The Why And How of Front-End Architecture

Feel the rhythm, feel the rhyme,
get on up, it’s front-end time

Who are you people anyway?

- **Wes Ruvalcaba**
  @wesruv
- **Carwin Young**
  @carwin
- **Sally Young**
  @justafish

Lullabot

Strategy, Design and Development

Front End Architecture

[frənt end ˈärkəˌtek(t)SHər]

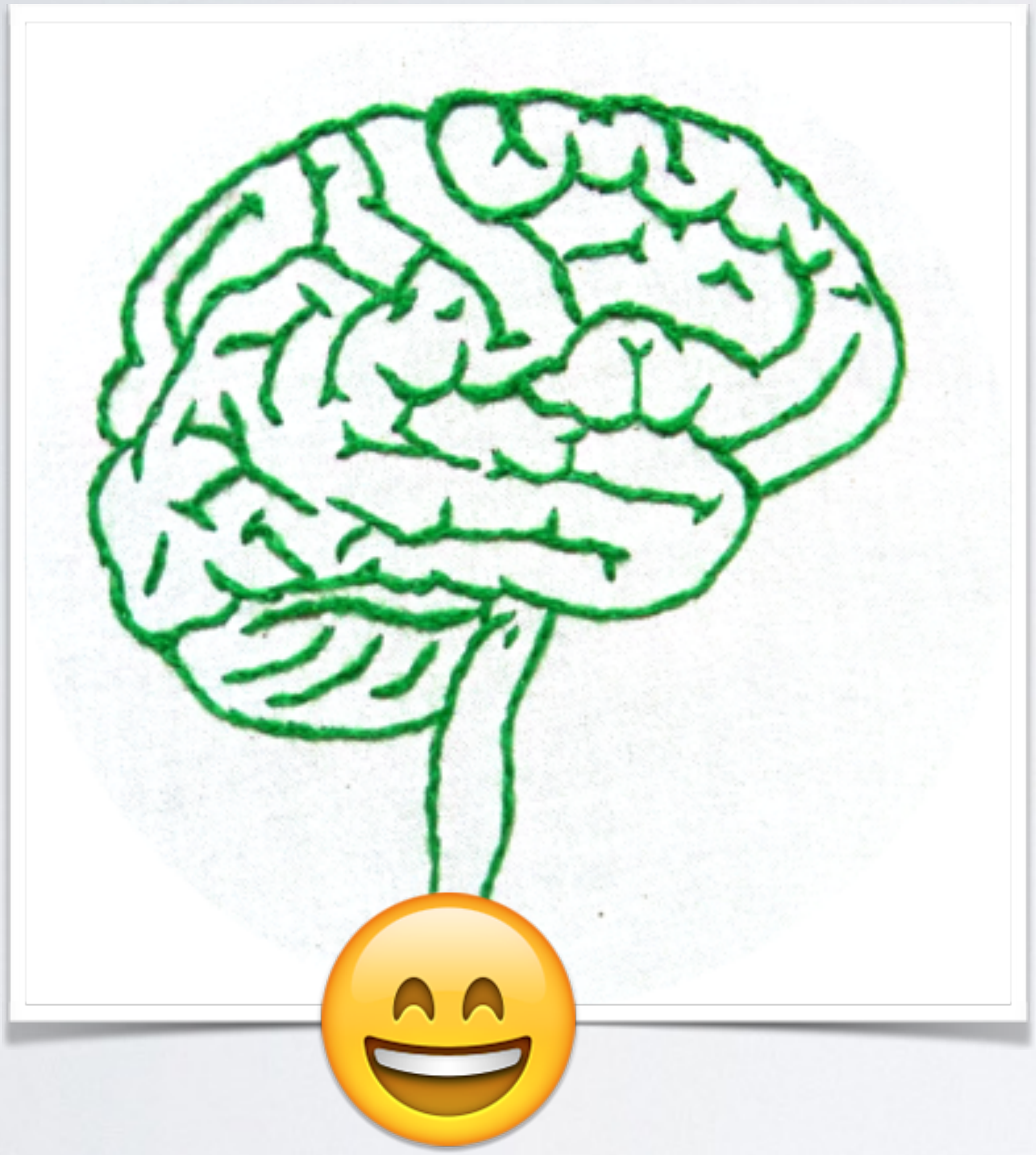
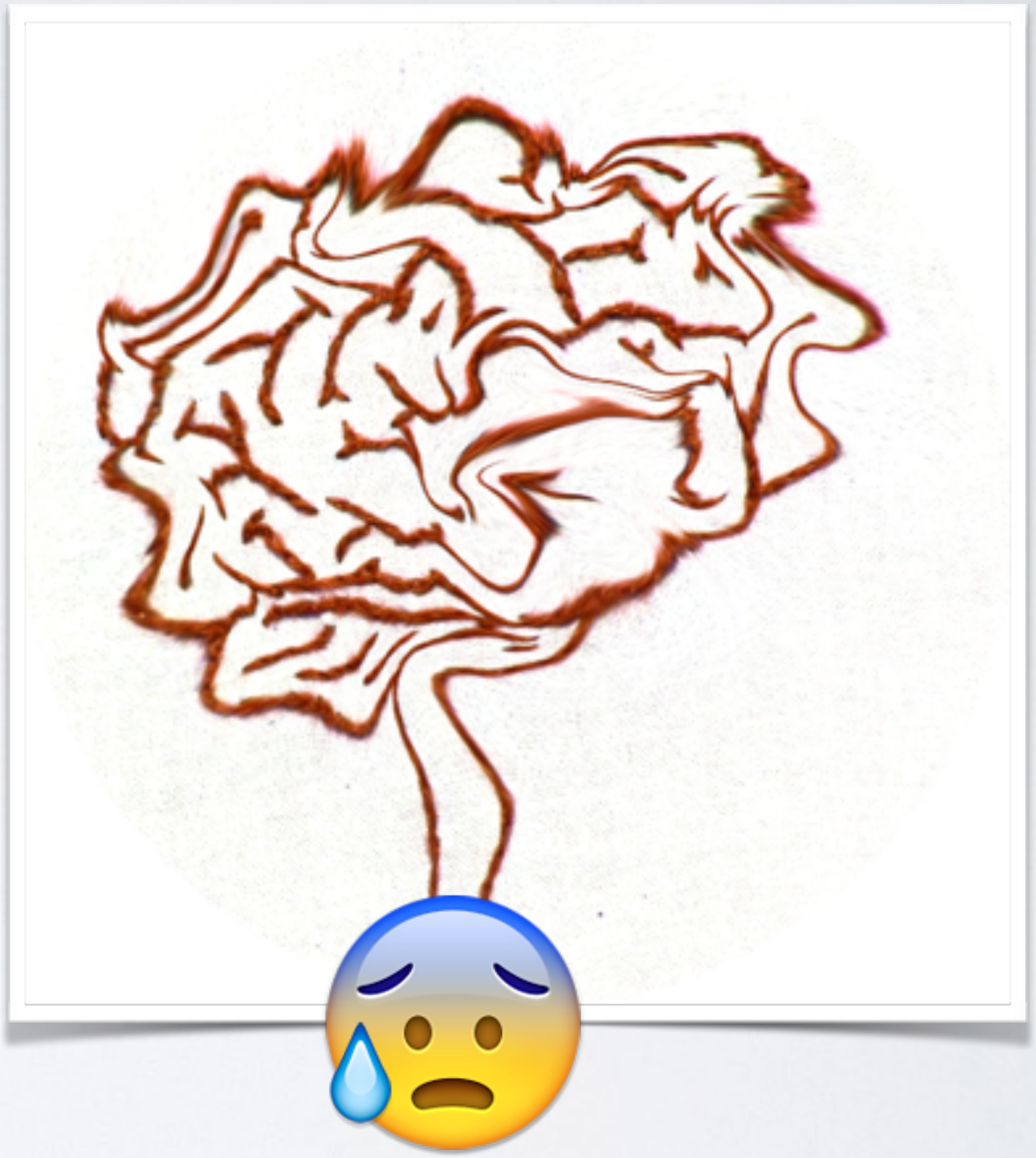
noun

