

LAB: Spinner, Cart & Store Subscriptions

The loading spinner one is quite simple. We already have a state item that's set to true anytime an async operation is carried out. This state item is called `showLoader`.

Cart items count is also simple -- just the length of the array as shown in the code below.

Furthermore, we need to create global subscriptions to our store mutations. That's the best way to listen to notifications mutations like `ADD_PRODUCT_SUCCESS`. When these notifications come in, we need to let the user know by showing a toaster.

```
// ./src/App
<template>
  <div id="app">
    <nav>
      <div class="container">
        <ul class="nav__left">
          <li><router-link to="/"><i class="fa fa-home"></i> Home</router-link></li>
          <li><router-link to="/admin"><i class="fa fa-user"></i> Admin</router-link></li>
        </ul>
        <ul class="nav__right">
          <li><router-link to="/cart"><i class="fa fa-shopping-cart"></i> Cart ({{cartItemsCount}})
        </ul>
      </div>
    </nav>
    <router-view></router-view>
    <div class="overlay" v-show="showLoader">
      <div class="loading-spinner">
        <div class="dot dotOne"></div>
        <div class="dot dotTwo"></div>
        <div class="dot dotThree"></div>
      </div>
    </div>
  </div>
</template>

<script>
import toastr from 'toastr'

import {
  ADD_PRODUCT_SUCCESS,
  UPDATE_PRODUCT_SUCCESS,
  REMOVE_PRODUCT_SUCCESS
} from './store/mutation-types'
export default {
  name: 'app',
  data () {
    return {
      cartItems: this.$store.state.cart
    }
  },
  created () {
    // Subscriptions for mutation
    this.$store.subscribe((mutation) => {
      if (mutation.payload) {
        switch (mutation.type) {
          case ADD_PRODUCT_SUCCESS:
            toastr.success('Product created.', 'Success!')
            break
          case UPDATE_PRODUCT_SUCCESS:
            toastr.success('Product updated.', 'Success!')
            break
          case REMOVE_PRODUCT_SUCCESS:
            toastr.warning('Product deleted.', 'Deleted!')
            break
        }
      }
    })
  },
  computed: {
    cartItemsCount () {
      // Cart count
      return this.cartItems.length
    },
    showLoader () {
      // Loading spinner
      return this.$store.state.showLoader
    }
  }
}
```

Getting Started		
1	Introduction	Free
2	Tools	Free
3	Our Task	Free
4	Hello World	Free
Vue Basics		
5	Vue CLI	Free
6	Components and the Vue Instance	Free
7	Template Syntax	Free
8	Conditional & List Rendering	Free
9	Styles & Classes	Free
Routing		
10	Single Page Apps	Free
11	Creating Routes	Free
12	Route Outlets & Links	Free
13	Nested & Dynamic Routes	Free
Forms		
14	Two Way Binding (Reactivity)	Free
15	Form Validation	Free
16	Handling Form Submissions	Free
API Backend		
17	Prelude	Free
18	Setup	Free
19	Provisioning a MongoDB Database	Free
20	Enabling CORS	Free
21	Schemas & Models	Free
22	Routes and Controllers	Free
23	Testing with POSTman	Free
Vuex		
24	The Current Problem	Free
25	State Management	Free
26	Getters	Free
27	Mutations	Free
28	Actions	Free
Using Store In Components		
29	State and Actions on Components	Free
30	LAB: Product List	Free
31	LAB: Product Details	Free
32	LAB: Admin Features	Free
33	LAB: Spinner, Cart & Store Subscriptions	Free