F21DV Coursework Part 1 Report

Jonathan Song Yang, Lee

H00255553

Contents

1 Introduction

The entirety of the F21DV four-part coursework is done in a way where it resembels a full static or a `node.js` server-side web application. The goal of this series of coursework is to demonstrate the understanding of `d3.js`.

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

1.1 General Setup

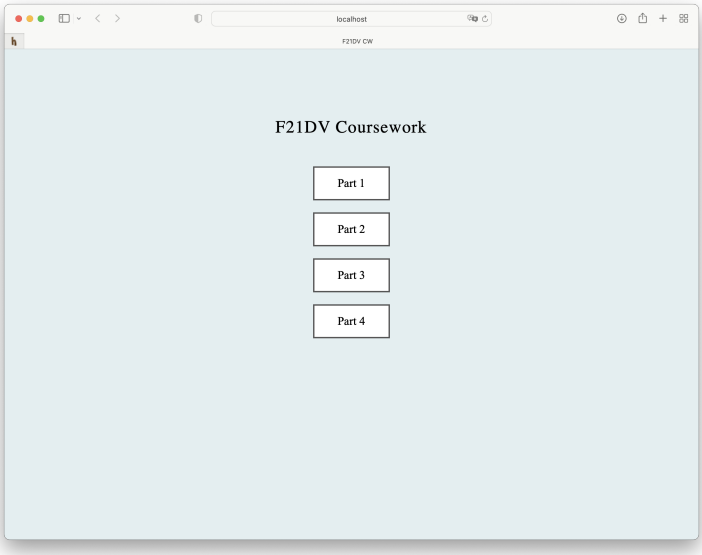

Figure 1.1: Main Index Page

The main index page of the web application would show buttons to access parts 1 to 4, as shown in figure 1.1, and for the case of Lab 1, upon clicking on the Part 1 button, a series of urls linking to different exercises would be shown, as seen in figure 1.1 below.

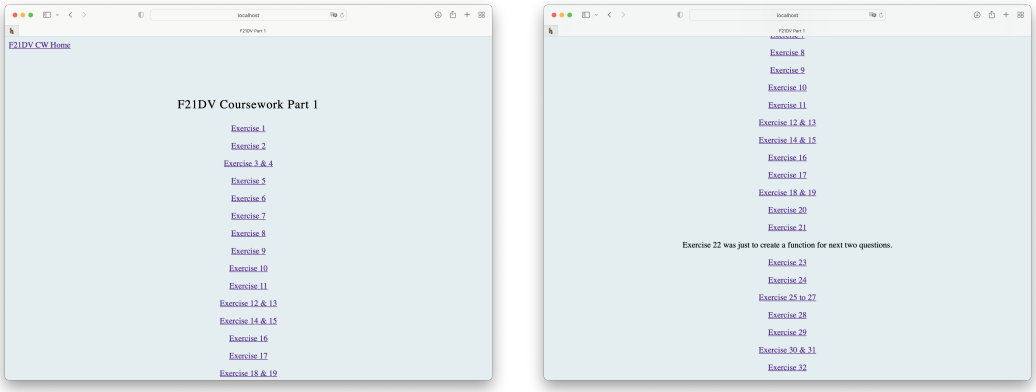

Figure 1.2: Main Index Page

To demonstrate the understanding of `d3.js`, this series of URLs were created systematically using `d3.js`, as shown in listing 1.1 abstact below.

```
1   // Array of numbers [0, 1, 2, ..., 32] defining number of exercises
2   const data = Array.from(Array(33).keys());
3
4   // Creating links to 32 Exercises Systematically.
5   d3.select('body').selectAll('p')
6       .data(data)
7       .enter()
8           // Append a <p> for each exercise and add an <a> for each <p>
9           .append('p').style('text-align', 'center')
10              .attr('class', d => 'task' + d)
11              .append('a')
12                  .attr('href', d => 'task' + d +'.html')
13                  .html(d => 'Exercise ' + d);
```

Listing 1.1: Systematic URL Creation

As the lab exercise would involve certain combinable exercises, such as exercises 30 and 31, I have also created a generalised function to help **combine** and **delete** certain exercises.

```
1   /**
2   * General Merge Function. If its just 2 consecutive exercise, ignore 3rd and 4th
    parameter.
3   * However, if merge is for a range of exercises, spanning more than 2, only enter
     the
4   * first and last exercise number.
5   * @param {*} first first exercise to merge
6   * @param {*} second second exercise to merge. Last exercise if there are more
    than 2 exercises.
7   * @param {*} cond1 used to change exercise <number> to Exercise <number> to <
    number>
8   * @param {*} cond2 used to rename html file to task<first>n<second>.html
9   */
10  function mergeTask(first, second, cond1 = ' \& ', cond2 = 'n') {
11      d3.select('.task' + second).remove();
12      d3.select('.task' + first + ' a').html('Exercise ' + first + cond1 + second)
13                                          .attr('href', d => 'task' + first + cond2
    + second + '.html');
14  }
```

