COMP 10259: Assignment 4

Overview

The goals of this assignment are:

1. Practice creating a web application using SVG and JavaScript.
2. Practice creating a web application using JavaScript event handling.
3. Practice responding to events through JavaScript DOM manipulation.
4. Practice the concept of state and variables within a web app.

If something is not specified, then you have flexibility to be creative. If you are uncertain, please post a question in the discussions area. Please note that there are many solutions on the Internet that might work as starter code for this assignment however, you are required to write all code yourself from your own ingenuity. Please see the Academic Integrity Policy or ask a question if you are unsure. Violations will be treated as academic offenses.

NOTE: if the assignment is not submitted to both CSUnix and to Canvas it will receive a grade of 0.

Content

In this assignment you are being asked to create a simple graphical application or a game. Since we have no back-end support, we will need to work 100% client side. You are not expected to retain any information between page loads – high scores, etc. will reset every new page load.

Do not over complicate the program itself, it can be as simple as a heads/tails coin flipping game, a dice game[1], memory matching puzzles[2], a typing tutorial, HTML tutorial (remember Super Markup Man?) or some other simple game that you wish to create. Perhaps there's even a Happy Chicken assignment in your Java course that can serve as inspiration. The goal here is to practice programming, not to create a AAA multi-million dollar hit game. The artistic standard is "good enough" not "gallery ready". A demo video is available at https://youtu.be/oB0HepB7sNs

Required Elements

- Your program must meaningfully make use of SVG elements.
  - These elements must be interactive using at least 2 types of JavaScript mouse events (your choice of specific events). You may change 1 mouse event to a keyboard event if you wish.
  - These elements must have the ability to be removed: they must disappear from the screen and be removed from memory.
  - These elements must be creatable by some mechanism. Suggestions include an add button or a timer that "spawns" them. After creation they should appear on the screen.
  - You may have an initial layout of hard coded elements if desired, but all other adding and removing of elements must be dynamic using JavaScript.
- Style your SVG elements using CSS in addition to the basic XML parameters available.

[1] See http://www.activityvillage.co.uk/dice-games for some inspiration.
[2] See https://www.mindgames.com/game/Memory+Trainer for some inspiration.

- Animate at least 1 SVG element using JavaScript. The animation must be obvious and noticeable. Do not use JavaScript to add a CSS animation to an element, nor use an animated GIFs, etc., to fulfil this requirement.
- Use standard HTML elements to establish user inputs and criteria for the program. Use at least 1 input element without a form to accomplish this requirement, in addition to any other elements you wish. For example: change the colour of all the circles, make all the squares twice as large, make the dice 8 sided instead of 6, make the animation move twice as quickly, etc. The specifics will depend on what kind of program you're creating.
- Track the state in some meaningful way – for example, a score, number of tries remaining, count of circles created; something that is connected to your application.
- Continue to use good HTML structure by creating a header and footer around your main content. The header or footer must contain your name and a copyright notice for 2023. The page should be titled and appear neat and complete.

You may use Bootstrap 5 or multimedia elements, including CSS animations to make your project visually appealing and interesting. You may not use jQuery.

- The app must run from beginning to end with no bugs or errors in the Console.
- All JavaScript and CSS code must be in separate, external files.
- The page should look ok on a phone (i.e. maximum width 320px) and on desktops as well.
- All files must be documented according to "Documenting JavaScript Programs" on the LMS.

Handing In

To eligible for grading the following steps must be done:

1. Name your html file as **index.html** and save all files in your CSUnix **public_html/private/10259/a4** directory.
2. Validate your HTML and CSS files are standard compliant
    a. Check your HTML and CSS files using the W3C validators. Ensure that there are no errors. Minor warnings will not result in a penalty, but any serious warning should be corrected. Please use the discussion group to get help to resolve the warnings and errors, or to clarify if you are uncertain.
3. Make sure you also follow the instructions in the Assignment Documentation Standard, including the student's name a number in each file you submit, including your CSS files.
4. Submit your work to the LMS by copying each of your html, css, and js files to add a ".txt" extension to them. For example "index.html" becomes "index.html.txt". Do not archive your files (no renamed zip files). Do not upload any multimedia files, i.e., images, videos, music, 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.