Not Quite a Library

Reusing Components Before You’re Sure How

By Dylan Hughes

Who’s this guy?

Agenda

1. What is a component library?
2. Why is a component library useful?
3. Choosing what goes in
4. Preparing components
5. Sharing strategies
6. Considerations
7. Summary
8. Questions

Disclaimers

1. Assumed Vue knowledge
2. I am bad at automated deploys
3. Product organization centric

What is a component library?

tl;dr: a place to put intentionally shared components

Should answer the following questions:

1. What components do we have available?
2. Where do I get them?
3. How do they work?

Other Names:

1. UI Library
2. Pattern Library

Some examples:

1. BootstrapVue
2. Lonely Planet
3. Salesforce Lightning
4. IBM's Carbon
5. AwesomeVue
6. VueMaterial

Component Library vs. Style Guide

Style Guide:

"*A style guide is a collection of pre-designed elements, graphics and rules designers or developers should follow to ensure that separate website pieces will be consistent and will create a cohesive experience at the end.*"
-Derek Bradley

Thoughts:

1. Style guide !== component library
2. Style guide should *inform* your component library
3. A style guide contains design elements that should make up your components
4. Things get blurry when designers write code

Why?

1. DRY
2. Facilitates reusability
3. More time in production
4. Ease of maintenance
5. Enables rapid prototyping

Choosing what goes into the library

Different Perspectives

Design Perspective

1. Design elements inform components
2. Ensure design consistency

Team Perspective

1. “Didn’t ____ build that already?”
2. Third time’s the charm
3. Ensuring consistency

Code Perspective

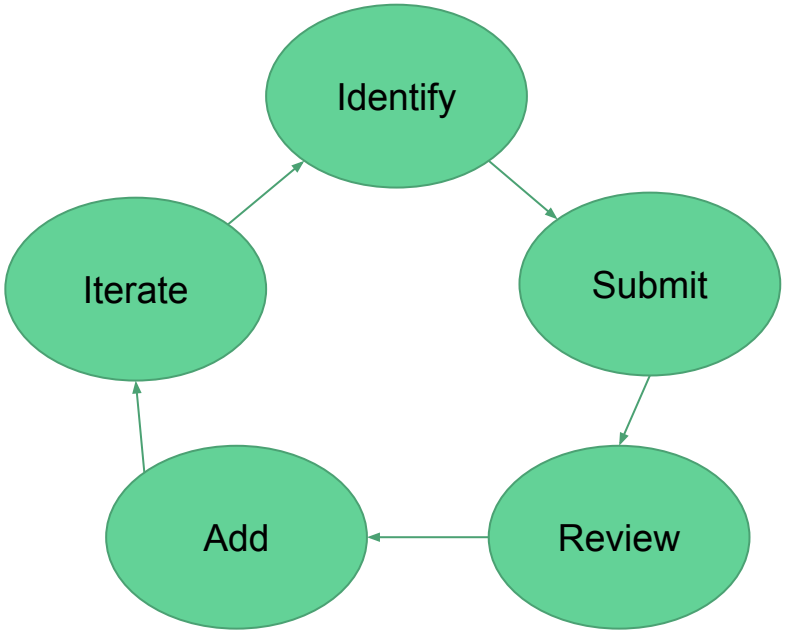
1. Adding non-breaking changes
2. Adding new functionality that benefits others
3. Fixing bugs in existing components

User Perspective

1. User expectations

Empowering Contribution

1. Use existing leadership structures & processes for review
    a. Existing positions (Principle Engineers, Frontend Engineers, Random Cabal, etc.)
    b. Github teams
2. Create a loose group if necessary
3. Everyone should feel able to contribute



Generic Product Company, Inc

Structure

1. 3 core products
    a. ToDo App
    b. Todo Analytics
    c. Document Sharing
2. 3 Product Teams
    a. Product Manager
    b. Designer
    c. Dev Lead
    d. 2-5 Software Engineers
3. Frontend Apps Deployed & Built Separately

Directory Structure

frontends
    todo-app
    todo-analytics
    document-sharing

Preparing components

Code

Sharing Strategies

1. Shared Folder

How: Use es6 imports to include components from a shared folder inside your top level directory

Code

Pros

1. Cheap
2. Quick
3. Almost no maintenance

Cons

1. Difficult to version
2. “Can’t” share outside the monorepo
3. Potential for side effects
4. Format doesn’t encourage documentation

2. npm Modules

How

1. Public/Open Source
    a. Least appealing option for many companies
    b. Not going to work for GPC, Inc
2. Pay for npm
3. Host our own npm registry
    a. Verdaccio
    b. Nexus Repository

Our SearchBar

Finding Your Private Component

1. If you set up a private registry

```
npm config set registry <registry url>
```

(Can also use an .npmrc file)

```
npm adduser [--registry=url] [--scope=@orgname]
```

```
npm install @org-name/component-name
```

Docs

Our Component

Code

Pros

1. Versioned
2. Frontend applications can live in separate repositories
3. Facilitates documentation

Cons

1. Requires work or money
2. Smaller companies may find $7/user/month very worth it
3. Larger companies may find it easier to deploy their own registry into existing service architecture
4. Does your company have more money or time?

What would it take to get to a library?

Considerations

1. Build or buy
2. Ownership
3. Living documentation
4. Testing
5. Private vs. Public
6. Overhead

Summary

- What is a component library?
  - A place to intentionally share components
  - Different from a style guide
- Why is a component library useful?
  - DRY
  - Facilitates reusability
  - More time in production
  - Ease of maintenance
  - Enables rapid prototyping
- Choosing what goes in
  - Different perspectives
- Preparing Components
  - Component API
  - Free of side effects
- Sharing Strategies
  - Shared folder
  - List of npm modules
  - Dedicated software
- Considerations
  - Build or buy
  - Ownership
  - Living documentation
  - Testing
  - Private vs. Public
  - Overhead

Questions?

Links / Contact

- Dylan Hughes
  - Twitter: @dylanbhughes
  - Github: dylanbhughes
  - Email: dhughes@optoro.com
- Links
  - Code
  - Verdaccio
  - Nexus Repository
  - npm pricing
  - Generic Product Company, Inc npm org
- Optoro
  - optoro.com/careers

Further Learning

1. Pattern Library
2. Egghead - Reusable Vue Components
3. Reusable Components
4. Standalone Components
5. vue-cli
6. npm orgs