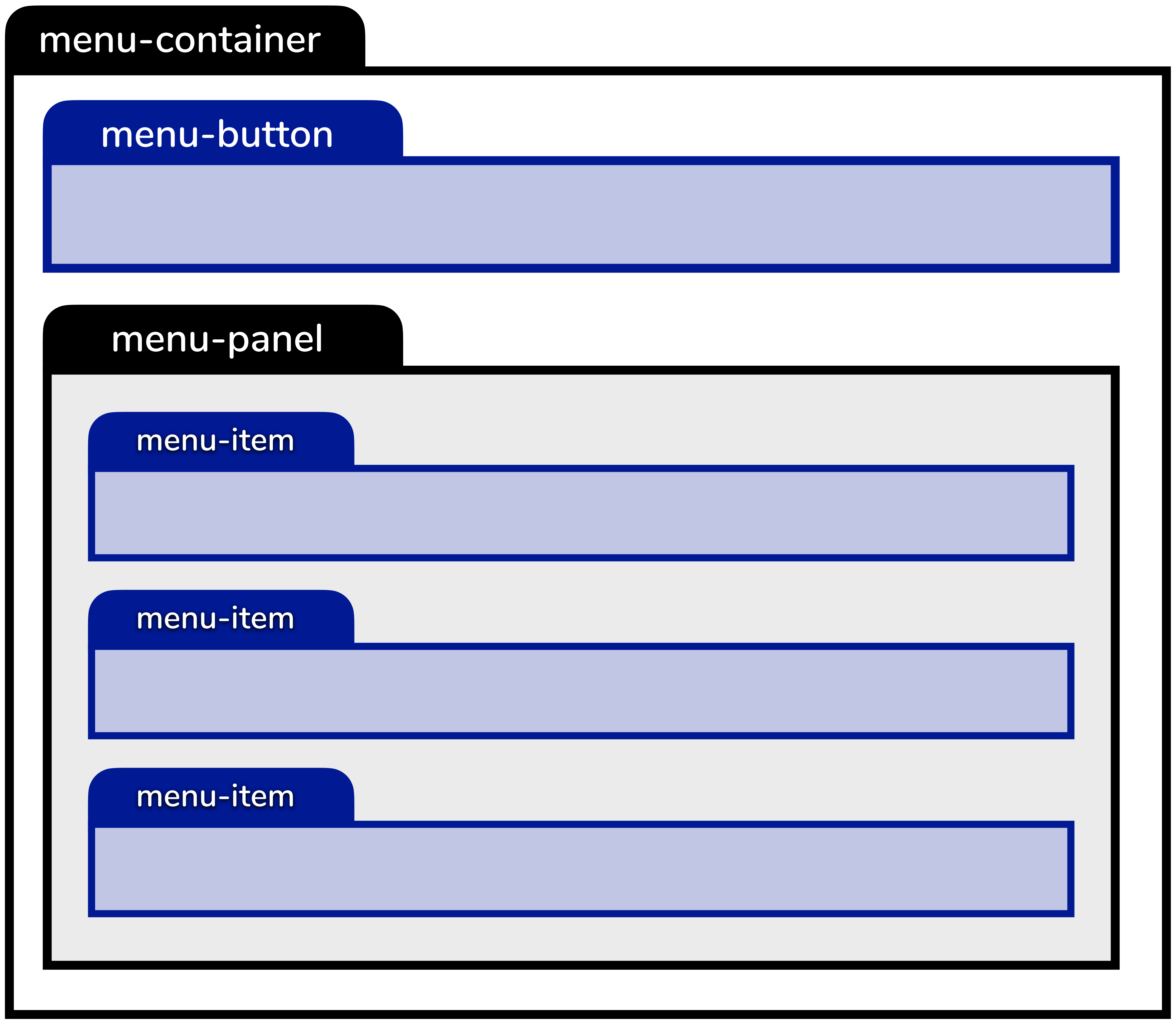
My Bookstore Hamburger Menu

Hamburger Menu with Submenu Dropdown using HTML, CSS, and Vue.js

Dr. K, Summer 2021

Menu Structure



```
<template>
  <div class="main-menu-container">
    <button class="button icon-button main-menu-button">
      <i class="fas fa-bars"></i>
    </button>
    <ul class="main-menu-panel">
      <router-link to="../category/classics" tag="li" class="main-menu-item">
        Classics
      </router-link>
      <router-link to="../category/fantasy" tag="li" class="main-menu-item">
        Fantasy
      </router-link>
      <router-link to="../category/mystery" tag="li" class="main-menu-item">
        Mystery
      </router-link>
      <router-link to="../category/romance" tag="li" class="main-menu-item">
        Romance
      </router-link>
    </ul>
  </div>
</template>
```

Hover Dropdown

```
.main-menu-panel {
  display: none;
  position: absolute;
  right: 0;
  background-color: #4e4e4e;
  color: white;
  cursor: pointer;
}

.main-menu-container:hover .main-menu-panel {
  display: block;
}
```

By default, don't display the panel

The right edge of the panel should be aligned with the right edge of the container

