JavaScript Lab Assignment

Due: February 1st, 6:00 pm
Total points: 20

Document structure

- **Introductory Materials**
  - TA Session + Slides + Software
  - What is JavaScript?
  - Additional Resources
- **Lab Assignment (20 points)**
  - Stencil code
  - Exercise 1 (5 points)
    - Description
    - Deliverables
  - Exercise 2 (10 points)
    - Description
    - Deliverables
  - Exercise 3 (5 points)
    - Description
    - Deliverables

Introductory Materials

In-class TA Session + Slides + Software

The presentation slides from the in-class TA session will be posted on Blackboard. See lecture recording from 01/25 for the TA session on Blackboard, echo 360. The only software you'll need for this lab is **Google Chrome** and a **text editor**, such as Sublime, Atom, or VSCode.

What is JavaScript?

JavaScript is an object-oriented scripting language used on the client-side of web browsers. JavaScript helps to make webpage content dynamic and interactive, leading to the creation of friendlier interfaces for users.

Goal of the Lab

This lab aims to provide you with a simple introduction to JavaScript syntax. You will learn how to

declare variables, **write functions**, **create classes**, and **edit HTML content**. Understanding these topics will give you a good baseline for JavaScript programming and will help expedite your understanding of tools you can use to develop a user interface for group projects later in the semester.

Additional Resources

Below are some additional resources to help you get started with the lab exercises.

Printing to the Console

Printing to the console is a critical tool to success in JavaScript development. First, open up the Chrome console, which can be done in either of two ways:

- Press on the button that looks like a stack of three dots on the upper right-hand corner of this webpage. Navigate to More Tools and open Developer Tools.

OR

- Right-click your webpage and select “Inspect”.

You will use this console to write JavaScript code that will directly modify and interact with this web page. You will type your code into the console, then press Enter to run it. To try this out, enter the following code into Chrome’s console:

```
console.log("Hello World!");
```

This will print out the line “Hello World!” directly beneath the code you just ran. console.log() is the JavaScript command for printing statements, and is equivalent to System.out.println() in Java.

Note that beneath the "Hello World!" line there is one other line that gets printed: a light grey line stating "undefined". After executing your print statement, Chrome attempts to return some sort of value for your program. The “undefined” keyword signals that nothing has been returned; it is the JavaScript equivalent of null.

Variables

JavaScript stores data in variables. You can declare variables in JavaScript using the keywords “const” and “let”. **You should use “const” when the variable’s value will never be reassigned, and “let” when the variable’s value might be reassigned**. Note that “let” does not only indicate that the variable's value may change, but also that the variable will only be used in the block that it's defined in. This means that a variable declared with “let” may not always be accessible by the entirety of your code. It’s standard practice to use “const” to declare variables whenever applicable. We recommend you use “const” as often as your code allows.

We do not recommend the use of “var” to declare variables, as this keyword offers the least amount of information about the data stored in the variable. It has confusing behavior, and using this keyword

is generally seen as bad practice.

Note that JavaScript doesn't care what type of data the variable is, and you, therefore, do not need to explicitly declare a variable's type when declaring the variable. JavaScript will automatically detect a variable's type depending on the value associated with that variable. You can check this by running "typeof [VARIABLE NAME];" into Chrome's console.

In order to get comfortable using and declaring variables, we're going to run a couple of tests within Chrome's console. Run the following lines:

```
const milk = "Almond";
const sweetened = false;
let quantity = 50;
```

These variables are saved locally within the console. To test this, try running

```
console.log(milk);
```

The printed line should display "Almond". Now try running the following code:

```
milk = "Coconut";
```

The console should return an error stating "Uncaught TypeError: Assignment to constant variable". This is because we tried to reassign the value of a variable we declared with the keyword "const". Now try running the following code:

```
quantity += 25;
console.log(quantity);
```

The console should not return an error, as we declared quantity with "let". Instead, the console should return a value of 75.

In addition to the primitive data types we've been declaring thus far, JavaScript also has variable objects. Objects are variables that contain information in the form of key-value pairs. JavaScript objects are very similar to hashmaps, except they're able to store variables of different data types.

Try running the following code:

```
const almondMilk = {
name: "Almond",
sweetened: false,
quantity: 50
};
```

This creates an object called milk. You can directly change the values in this object by running the following lines of code:

```
almondMilk.quantity += 25;
almondMilk.sweetened = true;
console.log(almondMilk.quantity);
console.log(almondMilk.sweetened);
```

console.log(almondMilk.quantity) should return 75, just as
console.log(almondMilk.sweetened) should return true.

Note that although we declared almondMilk with "const", we were still able to change the values contained within player. In this case, cereal itself is the variable that we cannot reassign to a new object.

Functions

Functions in JavaScript are essentially objects. You declare them like this:

```
function pourCereal(cerealName) {
console.log(cerealName + " has been poured!");
};
```

Depending on whether a function needs access to data from another source, a function may or may not have arguments. In the example above, the function named pourCereal takes in a single argument named cerealName .

To call this function, simply use its name and include the appropriate arguments, like so:

```
const myFavoriteCereal = "Frosted Flakes";
pourCereal(myFavoriteCereal);
```

In JavaScript, functions may return one of three things:
1) a variable
2) another function
3) the value "undefined".

