**Lab 3b tips**

Here are some tips from Dr. Zappala, Dr. Clement, and the TA's:

1) First, an overview of all the different files in a Vue CLI project and how they relate to this app:

* main.js -- global data structures (products, cart)
* App.vue -- code that goes on all your pages, like a menu or a footer
* router/index.js -- configure which view to use for each URL
* views/Home.vue -- view/component for the home page. Shows a search box and retrieves the product list from the global data structure. Uses the ProductList component to display a list of all products.
* views/Browse.vue -- view/component for the browse page. Shows a menu and keeps track of which menu item the user clicks on. Based on the menu item, calculates a subset of products that are from that country. Uses the ProductList component to display a list of the products for the selected country.
* components/ProductList.vue -- component that takes a list of products (in a props list) and displays them on the page. The component that uses the ProductList as a child component has to import it and pass it a list of products using a prop (from its props list).

2) Now, let's walk through the functionality you're adding for the cart:

*Modifying the "Add to Cart" button so that it adds the product to the cart. There is already a cart property in the global data object that you can use to store an array of products. Don't worry about quantities. We'll just assume that users need to add the same product more than once in order to buy multiples.*

Since the "Add to Cart" button is in ProductList, you want to work there. Use a Vue "@click" directive to add an event handler to the "Add to Cart" button. This event handler is a function, listed in the `methods` section for ProductList, that takes a product as a parameter and adds it to the global cart array.

*Modifying the menu so that it shows the number of items in the cart. You may want to use a computed property in App.vue to do this.*

Since the menu is in App.vue, you want to work there. You will need to add a "script" section to this component since it doesn't have one already. Use a computed property to get the number of items in the cart. You can calculate this using the length of the array. Once you have a computed property, e.g. "numberOfItems()", then you can modify the `template` section in this component to use this property like you would any other property.

{{ numberOfItems }}

*Adding a Cart view that is viewed when the user clicks on the Cart menu item. This view should show all the products in the cart. It should include a Remove button to remove items from the cart. It should also show a message when the cart is empty. Don't use the ProductList here since you won't have an Add to Cart button. And you should make the shopping cart view look different from the product listing so that the user is not confused.*

Take a look at how the Browse view is configured. There is a menu item in App.vue. When this is clicked (router-link), it goes to router/index.js to find the matching path for "/browse" and is configured to use the Browse.vue component to handle that path. You need to do something similar for a Cart view. Here are the general steps:

1. Copy either Home.vue or Browse.vue  to make Cart.vue. Modify it for the cart view including using this.$root.$data.cart as the computed value of the cart and the template will use a new component instead of ProductList.

2. Add the path for Cart.Vue in the router/index.js file using the others as examples.

3. Copy the ProductList component to create a CartList component.  Be sure to change the loop to go through the cart instead of the products list.

4. Change the addItem button and function to a remove item button and function.  This can be done by using indexOf() on the item passed in and splice() to remove it from the cart list.

5. Add a v-if v-else to display a message when the cart is empty.

6. Change the styling for the CartList component to make it look different from the product listing.

3) Installing on Digital Ocean:

Running "npm run build" basically creates a production folder for your website in the dist folder. This is what is moved to the var/www/motherearth.domainname.com.  Nothing on the home directory will show on your websites but anything inside the var/www/ will.  Having a production version allows the code to be more organized but any changes made to the original files will need to be reapplied by running build and copying over the folder again.