Creating a product configurator with Vue #2

Part 2 - The state store

Pim Hooghiemstra
May 31, 2020

In this post I'll describe the setup of the Vuex store for the product configurator that was introduced in part 1. The product configurator is part of a fictive webshop where customers may configure and order personal postcards.

Since usability becomes more and more important these days, this blog series covers a product configurator from concept to implementation.

Photo by Melissa Rosas on Unsplash

Vuex

I'll use Vuex (with modules) to handle state. Although a product configurator seems to be a relatively small part of a webshop, using Vuex will result in a better maintainable codebase when the app grows and becomes more complex.

Moreover, Vuex is rather opinionated about how you should implement certain features which always goes over a bit more than a single blog post. Implementing this kind of features:

The code for this post can be found on Github, release v0.1.

A quick note on the code samples: After creating this release and writing this post, I continued with the actual implementation of the product configurator, which will be described in the next post. I slightly changed the code of the application to get it all working. I have not changed the code in this release accordingly, so code samples in this blog post may be outdated. Nevertheless, they do serve my purpose to explain the general Vuex setup.

If you follow along, simply flush the Github repository and run npm install and npm serve.

Below is the sketch of the product configurator we're building (the concept is described in detail in the previous post).

The following actions may take place during the configuration of the postcards:

- add a postcard
- remove a postcard
- update the total price
- reset a configuration for a given postcard
- add to cart and proceed to the checkout

These will become the actions for the Vuex store. I'll discuss the setup of this part of the store in a minute.

Vuex modules

First a word on Vuex modules. For a small project like this it might not be necessary, but your future self will be thankful if the project grows 😉

Moreover, setting up the modules is not very hard. But what modules do we have in this project?

Clearly there is a configurator module consisting of all aspects related to postcards. Let's imagine we also have accounts in the webshop. An account, such login, logout and so on is a good candidate for another module. In addition, the cart could be a module. However, the latter is out of scope for the current post, so let's assume we only need modules for the configurator and account. The folder structure for the configurator are described above, for the account module I assume to the following set of actions:

Vuex modules implemented

Implementing the store with modules is quite easy. However, instead of the store.js file that comes with Vuex by default, we need a /store folder holding an index.js file and a modules folder that holds 'account.js' or 'configurator.js'. Each module contains its own state, actions, mutations, getters and (possibly) submodules if required in the project. The store/index.js file looks like this:

```
// store/index.js
import Vue from 'vue'
import Vuex from 'vuex'
import account from './modules/account'
import configurator from './modules/configurator'

Vue.use(Vuex)

export default new Vuex.Store({
  modules: {
    configurator,
    account
  }
})
```

store/index.js hosted with ❤ by GitHub view raw

The store itself is imported in main.js. This happens by default when adding Vuex to the project during the setup with @vue/cli.

```
// main.js
import Vue from 'vue'
import configurator from './modules/configurator'
import App from './components/App'

Vue.use(configurator)

new Vue({
  render: h => h(App),
  store: store,
}).$mount('#app')
```

main.js hosted with ❤ by GitHub view raw

Configurator module

The heart of the configurator module contains a products array and a price. We initialize the price to € 1.50. A single product is initialized in the products array. It's an object with a unique id and a config object holding all the properties a postcard may have:

```
// store/modules/configurator.js
const state = {
  products: [
    {
      id: uniqueId(),
      config: {
        headline: 'Merry Christmas',
        mainText: 'Dear Santa,\n\n\n
```