ES6 Syntax

JavaScript ES6 introduced new syntax which makes function declarations simpler. Here is an example of the pourCereal function written with ES6:

```
const pourCereal = cerealName => {
console.log(cerealName + " has been poured!");
};
```

OR even without the {} like so:

```
const pourCereal = cerealName => console.log(cerealName +
"has been poured!");
```

Fun tip: ES6 also introduced **template literals**! These allow you to use variables inside of a string instead of having to use + to concatenate them, as shown below. The pourCereal function could further be simplified to:

```
const pourCereal = cerealName => console.log("{$cerealName}
has been poured!");
```

Familiarizing yourself with ES6 will help you learn React later in the semester!

Classes

JavaScript provides a clear and simple syntax for declaring classes and using inheritance. They are designed to be easy to use and are useful in approaching JavaScript from a more object-oriented perspective. To declare a class, use the "class" keyword with the desired name of the class.

In this example, we will create a class named "Student".

```
                class Student {
                constructor(name, year, favoriteCereal) {
                this.name = name;
                this.year = year;
                this.favoriteCereal = favoriteCereal;
                this.id = "13877865";
        }
}
```

You can create new instances of the Student class by running the below code:

```
const carlos = new Student("Carlos", "junior", "chex");

console.log(carlos.name + " is a " + carlos.year + " and
loves " + carlos.favoriteCereal + ".");
```

Lab Assignment (20 points)

Stencil Code

Please download the stencil code and unzip the folder.

Exercise 1 (5 points)

To get more practice with JavaScript, we're going to be making a webpage that displays a cereal box and name on the click of a button.

Description

Here's what you'll need to do:

1. You'll be working with **cereal1.html** and **cereal1.js**. Follow the TODOs outlined in **cereal1.js**. You do not need to edit **cereal1.html**. Remember, your goal is to show the image and name of a cereal on the page when the "Display Frosted Flakes" button is pressed.

2. To see the live results of the webpage you are creating, open the HTML page in a web browser. You can double-click on the HTML file in your directory.

Deliverables for Exercise 1 (5 points)

- Updated JavaScript file Cereal1.js **(3 points for working functionality)**
- Screen capture of the webpage that displays a cereal box and name on the click of a button. The screen capture should show the following: **(2 points)**
    - The page **before the button** is pressed (you can just refresh the page). An empty shelf should be shown.
    - **When the" button is pressed**, an image of Frosted Flakes should be displayed, and the text "Frosted Flakes" should be shown above the image.
    - Place a **breakpoint** in some function of the **Cereal1.js** file using Chrome Developer Tools and demonstrate how that breakpoint gets hit by interacting with the webpage.

Exercise 2 (10 points)

Now that you've mastered the basics, you're going to use your new skills t create a website that generates a random cereal when the button is pressed and keeps a tally of how many times each cereal was generated.

Description

Here's what you'll need to do:

1. Now you'll be working with **cereal2.html** and **cereal2.js**. Follow the TODOs in **cereal2.js** to generate a random cereal when the button is pressed and update the tally board to record how often each cereal is suggested.

Helpful Tips:

- The syntax for writing conditional statements is linked here.
- Check this resource for how to access an element in a list/array.

Deliverables for Exercise 2 (10 points)

- Updated JavaScript file Cereal2.js **(7 points for working functionality)**
- Screen capture of the webpage that generates a random cereal when the button is pressed and updates the tally board to record how often each cereal is suggested. The screen

capture should show the following: **(2 points)**
    - The page before the button is pressed (where an empty shelf is shown).
    - Clicking the button should show a cereal box, display the name of the cereal, and increment the corresponding counter on the left.
- ES6 syntax for the changeCereal() function. **(1 point)**

Exercise 3 (5 points)

Now you're going to improve your incrementCerealCount() function by taking advantage of the power of the map function.

Description

Here's what you'll need to do:

Make a copy of cereal2.html and cereal2.js and call it **cereal3.html** and **cereal3.js**. In **cereal3.js**, re-implement your incrementCerealCount() function so that it maps the current count of each cereal to the text of the corresponding td element. **This solution should not require an if statement. The function body for incrementCerealCount() does not need to be longer than two lines of code.**

Helpful tips: You're going to need to update the td element of a Cereal without knowing whether it's Frosted Flakes, Captain Crunch, or Lucky Charms. Think about how you can augment the Cereal class to do this!

Deliverables for 2 (5 points)

- Updated JavaScript file Cereal3.js **(4 points for working functionality)**
- Screen capture of the webpage that generates a random cereal when the button is pressed and updates the tally board to record how often each cereal is suggested. The screen capture should show the following: **(1 point)**
    - The page before the button is pressed (where an empty shelf is shown).
    - Clicking the button should show a cereal box, display the name of the cereal, and increment the corresponding counter on the left.

Final Deliverables

- Deliverables for Exercises 1,2, and 3 (as listed above) should be submitted in **one zip folder** (javascript_lab.zip) by **February 1, by 6 PM.**

Course Policies

- All students should follow standards of conduct as discussed in UIC's disciplinary policies (see syllabus for more details). As per the course policies, you should be doing this assignment independently without discussing it with other students in the class or elsewhere. Do not copy and paste code from online sources.