F21DV Dynamic and Interactive Lab Report

Kyle Dick 4th Year Software Engineering MEng

This document contains discussions regarding the implementation of Lab 2 solutions to the provided exercises. Some exercises will have their documentation merged due to the nature of their implementation.

Section 1: CSS Effects/Animations

Exercise 1

The code used for this exercise is an updated version of the line graph code from the first lab. The main difference between the ideas in the implementation is that the 'markers' which are made to pulse on a mouseover are handled by the CSS provided through the index page. This is achieved by assigning a class to the markers which the CSS can communicate with.

Exercise 2

Reusing the code from lab 1 regarding the filtering of data, this displays a set of 5 elements with the shapes determined by the contained value. Using the CSS example of pulsar from the first exercise, the pulsar effect is added to the text values of the elements rather than the shapes themself. When hovered over the values are displayed.

Section 2: Events

Exercises 3, 4 and 5

The size changing properties of both the rectangle and the circle were achieved by defining their sizes previously and manipulation based on those. Mouseover events on the circle and rectangle detect when to grow and mouseover used to know when to shrink.
The svg also has a mousemove event which when called creates a text element within the svg which follows the mouse with each movement, slightly offset.

Section 3: Events

Exercise 6, 7 and 8

Starts with the creation of a div within the body of the page, using the style function of d3 the width, height and background color are set. An attached mouse over event then monitors for when the mouse hovers over this div, when this is done the style function is called and the previously defined attributes are edited to grow the shape into a red square from a smaller blue square. A second after this change the color will then also transition to shrink back to its original size with the difference of now being green in color.

Exercise 9

This exercise required the creation of three divs with a transition, each requiring a different ease method. In order to achieve this, a function was created which constructs the div and appends it to the DOM while also taking the parameter, of type d3.ease, and applying the correct ease to the transition

Exercise 10 and 11

Using the circle example from exercise 3, 4 and 5 the easing effect was added to the growing and shrinking effect of the circle. An additional text element is created which also has a hover effect which enlarges it and changes color only while hovering over.

Exercise 12, 13 and 14

Created three rectangles from div elements, each grows to a specified height over 2 seconds, waiting for the bar on its left to finish before starting. Once they are all at their max height they begin shrinking from right to left, waiting for their neighbour on the right. While the bars are going to their height they transition to red, and back to blue as they shrink.
This was achieved by using a function named Update which is called once. When called it creates the bars and using delay functions waits for a specified amount of time before doing their grow and shrink transition.

Exercise 15, 16 and 17

Creates a bar chart using data from an external csv file. The chart includes a title and a labelled axis for both x and y. The bars shift from a blue to red hue as they progress rightwards. On page load the chart is blank and the bars load in by growing downwards from their maximum y coordinate. Hovering over the bars grows them slightly and displays their exact value above the bar.
The coloring is handled by using a linear scalar which aims to later create the first element as blue and the final element as red with the intermediate elements showing a hue shift between these values.
From here the axis, and bars are created with the bars including a ease transition to their height attribute being defined such that it grows downwards.
Finally a function is created for both the mouseover effect of growing the bars then displaying the text and for the mouseout effect where the text is disappeared and the bar returns to its original size.
An external CSS file is also used in this example to create the highlight effect. When the mouseover function is called it changes the specified div to the highlight class, then mouseout removes the class from the object.

Exercise 18, 19, 20, 21 and 22

A bar chart with three buttons, each button associated with a data set. Pressing a button transitions the data on the chart to accurately represent the data. Dataset 2 includes one more element than dataset 1 and 2. This is accounted for as the extra element is transitioned in and out when needed. The axis' of the chart is adjusted to properly reflect the data represented. Each chart also has its own associated color, created by an ordinal scalar.

The switching of data is handled by a join() function call which handles exit, update and exit events. Two additional functions are included which show and hide the value represented by each bar when hovered over.

Exercise 23

Extending the previous exercises to a line chart instead of a bar chart. Utilises the line chart code from the first exercise and as such when combined with the CSS file the points can be hovered over which causes them to pulse.

Exercise 24, 25 and 26

Commenting explains the effect of the exercises in the javascript file. That commented information is included below:
Exercise 24
The output is [16.2, 34.4, 5.2].
The parameter of 0.2 means that these values are a 20% increase towards the next value.
For example, value 20 and value 1
There is a difference of 19 between these values.
19 * 0.2 = 3.8
20 - 3.8 = 16.2

Exercise 25
The output value is rgb(128,64,0)
The value of red is rgb(255,0,0)
The value of green is rgb(0,128,0)
By interpolating 0.5, the returned value is the halfway point between these two colors.
The R value is halved to 128 (realistically it should be 127.5 but RGB only accepts int)
The G value's final value would be 128 rather than 255, so its 0.5 value is 64.

Exercise 26
Below shows interpolation on dates.
By creating new dates, the interpolation starts at midnight of that day.
Hence calling 0.5 on the example would return Midnight of Monday the 21st of Febuary 2022

Exercise 27

Displays a pie chart to represent two different data sets which can be transitioned between by using the buttons provided. This is done by manipulating the arc functions through the join() function. When an update is called it changes the arc by the new values.

Exercise 28, 29, 30, 31 and 32

This final implementation uses forces along with circle elements contained in an svg. The forces apply a spread to the circle elements with the circle elements representing points of a data set. A mouseover event is attached to each of the elements which is able to display the

value of the circle when it is hovered over. A set of three functions then handles the dragging functionality of the elements. A function for the click event which adds ‘heat’ to the simulation which has cooled at this point causing all the simulation to pause. Adding this ‘heat’ causes elements to once again react to forces. The dragging function is similar to the function used to have text follow the mouse from a previous exercise. Due to the simulation being active once again, the elements will move if pushed by this element. This is due to the collision force being enabled. Finally the drag Ended function called which removes heat from the simulation to settle it.