The design for how to work on the Front End of a project. A strategy that helps developers implement and collaborate; and what standards, libraries and tools are being used.

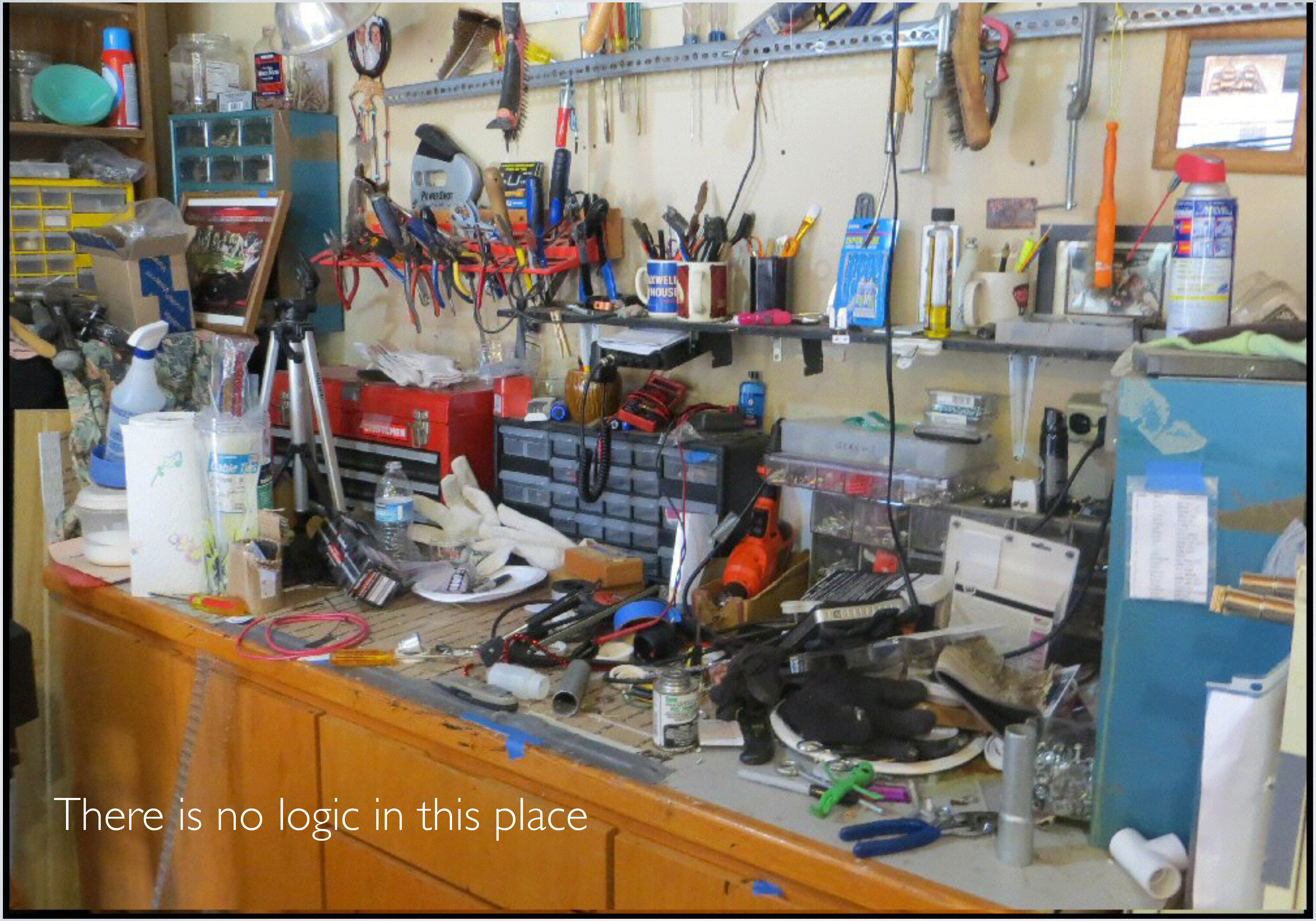
The Byproduct is

(something like)

- Coding standards
- Documentation, Style Guides, etc.
- Implementation Guidelines
- File Organization
- Tools for Building/Testing/Processing
- Included Libraries, Plugins, etc

Organization & Planning



There is no logic in this place

Some kind of plan is better than no plan at all.

Where should I put:

- This fancy new template file?
- This Sass code?
- Custom JavaScript?
- JS Library?

So organized

How to plan

Things to organize:

- Templates
