mason gem

January 2016

Goal: I want to make a gem that has all my shared ruby code, so that I can easily use it with every Ruby project I do. Since I have never built a Ruby gem before, I will document the process, as I am wont to do.

Work Log:

2016-01-09: figure out how to conveniently work on the gem and a project using it at the same time
2016-01-09: figure out how to use the gem by referencing its GitHub repo
2016-01-09: create My Little Pony® Baby's First Gem™

Links:

RubyGems guide to how to make your own gem (useful)

Ways to specify a local gem in Gemfile

以上

create My Little Pony® Baby's First Gem™

2016-01-09

Minimally, it is just 2 files in a simple hierarchy:

```
[mason@MacBook-Pro-No mason-gem]$ tree
.
├── lib
│   └── mason.rb
└── mason.gemspec
```

The gem spec can apparently be complicated, according to the docs, but this simple one suffices for now:

```
[mason@MacBook-Pro-No mason-gem]$ pygmentize mason.gemspec

Gem::Specification.new do |s|
  s.name        = 'mason'
  s.version     = '0.0.1'
  s.date        = '2016-01-08'
  s.summary     = "Mason!"
  s.description = "A personal utility gem by and for Mason."
  s.authors     = ["Mason Mark"]
  s.email       = 'mason@masonmark.com'
  s.files       = ["lib/mason.rb"]
  s.homepage    = 'http://masonmark.com'
  s.license     = 'MIT'
end

[mason@MacBook-Pro-No mason-gem]$
```

...and the code itself:

```
[mason@MacBook-Pro-No mason-gem]$ pygmentize lib/mason.rb

class Mason

  def boogie
    puts "w00t w00t"
  end

end

[mason@MacBook-Pro-No mason-gem]$
```

The gem spec and source code are ready; next, build the gem:

```
[mason@MacBook-Pro-No mason-gem]$ gem build mason.gemspec
  Successfully built RubyGem
  Name: mason
  Version: 0.0.1
  File: mason-0.0.1.gem
[mason@MacBook-Pro-No mason-gem]$


[mason@MacBook-Pro-No mason-gem]$ ls -l
```

```
drwxr-xr-x  3 mason  staff   102 Jan  8 20:58 lib
-rw-r--r--  1 mason  staff  4096 Jan  9 10:03 mason-0.0.1.gem
-rw-r--r--@ 1 mason  staff   378 Jan  9 09:52 mason.gemspec
[mason@MacBook-Pro-No mason-gem]$
```

Cool! The gem is built.

Now how can we use it? Well, if we were lame, we could install it globally on the local machine like this:

```
[mason@MacBook-Pro-No mason-gem]$ sudo gem install ./Mason-0.0.1.gem
Password:
Successfully installed mason-0.0.1
Parsing documentation for mason-0.0.1
Installing ri documentation for mason-0.0.1
1 gem installed
[mason@MacBook-Pro-No mason-gem]$
```

And then use it like this:

```
[mason@MacBook-Pro-No mason-gem]$ irb
irb(main):001:0> require 'mason'
=> true
irb(main):002:0> foo = Mason.new
=> #<Mason:0x007fb1a2a253b8>
irb(main):003:0> foo.boogie
w00t w00t
=> nil
irb(main):004:0> ^D
[mason@MacBook-Pro-No mason-gem]$
```

But that's not what we are trying to accomplish, so:

```
[mason@MacBook-Pro-No mason-gem]$ sudo gem uninstall Mason
Password:
Successfully uninstalled Mason-0.0.1
[mason@MacBook-Pro-No mason-gem]$
```

What we actually want to do is require this gem in a Gemfile used with Bundler. The good and easy way to do that is to put it on my own Github, since Bundler has built-in support for referencing GitHUb-hosted gems in the Gemfile.

But even easier at this stage (since this gem isn't yet on GitHub) is just requiring it locally via absolute path. Here is the working setup with a throwaway test project:

The minimum setup to use the gem in isolation via Bundler is a Gemfile to specify the

gem, and a ruby script to run.

```
    "mason"  path  "../mason-gem/"
  # path is to parent dir, not gem itself
```

For this exercise, we've specified only the mason gem. Then we require it in the source code for our project:

```
#! /usr/bin/env ruby

require 'mason'

      Mason
puts "Here's a Mason instance: #{   }"
puts "Will it boogie?"
```

Next, run it with Bundler. It seems that no 'bundle install' command is even necessary with this minimal setup.

