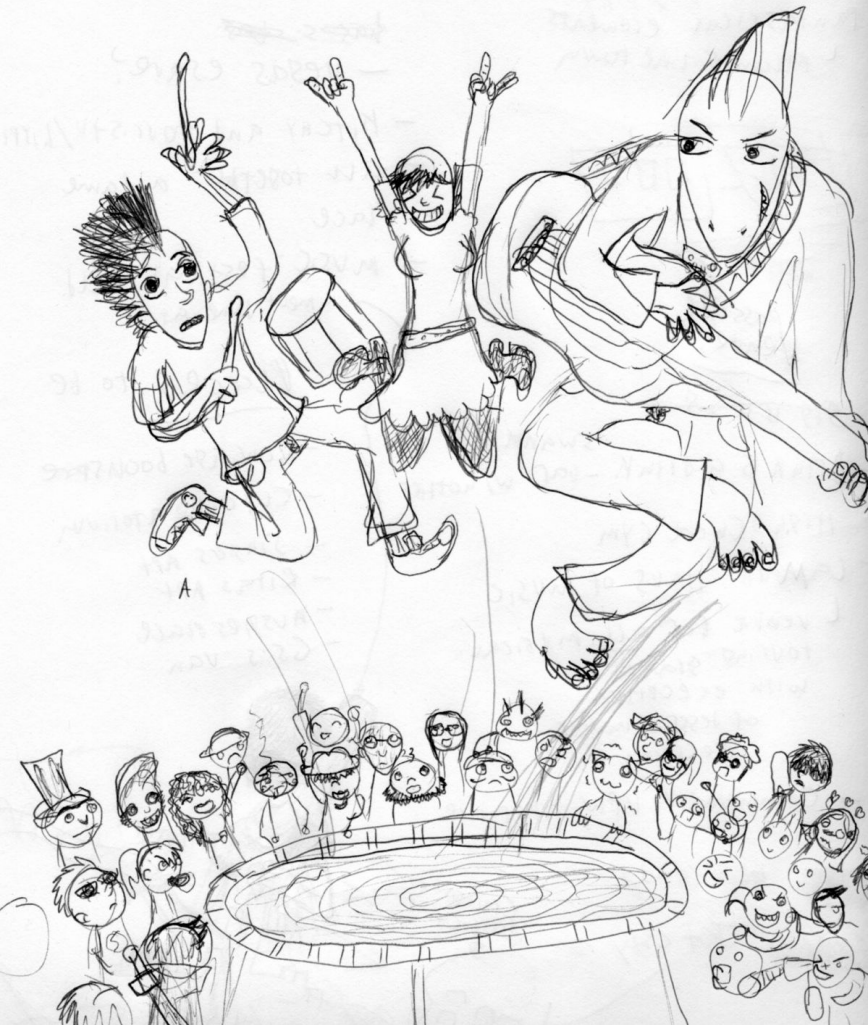
au yar





Contributing

1. Fork it
2. Create your feature branch (`git checkout -b my-new-feature`)
3. Commit your changes (`git commit -am 'Added some feature'`)
4. Run the tests (`bundle exec rspec`)
5. Push to the branch (`git push origin my-new-feature`)
6. Create new Pull Request



Evil Rails

Add a way for Users in an organization's app to see what committees they are on

This time we had Rails, Rspec, ActiveRecord to push Evil Pair toward.

Evil Rails

```
committee_spec.rb x
1 require "spec_helper"
2
3 describe Committee do
4   it "should have users" do
5     expect(described_class.new.users.count).to eq 0
6   end
7 end
8
```

```
committee_spec.rb x  committee.rb x
1 class Committee
2   attr_reader :users
3
4   def initialize
5     @users = []
6   end
7 end
8
```

Committee

should have users

Finished in 0.06165 seconds (files took 2.93 seconds to load)

1 example, 0 failures

Evil Rails

```
committee_spec.rb x  committee.rb x
1  require "spec_helper"
2
3  describe Committee do
4    let(:subject){ described_class.new }
5    let(:user) { User.create(username: "testerton", email: "testerton@aol.com") }
6
7    it "has users" do
8      expect(subject.users.count).to eq 0
9    end
10
11   it "can get more users" do
12     subject.users << user
13     subject.save!
14     expect(subject.users.first).to eq user
15   end
16 end
17
```

Evil Rails

```
committee_spec.rb x  committee.rb x
1  class Committee
2    attr_reader :users
3
4    def users
5      @users = []
6    end
7
8    def save!
9      puts "=^..^="
10   end
11 end
12
```

