Rspec expect document:

https://github.com/rspec/rspec-expectations#typesclasses
Respec Should be kind of https://www.rubydoc.info/gems/rspec-expectations/2.0.1/RSpec/Matchers

Rspec 教程：https://ihower.tw/rails/testing-cn.html

Rspec should eql:
https://relishapp.com/rspec/rspec-expectations/v/3-9/docs/built-in-matchers

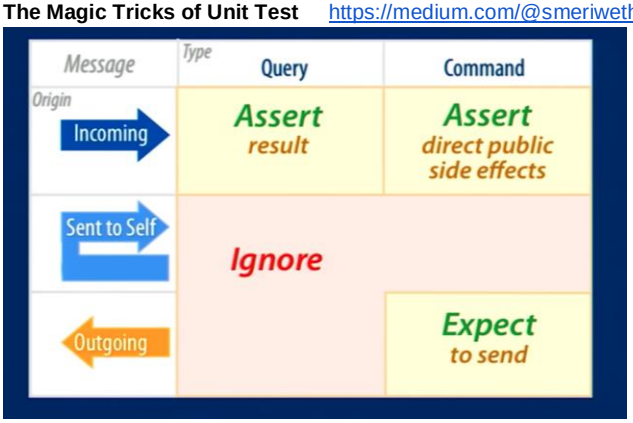
Rspec https://semaphoreci.com/community/tutorials/getting-started-with-rspec
Rspec: https://relishapp.com/rspec/
Ruby metaprogramming : https://www.toptal.com/ruby/ruby-metaprogramming-cooler-than-it-sounds

Scrum 介绍: https://www.youtube.com/watch?v=XU0llRltyFM

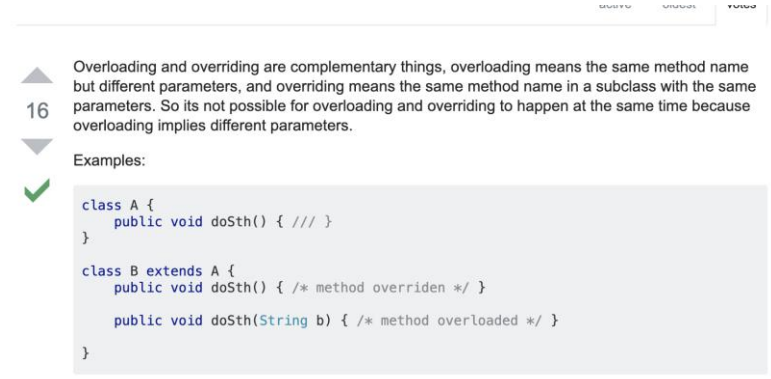
设计模式：（这份文件里的UML图没法显示。。可能还是找别的参考一下比较好。）
https://gof.quanke.name/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1%E8%AE%BE%E8%AE%A1%E5%8E%9F%E5%88%99.html
有图的设计模式：
https://juejin.im/post/5bc96afff265da0aa94a4493#heading-10

Command pattern
https://www.cnblogs.com/java-my-life/archive/2012/06/01/2526972.html

The Magic Tricks of Unit Test https://medium.com/@smeriwether/the-magic-tricks-of-unit-testing-28ce0b300cee



Override and Overload



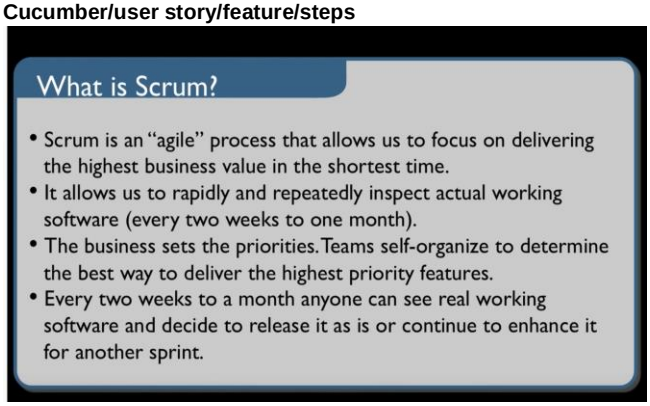
Focus on messages

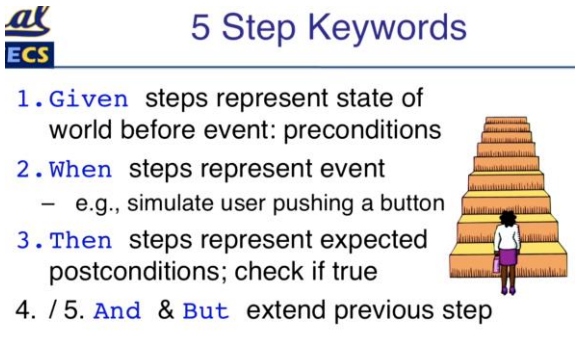
Incoming:
the object under test receives messages from others
Not let the outside see

