How I got Vue and Vuetify to work in Rails

Steps to generate a working Rails app with Vue:

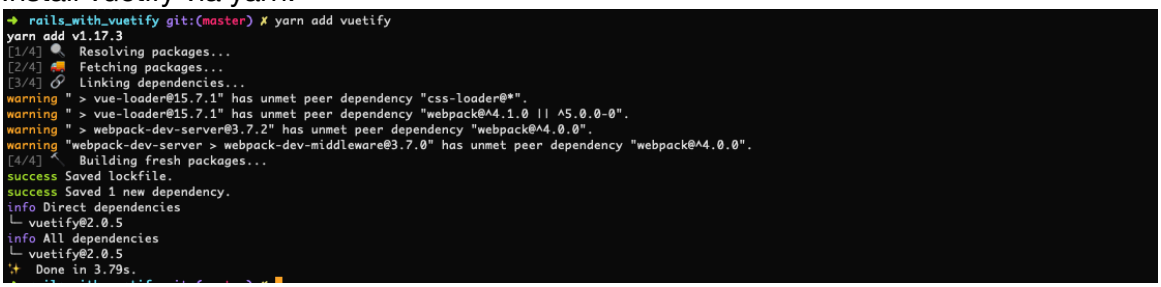
1. rails new <app name>
2. Add to the gemfile and run ‘bundle'

```
gem 'webpacker'
gem 'foreman'
```

3. Install webpacker, vue, and yarn

```
$ bin/rails webpacker:install
$ bin/rails webpacker:install:vue
$ bin/yarn install
```

4. Install vuetify via yarn.



5. Create Procfile for foreman (in the rails app root dir):

```
backend: bin/rails s -p 3000
frontend: bin/webpack-dev-server
```

6. Generate a controller and action that we can use to verify we have things installed.

```
➜  rails_with_vuetify git:(master) ✗ rails g controller landing index
Running via Spring preloader in process 37294
      create  app/controllers/landing_controller.rb
       route  get 'landing/index'
      invoke  erb
      create    app/views/landing
      create    app/views/landing/index.html.erb
      invoke  test_unit
      create    test/controllers/landing_controller_test.rb
      invoke  helper
      create    app/helpers/landing_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/landing.coffee
      invoke    scss
      create      app/assets/stylesheets/landing.scss
```

7. Modify the application.html.erb template in two ways: replace javascript_include_tag 'application' with javascript_pack_tag 'application' and insert a div around the yield statement that will give us a unique element we can point our Vue instance to in a bit.

```
<!DOCTYPE html>
<html>
  <head>
    <title>RailsWithVuetify</title>
    <%= csrf_meta_tags %>
    <%= csp_meta_tag %>

    <%= stylesheet_link_tag    'application', media: 'all', 'data-turbolinks-track': 'reload' %>
    <%= javascript_pack_tag 'application', 'data-turbolinks-track': 'reload' %>
  </head>

  <body>
    <div id='vue-target'>
      <%= yield %>
    </div>
  </body>
</html>
```

8. I like to rearrange the default folder structure that was setup by installing Vue. In particular, I like a components directory where I put all the components, even app.vue. I moved app.vue into the new components directory. I also deleted hello_vue.js because we're going to do that ourselves in a different spot (application.js). My structure looks like this:

9. Now modify the app/javascript/packs/application.js to load Vue and a basic test component.

```
/* eslint no-console:0 */
// This file is automatically compiled by Webpack, along with any other files
// present in this directory. You're encouraged to place your actual application l
// a relevant structure within app/javascript and only use these pack files to ref
// that code so it'll be compiled.
//
// To reference this file, add <%= javascript_pack_tag 'application' %> to the app
// layout file, like app/views/layouts/application.html.erb


// Uncomment to copy all static images under ../images to the output folder and re
// them with the image_pack_tag helper in views (e.g <%= image_pack_tag 'rails.png
// or the `imagePath` JavaScript helper below.
//
// const images = require.context('../images', true)
// const imagePath = (name) => images(name, true)

// console.log('Hello World from Webpacker')

// import the Vue object
import Vue from 'vue'

// import the base app component... Note that I modified the directory
// structure and location of files a bit to fit what I like...
// all my components in a single components/ folder. So,
// I created javascript/components/ and I moved the app.vue file into it.
import App from '../components/app.vue'

// there are several ways to register components for use...I'm going to
// register them here globally and name them with my own convention of
// "-component"
Vue.component('app-component', App)

// Now we need to create the Vue object and tell it to bind to the
// html element we've created in the application.html.erb template. We'll
// do that by adding an event listener that will trigger the Vue object
// creation when the page's DOM is finished loading.
document.addEventListener('DOMContentLoaded', () => {
  const vue_app = new Vue({
    el: 'div#vue-target'
  })
})
```

