PER SCHOLAS

ALAB 316.1.1:
DOM Manipulation (Part One)

Version 1.0, 6/15/23
Click here to open in a separate window.

Introduction

This lab is the first of a two-part assignment in which you will manipulate the DOM using various tools and techniques. This portion of the activity focuses on adding static elements created with JavaScript. In Part Two of this lab activity, we will make these elements dynamic and interactive.

Objectives

- Manipulate the DOM using JavaScript.

Resources

This lab uses CodeSandbox. If you are unfamiliar with CodeSandbox, or need a refresher, please visit our reference guide on CodeSandbox for instructions on:

- Creating an Account.
- Making a Sandbox.
- Navigating a Sandbox.
- Submitting a Sandbox link to Canvas.

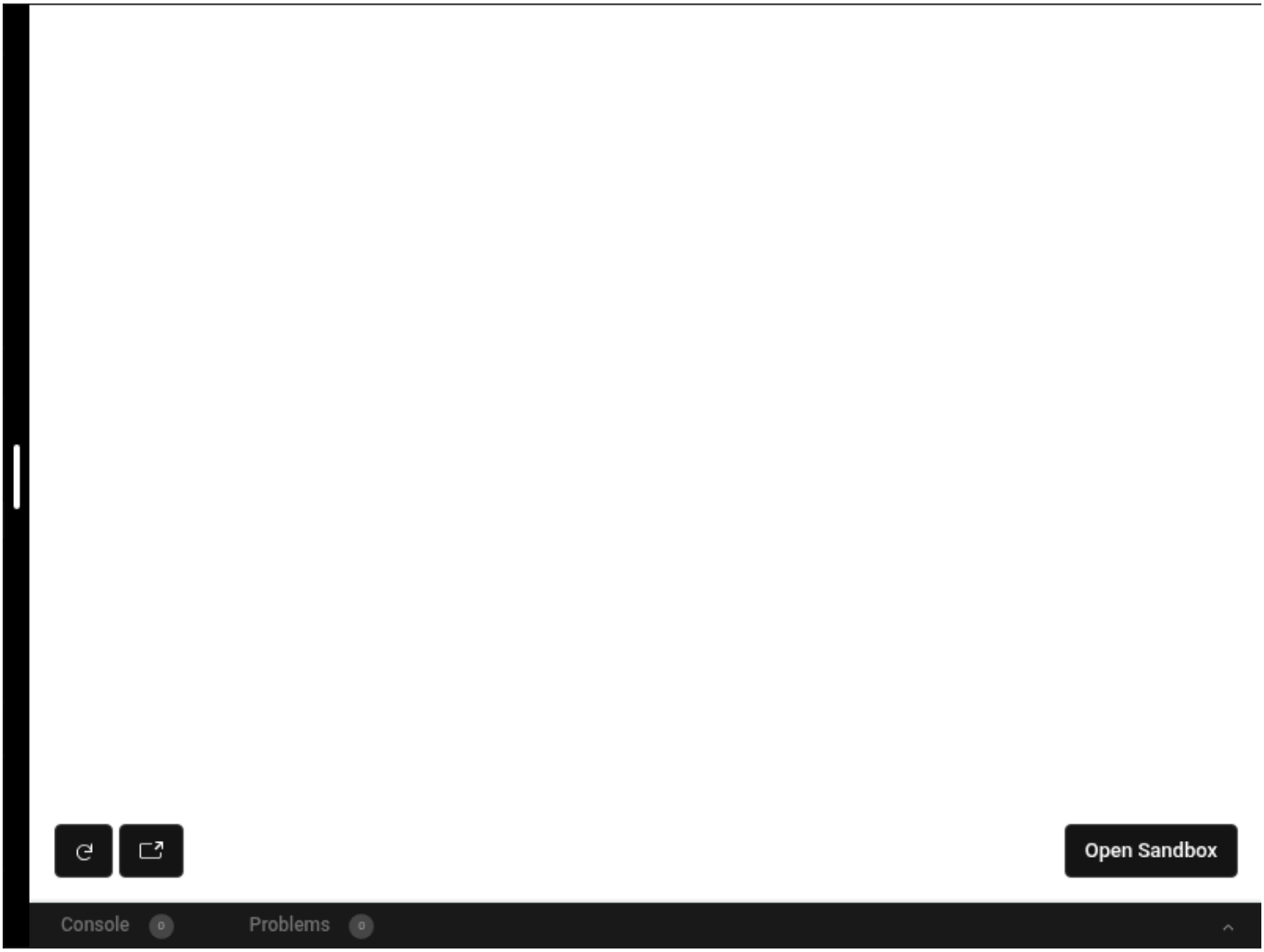
Submission

Submit your completed lab using the **Start Assignment** button on the assignment page in Canvas.

Instructions

You will begin with a pre-configured CodeSandbox:

- **ALAB DOM Manipulation (Part One)**



Fork the CodeSandbox above to get started.

Do not modify any of the contents of the index.html or styles.css files. Your goal in this lab is to demonstrate DOM manipulation through JavaScript. As such, directly modifying the HTML or CSS files is counterproductive.

Part 1: Getting Started

Explore the existing structure of the provided CodeSandbox to familiarize yourself with important aspects such as current DOM structure, element IDs, and available CSS classes.

Take five minutes to familiarize yourself with CSS Custom Properties (variables) - they are an amazing addition to CSS. If you are familiar with using variables with SASS/LESS pre-processors, CSS Custom Properties are similar but far more powerful because they are dynamic (their values can be changed during runtime) - and they are built into the CSS language!