Listing 1.2: Systematic URL Removal

Listing 1.2 shows a function that renames the href of combined exercises, and removes and rename a pre-existing href. For example, we have initially Exercises 30 and 31. The function first removes the

exercise 31's <p>, and then renames the original html href file from task30.html to task30n31.html.

```
1     // Remove task function
2     function removeTask(task, message) {
3         d3.select(`.task${task} a`).remove();
4         d3.select(`.task${task}`).append('p')
5                                  .text(message);
6     }
```

Listing 1.3: Systematic ¡a¿-text Replacement

Listing 1.3 shows a generalised function that replaces the exercises URL with a message string. For example, in exercise 22, we were asked to create a generalised SVG and lines function. This function will now remove the href from the <p> element, and replace it with a message string.

1.2 Individual Exercise HTMLs Setup

Within each exercise's own HTML `body`, they all inherit a generic CSS setting for a div called `.answerCenter` which acts as the center element for the presentation of answers for each exercise. There is also a generic button cssd style called `.buttonOri` and `.button` that handles the CSS transition of all the buttons.

For each exercise, I have also used a generalised function in `functions.js` to create the `<div>`s and buttons so that the styles remain consistent across all exercises.

```
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    }
15
16    /**
17     * Creates a button to execute action for each task.
18     * @param {*} exerciseNumber
19     */
20    export function createButton(exerciseNumber) {
21        d3.select('body')
22            .append('div').attr('class', 'container')
23                .append('div').attr('class', 'center')
24                    .append('button')
25                        .attr('class', 'buttonori')
26                        .text('Click for Task ' + exerciseNumber)
27    }
```

Listing 1.4: Systematic div and button creation

Listing 1.4 shows the systematic approach used to create the answer container, as well as buttons to execute exercises actions. However, not all exercise would use the button, since not all exercise is in need of an action.

2 Exercises

2.1 Exercise 1

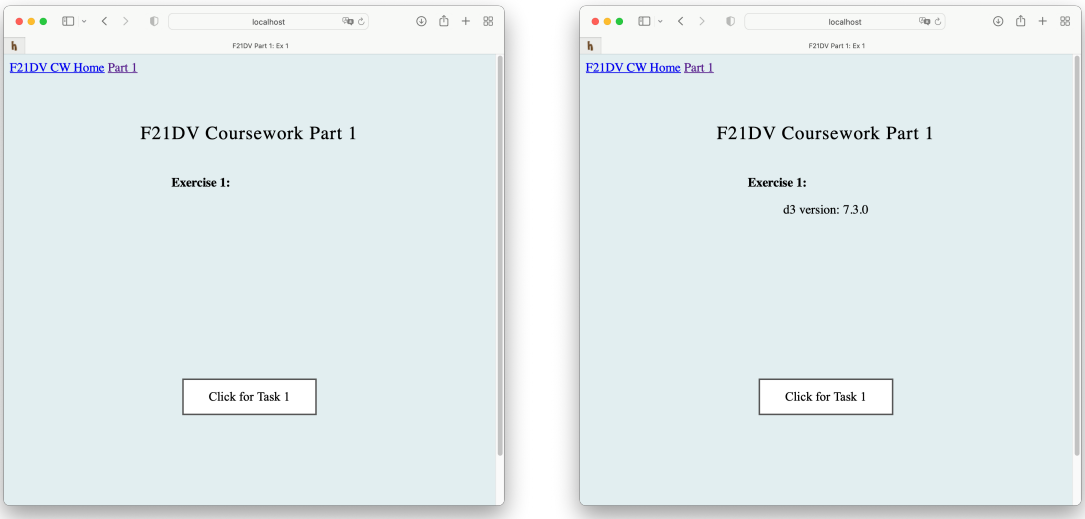

Figure 2.1: Exercise 1

Figure 2.1 shows the outcome after clicking on the action button. The answer container space would print the d3 version number upon clicking the task button.

```
1  // js script for part 1 Exercise:
2  const ex = 1;
3
4  // Create Divs and button systematically using a genral function.
5  import {createDiv, createButton} from './functions.js';
6  createDiv(ex);
7
8  // Create empty <p> to print version.
9  d3.select('.answerCenter')
10     .append('p')
11         .attr('id', 'task' + ex)
12         .attr('position', 'absolute')
13         .style('text-align', 'center');
14
15 // Button for task action.
16 createButton(ex);
17
18 // Button action: shows d3 version in <p> field.
19 d3.select('.buttonori').on('click', function(){
20     d3.select('#task' + ex).text('d3 version: ' + d3.version);
21 });
```

../../public/js/part1/task1.js

Upon clicking the button, using d3, the button action replaces the empty <p> with "d3 version: 7.3.0". For the remaining exercises, the button action would do a similar thing as well. It would select the desired object for modification, and through a button on-click event listener, modify the selected object.

