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.

```
1     /**
2     * Create div's for each question systematically.
3     * @param {*} exerciseNumber Task number.
4     */
5    export function createDiv(exerciseNumber) {
6        d3.select('body')
7            .append('div')
8                .attr('class', 'container')
9                .append('div')
10                   .attr('class', 'answerCenter')
11                   .append('p')
12                       .append('strong')
13                           .text('Exercise ' + exerciseNumber + ':')
14   }
```

Listing 1.1: Old Method

```
1     /**
2     * Similar to createDiv(). Was told to look into
3     * grid instead of using hard coded div settings.
4     * This one focuses on that, and will be used starting
5     * from lab2.
6     * @param {*} exerciseNumber
7     */
8    export function createAnswerDiv(exerciseNumber) {
9        d3.select('body')
10           .append('div')
11               .attr('class', 'grid-container')
12               .append('div')
13                   .attr('class', 'title-grid')
14                   .append('p')
15                           .append('strong')
16                               .text('Exercise ' + exerciseNumber + ':')
17       d3.select('.grid-container')
18           .append('div')
19           .attr('class', 'answer-grid')
20   }
```

Listing 1.2: New Method

```
    /* 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

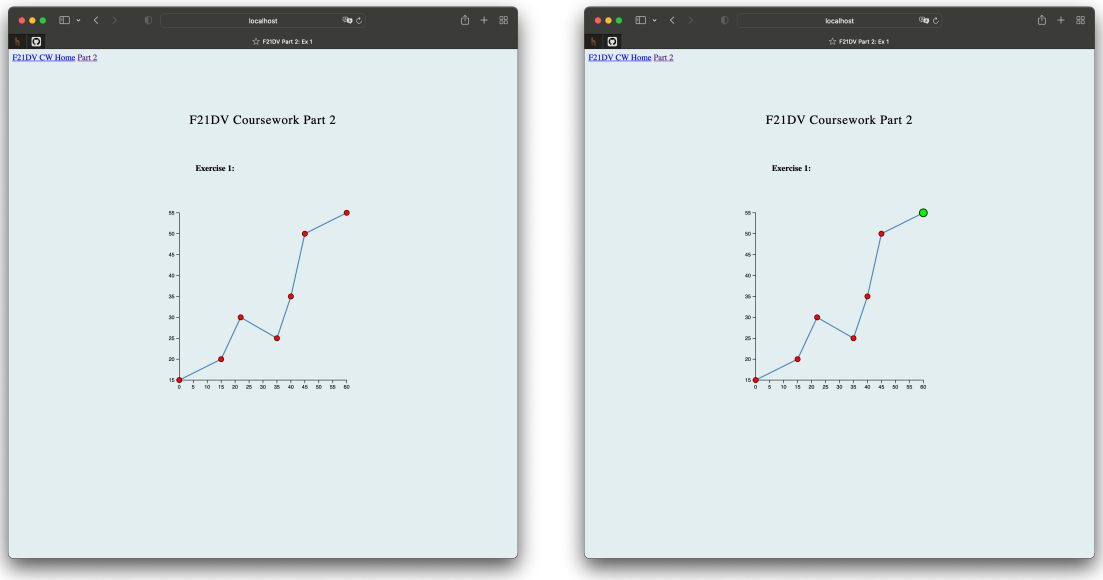

Figure 2.1: Exercise 1

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

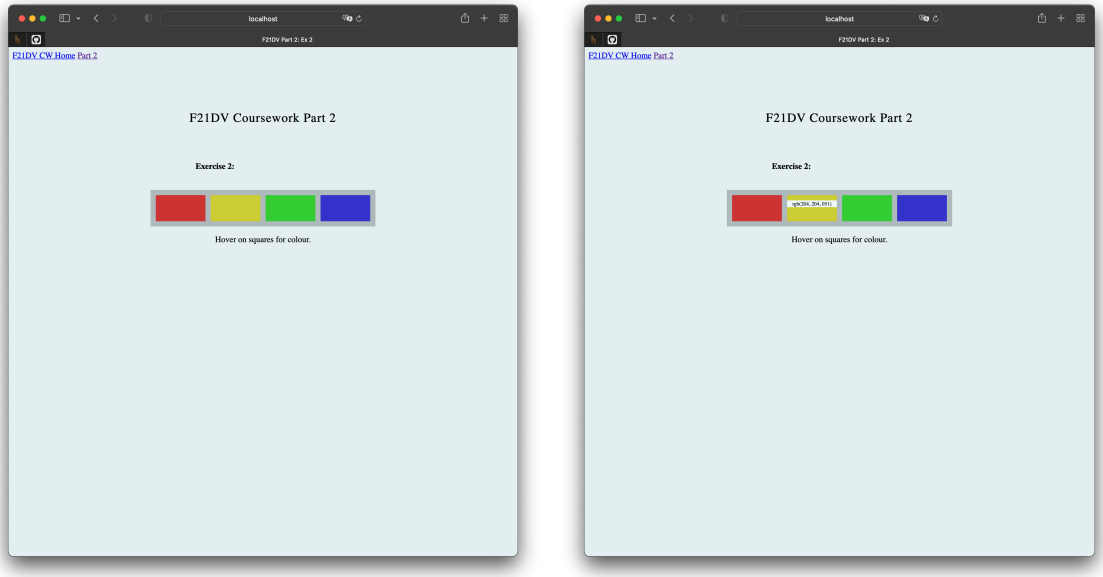

Figure 2.2: Exercise 2

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

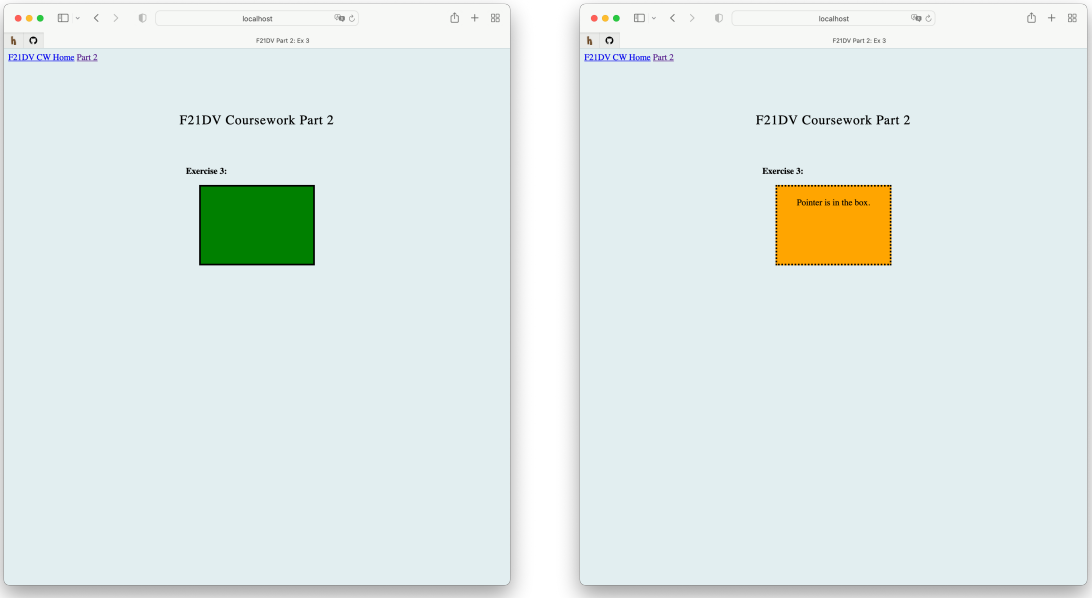

Figure 2.3: Exercise 3

```
1  // .js script for exercise:
2  const ex = 3;
3
4  // Imports of functions.
5  import { createAnswerDiv } from '../functions.js';
6
7  // Creating base <div>s systematically.
8  createAnswerDiv(ex);
9
10 // Adding the original div.
11 d3.select('.grid-container')
12     .append('div')
13         .attr('class', 'answer-grid-small')
14         .style('width', 'auto')
15         .style('height', '100px')
16         .style('background-color', 'green')
17         .style('border-style', 'solid');
18
19 // Change colour, border and text, using .on() mouse over and out.
