PER SCHOLAS

ALAB 316.2.1:

Interacting with the Browser

Version 1.0, 6/16/23

Click here to open in a separate window.

Introduction

This lab will have you demonstrate your understanding of the Browser Object Model (BOM) and your ability to reference documentation to tackle unfamiliar problems by creating a simple prompt-and-response game. The instructions below outline the functionality that the game should have, but the content is up to you - be creative!

Objectives

- Interact with the BOM using JavaScript.
- Use user input to dynamically alter the DOM.

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.

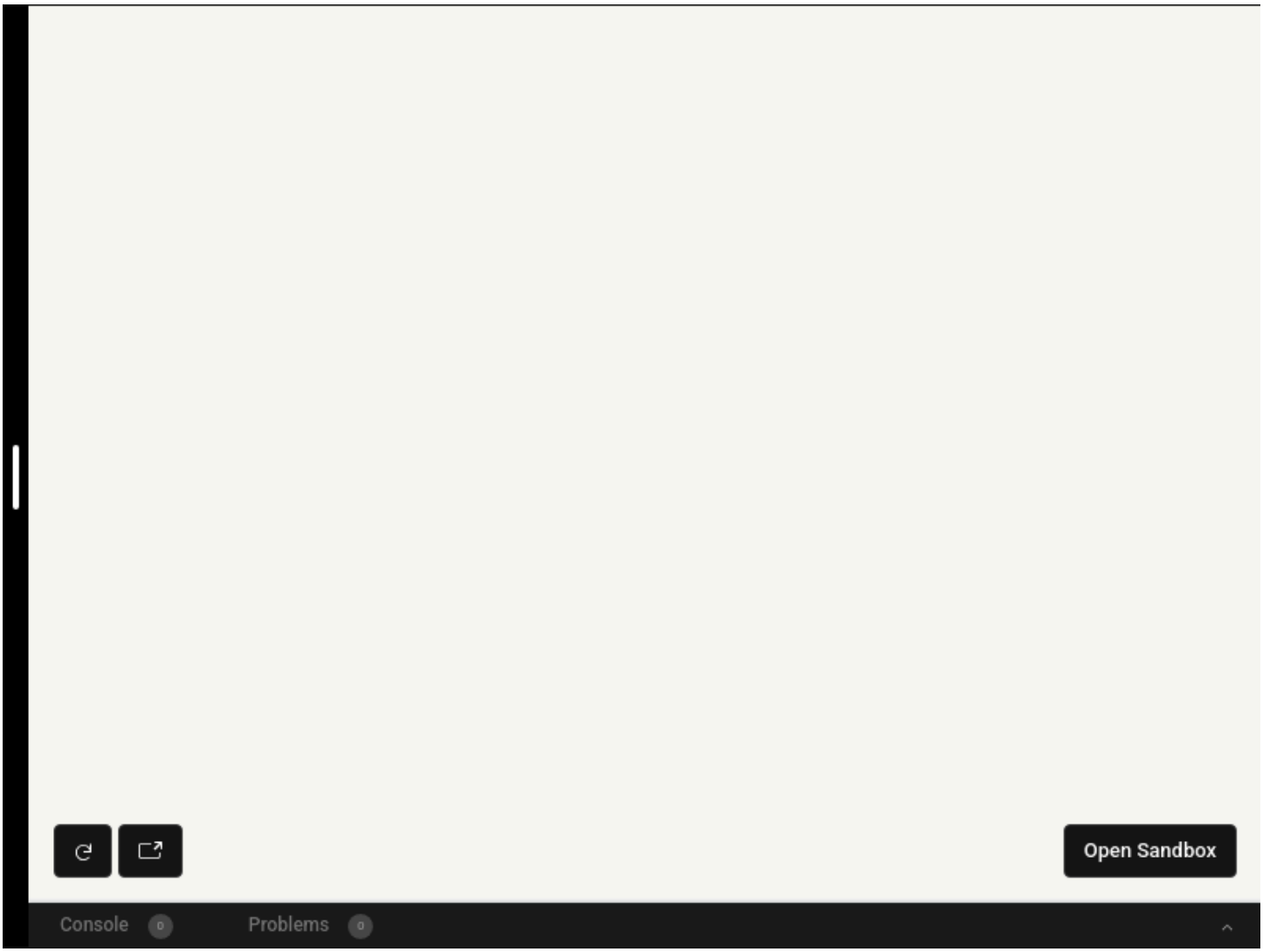
Submission

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 Interacting with the Browser**



Fork the CodeSandbox above to get started.

Your goal in this lab is to demonstrate interaction with the BOM and manipulation of the DOM through JavaScript, so directly modifying the HTML or CSS files to statically add dynamic content is counterproductive. Only modify those files if you need to add permanent elements.

Part 1: Requirements

You will create a simple guessing game. **Using window methods**, you will give and receive information from the user in order to direct them toward the correct answer in a limited number of guesses.

Here is a list of requirements for easy reference:

- Create a simple guessing game that pushes users toward the correct answer in some iterative way. The game does not need to be practical or complicated.
- Use **window object** methods to gather input from the user and display information to the user.
- Use DOM manipulation to give a visual indication of the game's progress in some way.

Part 2: Examples and Hints

We recommend starting simple, with numbers. Since numbers can be easily compared with conditional logic, alerting the user to the state of their last guess becomes relatively easy.

For example, here's a complete, working version of this theoretical number guessing game.

The answer to your game can be static or dynamic. You can set one answer that is always the answer, regardless of page reloads or other conditions, or you could randomly generate the answer using something like Math.random() and/or a list of set answers.

The method by which you push your user toward the correct answer can also be unique. For instance, here's a game that has the user guess a the breed of a cat by showing a blurred picture, which becomes progressively less blurry as the user guesses.

- The game uses DOM manipulation to change the CSS attributes of the picture based on the remaining number of guesses.

Again, your game does not need to be practical or complicated. You must only show an understanding of how to use the BOM and the DOM to create or manipulate content.

Part 3: Building the Game

Once you have an idea in mind, begin building. Remember that this can be a very simple game!

Along the way, you *will* encounter an issue with rendering while using alerts and prompts from the window object. These issues come into play due to a concept known as the Event Loop, which is beyond the scope of this module. For now, just understand that having a prompt or alert window open blocks anything from happening on the webpage itself.

As a temporary workaround, you can *delay* your prompts and alerts using another window method, **setTimeout**. If you are unfamiliar with setTimeout, here is a link to the documentation.

Here is how the workaround might look:

```
// Do some rendering tasks, manipulating the DOM
someDiv.style.backgroundColor = "blue";

// Delay a prompt to the user with setTimeout()
setTimeout(() => {
  alert("The sky is blue!");
}, 500);
```

This is not a requirement. If your program does not work as intended due to the blocking behavior of window.alert and window.prompt, that is okay! You will have the opportunity to better understand this behavior and mitigate it using different tools and techniques in the future.

Part 4: Completion

With your game finished, and tested, share your work with a peer. Discuss the differences in your approaches in order to form a better understanding of the variety of techniques available, even for a simple program like this one. Development is equal parts knowledge and creativity!

Do not forget to submit the link to your completed program using the submission instructions at the top of this document.