Boogie. Of course, it won't run without Bundler, because the gem is not installed on the local system:

That's as it should be.

For my next stupendous trick, I think I will put this gem on Github and then use it from

there.

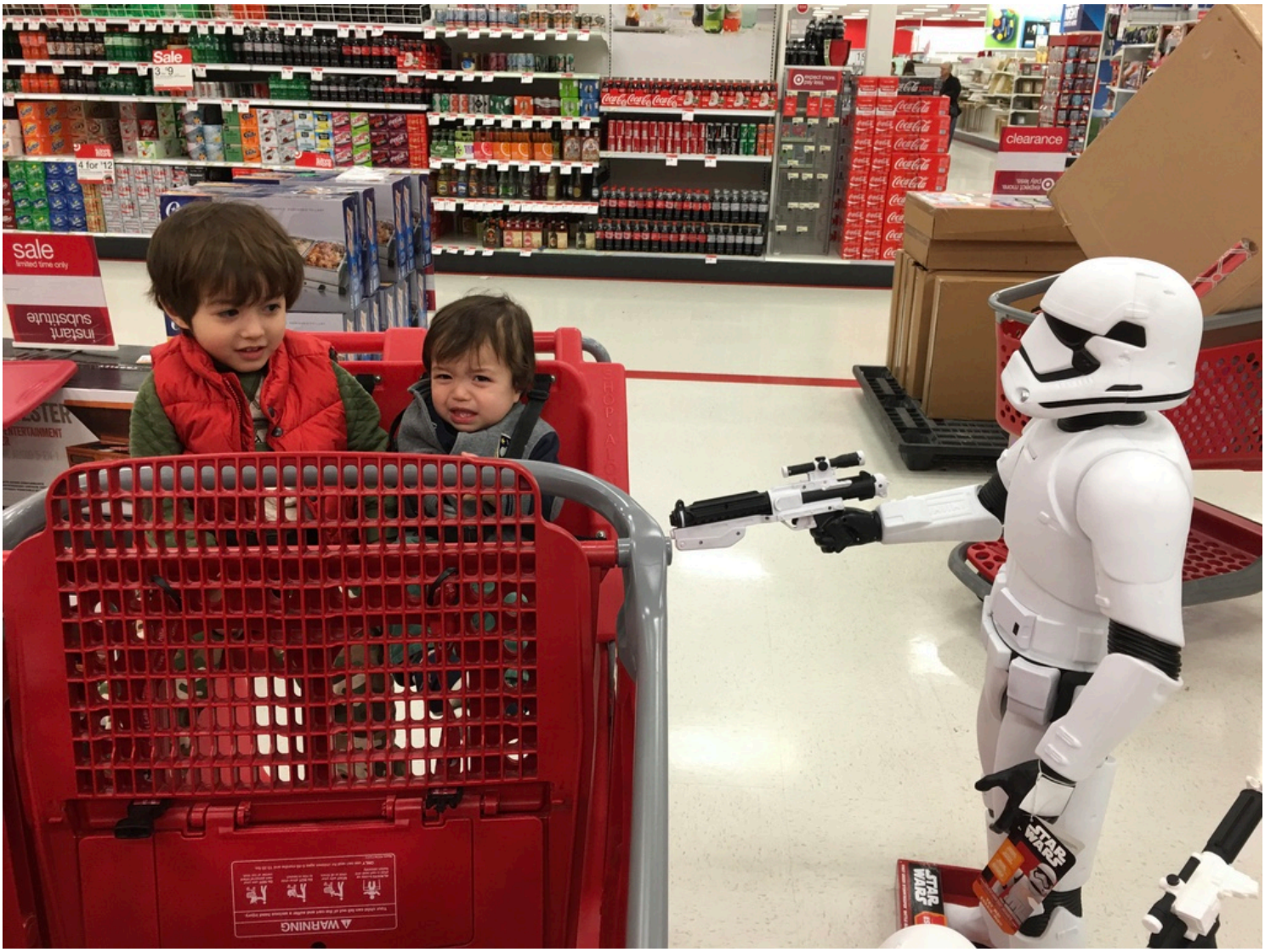
Sale
sale
limited time only
instant substitute
clearance
WARNING
STAR WARS

figure out how to conveniently work on the gem and a project using it at the same time

2016-01-09

Oh, cool. If using the local path way of referencing the gem in the Gemfile, it looks like you don't even have to actually do a `gem build`.

