LAB: Product List

NEXT LAB: PRODUCT DETAILS

Products List, Item and Button

The product list is composed with product item. The list is the parent of item and iterates over an array of products from the state. While, the item receives single products from the list and displays them properly.

This is what the product item looks like:

JS

```
// ./src/components/product/ProductItem
<template>
  <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12">
    <div class="product">
      <router-link :to="'/details/'+product._id" class="product-link">
        <div class="product__image">
          <img
            :class="img-responsive" :src="product.image" alt="">
        </div>
        <div class="product__description">
          <div class="product__info">
           <small>{{product.manufacturer.name}}</small>
           <h4>{{product.name}}</h4>
          </div>
          <div class="product__price-cart">
            ${{product.price}}
          </div>
        </div>
      </router-link>
      <div class="product__action">
        <product-button :product="product" ></product-button>
      </div>
    </div>
  </div>
</template>

<script>
  import ProductButton from './ProductButton.vue'
  export default {
    name: 'product-item',
    props: ['product'],
    components: {
      'product-button': ProductButton
    }
  }
</script>
```

The component receives `product` from prop which is expected to be passed down via the parent product list component. We will create the `ProductList` component soon but before that, note that the `ProductItem` component is housing a `ProductButton` UI Component. Let's create that:

HTML

```
// ./src/components/product/ProductButton
<template>
  <div>
    <button v-if="!isAdding" class="button" @click="addToCart"><i class="fa fa-cart-plus"></i> Add
    <button v-else class="button button-danger" @click="removeFromCart(product._id)"><i class="fa
  </div>
</template>

<script>
  import {
    ADD_TO_CART,
    REMOVE_FROM_CART
  } from '../../store/mutation-types'
  export default {
    props: ['product'],
    data () {
      return {
        cart: this.$store.state.cart
      }
    },
    computed: {
      isAdding () {
        return this.cart.indexOf(this.product) < 0
      }
    },
    methods: {
      addToCart () {
        this.$store.commit(ADD_TO_CART, this.product)
      },
      removeFromCart (id) {
        this.$store.commit(REMOVE_FROM_CART, id)
      }
    }
  }
</script>
```

Same product is still passed down to the button component. If an item exists in the cart, it cannot be added again and the remove from cart button is shown. This is controlled with the `isAdding` computed method.

`addToCart` and `removeFromCart` are used to add and remove items from the cart respectively. You can see how this is not done right in the component, rather, we just commit to the store to help us do that.

Let's now have a look at the origin of the `product` prop on the above components -- `ProductList`:

JS

```
// ./src/components/product/ProductList
<template>
  <div>
    <div class="products">
      <div class="container">
        <template v-for="product in products">
          <product-item :product="product" ></product-item>
        </template>
      </div>
    </div>
  </div>
</template>

<script>
  import ProductItem from './ProductItem.vue'
  export default {
    name: 'product-list',
    created () {
      if (this.products.length === 0) {
        this.$store.dispatch('allProducts')
      }
    },
    computed: {
      products () {
        return this.$store.getters.allProducts
      }
    },
    components: {
      'product-item': ProductItem
    }
  }
</script>
```

The component's `created` lifecycle method is used to to make a call to the API by dispatching the `allProducts` action. Remember, when discussing actions, that the `allProducts` action sends a request to the server to retrieve a list of products.

The `products` computed method then use the `allProducts` getters to retrieve what's available.

The template then iterates over the list of `products` and renders them using the `ProductItem` component we saw earlier.

For `ProductList` to work, we need to import it to the Home component:

JS

```
// ./src/pages/Home
<template>
  <div>
    <div class="title">
      <h1><i class="fa fa-braille"></i> In Stock</h1>
    </div>
    <product-list></product-list>
  </div>
</template>

<script>
  import ProductList from '../components/product/ProductList.vue'
  export default {
    name: 'home',
    components: {
      'product-list': ProductList
    }
  }
</script>
```