Bow Before MiniTest

@adman65 - gh://twinturbo

MiniTest

vs

RSpec

vs

Test::Unit

vs

Cucumber

Forget Cucumber!

- Long term cucumber use is detrimental to your brain
- Makes you learn another syntax to write tests
- Managing large test suites is painful
- Small number of good use cases

MiniTest

vs

RSpec

vs

Test::Unit

vs

~~Cucumber~~

MiniTest & Test::Unit

- Test::Unit is MiniTest in Ruby 1.9
- Test::Unit is a compatibility layer around MiniTest
- This is for backward compatibility
- We can forget about Test::Unit too...

MiniTest

vs

RSpec

vs

~~Test::Unit~~

vs

~~Cucumber~~

MiniTest

vs

RSpec

Kampf!

Things a Test Framework Should Do

- Make it easy to write and maintain tests
- Run tests in random order
- Believe me, this will find bugs in your code
- Make you focus on testing your code and not learning the framework yourself

Basics: Writing Tests

```ruby
require 'minitest/autorun'

class TruthTest < MiniTest::Unit::TestCase
  def test_truth
    assert true
  end
end
```

```ruby
require 'rspec'

describe "Truth" do
  it "should be true" do
    true.should == true
  end
end
```

Speed: Timing 10K Tests

```ruby
require 'test_helper'

class TruthTest < MiniTest::Unit::TestCase
  10_000.times do |i|
    define_method "test_truth_#{i}" do
      assert true
    end
  end
end
```

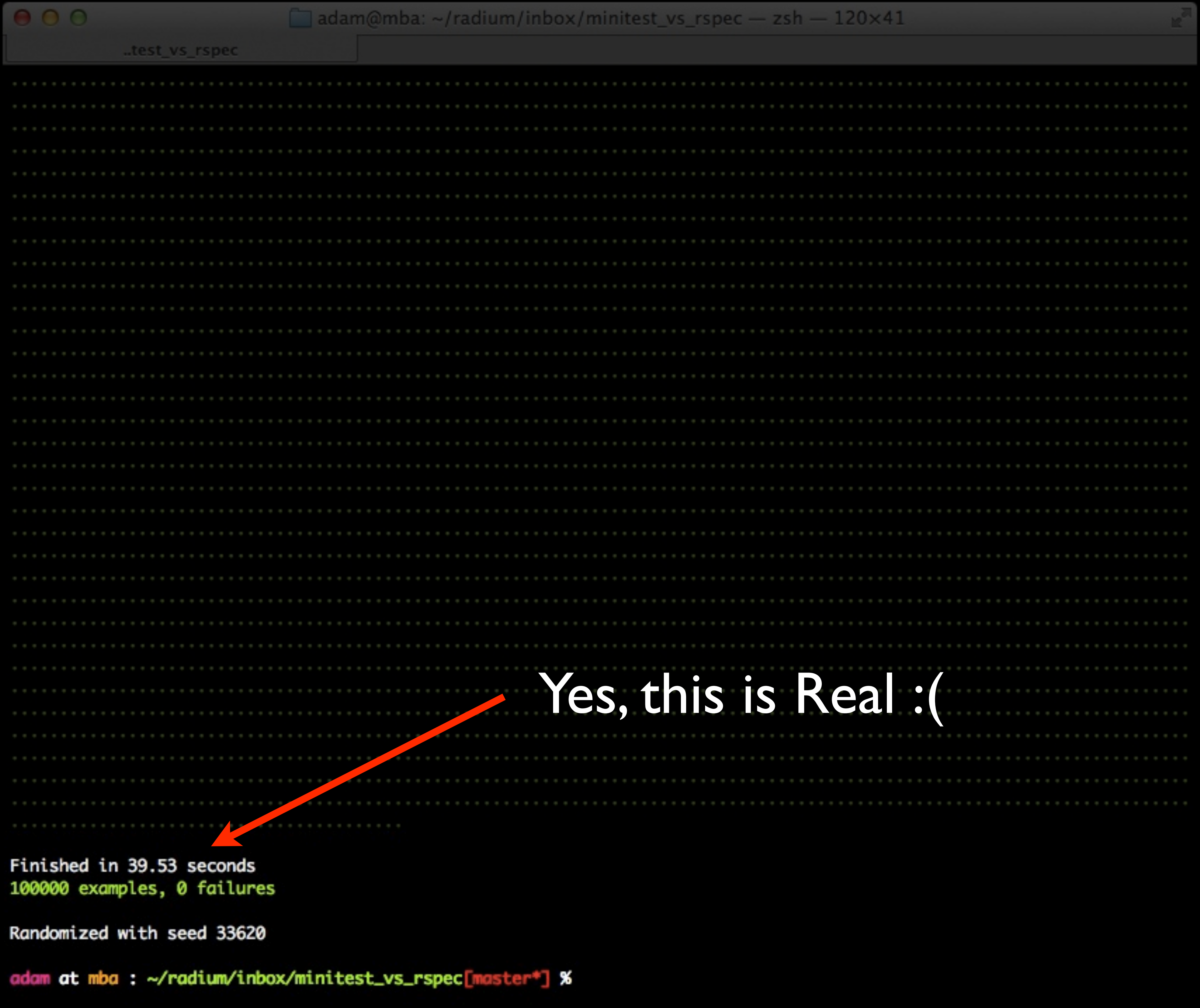
```ruby
require 'spec_helper'

describe "Truth" do
  10_000.times do
    it { should_not be_nil }
  end
end
```