Evil Rails

```
has users  
=^..^=  
can get more users
```

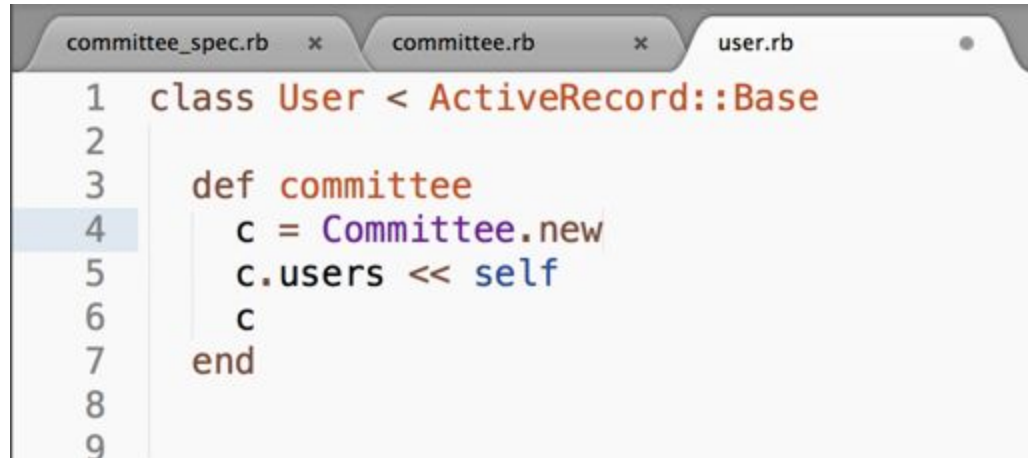
Evil Rails

```
committee_spec.rb x committee.rb x user.rb x
1 require "spec_helper"
2
3 describe Committee do
4   let(:subject){ described_class.new }
5   let(:user) { User.create(username: "testerton", email: "testerton@aol.com") }
6   it "has users" do
7     expect(subject.users.count).to eq 0
8   end
9
10  it "can get more users" do
11    subject.users << user
12    subject.save!
13    expect(subject.users.first).to eq user
14
15    #expect(subject.persisted?).to be_truthy
16    expect(user.committee).to eq subject
17  end
18 end
19
```

Evil Rails

```
committee_spec.rb x committee.rb x user.r
1 class Committee
2   attr_reader :users
3
4   def self.users
5     @@users ||= []
6   end
7
8   def self.save!
9     puts "=^..^="
10  end
11
12  def self.new
13    self
14  end
15 end
16
```

Evil Rails



```
committee_spec.rb x committee.rb x user.rb
1 class User < ActiveRecord::Base
2
3   def committee
4     c = Committee.new
5     c.users << self
6     c
7   end
8
9
```


Evil Rails

```
Committee
```

```
  has users
```

```
=^..^=
```

```
  can get more users
```

```
Finished in 0.11309 seconds (files took 2.82 seconds to load)
```

```
2 examples, 0 failures
```

Evil Rails

```
Committee
=^..^=
  can get more users
  has users (FAILED - 1)

Failures:

  1) Committee has users
     Failure/Error: expect(subject.users.count).to eq 0

       expected: 0
       got: 2

       (compared using ==)
     # ./spec/models/committee_spec.rb:7:in `block (2 levels) in <top (required)>'

Finished in 0.18487 seconds (files took 3.93 seconds to load)
2 examples, 1 failure
```



Learning from Evil



Learning from Evil

WWMGFDTST

Tips and Tricks



untitled

```
1 a = ["my", "cool", "words"]
2 a.map do |string|
3   string*2
4 end
5
6 #try to do something with `string` outside of the block
7 string == "Mathilde"
```

untitled

```
1 a = ["my", "cool", "words"]
2 a.map! do |string|
3   string*2
4 end
5
6
7
```

Impress your friends! How to be Evil

Metaprogramming

Monkey-patching methods

All the stuff they told you not to use (`@@class_variables`, for example!)

A hand-drawn illustration of a plant with red flowers and a central seed pod, overlaid with a semi-transparent grey box containing text. The drawing is done in a sketchy, colored-pencil style. The background is a light yellowish-green. The text is white with a red highlight on the word 'moral'.

The **moral** of the
story



Thank you!



[@MathildeMouw](#) [#sfdotrb](#)