- Preprocessor files (Sass / Less)
- CSS Properties (masochists only)
- JavaScript libraries, helper functions, etc…
- Literally whatever else you have going on

EVERYTHING.

HTML, Templates, and preprocess

You can organize this!

Do these functions or files rely on code provided by a module?

- On a large project, you should probably keep it with that module
- On a really small project, it might be better all in one place like the theme

Small Decisions Eat Lots of Time

```
while $decisions < $over_engineering {
  $developer_sanity++;
}
```

JavaScript

You already know this

CSS

As front-endy as it gets

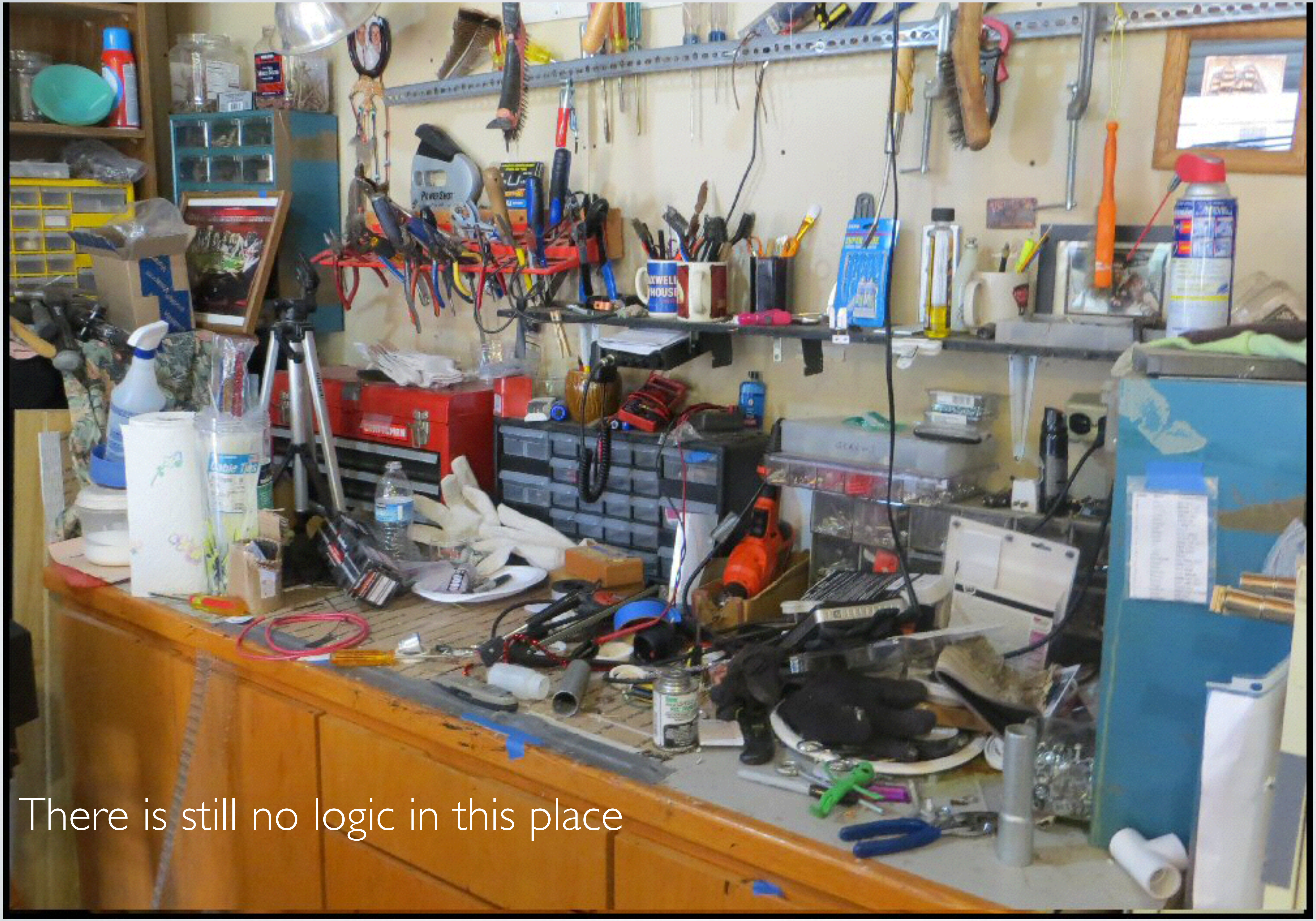
“Some kind of plan is better than no plan at all.”

