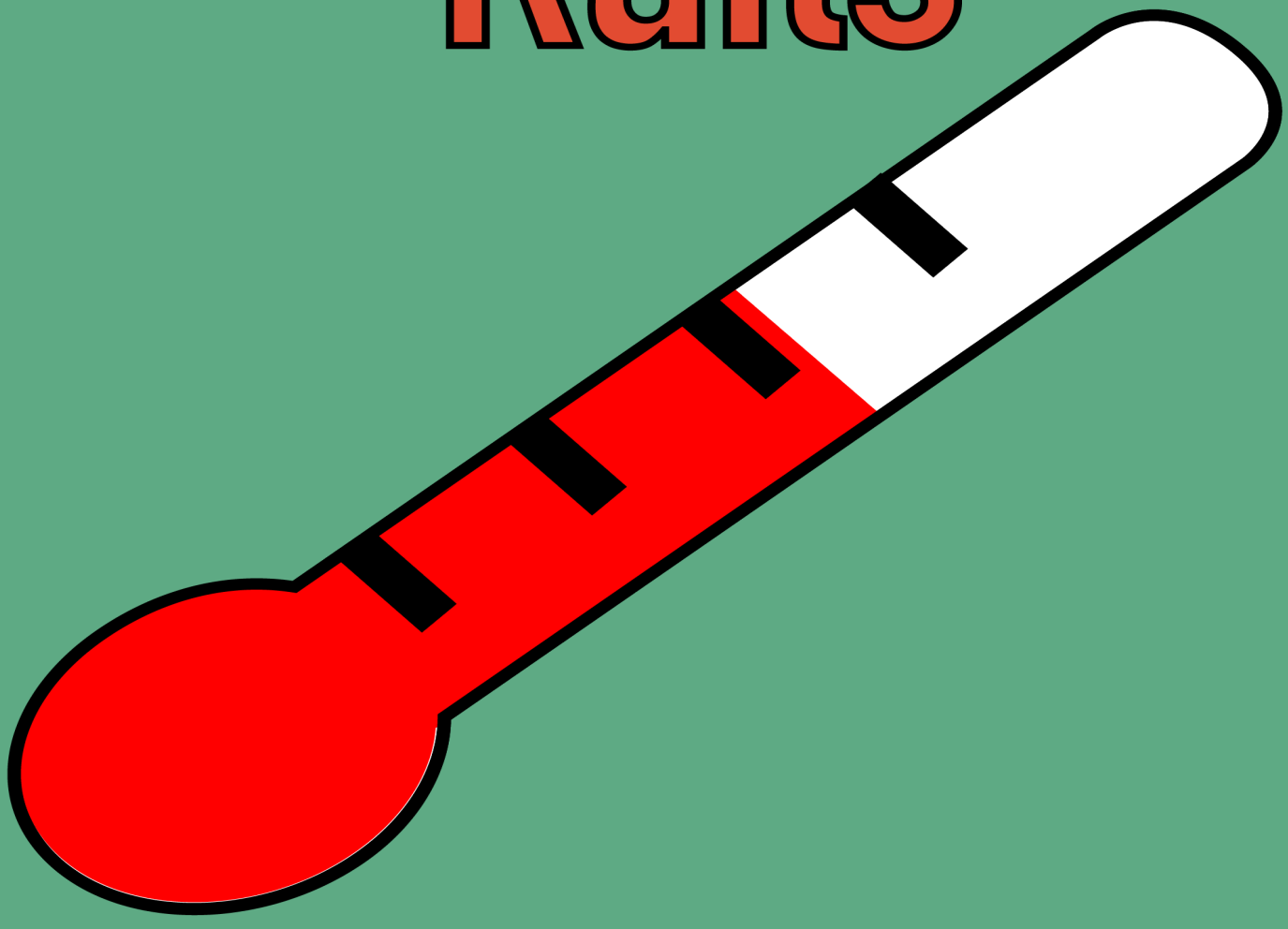
Ember.js Testing On Rails



Ember.js - Testing on Rails

Martin Feckie

This book is for sale at http://leanpub.com/emberjs-testingonrails

This version was published on 2014-05-05

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Introduction

A Bit About My Motivation

I've been very interested in the rise of Ember.js and front end web frameworks in general. I recently made an app for a customer using Angular.js for front-end rendering. There is a very clear focus on testing in Angular and I found it really easy to build a test suite. Integrating with Rails was much more of a challenge however. I began to look at Ember and gave it a good go. At the time I found that persisting data was a real challenge as was integrating with Rails, this was a big disappointment and in the end I kept going with Angular and was able to produce a well tested app I was happy with. Angular, however, is much more free form than Ember and presents lots of opportunities for differences of opinion and approach. Not necessarily a bad thing, but if I'm producing an app I'd like another developer to be able to come along and have a very good idea of where to find things and Ember presents a much higher opportunity to achieve this.

I've continued to watch the development of Ember and LOVE where it's going, the passion and talent of its developers is amazing and I'm so grateful for what they are doing. Most of my initial difficulties around persistence have been well and truly resolved. I'm still, however, frustrated with the lack of documentation about testing beyond integration tests. There are heaps of blog posts our there and people have put a lot of work into finding ways to test Ember apps. The problem is, with all the API changes, many are outdated and simply don't apply any more. That shouldn't be taken as a criticism of any of the work the authors have put in, I'm truly grateful they did.

I had desperately hoped to find a book that would tell me how to do test driven development with Ember, but it still hasn't eventuated. I've spent months playing around with different configurations and setups with Rails and Ember and came to the realisation that in creating my own setup I've learned a lot. As a result I thought I would try and write the book I would like to have read!

I'm not gonna suggest for a minute that everything I've put forward is perfect, I'm sure it's not and there will definitely be others who will come up with better ideas in the future. I'm content that what I've put forward is a good starting point. I'm very open to feedback and dialogue and want to create something that helps. If I succeed in helping, please tell others, if not please tell me!

A Note on Copyright

If you bought this book or were given a copy by me, skip this!

