# Practice What You Learned


# DOM "Menu" Lab - Part 2

## Intro

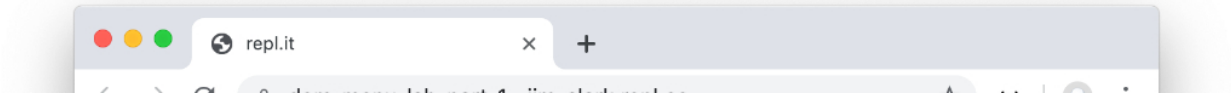
In the *DOM Events* lesson we saw how to run a function, i.e., an event listener, when an event, such as a click, was dispatched.

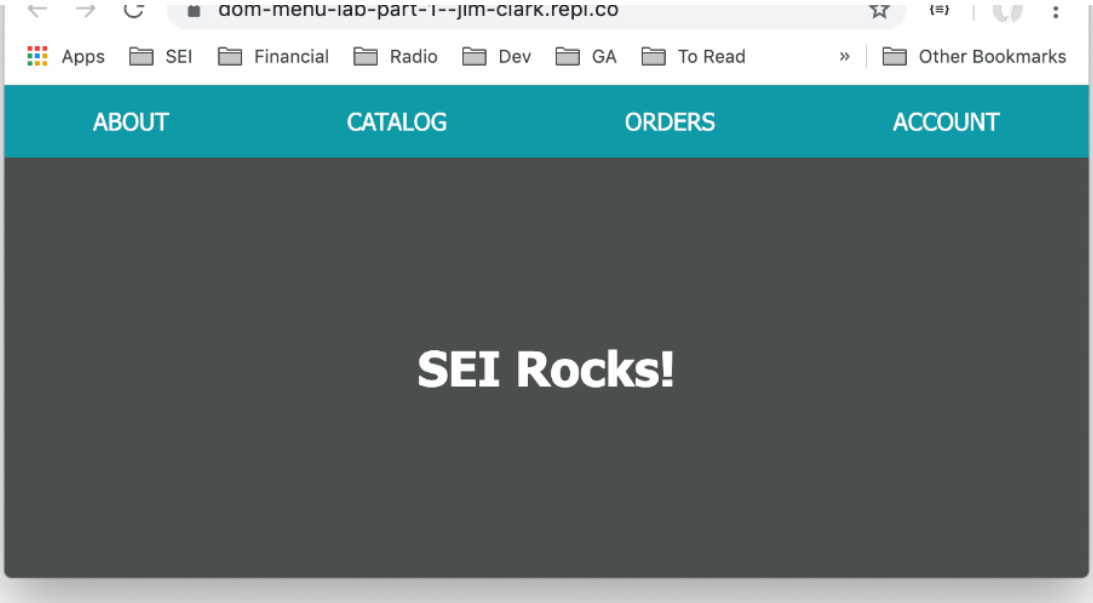
This lab continues where Part 1 left off and provides practice defining event listeners used to manipulate the DOM in response to user interaction. It also provides additional practice styling DOM elements dynamically using JavaScript.

### This lab is a deliverable.

## Setup

1. Continue to use the "DOM Lab" HTML/CSS/JS Repl you created in Part 1. This is what you should have thus far:






2. Insert an additional `<nav>` element within the `<header>` element in **index.html**:

```
<header>
  <nav id="top-menu"></nav>
  <!-- Add the <nav> element below -->
  <nav id="sub-menu"></nav>
</header>
```

Note: Other than the above changes, **DO NOT** modify **index.html** in any way.

3. Add the following CSS to the bottom of **style.css**:

```
header, #top-menu {
  position: relative;
}

#top-menu {
  z-index: 20;
}

#sub-menu {
  width: 100%;
  z-index: 10;
  transition: top 0.5s ease-out;
}

#sub-menu a:hover {
  background-color: var(--top-menu-bg);
}

```

```
19  nav a.active {
20    background-color: var(--sub-menu-bg);
21    color: var(--main-bg);
22  }
```

Note: Other than the above changes, **DO NOT** modify **style.css** in any way.

## Tasks

Tasks 1.0 thru 3.1 were completed in Part 1.

**Task 4.0**

Select and cache the `<nav id="sub-menu">` element in a variable named `subMenuEl`.

**Task 4.1**

Set the height `subMenuEl` element to be `100%`.

