F21DV Coursework Lab 2 Report

Jonathan Song Yang, Lee (H00255553)

Demonstrated to: Amit Parekh (11/02/2022)

Contents

1 Introduction

Lab 2 focuses on using `d3.js` for dynamic and interactive visualisation concepts. It was meant to be a step-up from Lab 1, where we were taught the basics of d3.js.

Github Repo: https://github.com/jonleesy/F21DV-Coursework

Github Pages: https://jonleesy.github.io/F21DV-Coursework/public

1.1 Set-up

The set-up for this lab is the same as the Lab 1, but instead of using hard coded div properties, I have implemented CSS grid for aligning divs and webpage objects, as recommended by my lab 1's lab helper.

```
  /**
  * Create div's for each question systematically.
  * @param {*} exerciseNumber Task number.
  */
export function createDiv(exerciseNumber) {
    d3.select('body')
        .append('div')
            .attr('class', 'container')
            .append('div')
                .attr('class', 'answerCenter')
                .append('p')
                    .append('strong')
                        .text('Exercise ' + exerciseNumber + ':')
}
```

Listing 1.1: Old Method

```
  /**
  * Similar to createDiv(). Was told to look into
  * grid instead of using hard coded div settings.
  * This one focuses on that, and will be used starting
  * from lab2.
  * @param {*} exerciseNumber
  */
export function createAnswerDiv(exerciseNumber) {
    d3.select('body')
        .append('div')
            .attr('class', 'grid-container')
            .append('div')
                .attr('class', 'title-grid')
                .append('p')
                        .append('strong')
                            .text('Exercise ' + exerciseNumber + ':')
    d3.select('.grid-container')
        .append('div')
        .attr('class', 'answer-grid')
}
```

Listing 1.2: New Method

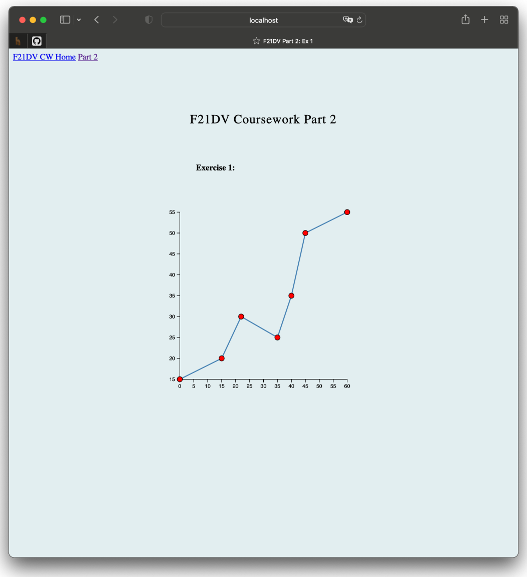
```css
/* Part 2 onwards CSS using grid */
.grid-container {
    display: grid;
    grid-template-columns: auto 100px 100px 100px 100px auto;
    grid-template-rows: 60px auto auto auto auto auto;
    grid-gap: 10px;
    /* background-color: #262c3046; */
    padding: 10px;
}

.grid-container > div {
    background-color: rgb(226, 238, 240);
    padding: 20px 0;
}

.title-grid {
    grid-area: 1 / 2 / 1 / 6;
    text-align: left;
    text-indent: 20%;
}

.answer-grid {
    grid-area: 2 / 2 / 6 / 6;
    text-align: center;
    align-content: center;
}

.answer-grid-small {
    grid-area: 2 / 3 / 6 / 5;
    text-align: center;
    align-content: center;
}
```

Listing 1.3: New CSS Method

Looking at listing 1.1 and 1.2, the only difference is with the new grid (`grid-container`) being added, then adding a smaller answer fiv with class `answer-grid` afterwards. The preperties of these grids above are shown in listing 1.3.

2 Exercises

2.1 Exercise 1



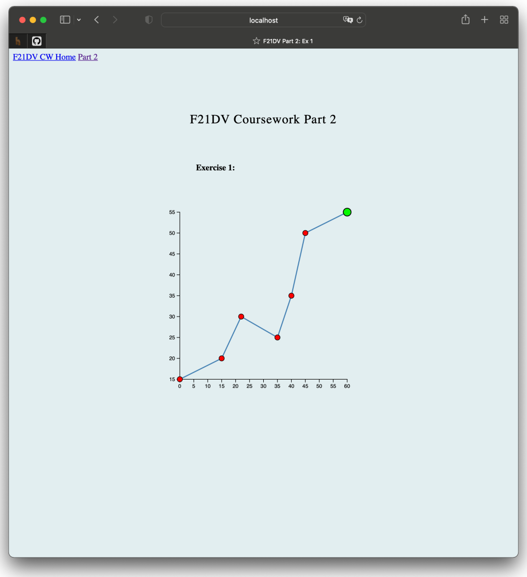


Figure 2.1: Exercise 1

Figure 2.1 shows a line chart with data points plotted on it. The right figure shows what happens when the mouse hovers upon a data point, the data point would pulse between red and green. **Can't seem to take a screenshot and capture the pointer.*

