Implementing the Behavior of Dropdown Menu

This lesson is a step-by-step guide on how to implement a working dropdown menu in JavaScript.

WE'LL COVER THE FOLLOWING

Toggling submenu visibility

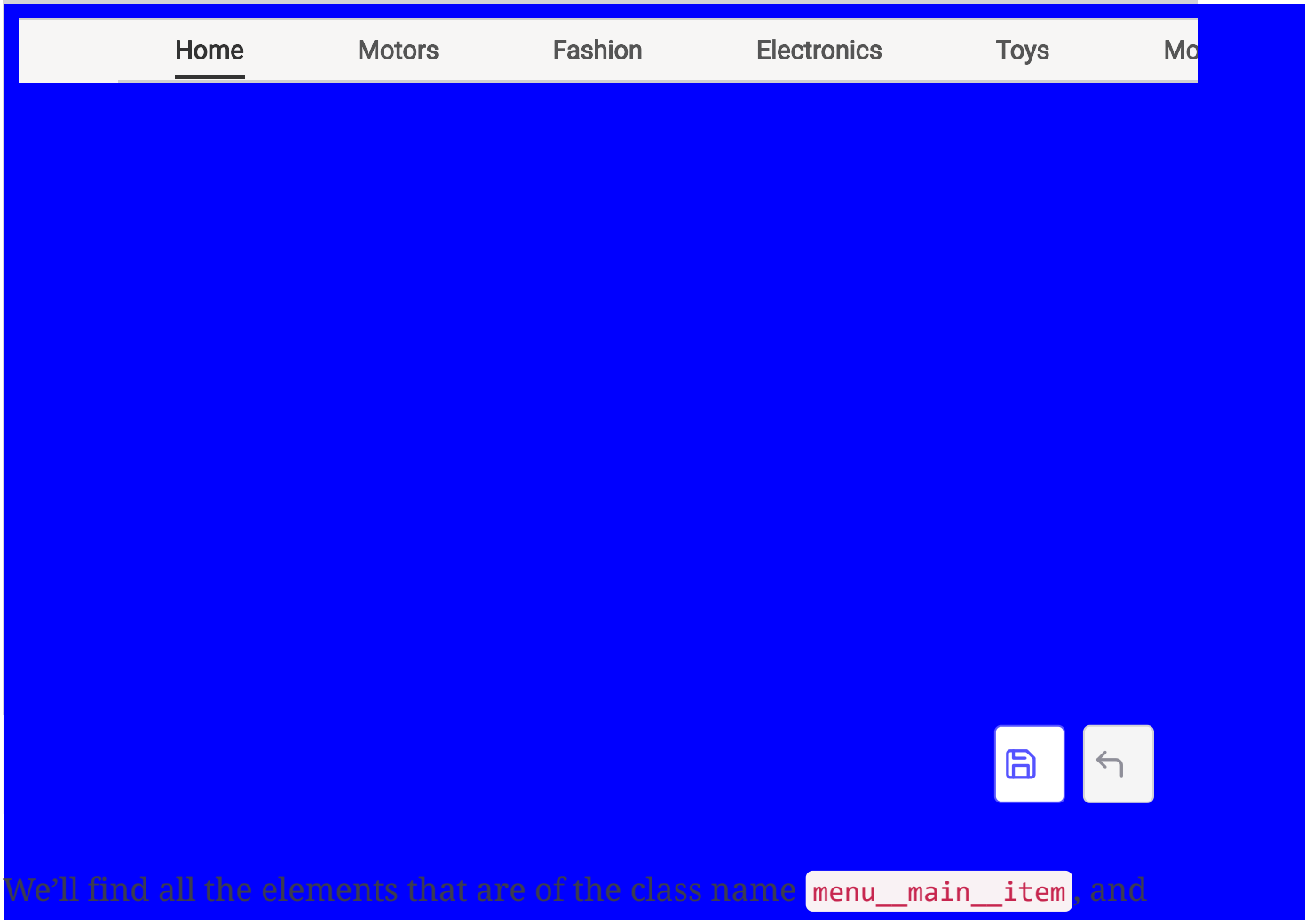
First thing's first: the submenu shouldn't be visible when a menu item isn't hovered over. Let's set the `display: none` property to be the default state, and when a menu item is hovered, it toggles back to `display: block`.