–Me, earlier in this talk

CSS Methodology Types

- **Component** - highly modular, discrete chunks of CSS / Markup
- **Utility** - the lego version of CSS, individual classes that do very few things (think .underline, or .red)
- **Hybrid** - a mix of the two (good for the environment)

- WOULD YOU RATHER MODIFY MARKUP?
  Utility Methodologies++
- WOULD YOU RATHER MODIFY STYLES?
  Component Methodologies++
- WANT TO PLAY IT SAFE?
  Drupal 8 already has standards. Use those.



There is still no logic in this place

Set aside time for architectural decisions

& keep them simple

“Some kind of plan is better than no plan at all.”

–Someone Great, earlier in this talk

Build tools

Why Adopt Build Tools?

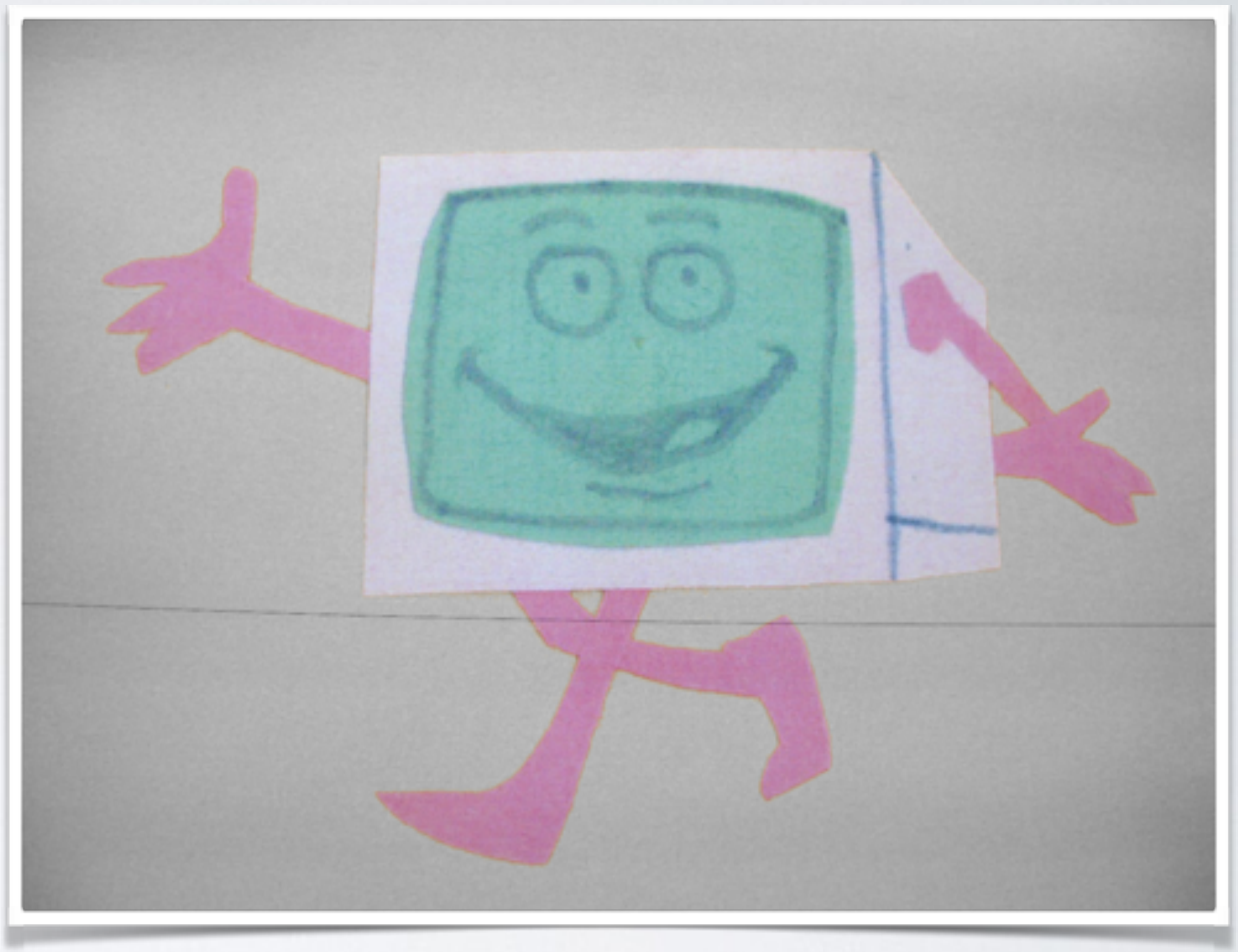
What are the advantages?