2.2 Exercise 2

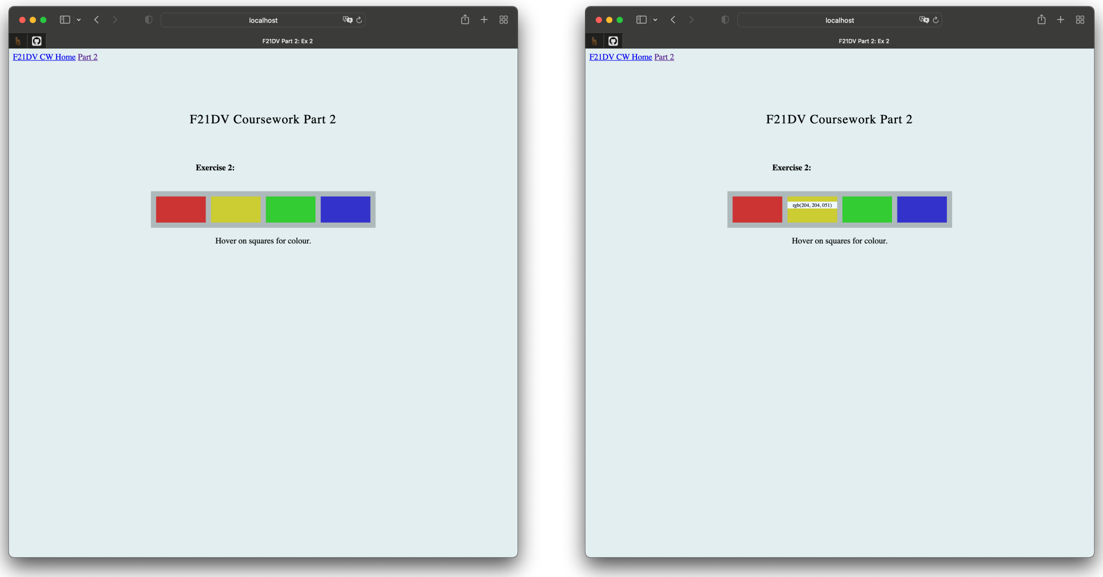


Figure 2.2: Exercise 2

Figure 2.2 shows 4 blocks with different colours. The blocks are just a div defined within a grid. Each block also has a fill colour defined using the `Rgb()` colour method, as shown in line 10. Upon hovering the mouse on the div, the div would show a text element displaying the Rgb colour. This effect is done using the css `:hover` method.