10. Add a default root route to the routes.rb file that points to our landing/index page:

```
Rails.application.routes.draw do
  root to: 'landing#index'
end
```

11. Fire up the browser and point it toward localhost:3000. Uh oh…we get this error:

To fix this, we need to tell the system to load a slightly different vue file. Change line

21 to look like this:

```
// import the Vue object
import Vue from 'vue/dist/vue.esm'
```

Save and refresh, and you should get a screen that looks like this:

Hello Vue!

12. Actually, that fix to pull in the build version of Vue won't work for long. There's a better way (or at least the only way I found). Go back to application.js and set the import Vue line to reference just 'vue' again. Then, find the environment.js file in <rails root>/app/config/webpack/. Add the code that is in lines 10-17 below.

```
const { environment } = require('@rails/webpacker')
const { VueLoaderPlugin } = require('vue-loader')
const vue = require('./loaders/vue')

environment.plugins.prepend('VueLoaderPlugin', new VueLoaderPlugin())
environment.loaders.prepend('vue', vue)


// see https://stackoverflow.com/questions/53582944/how-to-properly-install-vuetify-for-rails
const resolver = {
  resolve: {
    alias: {
      'vue$': 'vue/dist/vue.esm.js'
    }
  }
}
environment.config.merge(resolver)



module.exports = environment
```

Now, let's add Vuetify to the app:

1. Install Vuetify using yarn:

```
➜  rails_with_vuetify git:(master) ✗ yarn add vuetify
yarn add v1.17.3
[1/4] 🔍  Resolving packages...
[2/4] 🚚  Fetching packages...
[3/4] 🔗  Linking dependencies...
warning " > vue-loader@15.7.1" has unmet peer dependency "css-loader@*".
warning " > vue-loader@15.7.1" has unmet peer dependency "webpack@^4.1.0 || ^5.0.0-0".
warning " > webpack-dev-server@3.7.2" has unmet peer dependency "webpack@^4.0.0".
warning "webpack-dev-server > webpack-dev-middleware@3.7.0" has unmet peer dependency "webpack@^4.0.0".
[4/4] 🔨  Building fresh packages...
success Saved lockfile.
success Saved 1 new dependency.
info Direct dependencies
└─ vuetify@2.0.5
info All dependencies
└─ vuetify@2.0.5
✨  Done in 3.79s.
```

2. My package.json file now looks like this:

```
{
  "name": "rails_with_vuetify",
  "private": true,
  "dependencies": {
    "@rails/webpacker": "^4.0.7",
    "vue": "^2.6.10",
    "vue-loader": "^15.7.1",
    "vue-template-compiler": "^2.6.10",
    "vuetify": "^2.0.5"
  },
  "devDependencies": {
    "webpack-dev-server": "^3.7.2"
  }
}
```

3. Let's use the Vuetify hello world demo to see if we have things working. First, let's replace the app.vue component with one that uses Vuetify components. Code is here: https://github.com/iamshaunjp/vuetify-playlist/blob/lesson-2/todo-ninja/src/App.vue

```vue
<template>
  <v-app>
    <v-toolbar app>
      <v-toolbar-title class="headline text-uppercase">
        <span>Vuetify</span>
        <span class="font-weight-light">MATERIAL DESIGN</span>
      </v-toolbar-title>
      <v-spacer></v-spacer>
      <v-btn
        flat
        href="https://github.com/vuetifyjs/vuetify/releases/latest"
        target="_blank"
      >
        <span class="mr-2">Latest Release</span>
      </v-btn>
    </v-toolbar>

    <v-content>
      <HelloWorld/>
    </v-content>
  </v-app>
</template>

<script>
import HelloWorld from './components/HelloWorld'
export default {
  name: 'App',
  components: {
    HelloWorld
  },
  data () {
    return {
      //
    }
  }
}
</script>
```