Adds a layer of abstraction

- Work in compiled languages
- How you work ≠ how code is delivered
- Code can be DRYer and compartmentalized
- Can help bridge gap between skill levels

Automate tasks

Computers love repetitive menial tasks!



- Linting
- Repetitive command line tasks
- Minification
- Compiling
- Browser-prefixing
- Reloading the browser when a file changes

Task Runners

e.g. Grunt & Gulp

Common front End Task

Might be…

- Compile
- Autoprefix
- Minify
- Save the end product
- Reload browser

Caution

all who enter

If you’re new to build tools

- There’s a learning curve
- Getting your first setup will come with bumps
- The documentation for some isn’t great, look for articles

General Warnings

- There will be (some) maintenance cost
- Don’t Over-engineer
- Beware long build times!
  (But this can almost always be addressed)
- Front End Build Tools are still young, there will be change, but it is calming down
- Can increase developer specialization

How do we decide when we should (not) use Build tools

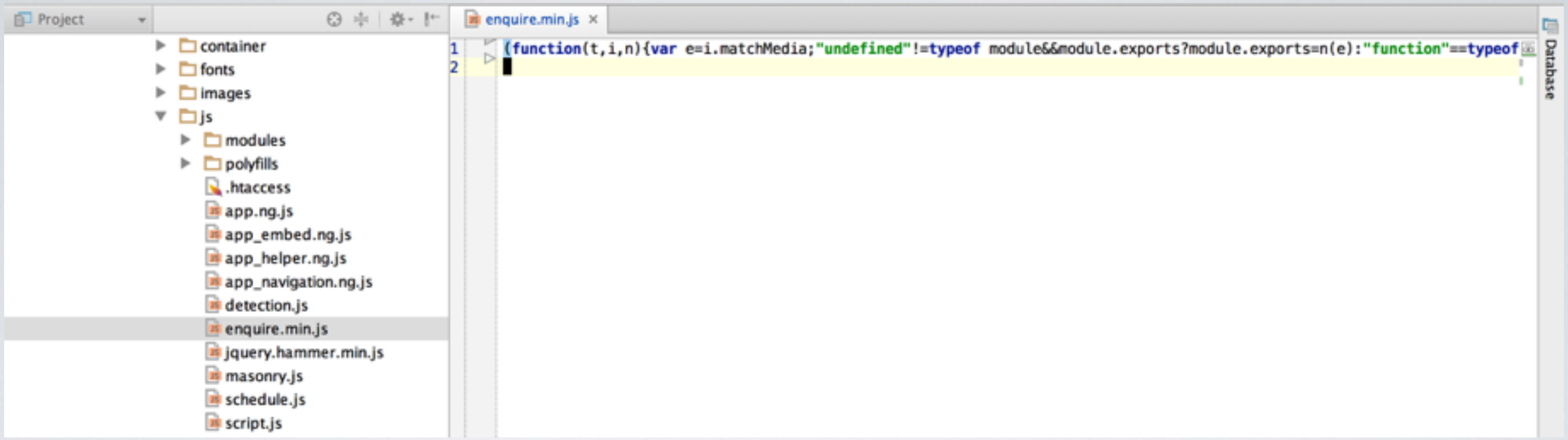
Small teams / Projects

- Set up time can be prohibitive
- It can hamper make cross-functional team members
- BUT if you have a common set of problems that build tools can solve, it can be really helpful

Larger Teams / Projects

- Build tools really help building for scale in team and code base
- Helpful to have a point person for maintenance
- Make sure it’s adding value, not frustration
- Watch for long build times
- Requiring a lot of command line knowledge
- Document, Document, Document!