Start the project by building a main content element using the following steps:

1. Select and cache the <main> element in a variable named mainEl.
2. Set the background color of mainEl to the value stored in the --main-bg CSS custom property.
    - **Hint:** Assign a string that uses the CSS var() function like this: 'var(--main-bg)'.
3. Set the content of mainEl to <h1>DOM Manipulation</h1>. There are a variety of ways to do this; pick whichever one that you think works best in this situation.

4. Add a class of flex-ctr to mainEl.
    - **Hint:** Use the Element.classList API.

Progress Check - Here is what the page should look like so far:

Part 2: Creating a Menu Bar

Next, create a blank menu bar that we can use to later add some interactivity to the page:

1. Select and cache the <nav id="top-menu"> element in a variable named topMenuEl.
2. Set the height of the topMenuEl element to be 100%.
3. Set the background color of topMenuEl to the value stored in the --top-menu-bg CSS custom property.
4. Add a class of flex-around to topMenuEl.

Progress Check - Here's what the page should look like so far:

Part 3: Adding Menu Buttons

Very often, you will be working with data provided by external sources in a variety of ways. For this project, copy the following data structure to the top of your index.js file; you will use it to create your menu buttons.

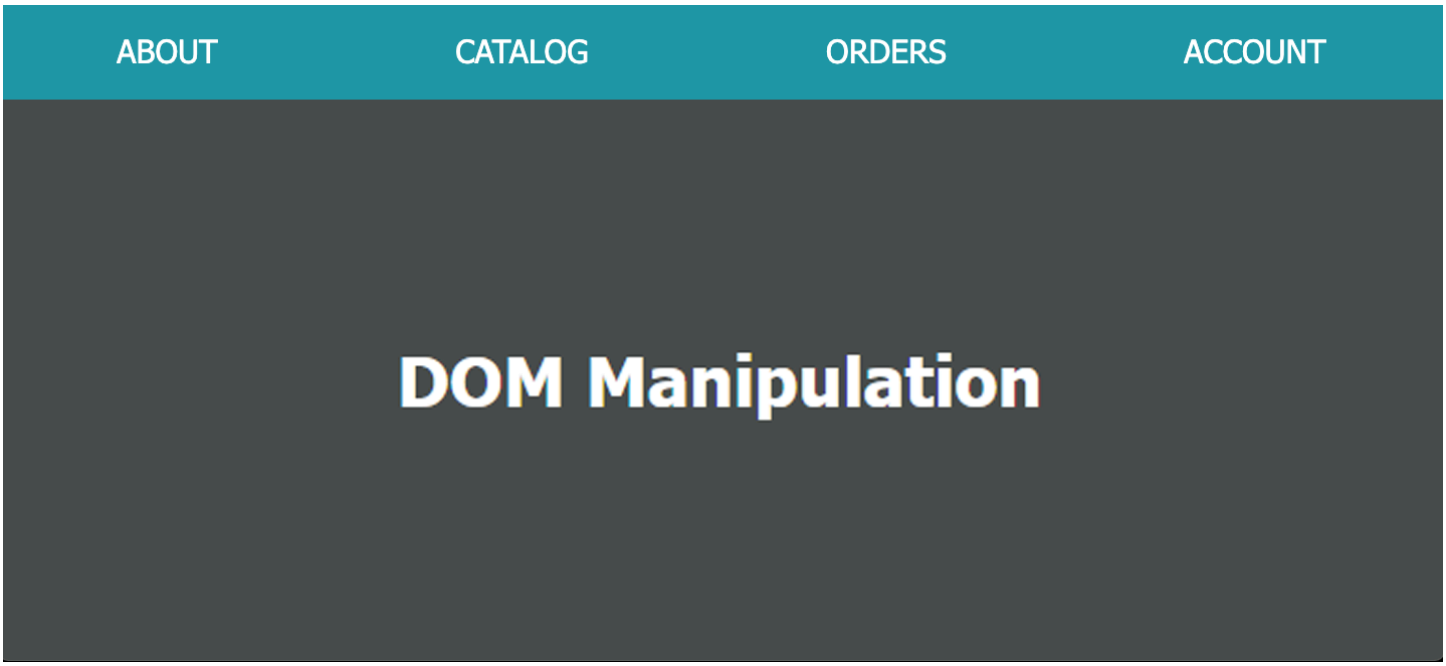
```
// Menu data structure
var menuLinks = [
  { text: 'about', href: '/about' },
  { text: 'catalog', href: '/catalog' },
  { text: 'orders', href: '/orders' },
  { text: 'account', href: '/account' },
];
```

If this data was provided by an external source, it would allow that source to control how our menu is built. We would simply implement the logic, and allow the data to decide what displays. This is not typically done with menus, but it can be done with *any* DOM element.

To continue:

1. Iterate over the entire menuLinks array and for each "link" object:
2. Create an <a> element.
3. On the new element, add an href attribute with its value set to the href property of the "link" object.
4. Set the new element's content to the value of the text property of the "link" object.
5. Append the new element to the topMenuEl element.

Progress Check - Here's what the page should look like so far:



Part 4: Adding Interactivity

With the basic structure of the page now generated purely with JavaScript, we have demonstrated the ability to manipulate the DOM in several fundamental ways.

In order to continue with this project, we must first explore how to add user interaction to DOM elements, which will be covered in a future lesson. For now, save your work so that you can return to it for Part Two of this lab activity.

Remember to submit the link to this part of the project to Canvas using the submission instructions at the beginning of this document.