ISC 250, Unit 6 Lab

Objects

The goal of this lab is to get a working understanding of how objects are used in programming, and in JavaScript.

Instructions

As with all assignments, **all lab documents must be accessible from your course webpage.**

To ensure this:

1. create a folder on your course website for this lab,
2. create and index.html file for the lab folder, and link to it from your primary course webpage's index,
3. create an HTML file for each lab task (e.g. task1.html, task2.html, task3.html, etc.), and link to them from this lab's index.html.
4. finally, to indicate you've completed the lab, submit a link to this lab folder's index on the associated blackboard assignment.

1. Create: A script to access and display object data

This page contains a display for nutrition facts for cereals. Unfortunately, right now it displays nothing at all, and its buttons don't work.

This script, on the other hand, contains the definition for an array (named "cerealData") of objects containing the nutrition information for a variety of cereals.

Your goal is to write a script to take information from the objects in the "cerealData" variable (which is, itself, an array) and display it for the user.

Things you'll need:

1. Some sort of overall "counter" variable to keep track of which entry in the cerealData array you're currently looking at. This will be a simple integer, starting at 0.
2. A function to update the viewer's HTML elements with the data from the cereal objects. Notice that the HTML ids are all the same as the property names of the cereal objects. Think carefully about what this function needs to do: (A) access the information in each property of the "current" cereal object, and (B) transfer that information into the elements on the page (think innerHTML).
3. Both a prev() and next() function, which should both: update the counter variable AND call the viewer update function. (Also consider how to prevent the user from going out of bounds-- an if-statement before updating the counter variable should help.)

In the end, you'll have a small interface for a user to interactively examine the nitrition facts of a variety of cereals.

2. Correct: Car battery replacement method

This page contains a little Car object that will drive when you click the "Drive!" button. Unfortunately, its internal Battery object runs out of juice quickly!

That's fine, we can just call the Car's "changeBattery()" method to grab one of our extras. Except... wait, it never runs out of batteries. What's going on?

Your goal is to correct the Car object's "changeBattery()" method to properly grab a battery from the Car's "batteries" array and install it in the "currentBattery" property.

1. Notice that the changeBattery() method is creating a new Battery() object every time it's called. While this is nice, it doesn't make much sense-- where are all of these batteries coming from???
2. To fix this, note how the "batteries" and "currentBattery" properties are used when the Car object is being defined.
    1. The Car object stores the array of Battery objects in its "batteries" property.
    2. Then, the Car calls that array's "pop()" method-- which simultanesously retrieves one of the objects AND removes it from the array. This is, essentially, using the pop() method to "move" one of the batteries to a new location.
3. First, explore how Arrays' "pop()" method works on your own. Then use the pop() function to properly and efficiently grab one of the batteries and install it.

Now the car's range should properly be limited by the number of batteries it has stowed.

3. Complete: Dice

This page contains some interface elements that let us create Die objects, which then display on the screen. We can even customize them to create custom, different, or even loaded dice.

But! These dice aren't rolling. Fill in the "rollDice()" function to roll all the dice displayed.

1. Notice the Die objects each already have a "roll()" method.
2. Notice also that the variable "dice" contains an array with all of the page's dice.
3. Write the function such that it iterates through all of the dice in the "dice" array and calls their "roll()" methods.

Now whenever you click the "Roll Dice" button, each die will display a face chosen at random.

Grading Rubric

Labs are graded as follows.

Each task is worth four (4) points, for a total of twelve (12) for the entire lab.

1. **Submitted (1 pt).** You'd be surprised...
2. **Attempted (1 pt).** Demonstrated an effortful attempt at solving the problem.
3. **Completed (1 pt).** Successfully solved the problem.
4. **Style (1 pt).** Good formatting of both the code (white space, indentation, clarity, comments) and the webpage (white space, readability, not utterly offensive to the eyes).