2.2 Exercise 2

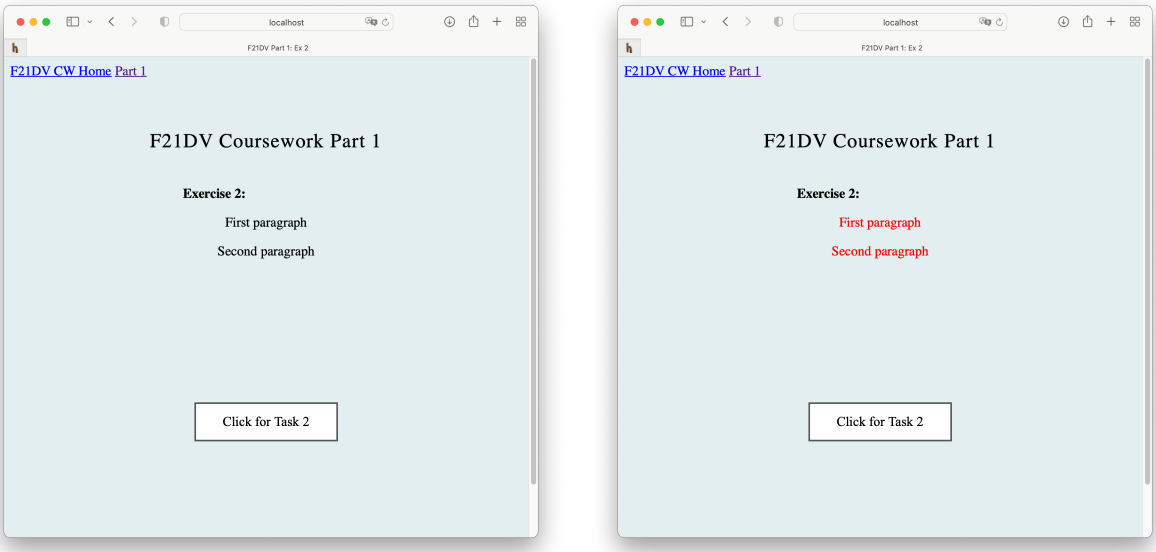

Figure 2.2: Exercise 2

Figure 2.2 shows that upon clicking the button, d3 would change the colours of the selected `<p>`s to red.

2.3 Exercise 3 & 4

Exercises 3 & 4 are one of the exercises where I have combined into one.

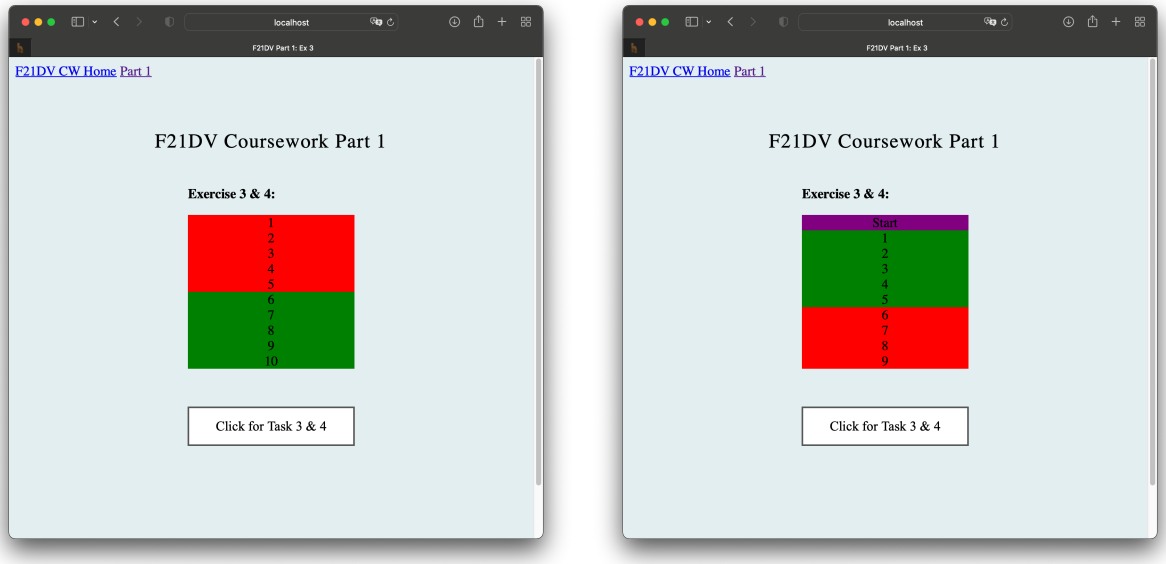

Figure 2.3: Exercise 3

Exercise 3 was to populate 10 div elements with numbers from 1 to 10, and to colour them according to their values. Task four asks then, to replace the first value, "1", with the text "start". However, I went ahead and implemented the change for the remaining divs as well, changing their values so that the divs would show start, 1, 2, ..., 9, whilst keeping the colour requirements. Upon clicking the button, the state of the div's would return to the original state again. Button is clickable multiple times.