Output

JavaScript

HTML

CSS (SCSS)



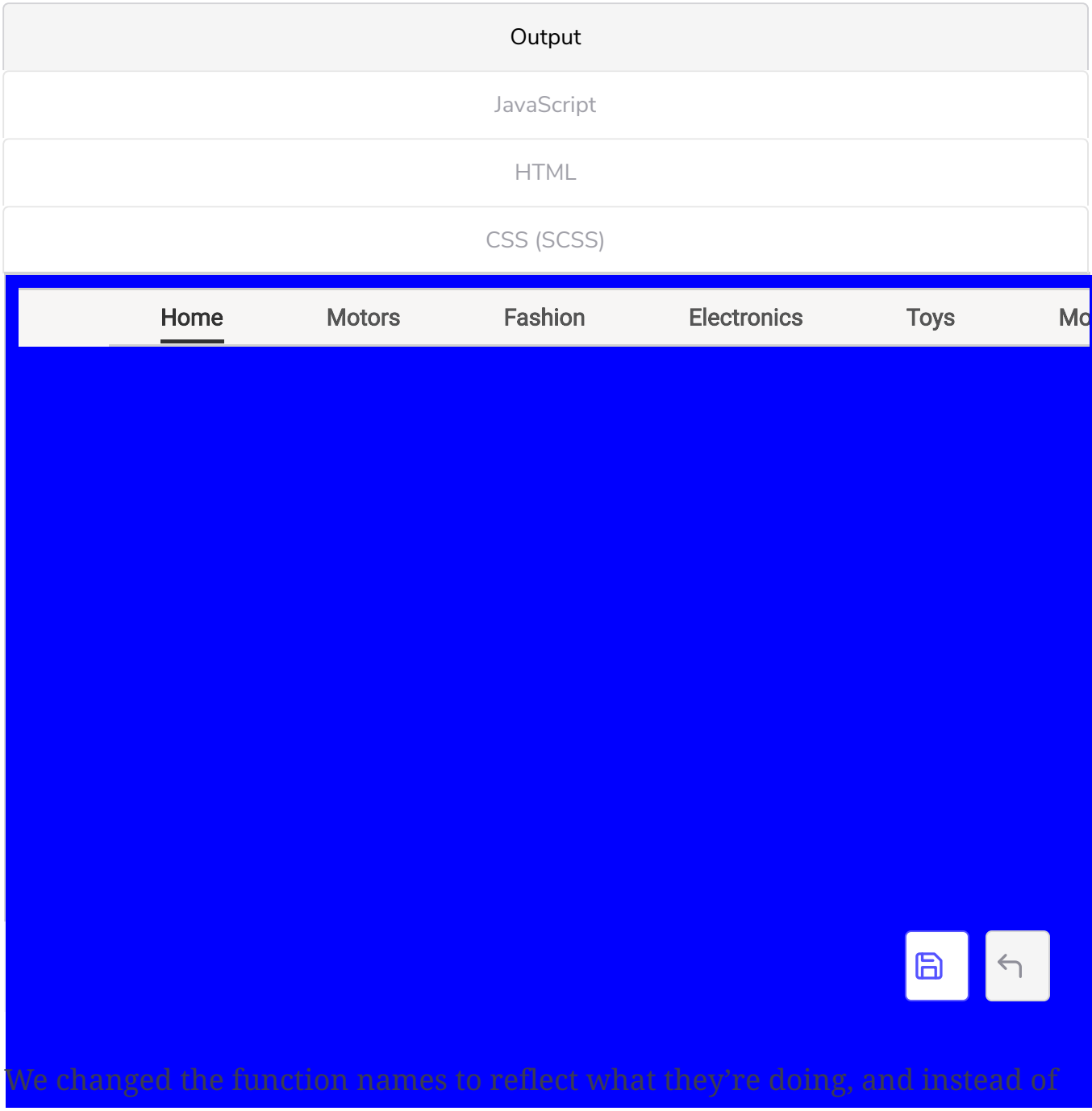
We'll find all the elements that are of the class name `menu__main__item`, and iterate through them so that we can bind functions to events. When one of these items get the event `onmouseenter` triggered, the function `menuItemEnter` is called. `menuItemEnter` finds the submenu by the class name, and sets the display property.

Do you see the issue here? We can't select anything in the submenu! That's because we're technically leaving the menu item when our mouse traverses to the submenu. To fix this, let's bind the mouse, leaving events to the submenu as well.

Hiding submenu correctly

By the way, this part is a little magical – how does moving the mouse away trigger the event it's supposed to? As the intro to this course says, the goal is to teach you how to build in components in plain JavaScript, so there are as little unknowns as possible, but unfortunately, we have to hit a wall unless you want to learn about computer architecture. But at least the APIs we're relying on that seem magical are ones provided by vanilla JavaScript (as opposed to a library built on top of it), which is the lowest layer available in web

development.



We changed the function names to reflect what they're doing, and instead of hiding the submenu on leaving an item, we hide it upon leaving the menu as a whole.

Retaining main menu item state

The next missing piece that's noticeable is that the menu items don't have the hovered white background when the submenu is displayed. We can address this by keeping the state of an "active" menu item where the active item is the one that has the white background and whose contents are in the submenu.