Change the Gemfile to reference the relative path to the gem's source code folder:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ pygmentize -l ruby Gemfile
# gem "mason", github: "masonmark/mason-gem"
  # When not actively working on the gem, referencing it via github is good.
  # However, when working on the gem itself, it's more covenient to reference it
  # via local path, because then there's no need to push the gem code to github
  # and then have to 'bundle install' in the project using it.

gem "mason", path: "../"
  # path is to parent dir, not gem itself

[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

Note that the built gem is not present:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$  ls -l ..
total 1392
drwxr-xr-x  8 mason  staff     272 Jan  9 12:17 README-mason-gem
-rw-r--r--@ 1 mason  staff  701669 Jan  9 12:17 README-mason-gem.pdf
drwxr-xr-x  6 mason  staff     204 Jan  9 12:17 README-mason-gem.vpdoc
-rw-r--r--@ 1 mason  staff      55 Jan  9 11:15 README.md
drwxr-xr-x  3 mason  staff     102 Jan  8 20:58 lib
-rw-r--r--@ 1 mason  staff     378 Jan  9 09:52 mason.gemspec
drwxr-xr-x  7 mason  staff     238 Jan  9 15:55 test-the-gem-bro
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

Also note that, just to be sure, I nuked the "vendor" dir in the test-the-gem-bro project, and also the Gemfile.lock file:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ ls -l
total 16
-rw-r--r--@ 1 mason  staff  417 Jan  9 15:53 Gemfile
-rwxr-xr-x@ 1 mason  staff  131 Jan  9 09:31 test.rb
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

But it still works:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ bundle exec ./test.rb
Here's a Mason instance: #<Mason:0x007f8c1212dc48>
Will it boogie?
w00t w00t
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

Now, the $0.64 question: if we change the gem code, and do nothing else, will the test project see the changes and run the new code? Yes:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ bundle exec ./test.rb
Here's a Mason instance: #<Mason:0x007ff1e2b347d8>
Will it boogie?
Sorry, the boogie feature has been removed in this new version.
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

Sick! So, this confirms that it's hella easy to work on a gem concurrently with a project that uses it. 👍🏾