20 d3.select('.answer-grid-small')
21     .on('mouseover', function() {
22         d3.select(this)
23             .style('background-color', 'orange')
24             .style('border-style', 'dotted')
25             .text('Pointer is in the box.');
26     })
27     .on('mouseout', function() {
28         d3.select(this)
29             .style('background-color', 'green')
30             .style('border-style', 'solid')
31             .text('');
32     });
```

../../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

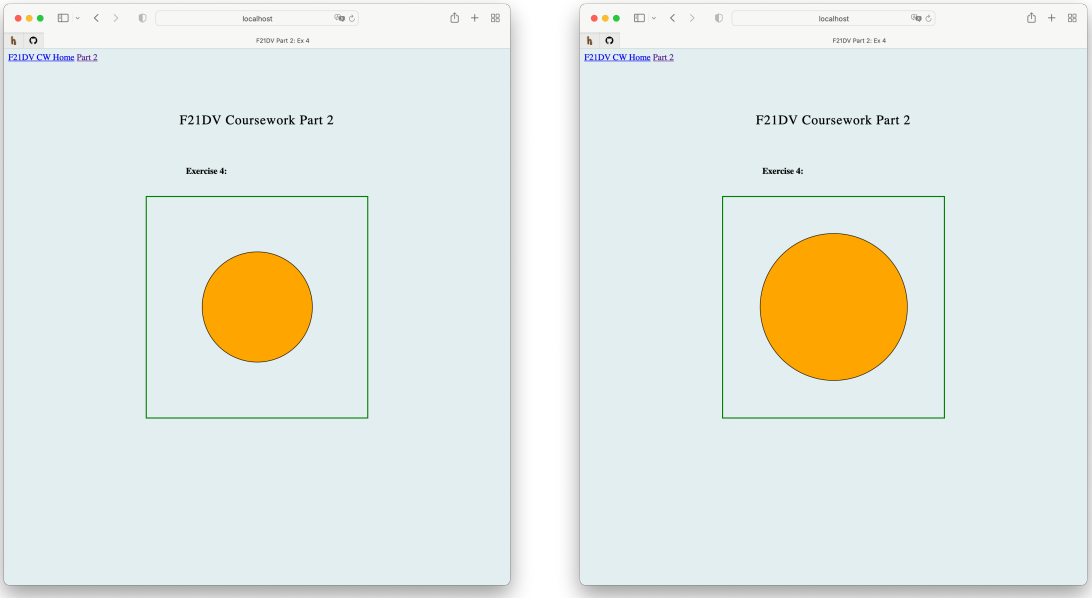

Figure 2.4: Exercise 4

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 5

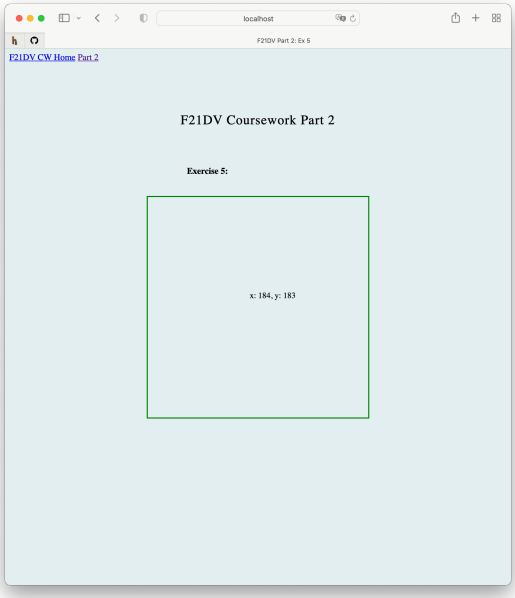

Figure 2.5: Exercise 5

Figure 2.5 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.

