

The Twelve Rules of Sinatra

Proven Techniques to Master Sinatra
The micro-web framework



by: RubyLearning

<http://rubylearning.com/blog/>

IMPORTANT

You Do NOT Have Rights to Edit, Resell, Copy or Claim Ownership to this Report!

However...You DO Have the Right to Pass this Report Along to Others Who Might Benefit from it!

Feel free to share it with your blog readers and give it away as a freebie.

ALL RIGHTS RESERVED: You do not have any rights to sell or profit from this report. All content is to remain unedited and all links must stay in tact as they are. You can not claim any type of ownership without express written permission from the creator and only the creator, Satish Talim. All rights to this report belong to the author only.

DISCLAIMER: All information contained within this report is strictly the views represented by the author, at time of publication. Said author can and does reserve the right to add to, change, alter or update the thoughts and opinions stated herein. Every attempt has been made to accurately substantiate all information in the said report. However, the author, his partners, affiliates make no warranty to nor do they take responsibility for any errors or exclusions that may be contained in within. The author does not offer legal or financial advice and anything inside this report should not be construed as such. It is recommended the reader contact the appropriate qualified professional for advice in these and any other areas should it be needed. It is the responsibility of the reader to know and adhere to any local (city, state, county, etc.) laws regarding the conducting of business.

Find more actionable tips and advice at:

<http://rubylearning.com/blog/introduction-to-sinatra-ebook/>

The Twelve Rules of Sinatra

Recently, I was reading Scott Adams' (of Dilbert fame) blog post "[Rule of Twelve](#)" where he stated:

The Rule of Twelve states that if you know twelve concepts about a given topic you will look like an expert to people who only know two or three. If you learn more than twelve concepts about a topic, the value of each additional one drops off considerably. Allow me to be the first to confess that twelve is not a magic and inviolable number.

He also wrote a follow-up post to support his statement: "[Twelve Rules of Energy Efficient Building](#)".

This made me wonder, could we apply the same "Rule of Twelve" to Sinatra? I talked to three Sinatra experts about this and here is their take on the *Twelve Rules of Sinatra* -

Here is what [Julio Javier Cicchelli](#) had to say:

1. **Unlearn Rails:** Sinatra doesn't require you to understand the MVC philosophy or it doesn't want you to learn any kind of convention. You're free to decide how you want to write your Sinatra application.
2. **Read the source code:** Sinatra is a small framework and DSL written in 2000 lines of code so you don't have any excuse for not reading it. I will help you understand how exactly Sinatra works, avoiding any magic.

3. Know Rack: Sinatra is a thin layer on top of Rack. The latter provides extreme flexibility and unprecedented extensibility that you can use as middlewares between your Sinatra application and the Web Server. Rack will unleash the power of your Sinatra apps, so use it wisely!

4. Test before you code: Misunderstood by most of the developers out there, Testing will assure your application does what it needs to be done by making you focus on its functionality, behavior or results. Ruby has made this very easy so please, write your test before you write your application using any of the available libraries (RSpec, Cucumber, Shoulda. etc.). Think of it as a long-term investment that you and your colleagues will benefit!

5. Master the REST concept: No other framework uses REST as easily as Sinatra but you should first focus on answer the following question: What's REST? It's not just unintelligent URL so please, read about it before start using it with your Sinatra application.

6. Use the Gems: probably the most underrated feature Sinatra provides, Sinatra can make use of the extensible gems without wrapping up as plug-ins. Of course, you can write and use Sinatra specific gems for very specific means but I will discuss that later.

7. Think of your project: Before start writing your test cases and the respective code, you should think of your application. Sinatra allows you to create virtually anything. Will it be a website, a web app, a service or something wicked? The answer to this question will let you focus on the development.

8. Befriend data support: Files, databases, caches, etc. Think of them as providers of the content and meta-content for your application. Ruby has gems that will interface any kind of data support and Sinatra/Rack will help you use them! Just be careful when uploading files from your app!

9. Deliver your ideas: Sinatra + REST will help you spread the content of your application easily. You just need to properly use one or multiple delivery formats, depending of the client that will get the content. It's important to know how to use XML, JSON, HTML, YAML and other formats for your purposes.

10. Understand Javascript: If you're going to build web sites or applications. Javascript will be the friend you require in order to make it fancy and interactive. Understanding Javascript is a huge advantage to build the front-end of your app. You should get to know further and add to your project libraries such as Prototype or JQuery. They'll definitely add value to your project!

