Vue.js + Webpack General Notes

SETUP:
https://cli.vuejs.org/guide/creating-a-project.html#vue-create

Using the Vue CLI (v3), install the CLI:

```
npm install -g @vue/cli
```

Create a new project using the CLI (for new folder):

```
vue create hello-world
```

Create a new project using the CLI (in current folder):

```
vue create .
```

Create a new project using the Vue UI:

```
vue ui
```

~~**SETUP:**~~
~~https://github.com/vuejs-templates/webpack~~
~~https://github.com/vuejs-templates/pwa~~

~~Using generic "webpack" template:~~

```
~~$ npm install -g @vue/cli~~
~~$ vue init webpack my-project~~
~~$ cd my-project~~
~~$ npm install~~
~~$ npm run dev~~
```

~~Using "PWA" template (change above "webpack" line to this):~~

```
~~$ vue init pwa my-project~~
```

~~**IMPORTANT:** Fix for hot reload if not working with PWA template:~~
~~https://github.com/vuejs-templates/pwa/issues/136~~

~~In /build/dev-server.js, find and **comment out** the following lines:~~

```
// force page reload when html-webpack-plugin template changes
compiler.plugin('compilation', function (compilation) {
  compilation.plugin('html-webpack-plugin-after-emit', function (data, cb)
{
    hotMiddleware.publish({ action: 'reload' })
    cb()
  })
})
```

BOOTSTRAP-VUE:
https://bootstrap-vue.js.org/

```
# With NPM:
npm i bootstrap-vue

# With Yarn:
yarn add bootstrap-vue
```

Then, register BootstrapVue plugin in your app entry point (main.js):

```
import Vue from 'vue'
import BootstrapVue from 'bootstrap-vue'
import 'bootstrap/dist/css/bootstrap.css'
import 'bootstrap-vue/dist/bootstrap-vue.css'

Vue.use(BootstrapVue);
```

Vue v3 import:

```
import Vue from 'vue'
import { BootstrapVue, BootstrapVueIcons } from 'bootstrap-vue'

import 'bootstrap/dist/css/bootstrap.css'
import 'bootstrap-vue/dist/bootstrap-vue.css'

Vue.use(BootstrapVue)
Vue.use(BootstrapVueIcons)
```

Vue Fontawesome:
https://github.com/FortAwesome/vue-fontawesome
https://fontawesome.com/how-to-use/on-the-web/using-with/vuejs

```
npm i --save @fortawesome/fontawesome-svg-core
npm i --save @fortawesome/free-solid-svg-icons
npm i --save @fortawesome/vue-fontawesome
npm i --save @fortawesome/free-brands-svg-icons
npm i --save @fortawesome/free-regular-svg-icons
```

In src/main.js:

```
import Vue from 'vue'
import App from './App'
import { library } from '@fortawesome/fontawesome-svg-core'
import { faUserSecret } from '@fortawesome/free-solid-svg-icons'
import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome'

library.add(faUserSecret)

Vue.component('font-awesome-icon', FontAwesomeIcon)

new Vue({
  el: '#app',
  components: { App },
  template: '<App/>'
})
```

In src/App.vue:

```
<template>
  <div id="app">
    <font-awesome-icon icon="user-secret" />
  </div>
</template>

<script>
export default {
  name: 'App'
```

```
}
</script>
```

Suggestion: create a fonts.js file in the root of /src and put all of the font imports there. Then add it to /src/main.js like this:

```
import './fonts.js'
```

In /src/fonts.js:

```
import Vue from 'vue'
import { library } from '@fortawesome/fontawesome-svg-core'
import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome'

import {
       faUserSecret
} from '@fortawesome/free-solid-svg-icons'

library.add(
       faUserSecret
)

Vue.component('font-awesome-icon', FontAwesomeIcon)
```

DYNAMIC CSS BACKGROUND IMAGES:

Use case: you want to apply an inline background-image to an element in a component template where the background image is a variable.