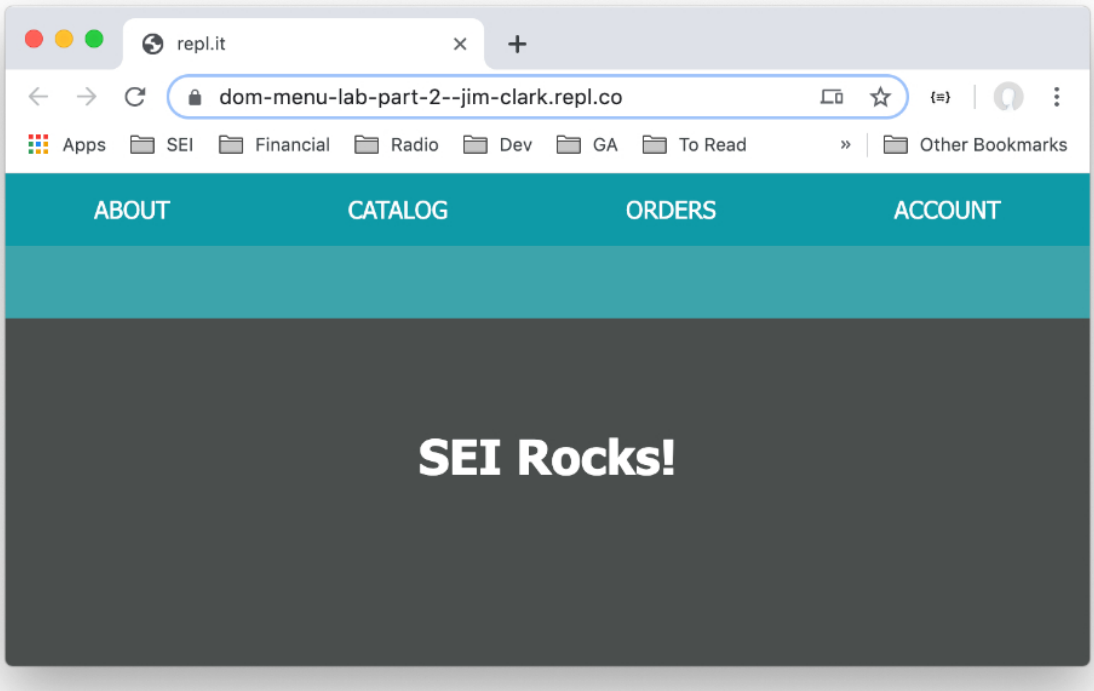
**Task 4.2**

Set the background color of `subMenuEl` to the value stored in the `--sub-menu-bg` CSS custom property.

**Task 4.3**

Add the class of `flex-around` to the `subMenuEl` element.

**Progress Check:**



**Task 4.4**

Set the CSS `position` property of `subMenuEl` to the value of `absolute` .

**Task 4.5**

Set the CSS `top` property of `subMenuEl` to the value of `0` .

**Task 5.0**

Update the `menuLinks` array in **script.js** to this:

```
var menuLinks = [
  {text: 'about', href: '/about'},
  {text: 'catalog', href: '#', subLinks: [
    {text: 'all', href: '/catalog/all'},
    {text: 'top selling', href: '/catalog/top'},
    {text: 'search', href: '/catalog/search'},
  ]},
  {text: 'orders', href: '#' , subLinks: [
    {text: 'new', href: '/orders/new'},
    {text: 'pending', href: '/orders/pending'},
    {text: 'history', href: '/orders/history'},
  ]},
  {text: 'account', href: '#', subLinks: [
    {text: 'profile', href: '/account/profile'},
    {text: 'sign out', href: '/account/signout'},
  ]},
];
```

**Task 5.1**

Select and cache the all of the `<a>` elements inside of `topMenuEl` in a variable named `topMenuLinks` .

Declare a global `showingSubMenu` variable and initialize it to `false` ;

**Task 5.2**

Attach a delegated 'click' event listener to `topMenuEl` .

The first line of code of the event listener function should call the event object's `preventDefault()` method.

The second line of code function should immediately return if the element clicked was not an `<a>` element.

`console.log` the content of the `<a>` to verify the handler is working.

**Progress Check**

Ensure that clicking **ABOUT**, **CATALOG**, etc. logs out **about**, **catalog**, etc. when a link is clicked.

Clicking anywhere other than on a link should do nothing.

**Task 5.3**

Next in the event listener, if the clicked `<a>` link has a class of `active`:

1. Remove the `active` class from the clicked `<a>` element.
2. Set the `showingSubMenu` to `false`.
3. Set the CSS `top` property of `subMenuEl` to `0`.
4. `return` to exit the handler.

**Task 5.4**

Next, the event listener should **remove** a class name of `active` from each `<a>` element in `topMenuLinks` - whether the `active` class exists or not.

**Hint:** Removing a non-existent class from an element does not cause an error, so just remove it!

**Task 5.5**

Next, the event listener should **add** a class name of `active` to the `<a>` element that was clicked.