When the mouse is anywhere over the container, display the panel

Box Shadow

```
.main-menu-panel {
  display: none;
  position: absolute;
  right: 0;
  background-color: #4e4e4e;
  color: white;
  cursor: pointer;
  box-shadow: 0 0 10px 0 #0008;
}
```

x and y offsets

blur and spread radius

rgba color:
#abcd is short for #aabbccdd

Data and Methods

```
<script>
export default {
  name: "HeaderDropdownMenu",
  data: function () {
    return {
      mainMenuOpen: false,
    };
  },
  methods: {
    toggleMainMenu() {
      this.mainMenuOpen = !this.mainMenuOpen;
    },
  },
};
</script>
```

Boolean variable with default value of false

Toggles the Boolean; "this" is required

@click and .main-menu-open

```
<template>
  <div class="main-menu-container">
    <button class="button icon-button main-menu-button"
    @click="toggleMainMenu">
      <i class="fas fa-bars"></i>
    </button>
    <ul class="main-menu-panel" :class="{ 'main-menu-open': mainMenuOpen }">
      ...
```

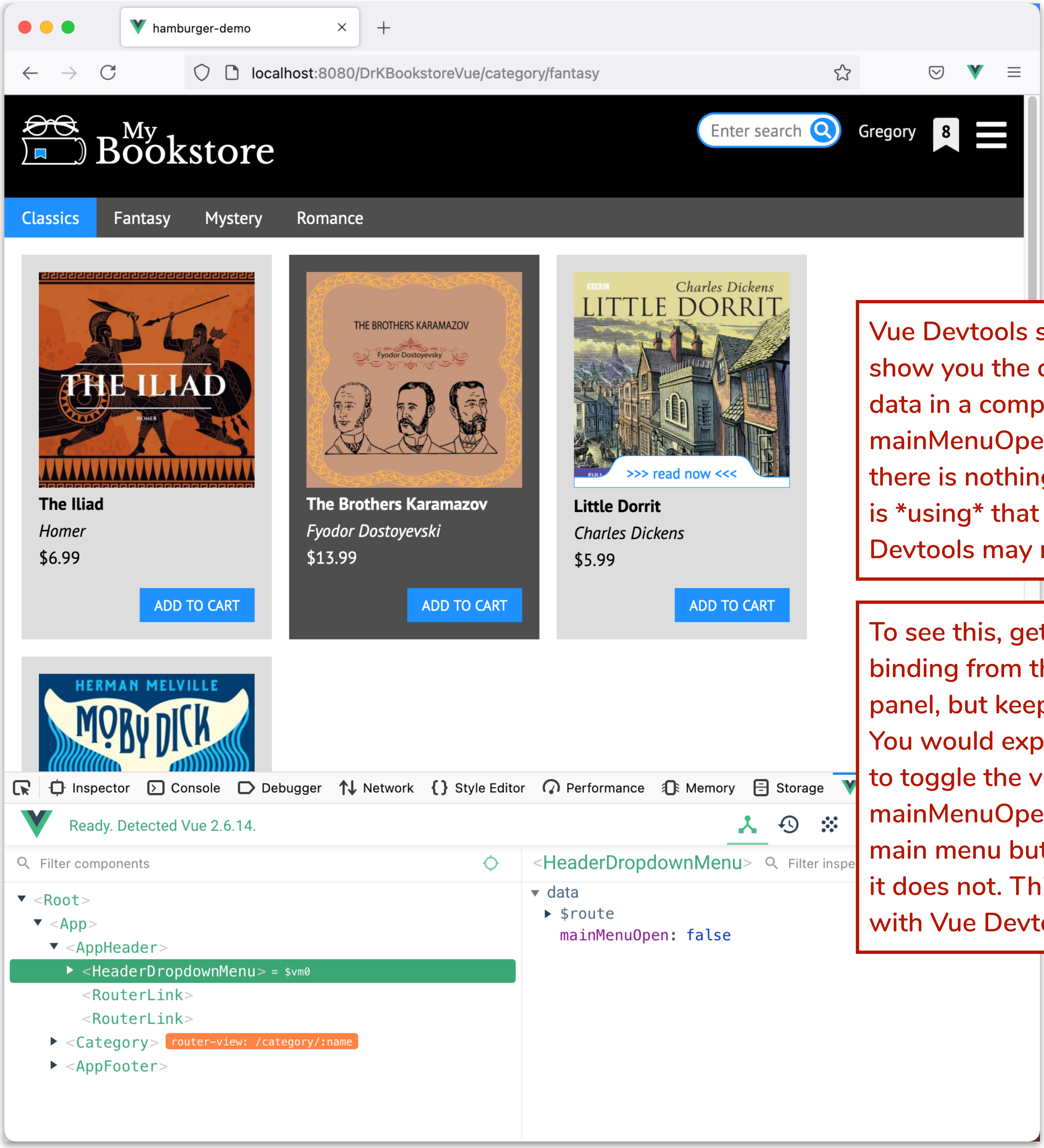
If data mainMenuOpen is true, include class main-menu-open

```
<style scoped>
.main-menu-container { ...

.main-menu-panel { ...

.main-menu-open {
  display: block;
}
```

If an element includes class main-menu-open, display it



Ready. Detected Vue 2.6.14.

```
▼ <Root>
  ▼ <App>
    ▼ <AppHeader>
      ▶ <HeaderDropdownMenu> = $vm0
        <RouterLink>
        <RouterLink>
    ▶ <Category> router-view: /category/:name
    ▶ <AppFooter>
```

```
<HeaderDropdownMenu>
▼ data
  ▶ $route
    mainMenuOpen: false
```

Vue Devtools should always show you the correct value for data in a component, like mainMenuOpen. However, if there is nothing on the page that is *using* that data, Vue Devtools may not work correctly.