Outgoing:
Sends messages
Can not let the inside see out

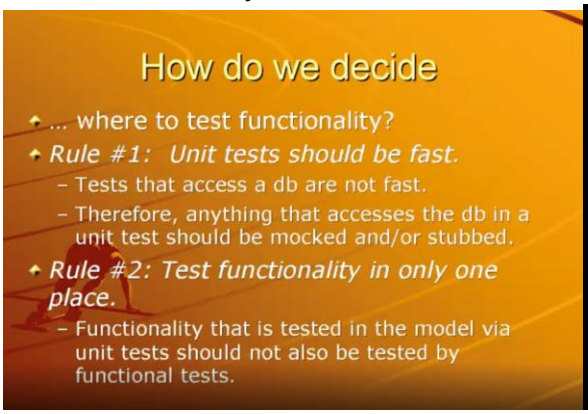
Sent to itself:
calling a private method

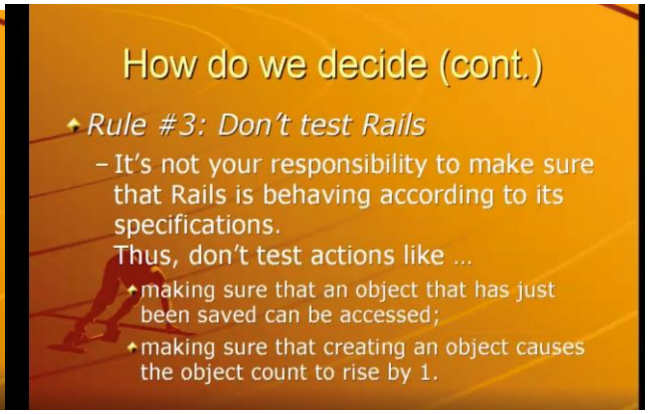
These three can be queries or commands
Query:
Return something/change nothing
Command:
Return nothing/change something
Smart User Story
https://www.visual-paradigm.com/scrum/write-user-story-smart-goals/
Rails原生test， self.should(base):
https://stackoverflow.com/questions/5160780/what-does-self-includedbase-do-in-ruby-on-rails-restful-authentication
Cucumber/user story/feature/steps





How to test functionality?





Controller Specs and Refactoring

Should & Should-not

- Matcher applies test to receiver of *should*

`count.should == 5`	Syntactic sugar for `count.should.==(5)`
`5.should(be.<(7))`	`be` creates a lambda that tests the predicate expression
`5.should be < 7`	Syntactic sugar allowed
`5.should be_odd`	Use `method_missing` to call `odd?` on 5
`result.should include(elt)`	calls Enumerable#include?
`result.should match(/regex/)`	

`should_not` also available

```
result.should render_template('search_tmdb')
```

Two new seam concepts

- `stub`
  - similar to `should_receive`, but not expectation
  - `and_return` optionally controls return value
- `mock`: create dumb "stunt double" object
  - stub individual methods on it:

```
m = mock('movie1')
m.stub(:title).and_return('Rambo')
```

  - shortcut: `m=mock('movie1',:title=>'Rambo')`

each seam enables just enough functionality for some *specific* behavior under test

Test Cookery #1

- Each spec should test *just one behavior*
- Use seams as needed to isolate that behavior
- Determine which expectation you'll use to check the behavior
- Write the test and make sure it fails for the right reason
- Add code until test is green
- Look for opportunities to refactor/beautify

Test techniques we know

```
obj.should_receive(a).with(b).and_return(c)
obj.stub(a).and_return(b)

d = mock('impostor')
```

Optional!

`obj.should` *match-condition*

Rails-specific extensions to RSpec:

```
assigns(:instance_var)
response()
render_template()
```