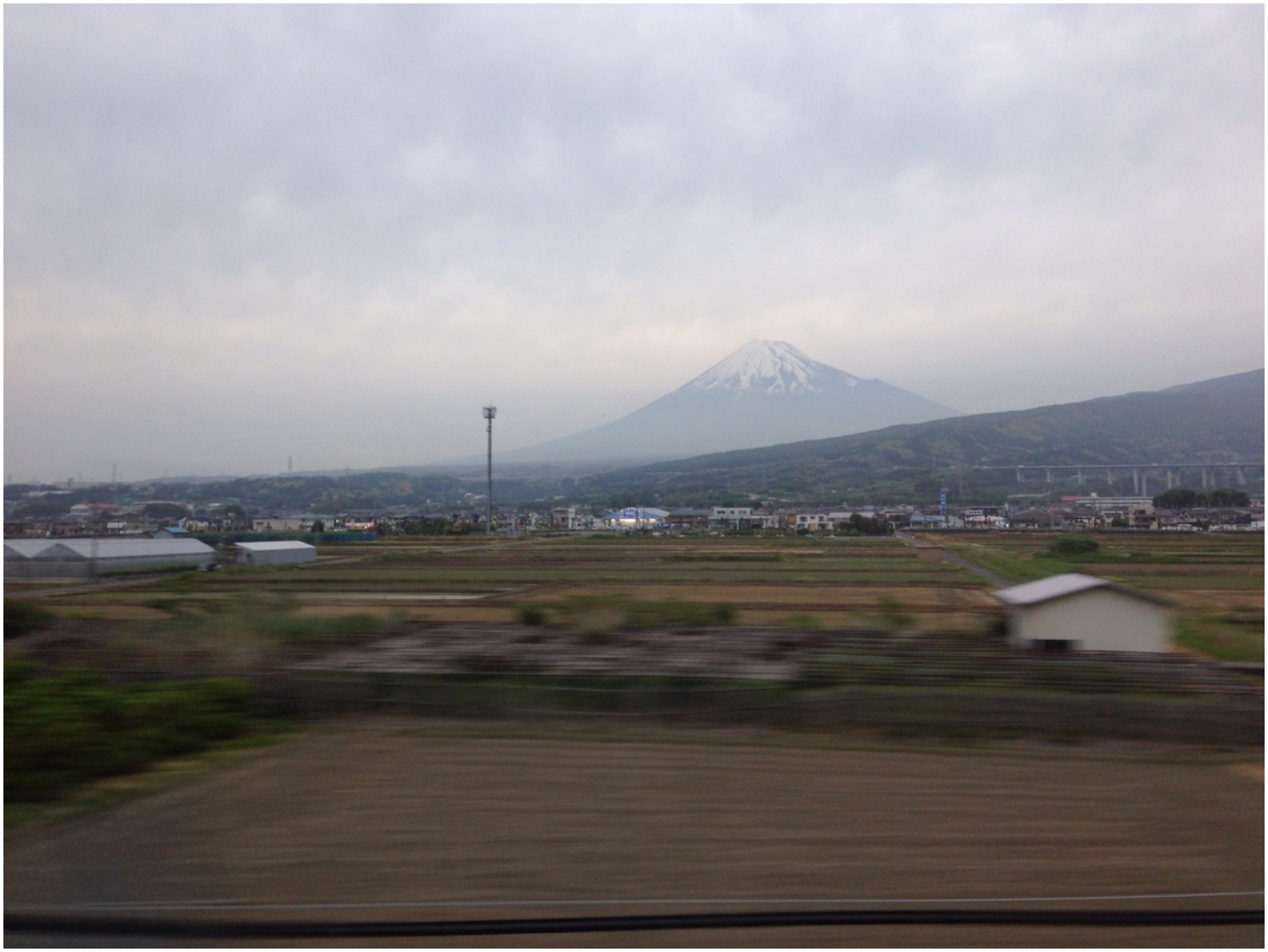

figure out how to use the gem by referencing its GitHub repo

2016-01-09

So, I moved the test project into the repo for the mason gem itself:

```
[mason@MacBook-Pro-No mason-gem (master)]$ ls -l
total 872
drwxr-xr-x  6 mason  staff     204 Jan  9 11:09 README-mason-gem
-rw-r--r--@ 1 mason  staff  431256 Jan  9 11:10 README-mason-gem.pdf
drwxr-xr-x  6 mason  staff     204 Jan  9 11:15 README-mason-gem.vpdoc
-rw-r--r--@ 1 mason  staff      55 Jan  9 11:15 README.md
drwxr-xr-x  3 mason  staff     102 Jan  8 20:58 lib
-rw-r--r--  1 mason  staff    4096 Jan  9 10:03 mason-0.0.1.gem
-rw-r--r--@ 1 mason  staff     378 Jan  9 09:52 mason.gemspec
drwxr-xr-x  5 mason  staff     170 Jan  9 10:12 test-the-gem-bro ← NEW!! 😮
[mason@MacBook-Pro-No mason-gem (master)]$
```

Then, I updated the Gemfile for the test project to reference the gem's GitHub repo, not the local path to the gem:

```
[mason@MacBook-Pro-No mason-gem (master)]$ cd test-the-gem-bro/
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ pygmentize -l ruby Gemfile

gem "mason", github: "masonmark/mason-gem"

# gem "mason", path: "../"
    # path is to parent dir, not gem itself
```

Obviously, this way does not work without a bundle install (unlike when we previously referenced the gem by its local path), because Bundler has to fetch the gem from Github:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ bundle exec ./test.rb
The git source git://github.com/masonmark/mason-gem.git is not yet checked out.
Please run `bundle install` before trying to start your application
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

That's OK though, because in a real project we always have to do this anyway:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ bundle install --path vendor/
bundle
Fetching git://github.com/masonmark/mason-gem.git
Resolving dependencies...
Rubygems 2.0.14 is not threadsafe, so your gems will be installed one at a time.
Upgrade to Rubygems 2.1.0 or higher to enable parallel gem installation.
Using mason 0.0.1 from git://github.com/masonmark/mason-gem.git (at master@1575c6f)
Using bundler 1.11.2
Bundle complete! 1 Gemfile dependency, 2 gems now installed.
Bundled gems are installed into ./vendor/bundle.
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

And now:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ bundle exec ./test.rb
Here's a Mason instance: #<Mason:0x007fe0aa28c1b0>
Will it boogie?
w00t w00t
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

Cool!