11. Turn into a Gem: Deployment can be done with libraries such as Capistrano or Vlad but I like the idea to package a complete solution within a gem. Learning how to wisely use the Rubygems library will let you master one of the required tasks of every Ruby programmer. Master this and you will be able to spread your application anywhere!

12. Help the Community: Sinatra is a relatively new framework and it is still ignored or misunderstood by most of the Ruby community. Undoubtedly, Rails is the king of the Ruby web frameworks but I think it's just a matter of time and effort of the few Sinatra developers to show the rest of the world what Sinatra is made of. Your idea can have

an impact in the Sinatra community so post your app on Github and share it! You will learn a lot from this experience.

Next, [Chris Strom's](#) take on this was -

1. **Security** - You are not a master of any web framework without a very solid understanding of web security. You should be familiar with the major attack vectors on the web (SQL injection, XSS, CSRF). You should understand how to mitigate or eliminate those attack vectors, both in principal and with Sinatra.

2. **Rack** - You do not need Rack when just getting started with Sinatra, but you definitely need to understand it, and how Sinatra fits into it, to become proficient with Sinatra.

3. **Sinatra source code** - It is lovely code and there is not too much of it. Read through it and learn a thing or two - about programming and about Sinatra.

4. **Rails** - Sometimes Sinatra is not the right choice. You should be familiar enough with Rails so that you might possess the wisdom to know when to use a particular framework.

5. **HTTP** - You should be comfortable enough with the HTTP protocol to run HTTP sessions from a telnet session. Every now and then, there is not substitute for a good telnet session when trying to troubleshoot.

6. **REST** - Rest is not strictly necessary for simple applications, but you should be very comfortable building REST-like APIs.

7. Ruby (enumerable, blocks, procs, lambdas etc.) - Maybe this goes without saying, but you cannot be a Sinatra master without first being proficient at Ruby. Read and understand the Enumerable documentation. Learn the difference between blocks and lambdas.

8. ORMs (besides ActiveRecord) - Chances are, if you develop enough Sinatra applications, you will need a backend data store. A master is familiar with several so that he or she is able to recognize the right tool for the job.

9. Testing (unit and integration) - If you are tossing together a prototype, you can do without testing. If you are doing anything other than learning, you must test. Driving development by example (BDD) will produce simpler, more elegant design. A combination of unit and integration testing (e.g. Cucumber) will catch regressions. I didn't say it first, but I repeat it as much as possible: robust, accurate, maintainable code is the goal. It is impossible to achieve that goal without testing.

10. Templating systems - You should have a good understanding of the templating options available to you (Haml, Erb, etc.)

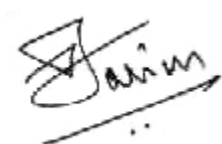
11. 3rd party API gems - Sinatra is great for building interfaces to 3rd party applications, but you need to be familiar with gem libraries that interact with those applications. If there are no such gems - you ought to be comfortable rolling your own.

12. Solid foundation of web principles (sessions, cookies, javascript) - If you are using Sinatra for end-user consumption, you need to be competent with the technologies of the browser.

Finally, [Jeremy Evans](#) had this to say -

1. Just like Rails, keep your controller/actions simple, and put most of your business logic in your models. This makes testing and code reuse easier.
2. Also like Rails, avoid excess logic in your views. Add helper methods that the views call to keep the views clean.
3. Unlike Rails, read the Sinatra source. The main part is a single file that's around 1000 lines of quite understandable Ruby code. Just reading it will probably make you a better programmer.
4. If you have a problem that you think other people probably have (e.g. a Rails-like flash), look first for a Rack middleware that handles it, rather than recreating the wheel.
5. Untested code will probably break sooner than later, so if you want the code to work in the future, write tests.

Best,

A handwritten signature in black ink, appearing to read 'Satish', with a long horizontal stroke extending to the right and a small mark below it.

<http://rubylearning.com/blog/>

If you have any questions, feel free to leave a comment on my blog post all about how to create an income producing product review [here](#).

My Sinatra eBook

The [Introduction to Sinatra eBook](#) is an easy-to-follow guide to everything on **Sinatra**. It is a complete step-by-step guide to quickly creating tiny web-applications and small services in Ruby. I focused entirely on practical advice with specific, actionable how-to tips, tools and techniques. Follow the easy steps and clear tactics to take action today.