4. Next, let's create a hello world component. Create helloworld.vue in the components directory and fill it with the code from here: https://github.com/iamshaunjp/vuetify-playlist/blob/lesson-2/todo-ninja/src/components/HelloWorld.vue No screen shot for this since it's a big file.


```
<template>
  <v-container>
    <v-layout
      text-xs-center
      wrap
    >
      <v-flex xs12>
        <v-img
          :src="require('../assets/logo.svg')"
          class="my-3"
          contain
          height="200"
        ></v-img>
      </v-flex>

      <v-flex mb-4>
        <h1 class="display-2 font-weight-bold mb-3">
          Welcome to Vuetify
        </h1>
        <p class="subheading font-weight-regular">
          For help and collaboration with other Vuetify developers,
          <br>please join our online
          <a href="https://community.vuetifyjs.com" target="_blank">Discord Community</a>
        </p>
      </v-flex>

      <v-flex
        mb-5
        xs12
      >
        <h2 class="headline font-weight-bold mb-3">What's next?</h2>

        <v-layout justify-center>
          <a
            v-for="(next, i) in whatsNext"
            :key="i"
            :href="next.href"
            class="subheading mx-3"
            target="_blank"
          >
            {{ next.text }}
          </a>
        </v-layout>
```

5. Also, download the logo.svg file from that same repository. Drop it into an ‘assets’ folder you create under the javascript folder.
6. Now we need to import and register the helloworld component. Back to the application.js file. And, we need to import Vuetify and tell Vue to use it.
7. Since I changed the name of the hello world component during the import and because I registered it globally, I need to update the app component a bit:

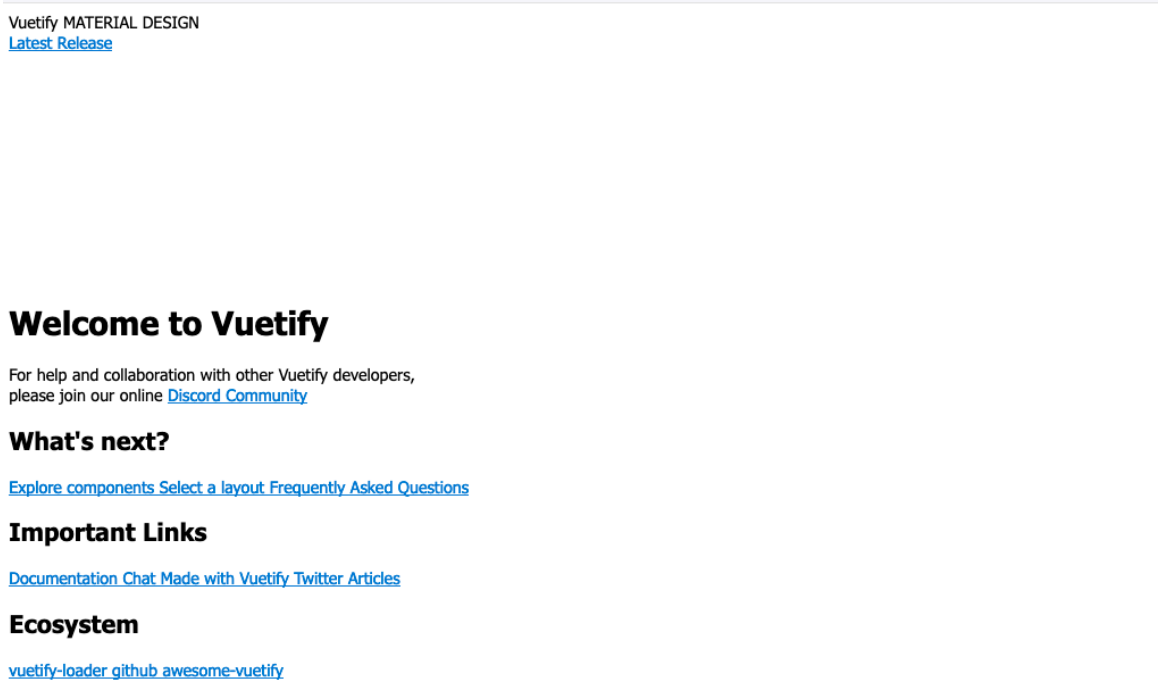
```
<template>
  <v-app>
    <v-toolbar app>
      <v-toolbar-title class="headline text-uppercase">
        <span>Vuetify</span>
        <span class="font-weight-light">MATERIAL DESIGN</span>
      </v-toolbar-title>
      <v-spacer></v-spacer>
      <v-btn
        flat
        href="https://github.com/vuetifyjs/vuetify/releases/latest"
        target="_blank"
      >
        <span class="mr-2">Latest Release</span>
      </v-btn>
    </v-toolbar>

    <v-content>
      <!-- Changed the name of the component -->
      <hello-world-component/>
    </v-content>
  </v-app>
</template>

<script>
  // Since the component has been registered globally, we don't need this
  // import HelloWorld from './components/HelloWorld'
export default {
  name: 'App',
  // Since the component has been registered globally, we don't need this
  // components: {
  //   HelloWorld
  // },
  data () {
    return {
      //
    }
  }
}
</script>