Results

# Tests	MiniTest	Rspec
1.000	0.069	0.428
10.000	0.730	3.92
100.000	8.345	39.53

Times in Seconds



```
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................................................................................................
........................................

Finished in 39.53 seconds
100000 examples, 0 failures

Randomized with seed 33620

adam at mba : ~/radium/inbox/minitest_vs_rspec[master*] %
```

Yes, this is Real :(

Analysis

- MiniTest is clearly faster on tremendously large test suites
- MiniTest is notably faster on smaller tests suites
- RSpec will continue to slow down rapidly because matchers create objects which trigger garbage collection
- Developers may notice a difference in typical test suites

Expressiveness:
should vs assert

Rspec is testing DSL

```ruby
describe Person do
  its(:phone_number) { should =~ /^\d+$/ }
end
```

MiniTest is Ruby

```ruby
class PersonTest < MiniTest::Unit::TestCase
  def test_phone_number_format
    person = Person.new
    assert_match person.phone_number, /^\d+$/
  end
end
```

Magic

- RSpec: less code & more magic
- MiniTest: more code & less magic

```ruby
# 1. describe Person detects that the described
# object is a class so `Person.new` is called
# implicitly before each test and assigned to
# `subject`
describe Person do
  its(:phone_number) { should =~ /^\d+$/ }
end
```

```
describe Person do
  # Call the `phone_number` method on subject
  # an assign it's return value to `subject`
  # inside the test block
  its(:phone_number) { should =~ /^\d+$/ }
end
```

```ruby
describe Person do
  # Call should on the implicit subject
  # so it can be tested against the regex
  its(:phone_number) { should =~ /^\d+$/ }
end
```

I don't like this complexity

Do I need to explain MiniTest?

Custom Matchers
vs
Custom Assertions

```ruby
def assert_valid(model)
  assert model.valid?, "Expected #{model} to be valid"
end
```

```ruby
# There is a large API for this!
# This is the most simple case

RSpec::Matchers.define :be_valid do
  match do |model|
    model.valid?
  end
end
```

Go read this file: https://github.com/rspec/rspec-expectations/blob/master/lib/rspec/matchers/built_in/base_matcher.rb

Writing custom assertions is easier and more understandable in MiniTest

```ruby
def test_vendor_comes_before_app
  # do stuff to build a file
  content = read "site/application.js"
  assert_includes content, "APP"
  assert_includes content, "VENDOR"
  assert content.index("VENDOR") < content.index("APP"),
    "Vendor should come before App!"
end
```

extract-method

```ruby
def test_vendor_comes_before_app
  content = read "site/application.js"
  assert_before content, "VENDOR", "APP"
end

def assert_before(source, first, second)
  assert_includes content, first
  assert_includes content, second
  assert content.index(first) < content.index(source),
    "#{first} should be before #{second}"
end
```

One Less Thing to Learn

You Don’t need to lookup a Matcher API, just write ruby methods

MiniTest will make your code better

Why?

- Minimal feature set: focus on writing code
- Removing a complex mocking/stubbing library makes you consider design
- Run tests in random order--this will usually find bugs in your test suite or code
- Can run your tests in parallel

If you do in fact, suck at writing tests....

```
class OrderDependentTest < MinitTest::Unit::TestCase
  i_suck_and_my_tests_are_order_dependent!

  def test_step1
    # ....
  end

  def test_step2
    # ....
  end
end
```

```ruby
require 'minitest/unit'
require 'minitest/hell'    # put your tests through the ringer
require 'minitest/autorun'

class ParallelTests < MiniTest::Unit::TestCase
  def test_multi_threading
    # ...
  end
end
```

tl;dr

Why MiniTest?

- Its faster and lighter than Rspec
- Its much easier to understand and extend
- Random & parallel tests out of the box
- Its part of the standard library so it’s available everywhere
- More stable and better supported than Rspec
- It will make your test suite better!

Why Rspec?

- $ rspec foo_spec.rb
- Running individual tests is trivially easy
- You like DSL’s and complexity

die Slides

http://speakerdeck.com/u/twinturbo