COMP 10259: Assignment 5

Overview

The goals of this assignment are:

1. Practice creating a web application using SVG and JavaScript.
2. Practice responding to events through JavaScript DOM manipulation.
3. Practice the concept of state and variables within a web app.
4. Practice designing and creating a larger, interconnected program.

If something is not specified, then you have flexibility to be creative. If you are uncertain, please post a question in the discussions area. <u>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.</u> Please see the Academic Integrity Policy or ask a question if you are unsure. Violations will be treated as academic offenses.

<u>**NOTE: if the assignment is not submitted to both CSUnix and to Canvas it will receive a grade of 0. The assignments are becoming sophisticated enough that testing the application is very important.**</u>

Content

In this assignment you are creating a simple graphical application. There are a few choices you can decide between. Check the required elements list and be sure to point them out using comments in your code.

In class we have looked at examples that demonstrate how to create SVG elements both in HTML as static elements we interacted with, and dynamic elements using JavaScript and manipulating the DOM tree.

Mixed into this we also added some application logic that controlled how our elements were created, or how they behaved.

Ideas

You are not required to create any of these ideas, they are just here to give you some inspiration. You may choose one of them if you wish but are free to explore your own idea. Either way, be sure to check the required elements list.

- A small game where objects move on the screen and when interacted with by the user they are "broken" into smaller objects a couple of times. The smallest ones can simply disappear. I'm thinking about the idea in the old game Asteroids where they split into smaller asteroids. You shouldn't worry about building the whole ship/shooting part of the game, instead you can use mouse events for the user to interact with your objects.
- A simulation where a user must press up to jump over oncoming objects or down to duck under other objects or accept input by detecting the mouse being moved up and down while a button is pressed (to avoid jitters when the button is not pressed). This can be very basic, high levels of fluid movement are not required. This can be like Flappy Bird or can be more like Donkey Kong (but without the ladders). Animate both the background, to simulate movement, and the foreground obstacles.

- Create a simple tree drawer – do not try to be complex, this problem can grow quite quickly. Your tree drawer should connect nodes based on some relationship the user can manipulate. For example, "make node parent of ……" with a select list of available nodes. Other relationships like "make node child of …" and "make node sibling of …" could work. Model some sort of relationship with your tool as a sample, a very small part of a family tree, or a couple of tic tac toe boards.

Required Elements

- Create SVG elements in your HTML code using XML markup, and dynamically using JavaScript
  - Use at least 1 ellipse or circle, 1 polygon, 1 textpath, and at least 1 curve in some meaningful way in your design
  - These of these elements can be used in a logo on your page that you have invented.
- Style your SVG elements using CSS
- Animate at least 1 SVG element using CSS animation
- Animate at least 1 SVG element using JavaScript to animate
- Use at least 2 different kinds of mouse events to interact with elements, for example all kinds of clicks are one kind, while all kids of mouseover style events are another kind. You may substitute 1 of your mouse events for a keyboard event if you wish.
- Use standard HTML elements to establish user inputs and criteria for the game. For example, in class we used buttons with click events to call functions that determined the number of houses. Use at least 1 input element without a form to accomplish this requirement, in addition to any other elements you wish.
- Track the state in some meaningful way – for example, a score, number of tries remaining, 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 2022. The page should be titled and appear neat and complete.

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/a5** directory. Note that paths are case sensitive, a5 is not the same as A5.
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. Ensure your JavaScript runs without errors output to the console. You may have debug statements output to the console without penalty (e.g. deliberate console.log() messages are ok)

4. 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.
5. 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

Technical requirements: 70%

Aesthetics and presentation: 20%

Documentation and validation: 10%

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