Component “Data”:

some_element_styles: {}

Either in component “data” or in some method:

some_element_styles.backgroundImage = 'url(\" + vm.first_place.lead_image + '\')';

In the template:

<div class="my_element" :style="some_element_styles">

AUTO LAUNCH BROWSER WINDOW FOR DEV:

- Open file: /config/index.js
- Change value for “autoOpenBrowser” from false to true under “dev”

ROUTER SETUP:

Some misc. Router modifications to consider:

- Adding “process.env.NODE_ENV === ‘production’” lets you do different things for dev vs. prod automatically
- “base” can be set/modified in the router for sites hosted in subfolders
- Using “mode: history” lets router navigation function correctly with “back button” behavior
- The “scrollBehavior()” function lets you change the scroll position when a route changes. In this case, we’re scrolling to the top when a route changes

```
import Vue from 'vue'
import Router from 'vue-router'
import HelloWorld from '@/components/HelloWorld'

Vue.use(Router)

let base = '/';
if(process.env.NODE_ENV === 'production') {
      base = '/myroot/';
}

export default new Router({
      mode: 'history',
      scrollBehavior() {
            return { x: 0, y: 0 };
      },
      base: base,
      routes: [
            {
                  path: '/',
                  name: 'HelloWorld',
                  component: HelloWorld
            }
      ]
})
```

SETTING UP SASS/SCSS SUPPORT:
https://vue-loader.vuejs.org/guide/pre-processors.html#sass

```
npm install -D sass-loader node-sass
```

In your webpack config (/build/webpack.base.conf.js):

```
module.exports = {
  module: {
    rules: [
      // ... other rules omitted

      // this will apply to both plain `.scss` files
      // AND `<style lang="scss">` blocks in `.vue` files
      {
        test: /\.scss$/,
        use: [
          'vue-style-loader',
          'css-loader',
          'sass-loader'
        ]
      }
    ]
  },
  // plugin omitted
}
```

ROUTER VIEW:
Small modification to the <router-view> element for outputting route components. Keeps data persistent.

```
<keep-alive>
    <router-view></router-view>
</keep-alive>
```

TRANSITIONS & ANIMATE.CSS:
https://github.com/daneden/animate.css
https://vuejs.org/v2/guide/transitions.html

```
npm install animate.css --save
```

In App.vue -> <style>

```
@import '../node_modules/animate.css/animate.css';
```

Sample usage in Vue.js

```
<transition
   name="custom-classes-transition"
   mode="out-in"
   enter-active-class="animated tada"
   leave-active-class="animated bounceOutRight"
 >
      <!-- SOME CODE HERE -->
</transition>
```

RESPONSIVE BREAKPOINT DETECTION:
https://github.com/AlexandreBonaventure/vue-mq

```
npm install vue-mq
```

In main.js entry point

```
import Vue from 'vue'
import VueMq from 'vue-mq'

Vue.use(VueMq, {
  breakpoints: { // default breakpoints - customize this
    sm: 450,
    md: 1250,
    lg: Infinity,
  }
  defaultBreakpoint: 'sm' // customize this for SSR
})
```

Customized example:

```
Vue.use(VueMq, {
      breakpoints: {
```

```
            mobile: 450,
            tablet: 900,
            laptop: 1250,
            desktop: Infinity
        }
})
```

Use `$mq` property

After installing the plugin every instance of Vue component is given access to a reactive $mq property. Its value will be a `String` which is the current breakpoint.

Eg: *(with default breakpoints)* `'sm'` => 0 > screenWidth < 450 `'md'` => 450 >= screenWidth < 1250 `'lg'` => screenWidth >= 1250

```
//Use it in a component
new Vue({
  template: `
    <h1>current: {{$mq}}</h1>
  `,
})
```

How to use vue-mq in CSS of component:

```
.my-title {
    &.mobile { font-size: 1em; }
    &.desktop { font-size: 3em; }
  }
```

STORE PATTERN:
Create “src -> store -> store.js”

```
import Vue from 'vue'
import router from '@/router';
import http from './modules/http_services'
import nav from './modules/mod_navigation'
import storage from './modules/mod_storage'

export default {
```

```
      debug: true,
      state: {
            clientid: ''
      },

      get_current_clientid: function() {
            return this.state.clientid;
      },

      goto_path: function(path_name, param) {
            let push_path = path_name;

            // APPEND TRAILING SLASH
            if(push_path.slice(-1) != '/') {
                  push_path += '/';
            }

            // IF PARAM SET, APPEND IT
            if(param) {
                  push_path += param + '/';
            }

            // console.log('push_path: ', push_path);

            router.push({
                  path: push_path,
            })
    },

      nav,
      storage
}
```

Add to main.js entry point

```
import store from '@/store/store';

const root = new Vue({
  el: '#app',
  router,
  template: '<App/>',
```

```
  data: {
      store: store,
      state: {
            env: process.env.NODE_ENV
      }
  },
  components: {
      App
  }
})
```

Create “src -> store -> modules” folder. Create modules that can be attached to store.js, such as “nav” and “storage” in the example above.