If you didn't buy this book and found it on a file sharing site, please have a think about it. This book is a labour of love for me and an attempt to share countless hours of experience with others. If you feel that your need is so great that you must read without paying, contact me and I will see what I can do to provide you with a legal copy that receives regular updates.

Karma is a wonderful thing and if you do choose to take my book without paying, then I wish you well and I hope that you find it useful and it helps you develop in your career. If so, please consider buying a copy for someone else who would benefit.

First There Was Rails

If you've even thought about purchasing this book, I'm sure you're familiar with Ruby on Rails. As a framework it's done much to drive best-practices in web application development, providing users with a truly open source framework very different from some of the proprietary systems on offer.

The true beauty of working on a Rails app is that any part of the framework that you don't like, you can change! Monkey patch away, but be warned, straying from the 'golden path' is something that can really cause you long term pain.

To combat some of the perils of free form development Rails provides us with a convention over configuration philosophy. If you do things 'the Rails way' then things tend to flow very easily and your pain is minimized!

One of the conventions that I love the most is the focus on testing. There are plenty that argue that the style of Test Drive Development provided by Rails is invalid. The argument goes that it doesn't follow the test first style recommend by many. When you run

```
1 $> rails generate model Thing name:string
```

you get pre populated tests for your newly generated model. Personally, I think that's a good thing for new developers and it's easily disabled by seasoned developers.

And so it was that all was good in the world, we rendered on the server, pushed to the client and everyone was happy. But then…

Along Came jQuery

Despite masses of critics, there can be no denying that javascript won the language wars (in terms of availability at least!). It's used on more devices, in more places that any other language. We can spend hours bike shedding the topics, or accept reality and find good ways to work with the language.

As the web has developed and javascript won the war, there became an increasing need to do more 'stuff' on the client side. We started to see people doing evil things with javascript - anyone remember the pop-up laden websites of the early 2000's? Not only did javascript facilitate annoying pop-up's but also helped malicious actions.

Having said all that, some people found really creative uses for javascript and one of the most successful early ideas was jQuery. jQuery provided developers with a straightforward way to interact with the Document Object Model (DOM). Developers being lazy (in a good way) found the **$** abstraction very useful.

