How opinionated is too opinionated?

Part of my love/hate relationship with Rails is that I think it too opinionated for its own good. It makes it easier for people to write apps, if you accept the opinions that DHH and the Rails community have baked into the platform over the years.

Don't make me think, or why I switched to Rails from JavaScript SPAs makes a comparison between Javascript and the need to manually select the tools and technologies that we use in our projects in contrast to Rails and its "convention over configuration" approach.

It made me think about the two opposite sides of the coin and which one is better, if you can make the comparison beyond personal preferences.

Two of the pillars of the Rails Doctrine: convention over configuration and The menu is omakase are at the core of how Rails works and why it works the way it does. They are illustrative of why a community would offer opinionated setting and some of their drawbacks.

Most of the time, we use the defaults Rails provide; from creating the application to generating models, views and controllers, to using the database, to using the framework. This works fine as long as we're ok with using these defaults.

But what happens if you want to use MongoDB or PostgresSQL instead of the default SQLite datbase for development or MySQL / MariaDB for production?

You have to edit your database configuration file and ensure that the proper adapter is installed before you can switch databases. You then have to run existing migrations to ensure that the database is in the correct state. And you're not sure how well will these other databases work with Rails, are you?

The problem with having the chef tell you what you will eat is that it may be good or it may be awful, there is no real one-size-fits-all.

Yes, people are building the same box, all Rails users share a common experience, and you can swap portions of Rails for other libraries but it's not intuitive and you're limited to the portions of the framework the community decided you can swap and how well they will work with the rest of rails.

Looking at the front end

For individual projects, we are the only one building the projects and should be comfortable with the code we use and the code we write.

But if we're going to share the projects or are building generic tools t hat we want others to use, being opinionated may be counterproductive. How easy would it be for people to use expand the tool or change it so it fits their own use? How will the opinionated nature of a project detract new uers from adopting it?

One example that I've been working on is Markdownlint. The defaults are good but there are additional options to make it work more consistently.

One of the problems with Markdown is that it is too flexible. For example, these are both valid ways to create headers in markdown:

This is one way I've always seen headers in Markdown; it's the one I see most often:

```
# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6
```

This is an alternative way to create headers in Markdown that I see in WordPress README files:

```
= header 1 =

== Header 2 ==
```

```
=== Header 3 ===

==== Header 4 ====

===== Header 5 =====

====== Header 6 ======
```

The same thing with bulleted lists. There are three different ways to create a bulleted list in Markdown. The following three examples all are valid and represent the same item

```
+ item 1
```

```
- item 1
```

```
* item 1
```

Instead of just telling users how to write Markdown I chose to write a Markdownlint configuration file that will give users warning and errors when publishing Markdown. Yes, this is more painful than if we were to do it in an editor (and I may still do that) but it is more efficient as it doesn't require every developer to install the rules and configure the editor.

```
{
  "heading-style": {
    "style": "atx"
  },
  "atx""ul-style": {
    "style": "asterisk"
  },
  "ul-indent": {
    "indent": 2
  },
  "no-hard-tabs": false,
```

```
  "no-reversed-links": true,
  "no-multiple-blanks": {
    "maximum": 5
  },
  "no-missing-space-atx": true,
  "no-multiple-space-atx": true,
  "blanks-around-headings": true,
  "heading-start-left": true,
  "no-duplicate-heading": {
    "siblings_only": true,
    "allow_different_nesting": true
  },
  "no-trailing-punctuation": true,
  "no-multiple-space-blockquote": true,
  "no-blanks-blockquote": true,
  "blanks-around-fences": true,
  "no-emphasis-as-heading": false,
  "fenced-code-language": true,
  "first-line-h1": true,
  "code-block-style": {
    "style": "fenced"
  },
  "code-fence-style": {
    "style": "backtick"
  },
  "emphasis-style": {
    "style": "asterisk"
  },
  "strong-style": {
    "style": "asterisk"
  },
  "whitespace": false,
  "no-inline-html": false
}
```

This is a simple example. We can do the same thing for PostCSS, Babel, ESLint, and other tools.

The question is, again, how oppinionated is too opinionated? How many of

those rules are really necessary? How strict should we be when enforcing rules?

Another example is Javascript linting. ESlint is a great tool to check syntax, lint, and enforce code style choices but whose choices? In large teams with large codebases like AirBnB, or Google, it makes sense to have a prescriptive set of rules that are enforced in the editor, when you submit pull requests and when the content is pushed to the working branch of the repo.

But do we do the same for our personal projects? Does it make sense to enforce the same rules in our personal projects?

For me it does. I always use Google's ESLint plugin to lint and enforce code style choices

At a higher level, opinionated tools may be good to have but they should always have a way to break the rules to suit the needs for your projects.