2.3 Exercise 3

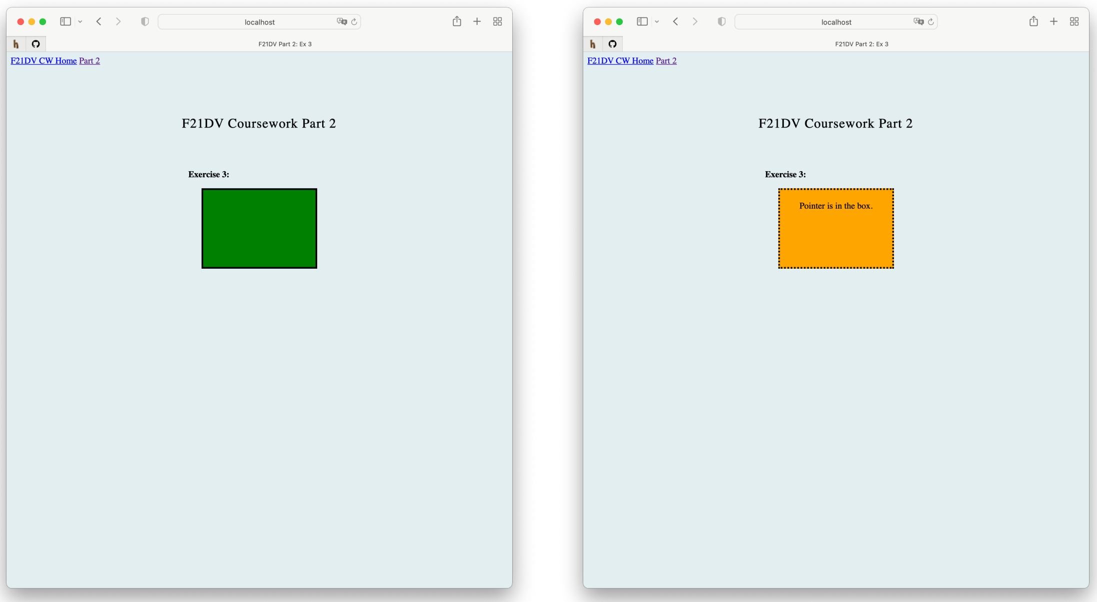


Figure 2.3: Exercise 3

```
// .js script for exercise:
const ex = 3;

// Imports of functions.
import { createAnswerDiv } from '../functions.js';

// Creating base <div>s systematically.
createAnswerDiv(ex);

// Adding the original div.
d3.select('.grid-container')
    .append('div')
        .attr('class', 'answer-grid-small')
        .style('width', 'auto')
        .style('height', '100px')
        .style('background-color', 'green')
        .style('border-style', 'solid');

// Change colour, border and text, using .on() mouse over and out.
d3.select('.answer-grid-small')
    .on('mouseover', function() {
        d3.select(this)
            .style('background-color', 'orange')
            .style('border-style', 'dotted')
            .text('Pointer is in the box.');
    })
    .on('mouseout', function() {
        d3.select(this)
            .style('background-color', 'green')
            .style('border-style', 'solid')
            .text('');
    });
```

../../public/js/part2/task3.js

Figure 2.3 shows a block thats green colour. Upon mouse hover on the box, the box now shows a text that says ‘Pointer is in the box’, its colour is now orange, and it has a new type of border. This is done using the `.on()` function.

2.4 Exercise 4

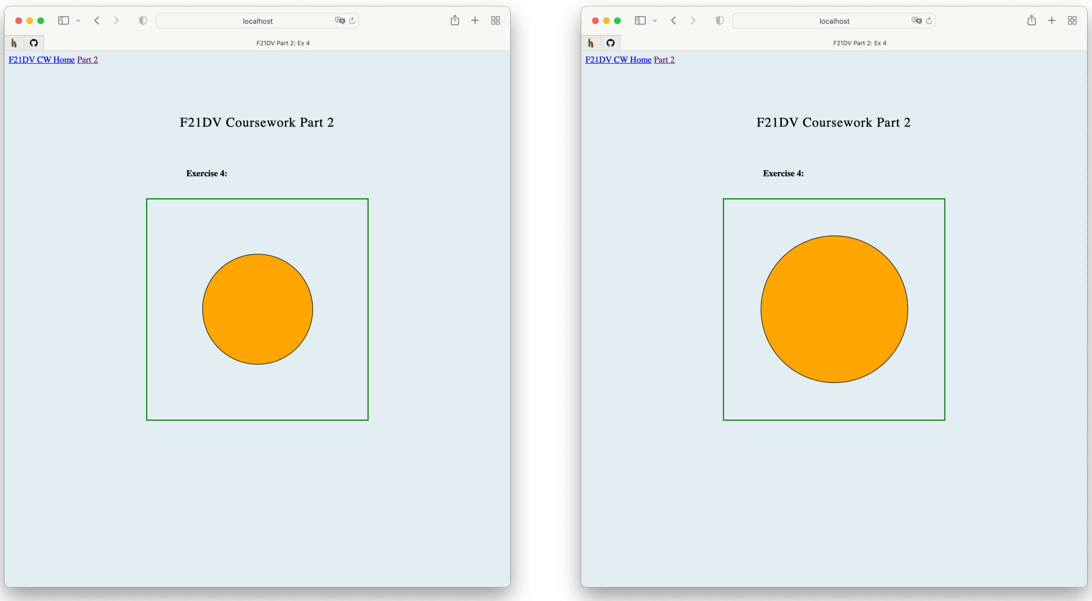
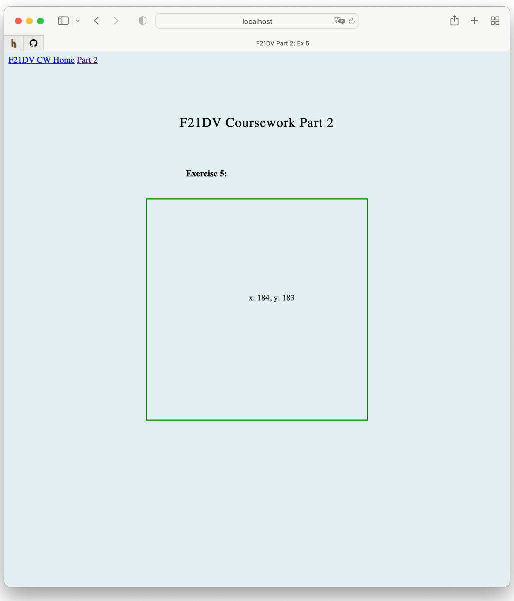


Figure 2.4: Exercise 4

Figure 2.4 shows an SVG object with a circle in the middle. Upon hovering on the circle, the circle enlarges. This time, the action was modeled using d3 transitions instead of css :hover method. I've added transtitions upon mouse hover and out.

2.5 Exercise 4



F21DV Coursework Part 2

Exercise 5:

x: 184, y: 183

Figure 2.5: Exercise 5

Figure 2.4 shows an emptey svg object, and upon mouse hover, it would show the corredinated of the mouse. There was also a pre-appended empty text box. To show the x-y coordinates, this is done using the event data of the mouse movement. Then using the data, I modified the ‘x’ and ‘y’ attribute of the text box.