Package Managers

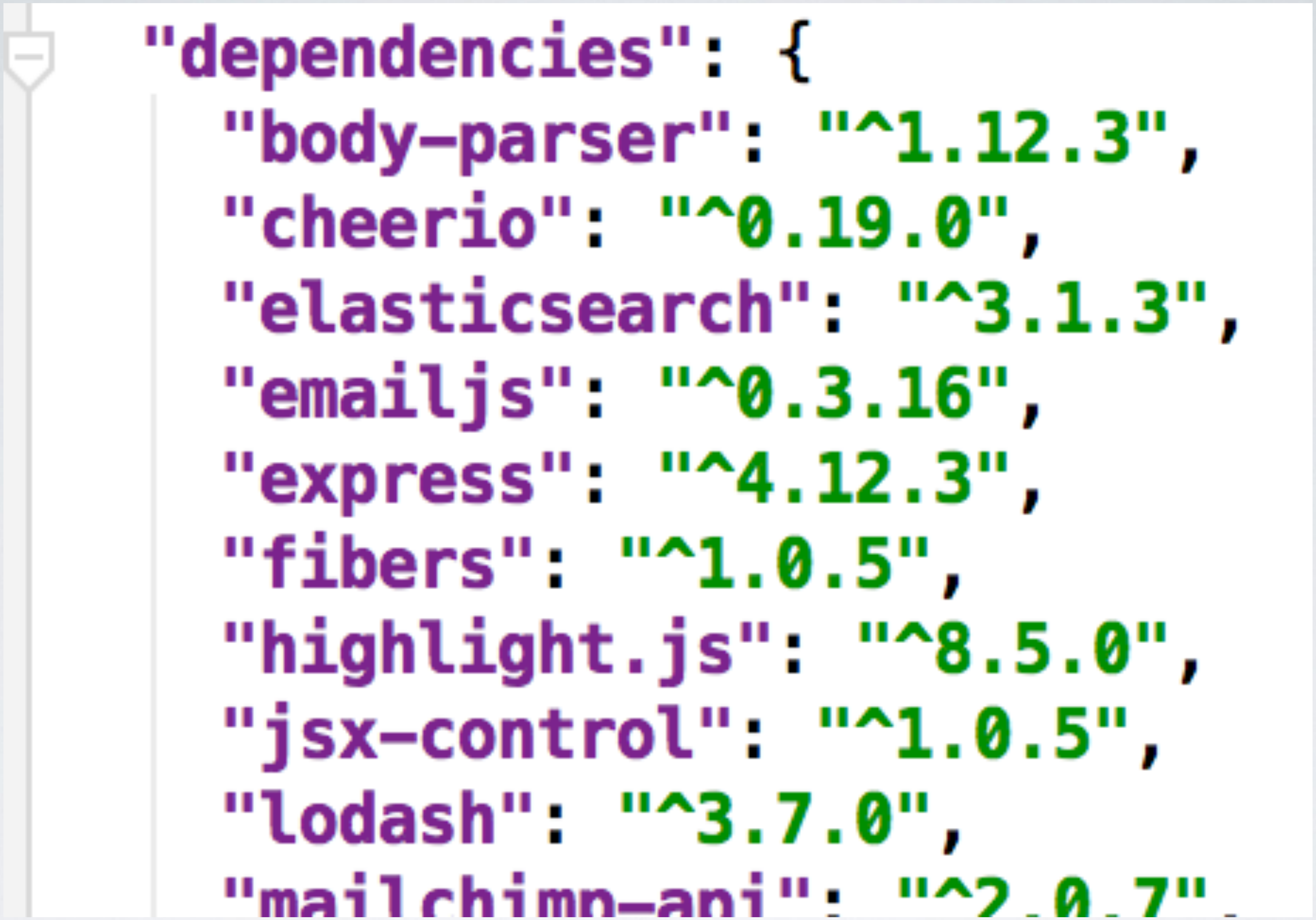


```
(function(t,i,n){var e=i.matchMedia;"undefined"!=typeof module&&module.exports?module.exports=n(e):"function"==typeof
```

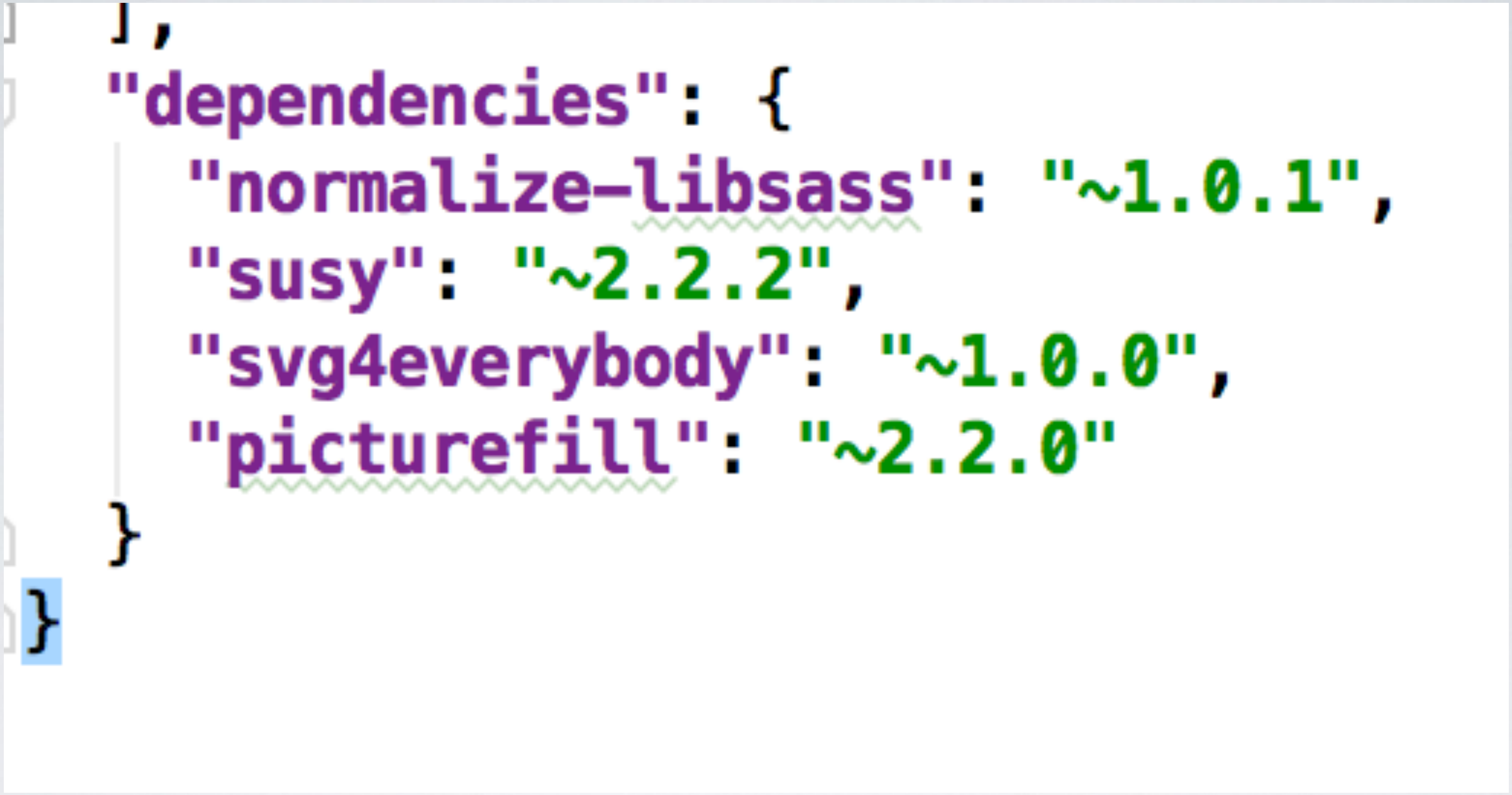
- What does this do?
- Where does it come from?
- What version is it?