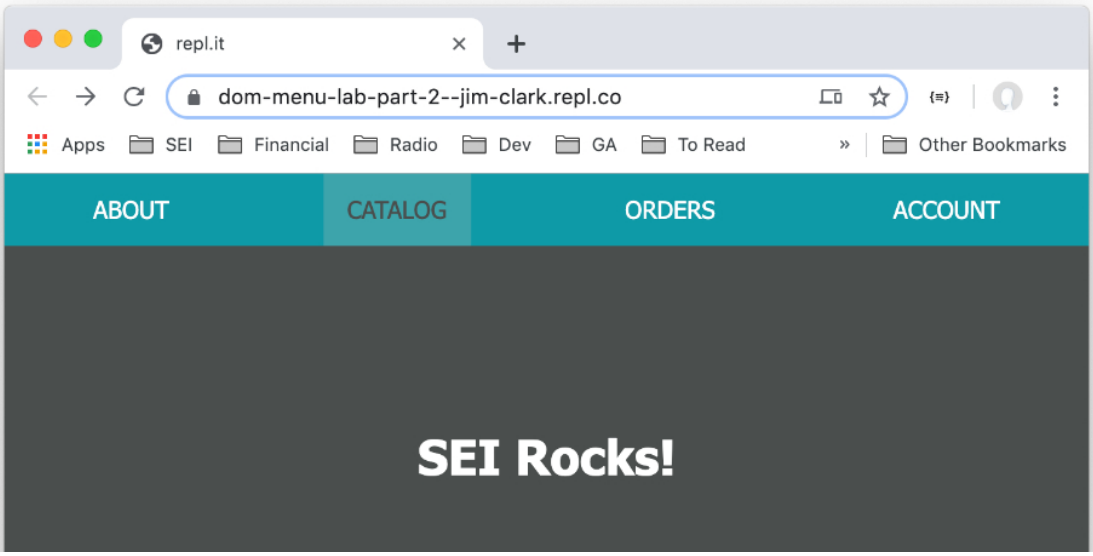
**Task 5.6**

Set `showingSubMenu` to `true` if the clicked `<a>` element's "link" object within `menuLinks` has a `subLinks` property (all do, except for the "link" object for **ABOUT**), otherwise, set it to `false`.

**Hint:** Saving the "link" object in a variable will come in handy for passing its `subLinks` array in Task 5.7

**Progress Check**

Clicking any of the links should make that link "active" and clear the others:



Clicking an "active" link should clear that link.

**Task 5.7**

Next in the event listener...

If `showingSubMenu` is `true` :

1. Call a `buildSubMenu` function passing to it the `subLinks` array for the clicked `<a>` element.
2. Set the CSS `top` property of `subMenuEl` to `100%` .

Otherwise ( `showingSubMenu` is `false` ):

1. Set the CSS `top` property of `subMenuEl` to `0` .

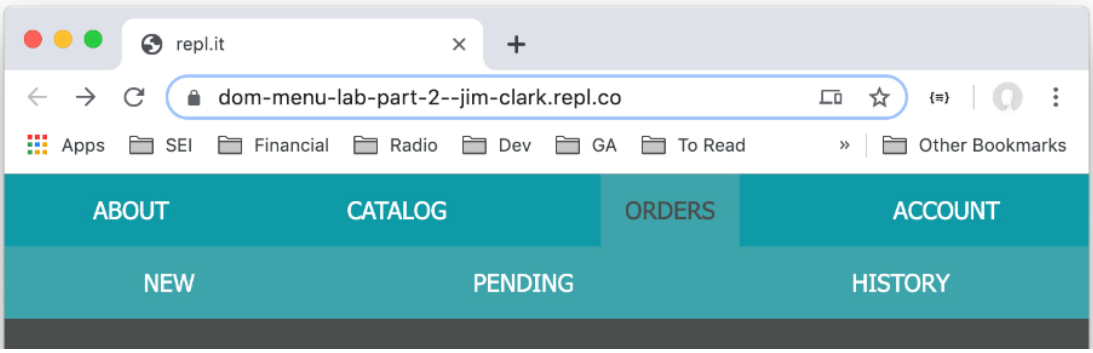
**Task 5.8**

Code the `buildSubMenu` function so that it:

1. Clears the contents of `subMenuEl` .
2. Iterates over the `subLinks` array passed as an argument; and for each "link" object:
    - Create an `<a>` element.
    - On the new element, add an `href` attribute with its value set to the `href` property of the "link" object.
    - Set the new element's content to the value of the `text` property of the "link" object.
    - Append the new element to the `subMenuEl` element.

**Progress Check**

Take the menu for a test drive!




**Task 6.0**

Attach a delegated 'click' event listener to `subMenuEl`.

The first line of code of the event listener function should call the event object's `preventDefault()` method.

The second line of code function should immediately return if the element clicked was not an `<a>` element.

`console.log` the content of the `<a>` to verify the handler is working.

**Task 6.1**

Next, the event listener should:

1. Set `showingSubMenu` to `false`.
2. Set the CSS `top` property of `subMenuEl` to `0`.

**Task 6.2**

Remove the class name of `active` from each `<a>` element in `topMenuLinks` - whether the `active` class exists or not.

**Task 6.3**

Update the contents of `mainEl` to the contents of the `<a>` element, within an `<h1>`, clicked within `subMenuEl`.

**Task 6.4**

If the **ABOUT** link is clicked, an `<h1>about</h1>` should be displayed.

**Congrats!**