Access store in a component. Can be used in any function, including computed, mounted, etc.

```
this.$root.store.nav.get_main_nav();
```

Sample module attached to store that uses “http_services”

```
import Vue from 'vue';
import http from './http_services'

const contests = {
      state: {
            master_contest_list: [],
            contest_sponsors: []
      },

  get_current_master_contest_list: function() {
    return this.state.master_contest_list;
  },

  get_batch_contest_voting_details: function(instance_contestid_list,
userid) {
    // console.log('instance_contestid_list: ', instance_contestid_list);
    if(instance_contestid_list instanceof Array) {
      return
this.fetch_batch_contest_voting_details(instance_contestid_list, userid);
    }
```

```
    else {
      return false;
    }
  },

  fetch_batch_contest_voting_details: function(instance_contestid_list,
userid) {
    let vm = this;
    let requested_set = 'batch_contest_voting_details';
    let options = {
      list: instance_contestid_list,
      userid: userid
    }

    return http.get_api_data(requested_set, options);
  }

}

export default contests
```

AXIOS WITH VUE.JS:
https://vuejs.org/v2/cookbook/using-axios-to-consume-apis.html
https://alligator.io/vuejs/rest-api-axios/

```
npm install axios --save
```

Usage

```
import axios from 'axios';

axios.get('http://jsonplaceholder.typicode.com/posts')
    .then(response => {
      // JSON responses are automatically parsed.
      this.posts = response.data
    })
    .catch(e => {
      this.errors.push(e)
```

```
    })
```

DYNAMIC/CUSTOM META DATA:

A plugin that allows you to specify custom meta data for pages/components for social and SEO purposes. Works best when combined with a pre-render solution.
https://github.com/declandewet/vue-meta

```
npm install vue-meta --save
```

In your router file:

```
import Vue from 'vue'
import Router from 'vue-router'
import Meta from 'vue-meta'

Vue.use(Router)
Vue.use(Meta)

export default new Router({
  ...
})
```

Sample usage in a component

```
<template>
  <div>
    <h1>{{{ title }}}</h1>
  </div>
</template>

<script>
  export default {
    name: 'post',
    props: ['title'],
    data () {
      return {
        description: 'A blog post about some stuff'
      }
    },
    metaInfo () {
      return {
        title: this.title,
```

```
        meta: [
          { vmid: 'description', name: 'description', content: this.description }
        ]
      }
    }
  }
</script>
```

USING STATIC ASSET IMAGES/FILES:

Use case example: you want to use an image in your /src/assets/ folder as a background-image for a div in a component.

In component script:

```
<script>

export default {
  name: 'info_large',
  props: [

  ],
  data () {
    return {
      banner_photo: require('@/assets/pexels-photo-532045_660w.png')
    }
  },
  computed: {

  },
  methods: {

  },
  components: {

  }
}
</script>
```

In component template:

```
<div class="banner_photo" :style="{backgroundImage: 'url(' + banner_photo + ')' }"></div>
```

DATEPICKER:

Simple but highly functional date picker plugin, similar to jQuery UI but not attached to jQuery at all.
https://github.com/charliekassel/vuejs-datepicker

```
npm install vuejs-datepicker --save
```

In component you'd like to use the date picker:

```
import Datepicker from 'vuejs-datepicker';

export default {
  // ...
  components: {
    Datepicker
  }
  // ...
}
```

Usage:

```
<datepicker></datepicker>
<datepicker :value="state.date" name="uniquename"></datepicker>
<datepicker v-model="state.date" name="uniquename"></datepicker>
<datepicker @selected="doSomethingInParentComponentFunction"
@opened="datepickerOpenedFunction" @closed="datepickerClosedFunction">
```

Custom Date Format:

```
<script>
  methods: {
    customFormatter(date) {
      return moment(date).format('MMMM Do YYYY, h:mm:ss a');
    }
  }
</script>
<datepicker :format="customFormatter"></datepicker>
```

MOMENT.JS

Easy way to install moment.js in a vue.js project globally

https://momentjs.com/

```
npm install moment --save
```

In your /src/main.js file:

```
var moment = require('moment')
const root = new Vue({
  el: '#app',
  router,
  template: '<App/>',
  data: {
      store: store,
      moment: moment,
      state: {
            env: process.env.NODE_ENV
      }
  },
  components: {
      App
  }
})
```

Usage in a component:

```
this.$root.moment().subtract(1, 'days').format('YYYY-MM-DD');
```

Simpler usage in a component:

```
import moment from 'moment'
```

You can then use moment() as you normally would in your JS.