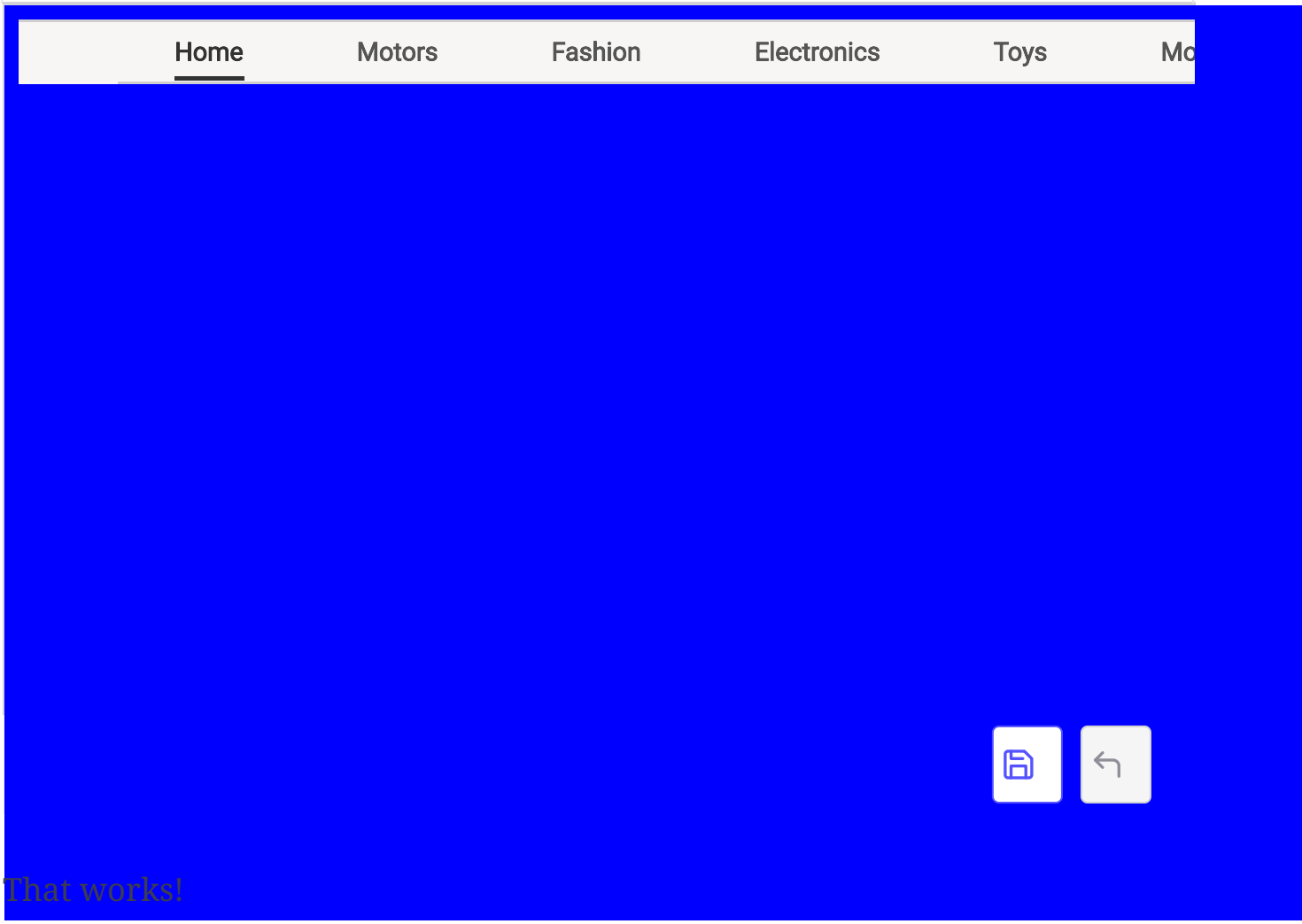
So for each menu item, let's add an "active" class to it upon the `mouseenter` event, and add to the CSS properties that match what it looks like when hovered over (even though we're not directly hovering over it).

Output

JavaScript

HTML

CSS (SCSS)



One small thing to note here is the syntax on line 18. We previously just had `onMenuItemMouseEnter`, but now we want to pass in the specific menu item as an argument. A common mistake is to do something like `menuItem.onmouseenter = onMenuItemMouseEnter(menuItem)`.

Q Why is the above line wrong?

If we had written the line like above, it would've called the function `onMenuItemMouseEnter` and used the return value as what's called when `onmouseenter` is triggered. Since the return value is `undefined`, this would've led to a runtime error.

If you're not familiar with ES6 (a version of JavaScript), the use of the `() =>` syntax just creates a function within the body as everything after the arrow. It's the equivalent of

```
function() {
  return onMenuItemMouseEnter(item);
}
```

This lets the `onmouseenter` event trigger the function correctly.

Deactivating main menu items

We still have a problem. We're not deactivating the active class, so when we hover over another menu item, it looks like we have two active menu items. Let's manage some states so that there's only one active item at any given time.

Note how the menu's `onmouseleave` function calls the `hideSubmenu` function which then updates the active menu item's `display` style property to `none`; ultimately hiding it from view.

Awesome! This looks close. Now, all that's needed is to toggle the content values based on which item is active, which means it's time to modify our server a little to accept data in the `get` parameters.