NPM for Fun And Profit

Tuesday, September 10, 13

Thanks for being here, and to Sport Ngin for sponsoring the event.

Tonight, I'd like to talk about NPM Modules for Fun and Profit

But Mostly Profit

~~NPM~~ Gems for Fun And Profit

Tuesday, September 10, 13

My experience is mainly in the Ruby world, so this was originally conceived as discussing Ruby Gems.

The themes that I'm going to discuss are universal, regardless of the technology being used.

Don't worry: I'm not going to have any source code on screen.

Tuesday, September 10, 13

NPM modules, of course, a way to share code between projects

http://npmjs.org/

Show of hands…

Who has used a module?

express, request, async, grunt?

Who has **made** a module?

If you didn’t keep your hands up, I hope that by the end of this I’ll have convinced you that now is the perfect time to start

Why write a module?

Obvious answers:
* Share code between projects
    * DRY, reduce duplication
* Open-source
    * Receive contributions
* Joy of sharing

Enough About You

Tuesday, September 10, 13

Let’s talk about me

Patrick Byrne

Developer in the Twin Cities:
* Developer since 2006
* Primarily work in Ruby with a healthy dose of shell

SPORT NGIN

Tuesday, September 10, 13

Formerly…

Tuesday, September 10, 13

Currently… Dribbble, social network for designers

If you haven’t heard of it, ask your designer friends

Enough About Me

Let’s talk about modules

* Skipping the boring, mechanical how-to stuff
* This is a really smart, good-looking crowd
    * Confident that you guys can figure that part out
    * You can Google

npm init

Tuesday, September 10, 13

Good place to start building a new module

Let's talk about:
* some of the less-obvious benefits
* how you can get so much more out of these projects if you think about them a little differently

This comes from my experience: working mostly on a few large legacy apps as part of a larger team in a mid-sized organization. Your experience might be entirely different, so please chime in with questions or additional suggestions.

OK Computer

Tuesday, September 10, 13

Configurable health-check gem that I wrote at Sport Ngin

https://rubygems.org/gems/okcomputer

Configurable application health-check for Rails apps

What Does That Mean?

Who here uses a load balancer?

* Simplest case is a lightweight URL on each app server to tell the LB that it’s up
* LB can remove down instances
* Primary purpose

So Much More

Tuesday, September 10, 13

* But you can also use it for application health checks (Pingdom for example) to alert you in case something is wrong
	* Fail if Delayed Job or Resque get too far backed up
	* Fail if an external service is unreachable (SendGrid, Twitter, payment gateway?)
* Check those regularly with something like Pingdom and alert you if something goes wrong with something like Pager Duty
	* or manually with `curl`

A Little History

* Had used fitter-happier in our primary app, just did ActiveRecord and basic up-check
* Created a second app that used Mongoid. No choice but to monkey-patch, since no customization available
* Added a Resque check, more monkey patching
* Third app needs these patches, and another

Stop the Madness!

We can do better than this. The idea for OK Computer is born.

* Provide the checks that we knew we needed
* Easily configure them on a per-app basis
* text, JSON, HTTP status codes for success and failure

OK, with all of that out of the way, let’s start talking about the ways that extracting parts of your applications into gems

Eliminate Duplication

I lied a little bit. An obvious benefit, but it’s SO IMPORTANT that I have to talk about it.

* Such a fundamental part of writing good software
* Write the module once, put it in all your apps
* If you find yourself copying and pasting support classes from project to project, STOP IT. You owe it to yourself, your team, your boss, your clients, your customers to extract it out and put it into a module.
* OK Computer: We created the kinds of checks we wanted, tested them thoroughly, to be enabled and configured in a per-app basis.

Greenfield Development

Another show of hands: How many of you work on a “legacy” application?
* You’re not the only developer, it’s been through a few Rails upgrades, a few versions of Ruby

The weight of past decisions factors into new development
* Libraries used
* Code conventions
* Testing style and toolchain

Can be a bit boring to feel constrained. Maybe not as exciting.

Everybody loves bright and shiny and new. Gets the blood pumping. New module excellent opportunity to start with fresh sheet of paper, unbound by the rest of your apps/team.

If on a single app, not saving duplication, moving code that’s not business logic into a module still provides benefit.

OK Computer: Brand new, exciting problem domain. Couldn’t stop thinking about it in that first rush of development.

Peer Pressure

If you’re building something that can be released to the public, you’re more likely to:
* Write a good README
* Provide sensible defaults
* Make it reasonably complete
* Try to impress anyone who will see it

If it was in your main app, you might be tempted to throw it together and move on. Call it “done”. Here you’re free to polish it and iterate independent of your main app

OK Computer: Our Fitter Happier monkeypatches were untested because “it’s simple enough” and wasn’t deemed worth of the effort. OK Computer was thoroughly test-driven, which lead to better code.

Opportunity for New Experiences

Depending on your org, you might only experience small part of the lifecycle. Here you own everything.

* Never built a Rails Engine before?
* Never written a test before?
* Never started a project from scratch?
* Never take something from idea to release?

Low risk here, plenty of opportunity to make yourself better.

Freedom to Experiment

Free to try new tools and techniques

FAR EASIER to try these in a small, stand-alone project than to change course in an existing codebase

Easier to convince yourself to try something new than to convince your team to try something new and unproven

This freedom comes in a couple different flavors…

Code

Use jasmine at work? But want to try mocha or qunit?
Can't bring your team on-board with TDD?
Think you have a better convention than your main app?
Want to put Sandi Metz's 4 rules to the test?
* http://robots.thoughtbot.com/post/50655960596/sandi-metz-rules-for-developers
* Classes can be no longer than one hundred lines of code.
* Methods can be no longer than five lines of code.
* Pass no more than four parameters into a method. Hash options are parameters.
* Controllers can instantiate only one object.
Want to try these SOLID principles you keep reading about in blogs?
* http://en.wikipedia.org/wiki/SOLID_(object-oriented_design)
* Single responsibility principle: a class should have only a single responsibility
* Open/closed principle open for extension, but closed for modification
* Liskov substitution principle: objects be replaceable with instances of their subtypes
* Interface segregation principle: many client-specific interfaces are better than one general-purpose interface
* Dependency inversion principle: depend upon abstractions rather than a specific implementation

OK Computer: README-driven development, TomDoc

Tools

Many tools are free for open-source projects, so you can test them out before paying for them in your core app:
* Code Climate (static code analysis)
* Travis CI (run tests on every commit and report results)
* Coveralls (test-coverage reporting)

Share Your Experience

The MOST IMPORTANT PART of the experiment. If what you tried was better, tell everyone.

Thoughtbot Bot Cave: Newsletter sharing results of their experience

What worked? What didn't? Why? Why not?

If something is better, you have ammunition to improve your existing app with these findings.

Recap

Tuesday, September 10, 13

OK, so what have we learned tonight?

Eliminate Duplication

Extract shared code to eliminate duplication

Greenfield Development

Tuesday, September 10, 13

Pay attention to the creative rush you get working on something brand new

Peer Pressure

Succumb to the pressure to pay close attention to the quality of the code that you release to the public

Opportunity for New Experiences

Wear all of the hats of software development and release

Freedom to Experiment

Please, please, please: take advantage of the opportunity to learn something new and bring the knowledge you gain back to your team

Questions?