NUMERAL.JS FILTERS:
https://github.com/Kocal/vue-numerals

Adds filter options for numeral.js in vue.js

```
npm install --save vue-numerals
```

Then add to /src/main.js:

```
import VueNumerals from 'vue-numerals';

Vue.use(VueNumerals, {
  locale: 'en'
});
```

Usage example within <template> of a component:

```
<template>
  <div>
    <!-- With french locale, it will display: `12 345` -->
    <p>{{ count | numeralFormat }}</p>

    <!-- With french locale, it will display: `12 345 €` -->
    <p>{{ count | numeralFormat('0,0[.]00 $') }}</p>
  </div>
</template>

<script>
export default {
  data: () => ({ count: 12345 }),
}
</script>
```

LOCALFORAGE FOR VUE (VUE-FORAGE):
https://github.com/MianSaleem/vue-forage#readme

```
npm i vue-forage
```

In /src/main.js:

```
import vf from 'vue-forage';

// Tell Vue.js to use vue-forage
Vue.use(vf);
```

In components:

```
// configure your local storage
this.$vf.config({
    name: 'vue-forage'
});

// change driver
// this.$vf.useWebSQLDriver();
// this.$vf.useIndexedDBDriver();
this.$vf.useLocalStorageDriver();
// this.$vf.setDriver(YOURDRIVER);

// SET ITEM
// this.$vf.setItem('key', 'value');
this.$vf.setItem('app', 'Vue Forage');

// or
this.$vf.setItem('app', { app: 'Vue Forage', version: '1.0.0', author: { name:
'John Doe', email: 'john.doe@mail.com' }});

// GET ITEM
// this.$vf.getItem('key'');

this.$vf.getItem('app');

// REMOVE ITEM
// this.$vf.removeItem('key'');

this.$vf.removeItem('app');

this.$vf.clear(); // delete everything

// Forage will stringify/parse the json object automatically.
```

OR use it as a module for store pattern. In this example, you would NOT add vf to your /src/main.js file.

Create /store/modules/mod_storage.js. Insert:

```
import Vue from 'vue';
import vf from 'vue-forage';

Vue.use(vf);

$vf.config({
  name: 'metrics'
})

const storage = {
      general_data_expiration_minutes: 5,
      state: {
            general_data_last_modified: 0
      },

      get_current_general_last_modified: function() {
            return this.state.general_data_last_modified;
      },

      retrieve_general_last_modified: function() {
            let vm = this;

            vm.get_item('general_data_last_modified')
              .then(function(vf_response) {
                // console.log('vf_response: ', vf_response);
                if(vf_response) {
                  // SAVE TO LOCAL STATE
                  Vue.set(vm.state, 'general_data_last_modified',
vf_response.general_data_last_modified);
                }
                else {
                  // UPDATE IN STORAGE
                  vm.update_general_last_modified();
                }
              })
      },

      update_general_last_modified: function() {
            let vm = this;
            let now = new Date();

            vm.set_item('general_data_last_modified', now.getTime())
```

```
              .then(function() {
                Vue.set(vm.state, 'general_data_last_modified',
now.getTime());
              })

      },

      get_item: function(item_key) {
            return $vf.getItem(item_key);
      },

      set_item: function(item_key, item_value) {
            // CHECK IF ITEM_VALUE IS AN OBJECT
            if(typeof item_value == 'object') {
              // APPEND "LAST_MODIFIED" TIMESTAMP
              var modified = new Date();
              modified = modified.getTime();
              item_value.last_modified = modified;
            }

            return $vf.setItem(item_key, item_value);
      },

      remove_item: function(item_key) {
            return $vf.removeItem(item_key);
      }

}

export default storage
```

UNDERSCORE:
https://github.com/HKskn/vue-underscore#readme

Installs a vue plugin for underscore.js

```
npm install vue-underscore --save
```

In /src/main.js/:

```
import Vue from 'vue';
import underscore from 'vue-underscore';
import App from './App';

Vue.use(underscore);

new Vue({
  ...App
}).$mount('#app');
```

Example usage in a component:

```
import {_} from 'vue-underscore';

let testArr = [{id: 1}, {id:2}];
let foundInfo = _.findWhere(testArr, {id:1});
```

FIX BUILD ERRORS RELATED TO ES6 TRANSPILING

If you get errors on build related to certain modules, it might be an issue with es6 code not getting transpiled. You need to tell Babel about the module(s) to that it can transpile.
https://medium.com/thetiltblog/how-to-transpile-es6-vue-plugins-with-webpack-c5d4286a7737

In /build/webpack.base.conf.js, add the module(s) to the “include” array:

```
{
        test: /\.js$/,
        loader: 'babel-loader',
        include: [resolve('src'), resolve('test'),
resolve('node_modules/vue-forage')]
},
```

In the example above, “vue-forage” was added to make Babel transpile it correctly.