Package Managers

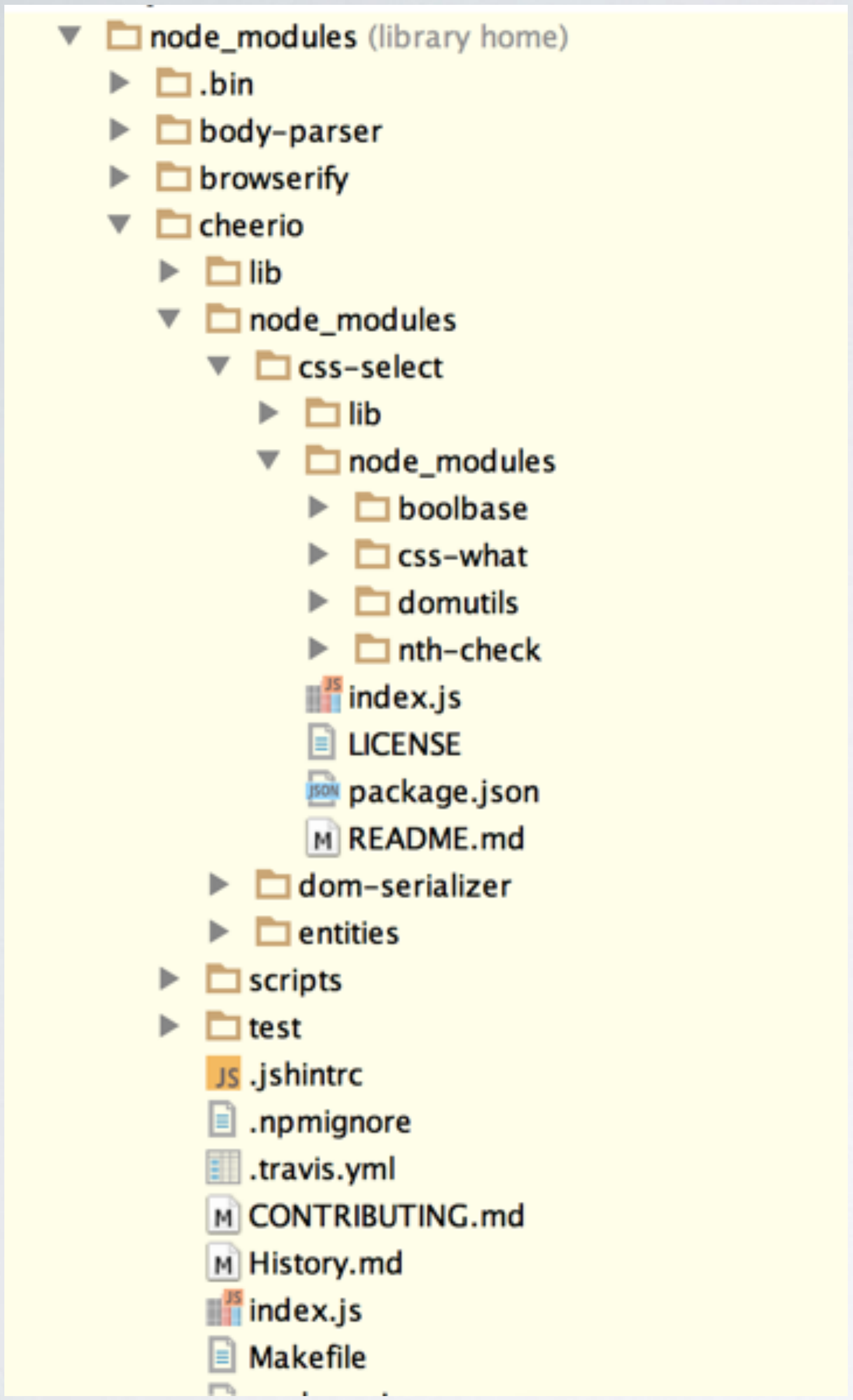
Used for external dependencies e.g. JavaScripts, CSS Frameworks



```
"dependencies": {
  "body-parser": "^1.12.3",
  "cheerio": "^0.19.0",
  "elasticsearch": "^3.1.3",
  "emailjs": "^0.3.16",
  "express": "^4.12.3",
  "fibers": "^1.0.5",
  "highlight.js": "^8.5.0",
  "jsx-control": "^1.0.5",
  "lodash": "^3.7.0",
  "mailchimp-api": "^2.0.7",
```