Javascript DOM selection vs jQuery

```
document.getElementByClassName('container')
document.getElementById('someID')
document.getElementByTagName('p')
// vs
$('.container')
$('#someId')
$('p')
```

In and of itself, the **$** abstraction doesn't do anything to speed up the interface, but does aid developers in providing a one stop shop for interacting with the DOM. Not needing to change methods speeds development so we don't have to do `document.getElementById` or `document.getElementByTagName`, we can simply adjust the call inside the **$** abstraction.

jQuery allowed developers to provide a bit more structure to their javascript, but once an application grew to a reasonable size, it became a fight to keep the code clean and developers would often experience call-back hell [1].

Javascript IV -Ember - A New Hope

I knew that Ember.js was going to be special the moment I looked at the early website. I can't believe how excited I was by the bindings! Oh the bindings! I type it here and it's updated there, live and I can persist it!!! Woo hoo.

It's not surprising that Ember is such a great framework, after all it's got Yeduda Katz (of jQuery, Rails and much more), Tom Dale. The other core contributors are amazing as are the hundreds of other who've made contributions. I've got a tremendous amount of respect for the Ember team, their commitment to getting it right and their willingness to acknowledge when things went wrong (early stage ember-data anyone?) is impressive.

[1] Call-back hell is a term used to refer to the problem to heavily nested code, often detected through it's particular 'triangular' shape. http://callbackhell.com/

I can also see the direction the project is going in and can see huge strides in performance and convenience with the release of 1.0 . The six week release cycle will see the speed of the framework improve and minor bugs and problems resolved (though already the benchmarks are impressive in comparison to other frameworks [2])

The learning curve for any framework following the convention over configuration pathway is almost always huge. This is certainly the case with Ember. The initial excitement of being able to do so much, with so little code soon gives way to the frustration of an error caused by a misnamed class or incorrect pluralisation.

Testing to the Rescue

Testing is a great way for us to explore the framework and I hope to provide a robust guide to allow you to get up and going with an environment that provides rapid feedback and some good strategies for testing your Ember.js apps.

A word of warning on the code examples

The Leanpub platform automatically creates a '\' at the end of long lines to indicate wrapping. These obviously shouldn't be copied and I haven't found a way to turn them off. Please be cautious when following the examples.

[2]Although artificial benchmarks are frowned upon, here's some interesting comparisons with backbone.js http://jsfiddle.net/jashkenas/CGSd5/. Much more impressive is the future with HTMLBars comparing with React.js, Backbone and raw javascript when animating elements. http://jsfiddle.net/Ut2X6/. HTMLBars is a very exciting potential improvement to Handlebars https://github.com/tildeio/htmlbars

Design Choices, the Golden Path and 'Why Didn't You Include / Use / Show X, Y, Z?"

I've put out a few releases of the book and so far have received positive feedback, however I've also had some questions that I feel are worth responding to.

Why Didn't You Use Third Party Libraries to Cover BDD, Such as Pavlov?

The philosophy I'm coming from with the book is that getting as 'close to the metal' as possible will give you the knowledge and confidence to use third party libraries because you will learn what they are abstracting away. Knowing the hooks they use and the methods they leverage will allow you to troubleshoot and make the trade offs you feel are worth it.

Why Don't You Use Factories Instead of Fixtures in Ember?

Basically, none of the things we're going to use fixtures with require any dynamic attributes. Fixtures are lightweight and provided for free by Ember. I chose not to add in the extra work to provide dynamic objects.

You Know That You Can Run Multiple Suites of Tests With Teaspoon?

Yes, in the first draft of the book I utilised this feature of Teaspoon, but found that there was a conflict with Guard that led to tests getting 'stuck' in one place, leading to false positive / negative notifications via Growl. As a result, I chose not to use the feature because I valued reliability higher than separation of unit and integration tests. The issue with 'sticking' may well get ironed out with future (or even current) releases of Guard / Teaspoon, so by all means use the feature yourself.

The Golden Path

I've chosen to use the same tools as much as possible as the Ember core team. If you prefer to use Jasmine, Mocha, Chai, Pavlov etc. then go for it. Using QUnit would not be my first choice, but it is well integrated with Ember, well supported on Stackoverflow and other sources of knowledge and provides a great baseline. This is really about getting started with 'training wheels', if you're ready to go without them, take them off an go!