2.6 Exercise 6

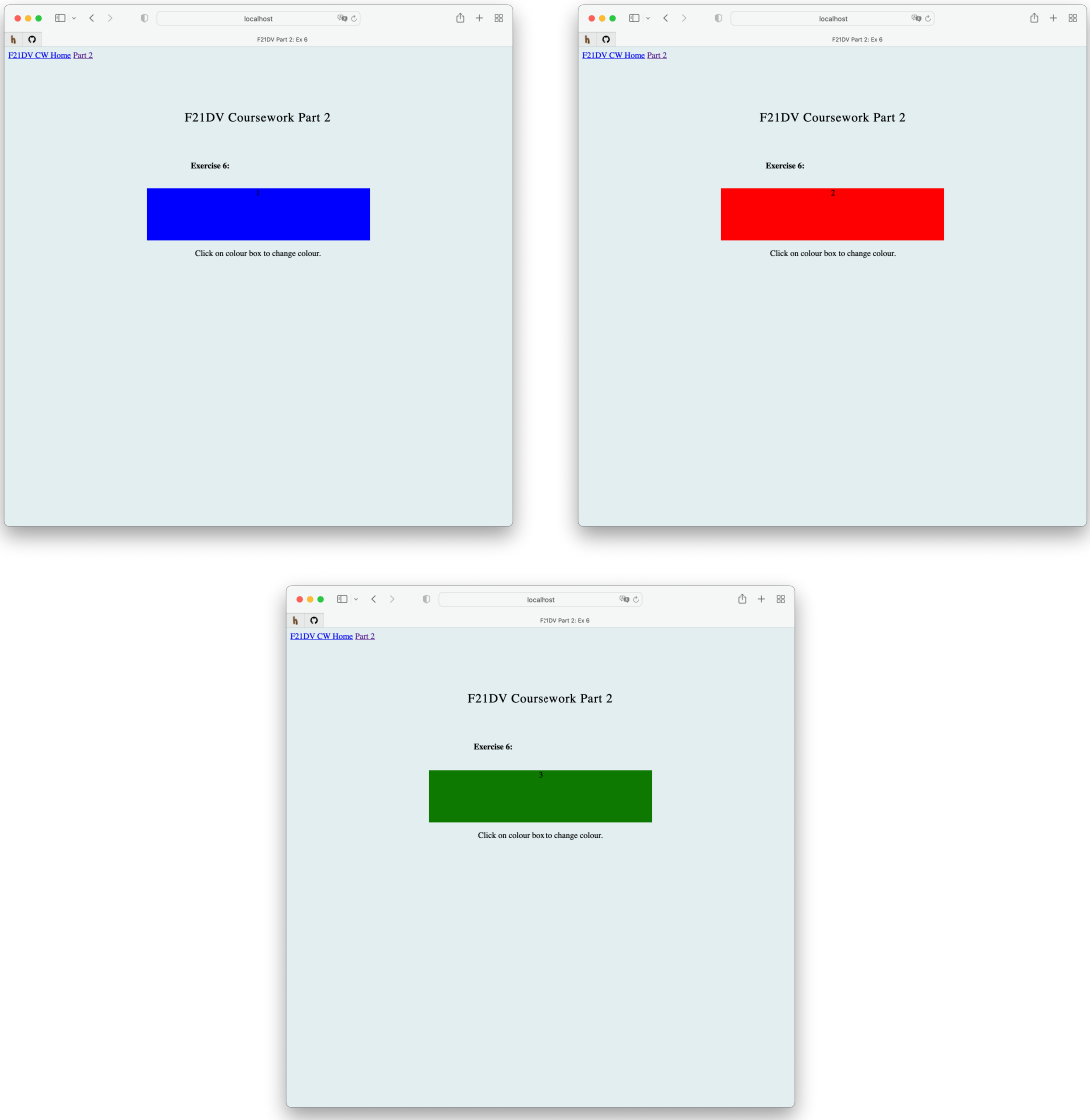

Figure 2.6: Exercise 6

```
1  // .js script for exercise:
2  const ex = 6;
3
4  // Imports of functions.
5  import { createAnswerDiv } from '../functions.js';
6
7  // Creating base <div>s systematically
8  createAnswerDiv(ex);
9
10 // Add original div.
11 d3.select('.answer-grid')
12     .append('div')
13         .style('width', 'auto')
14         .style('height', '100px')
15         .style('background-color', 'blue')
16         .text('1');
17
18 // Add Transition.
19 d3.select('.answer-grid div')
20     .on('click', function() {
```

```
21          d3.select(this)
22              // Ori -> 2(red)
23              .transition()
24                  .duration(1000)
25                  .style('background-color', 'red')
26                  .text('2')
27              // 2(red) -> 3(green)
28              .transition()
29                  .duration(1000)
30                  .style('background-color', 'green')
31                  .text('3')
32              // 3(green) -> Ori
33              .transition()
34                  .duration(1000)
35                  .style('background-color', 'blue')
36                  .text('1')
37      });
38
39  // Add Instructions.
40  d3.select('.answer-grid')
41      .append('p')
42      .text('Click on colour box to change colour.')
```

../../public/js/part2/task6.js

Figure 2.6 shows a coloured div with a number on it. On click, this div will transition to "2" and change to red, and then to "3" and then to green. Finally it will transition back to the original form. From listing above, we could see that this is done by using the transition upon mouse click, then changing the attributes of the div.