To see this, get rid of the class binding from the main menu panel, but keep the click handler. You would expect Vue Devtools to toggle the value of mainMenuOpen whenever the main menu button is clicked, but it does not. This is a known issue with Vue Devtools

Category Submenu

```
<div class="sub-menu-container">
  <button class="button icon-button sub-menu-button">
    <i class="fas fa-bars"></i>
  </button>
  <ul class="sub-menu-panel">
    <router-link to="../category/classics" tag="li" class="sub-menu-item">
      Classics
    </router-link>
    <router-link to="../category/fantasy" tag="li" class="sub-menu-item">
      Fantasy
    </router-link>
    <router-link to="../category/mystery" tag="li" class="sub-menu-item">
      Mystery
    </router-link>
    <router-link to="../category/romance" tag="li" class="sub-menu-item">
      Romance
    </router-link>
  </ul>
</div>
```

Modified Main Menu

- Classics => Categories
  - "to" stays the same (Classics category)
- Fantasy => Search
  - "to" goes to welcome page
- Mystery => Account
  - "to" goes to welcome page
- Romance => Cart (8)
  - "to" goes to welcome page

Category Submenu

```
  <ul class="main-menu-panel" :class="{ 'main-menu-open': mainMenuOpen }
    <li class="sub-menu-container">
      <router-link
        to="../category/classics"
        class="main-menu-item sub-menu-button"
      >
        Categories
      </router-link>
      <ul class="sub-menu-panel">
        ...
      </ul>
    </li>
    <router-link ... > Search </router-link>
    <router-link ... > Account </router-link>
    <router-link ... > Cart (8) </router-link>
  </ul>
```

The router-link cannot be the sub-menu container, because then all sub-menu items would link to the classics category

Now Categories is both a main-menu item and a sub-menu button

We loose tag="li", so this router-link element becomes an anchor tag <a> in the rendered HTML

Since style is scoped, we can ensure all link states in this component have color white

```
a,
a:active,
a:hover,
a:visited {
  color: white;
}
```

Submenu CSS

```
.sub-menu-panel {
  background-color: #333;
}

.sub-menu-item {
  display: inline-block;
  width: 100%;
  height: 100%;
  padding: 0.5em 3em;
  white-space: nowrap;
  text-decoration: none;
}

.sub-menu-item:hover {
  background-color: var(--primary-color-dark);
}
```

Background color is a little darker

top and bottom padding is tighter

hover background color is a little darker

Transitions

- Simplest type of animation
- A transition occurs when a CSS property on an element changes
- Transitions can only occur on properties that take real numbers
- The animated element exists before and after the transition

Main Menu Transition

```css
.main-menu-panel {
  position: absolute;
  right: -1rem;
  margin-top: 1rem;
  transform: translateX(100%);
  transition-duration: 500ms;
  transition-property: transform;
  transition-timing-function: ease-in-out;
  background-color: #4e4e4e;
  color: white;
  cursor: pointer;
}

.main-menu-panel.main-menu-open {
  transform: translateX(0);
  box-shadow: 0 0 10px 0 #0008;
}
```

JavaScript for Sub Menu Transition

```
<script>
export default {
  name: "HeaderDropdownMenu",
  data: function () {
    return {
      mainMenuOpen: false,
      subMenuOpen: false,
    };
  },
  methods: {
    toggleMainMenu() {
      this.mainMenuOpen = !this.mainMenuOpen;
      if (!this.mainMenuOpen) {
        this.subMenuOpen = false;
      }
    },
    toggleSubMenu() {
      this.subMenuOpen = !this.subMenuOpen;
    },
  },
};
</script>
```

HTML for Sub Menu Transition

```
<template>
  <div class="main-menu-container">
    <button class="button icon-button main-menu-button"
    @click="toggleMainMenu">
      <i class="fas fa-bars"></i>
    </button>
    <ul class="main-menu-panel" :class="{ 'main-menu-open': mainMenuOpen }">
      <li class="sub-menu-container">
        <div class="sub-menu-button" @mouseover="toggleSubMenu">
          <router-link to="../category/classics" class="main-menu-item">
            Categories
          </router-link>
        </div>
        <ul class="sub-menu-panel" :class="{ 'sub-menu-open': subMenuOpen }">
          ...
```

CSS for Sub Menu Transition

```css
.sub-menu-panel {
  max-height: 0;
  overflow-y: hidden;
  transition: max-height 400ms ease-in-out;
  background-color: #333;
}

.sub-menu-panel.sub-menu-open {
  max-height: 10em;
}
```