```

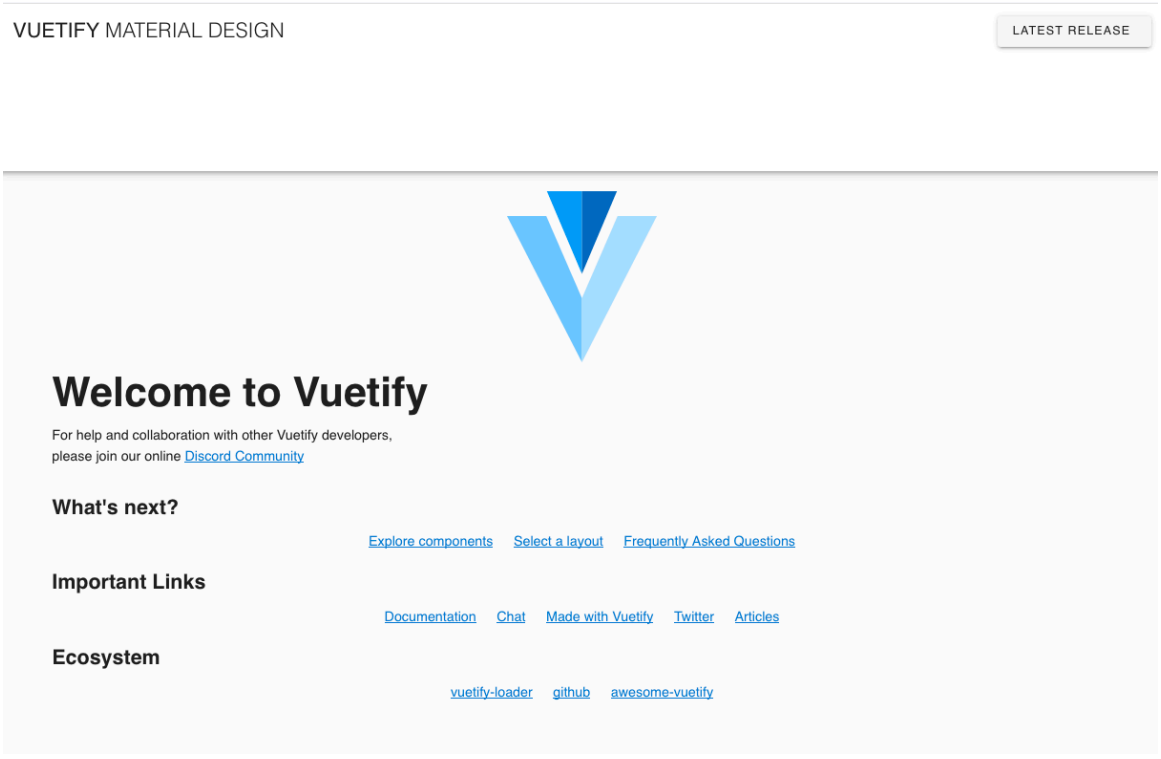
8. At the command prompt, run ‘foreman start’ to spin up both the Rails server and the webpack server. Visit localhost:3000 and you should see this:



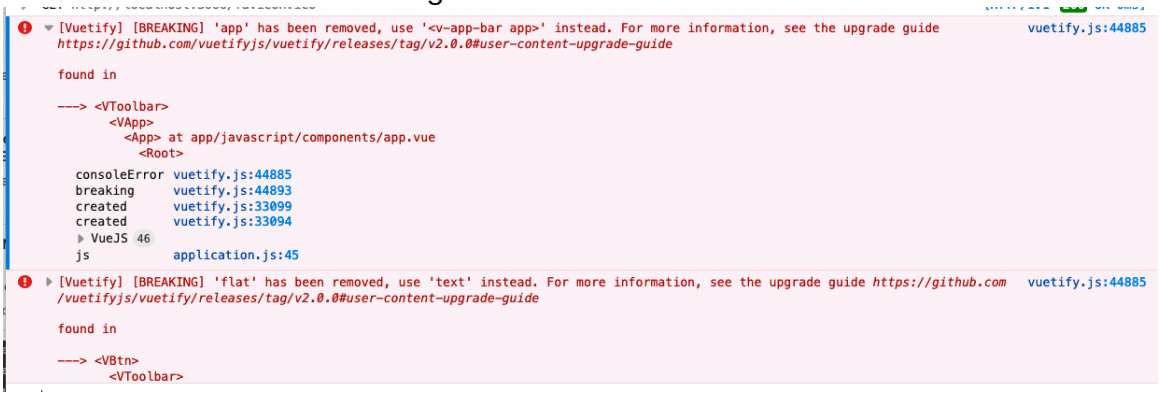
9. That doesn't look quite right…and it's because we need to manually import a Vuetify style sheet. Back in application.js, add a line to import the 'vuetify/dist/vuetify.min.css' file:

```
19
20  // import the Vue object
21  import Vue from 'vue'
22  import Vuetify from 'vuetify'
23  import 'vuetify/dist/vuetify.min.css'
24
```

10. Refresh, and now it looks a bit better:



11. If you look at the browser console, you'll see that this code is using some older Vuetify statements and we should change them:



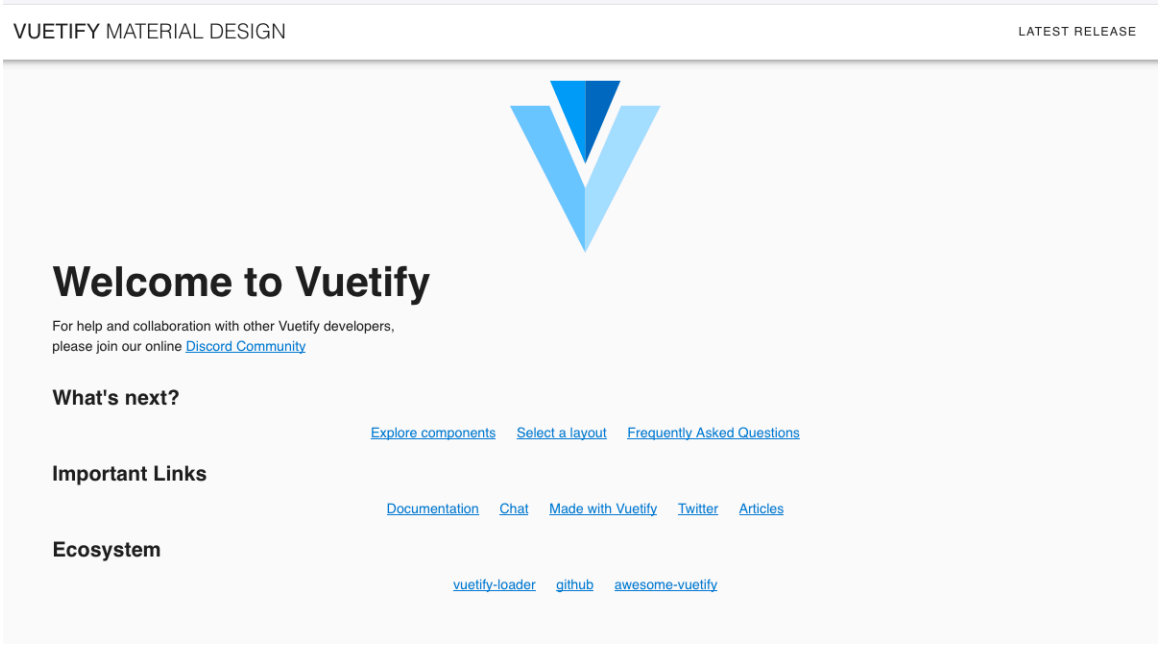
So, change v-toolbar to v-app-bar (and its closing tag) and change 'flat' to 'text' in the v-btn.

```
<template>
  <v-app>
    <v-app-bar app>
      <v-toolbar-title class="headline text-uppercase">
        <span>Vuetify</span>
        <span class="font-weight-light">MATERIAL DESIGN</span>
      </v-toolbar-title>
      <v-spacer></v-spacer>
      <v-btn
        text
        href="https://github.com/vuetifyjs/vuetify/releases/latest"
        target="_blank"
      >
        <span class="mr-2">Latest Release</span>
      </v-btn>
    </v-app-bar>

    <v-content>
      <!-- Changed the name of the component -->
      <hello-world-component/>
    </v-content>
  </v-app>
</template>

<script>
  // Since the component has been registered globally, we don't need this
  // import HelloWorld from './components/HelloWorld'
export default {
  name: 'App',
  // Since the component has been registered globally, we don't need this
  // components: {
  //   HelloWorld
  // },
  data () {
    return {
      //
    }
  }
}
</script>
```

