NEXT: PRELUDE

Our project has a product form on the admin section (/admin/new). Right now it just has static text. This is a good time to build this product form, use it to collect data from the user, and assemble this data.

You could go right into the New component in the admin directory and start writing this form, however, let's slow down and give this some thought. We have two forms that might look the same, New and Edit component forms. Therefore, to uphold DRY principles (don't repeat yourself), it would be best it we make a ProductForm component that can be shared by New and Edit components.

Move right into src/components/products and create a ProductForm.vue file:

```
HTML
<!-- ./src/components/products/ProductForm.vue -->
<template>
    <form @submit.prevent="saveProduct">
         <div class="col-lg-5 col-md-5 col-sm-12 col-xs-12">
              <div class="form-group">
                  <label>Name</label>
                   <input type="text" placeholder="Name" v-model="model.name" name="name" class="form-control</pre>
              </div>
              <div class="form-group">
                  <label>Price</label>
                   <input type="number" class="form-control" placeholder="Price" v-model="model.price" name='</pre>
              </div>
              <div class="form-group">
                  <label>Manufacturer</label>
                   <select type="text" class="form-control" v-model="model.manufacturer" name="manufacturer"</pre>
                       <template v-for="manufacturer in manufacturers">
                            <option :value="manufacturer._id" :selected="manufacturer._id == (model.manufacturer &</pre>
                       </template>
                  </select>
              </div>
         </div>
         <div class="col-lg-4 col-md-4 col-sm-12 col-xs-12">
              <div class="form-group">
                  <label>Image</label>
                  <input type="text" lass="form-control" placeholder="Image" v-model="model.image" name="image" name="imag
              </div>
              <div class="form-group">
                  <label>Description</label>
                  <textarea class="form-control" placeholder="Description" rows="5" v-model="model.descripti
              </div>
              <div class="form-group new-button">
                   <button class="button">
                       <i class="fa fa-pencil"></i></i>
                       <!-- Conditional rendering for input text -->
                       <span v-if="isEditing">Update Product</span>
                       <span v-else>Add Product</span>
                  </button>
              </div>
         </div>
     </form>
</template>
<script>
     export default {
         props: ['model', 'isEditing'],
    methods: {
        saveProduct () {
              this.$emit('save-product', this.model)
```

Lots of HTML code here but everything is just basic form stuff. We have a few controls that are bound to a model. The model is an object that is received via the parent component (New or Edit). This is done by passing the values as attributes to the ProductForm component as shown in the code below, and explicitly declaring them using the props property. The isEditing flag is used to toggle what is displayed on the submit button.

</script>

We use the submit event to send the form data to JavaScript for processing. The prevent modifier calls the preventDefault() method so as to disable the submit event from reloading the page. The event is handled with saveProduct which emits a save-product component event for the parent component to listen to and act accordingly

parent component for now.

The manufacturers' list is an array of manufacturers which we can create in the

```
HTML
<template>
  cproduct-form @save-product="addProduct" :model="model" :manufacturers="manufacturers">
 </product-form>
</template>
<script>
  import ProductFrom from '@/components/product/ProductForm.vue'
  export default {
   data () {
      return {
       model: {},
        manufacturers: [
            _id: 'sam',
            name: 'Samsung',
            _id: 'apple',
            name: 'Apple',
         },
        1,
   },
   methods: {
      addProduct (model) {
       console.log('model', model)
   },
   components: {
      'product-form': ProductFrom
</script>
```

We are now passing model and manufacturers to ProductForm via it's New parent.

isEditing will be false by default so Add Product text will be shown on the button.

We are also listening to the save-product event which the child component is emitting. The event is then handled with the addProduct method. For now, just

some console.log stuff going on.

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

31 LAB: Product Details

32 LAB: Admin Features

33 LAB: Spinner, Cart & Strore Subscript