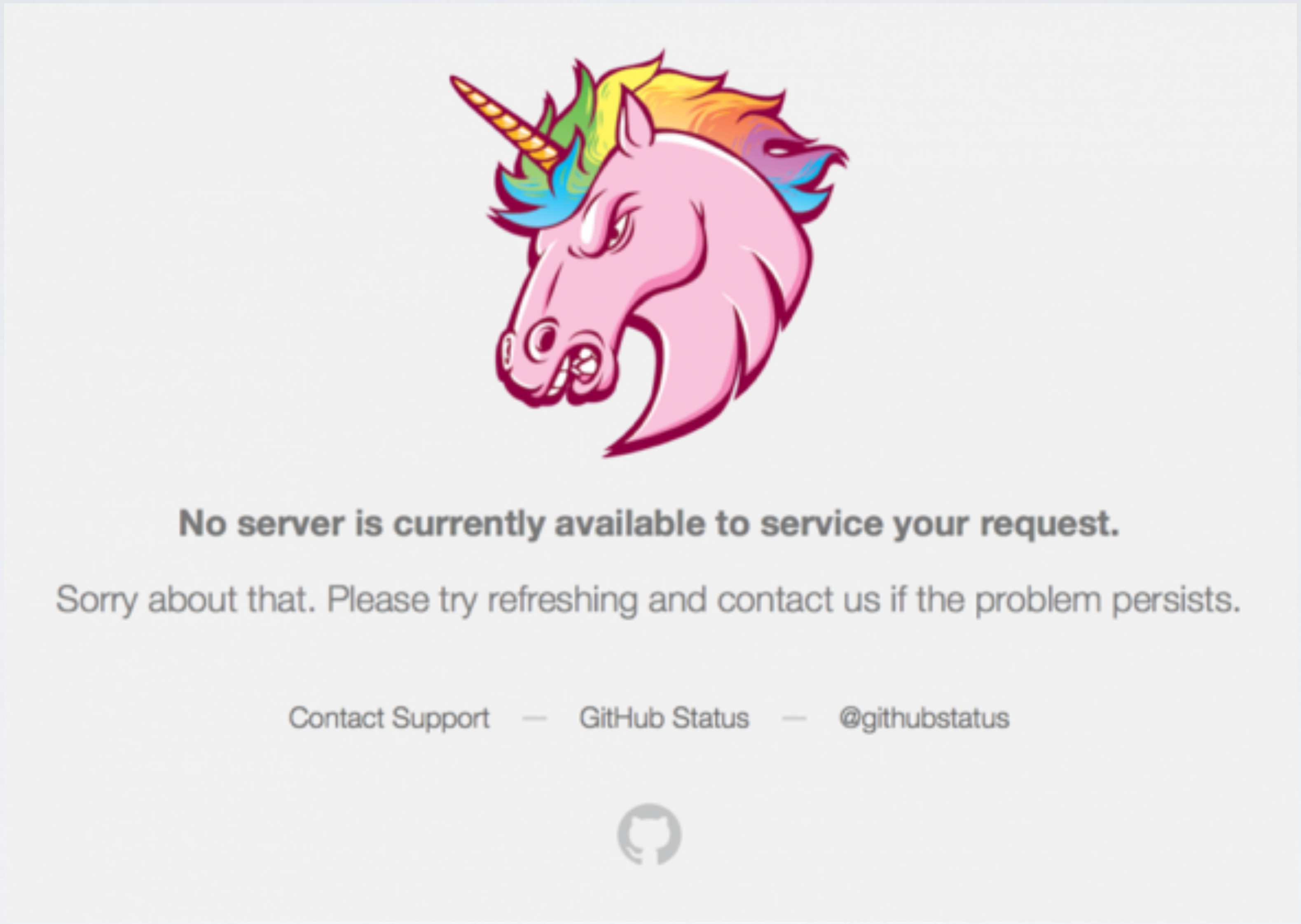


```
  ],
  "dependencies": {
    "normalize-libsass": "~1.0.1",
    "susy": "~2.2.2",
    "svg4everybody": "~1.0.0",
    "picturefill": "~2.2.0"
  }
}
```



```
▼ node_modules (library home)
  ▶ .bin
  ▶ body-parser
  ▶ browserify
  ▼ cheerio
    ▶ lib
    ▼ node_modules
      ▼ css-select
        ▶ lib
        ▼ node_modules
          ▶ boolbase
          ▶ css-what
          ▶ domutils
          ▶ nth-check
          index.js
          LICENSE
          package.json
          README.md
      ▶ dom-serializer
      ▶ entities
    ▶ scripts
    ▶ test
    .jshintrc
    .npmignore
    .travis.yml
    CONTRIBUTING.md
    History.md
    index.js
    Makefile
```

Downsides


No server is currently available to service your request.

Sorry about that. Please try refreshing and contact us if the problem persists.

Contact Support — GitHub Status — @githubstatus

Testing

- Visual Testing including Regression Testing
- Unit tests
- Code sniffing, linting etc

Front End Frameworks

e.g. Bootstrap, Susy Grids, Yeti, Foundation

Benefits

- A lot of testing, grunt work, and coding already done
- Great Documentation done
- A lot of support
- Built to help devs of all Front End skill levels build interfaces
- A lot of FE Arch decisions made for you

Downsides

- Stick to the design they give you… or else!
- Code bloat *could be* an issue
- Specificity wars
- A lot of FE Arch decisions made for you

When might you adopt a framework?

- Pragmatism over idealism; crunches in time, team abilities, QA or other factors
- Supporting a lot of other devs that aren't as front end savvy

TL;DL

Some kind of plan is better than no plan at all

Build tools are good… (probably?)

Package managers… yes please!

Testing is good… but people are good too!

Frameworks - use at your own risk

Q & A