12. Refresh and now the app bar at the top looks better, and the link to Latest Release looks flat.



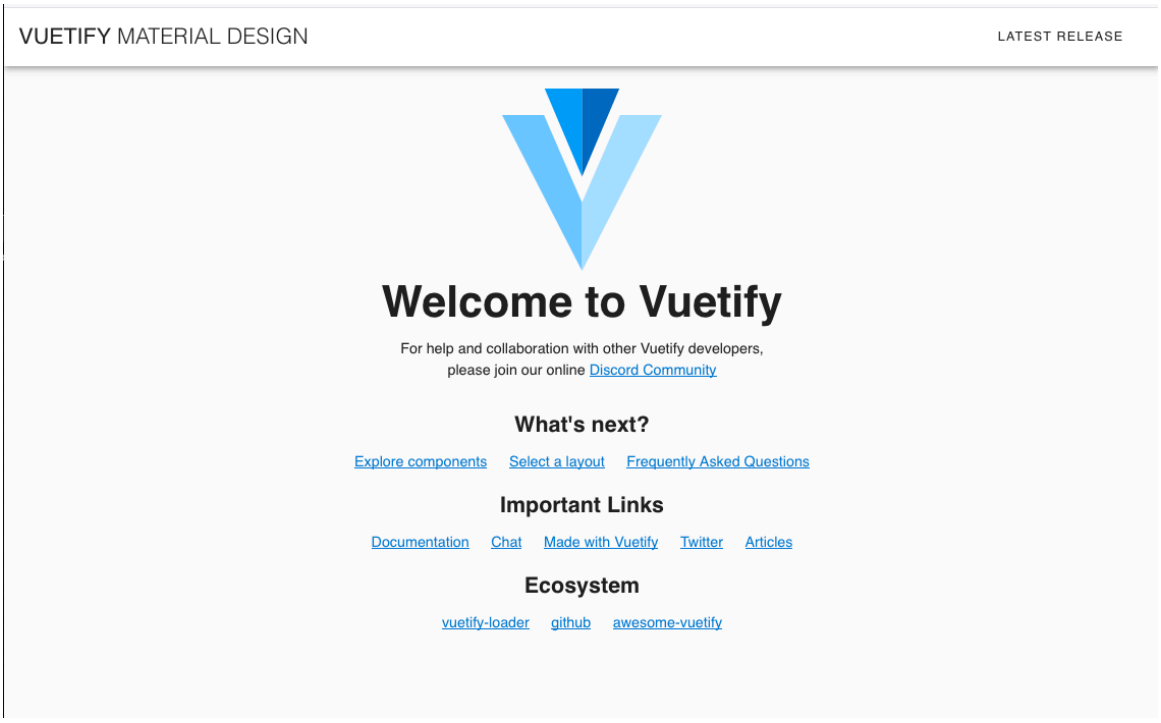
13. Last step…I like the text to be centered for this demo, so I'm going to change a directive on the layout. Line 4 of the helloworld.vue component directs Vue to center the layout on extra small screens, but I want it centered on all screens. To do this, I just change the directive from text-xs-center to text-center. From this:

```
1  <template>
2    <v-container>
3      <v-layout
4        text-xs-center
5        wrap
6      >
```

to this:

```
1  <template>
2    <v-container>
3      <v-layout
4        text-center
5        wrap
6      >
```

14. Now I get this:



Miscellaneous Notes:

- https://www.youtube.com/watch?v=2uZYKcKHgU0 is a series of Vuetify videos that were super helpful to me. All free, although you can sign up for a buck a month to help the author out.
- There are gobs of postings online about how to get css-loader and sass-loader and such to work properly. Even the Vuetify docs lead me down a bunch of rabbit holes. In the end, I found that it was better to stay very simple and just let Vue and Vuetify handle things…so there is no special importing of loaders and other tweaks.
- https://mkdev.me/en/posts/rails-5-vue-js-how-to-stop-worrying-and-love-the-frontend is a good post for how to get your Rails app up and running with Vue.
- My code is available at https://github.com/davidlbean/rails_with_vuetify