Introduction & Overview

In this assignment you will practice using JavaScript to respond to events and to read inputs and write outputs from/to the DOM. This assignment brings together the early course concepts in HTML/CSS and previous material on JavaScript coding to create a simple web application.

Though you might find solutions to these problems on the Internet, you are reminded that the only value these exercises have is as practice to you. Any detected incidents of academic integrity violations will be cited.

The goals of this assignment are:

1. Practice using JavaScript to detect and respond to events in a HTML/CSS environment.
2. Practice writing software solutions to problems using JavaScript.
3. Practice reading input from a web page using JavaScript.
4. Practice writing output to specified HTML elements using JavaScript.

All elements in your submission must be styled to some degree but it is not expected that you try to create a work of art. A neat, professional design will be very sufficient. If something is not specified, then you have flexibility to be creative. If you are uncertain, please post a question in the discussions area.

Details

Files and layout

Create 1 HTML file, 1 JavaScript file, and at most 1 CSS file. Name your HTML file index.html. You may pick appropriate names for your JavaScript and CSS files though you are encouraged to use all lowercase letters, and no spaces. You are encouraged to use a logical directory structure for your CSS, images, and other resource files. All CSS must appear in external style sheets. All JavaScript must appear in your external JavaScript file and ***must*** be loaded in your **head** element.

Content

You are expected to use your own creativity to design this web application and talents to implement it. An example solution can be viewed at https://youtu.be/8eQemjnD6O0 but you are welcome to arrange the required elements in any creative way you choose. Remember that being creative does not mean finding neat widgets on CodePen or elsewhere and applying them, we want you to have the experience of building the components yourself.

The page will follow the header/main/footer format that we previously used. A nav bar is not required here since there is only one page and it's not semantically appropriate to put miscellaneous controls there.

Inside the header place a h1 tag with a title for your page. Inside the footer place identifying information like your name, student number, and the current year.

You will require a total of 9 images with 3 themes to complete the assignment. Initially you will have 3 empty image tags, but once the page is loaded, call a function to choose 3 of the images to display, 1 from each theme, random inside the group. For example: I randomly picked 1 cat, 1 dog, and 1 star picture from my collection of images. Each time I refresh the page I get 1 cat, 1 dog, and 1 star, but not the same ones all the time.

They must appear evenly spaced, and responsive within the main element. Hint: This gets a ***lot*** easier if all your images are the same dimensions.

Elsewhere on the page you will have an input for the time until an automatic refresh will happen, a countdown timer, a manual refresh button, and a counter of how many times the images have been updated.

Images

Each of these images will be clickable using either an onclick attribute or by JavaScript addEventListener() method.

When an image is clicked the following effects take place:

- A transition animation plays.
- During the animation the image changes to a random image (from any category).
    - It is not required that you guarantee that all the images are unique, it is ok if an image repeats. If you wish to try to make this enhancement, please do!
- The automatic refresh timer is reset.
- The image counter is increased by 1.

Store your image filenames in a 3x3, 2D array for ease of access.

Making the animation play can be a bit tricky, and so you may use the following code to accomplish it. Be sure you change 'spin' to the name of your CSS Class that links to your animation.

```
function do_animation( event ) {
    target = event.srcElement;
    target.classList.remove('spin');
    setTimeout( () => {target.classList.add('spin');}, 0 );
}
```

Input

You will have 1 input element that accepts a number only. This number must be in the range [500,10,000]. Though you can use HTML attributes to try to enforce this range, the JavaScript must verify it before attempting to use the input. When the input element is changed, call a JavaScript function that validates the input, and if the input is numeric and is in the valid range, change the automatic refresh time, cancel any old refreshes, and restart the refresh cycle anew.

Randomize Button

When the randomize button is clicked, it will run a function that picks 3 random images from your collection and replace the current ones with them. These image changes do not activate your CSS animation, they just suddenly change. These 3 images do not have to be from different groups, nor do they have to be different images. Random allows for 3 of the same images to appear one click, and for 3 different images the next click. When this happens the automatic refresh timer is reset, and the number of images displayed total increases by 3.

Clock

A countdown timer until the next automatic refresh will exist, counting down in 1/10ths of a second. When the timer reaches 0 your web application will behave exactly as if the randomize button had been pressed. As the counter ticks down you will change the colour scheme of the timer window. It must start at white text/green background, transition to black text/yellow background, and finally to white text/red background before returning to the first state. You can decide at what point these transitions occur, but each should last long enough to be noticeable if the reset timeout is set to 2500ms.

Image Change Counter

Display, in real time, how mange times the image has been changed, as noted above.

Notes

This assignment relies heavily on timers. Do not attempt to do everything in one timer, it is far simpler to have different timers controlling different functions, especially if they have different time out values.

Do not wait until the last minute to do this assignment. There are many separate, interacting parts that you can work on independent of each other. I strongly recommend using ***problem decomposition*** strategies to break this into manageable chunks of work.

You have a great deal of liberty to implement the design in a way that you consider appealing if it meets the requirements.

Misc. Requirements

- Bootstrap, if used, must be version 5. Any other version will result in a grade of 0. You are allowed to augment your BootStrap CSS with your own custom elements, but it shouldn't be most of the styling.
- Use semantic elements wherever possible.
- Follow the suggestions in the HTML/CSS/JS Style Guides
- Validate your code using the HTML Validator (linked in module 0)

Submitting Your Work

Save all files in your CS Unix **public_html/private/10259/a3** directory, which will correspond to the https://csunix.mohawkcollege.ca/~sa#########/private/10259/a3 directory.

Copy all HTML, CSS, and original JavaScript files (not the Bootstrap files) and add ".txt" to the end of the file name of each file. Submit only these files in Canvas – no zips, no images, etc.

Evaluation

See the rubric posted on the LMS for details of the evaluation.

Please direct all questions to your professor, or to the discussion group on the LMS.