Jay Annadurai

MAD SP23

T. Faas

14 Feb 2023

Exercise 5: Arrays & Loops

[Link to GitHub](https://jannadurai.com/Exercises/Exercise%205/index.html)

**Introduction**

The HTML Array & Loop exercises iterate on the previous DOM exercises but emphasize creating and updating HTML DOM entities using loops and arrays. The goal of the exercise is to apply iterative and recursive algorithms to HTML DOM entities.

**Algorithms & Planning**

     All files contain a similar structure of potential JS Classes followed by any general JS which is then ultimately followed by any JS functions. Within the specific exercise set, all exercises have the same program flow of defining a HTML object as a JS object and using arrays and loops to reach the desired output.

**Reflection**

     Ex5.1 simply requires one div with a list of incremented numbers. It simply requires a loop to generate the inner HTML content and place that content within a created div. The HTMLasJS class was used to generate the div and each loop increment iterated upon the previous inner HTML content of the generated div. This exercise was fairly easy. The only change in the future would be to make it more visually appealing.

Ex5.2 is similar to Ex5.1 in that it is a div with a list of incremented numbers. The same logic applied as to the div generation and inner HTML generation. The distinction comes in form of the logic tree required to transform specific numbers to arbitrary strings based on their factors. Sequential conditional logic was used to perform the logic checks for the transformations. Other than that, this was a fairly streamlined exercise.

     Ex5.3 was perhaps the most difficult, but difficult of my own making. The original exercise calls for a previously-sorted array but I decided to generate a random array with a random order. As such, I needed an algorithm to sort through the elements of the randomized array. Creating a sorting algorithm was difficult but the one I’m most innately familiar with is the selective sort algorithm but I decided to try a bubble sort algorithm. The logic was not too difficult but there were small syntactical errors which were easily corrected. Once the bubble sort algorithm is applied to the randomized array, the sorted array is iterated through and a new HTMLasJS object is created for array element. Each object is created as a sub-object of the Window object. Some small conditional logic is applied to check for the top 3 fastest ‘times’ is executed so that the podium positions can have unique div colors. This exercise was the most difficult of the set, but it was well worth it. The most interesting aspect of it was calling methods with arguments in array notation to the window object. In the future, instead of reiterating the same inline style for each div element, class functionality can be built into the HTMLasJS class such that there can be a HTMLClassasJS class which either acts as a parent class or extends the functionality of the HTMLasJS class.

Ex5.4 utilized the same logic as Ex5.3—an array iterated through with each array element having a corresponding div element. In this case, it was actually easier than Ex5.3 as there’s no sorting mechanism necessary. The same remarks about extending property functionality from a super class or extending functionality to a child class in Ex5.3 apply here as well.

Overall, while the exercises in set 5 were not stunning, they demonstrated the functionality achieved from the HTMLasJS class initially built in exercise set 4. In that form, the exercises were a great success as there were only minimal modifications, mainly a setInnerHTML method, required to attain the functionality required to complete all exercises in set 5.