Elliot Whitehouse Lab 1: Demonstrated: 04/02/2022

F21DV Lab 1

Demonstrated to: Shuangjiang Xue

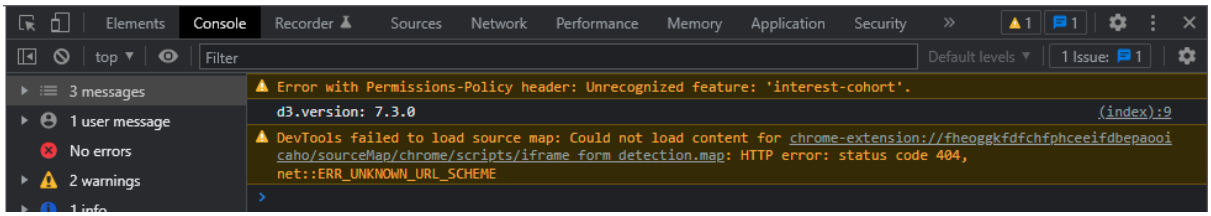
Progress:

Set up GitHub repository configured to use GitHub Pages in order to host the lab exercises directly on the repository.

Part 2 D3 Setup

Exercise 1.

The version number displayed in in the console output window is “d3.version: 7.3.0”

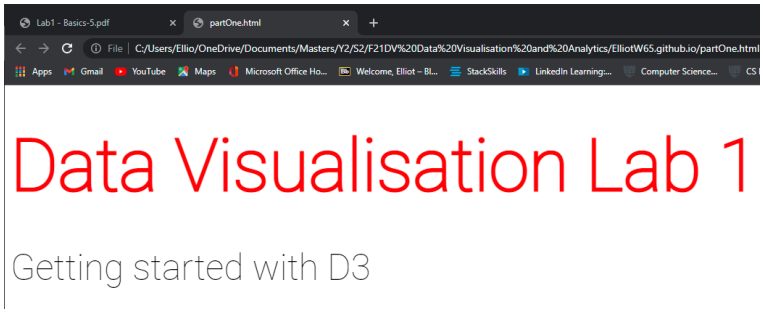


Exercise 2.

Changing different properties of the paragraphs using different selectors:

- select(“p”)
- select(“p1”) - using ID
- selectAll(“p”) - style all paragraphs

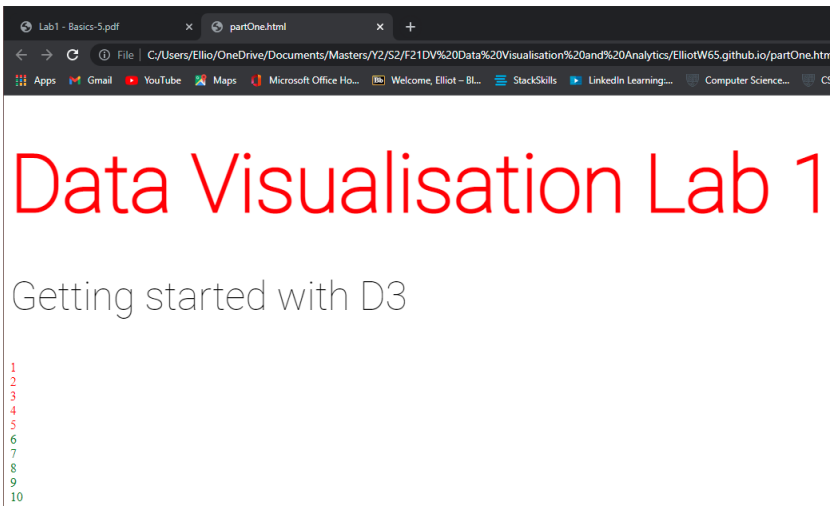
```
<p id="p1">First paragraph</p>
<p id="p2">Second paragraph</p>

<script>
    // Exercise 1: Display version number in console
    console.log('d3.version:', d3.version);

    // Exercise 2
    // Selecting the first paragraph
    d3.select("p").style("color", "red");
    d3.select("p").style("font-size", "100px");
    d3.select("p").style("line-height", "20px");
    d3.select("p").text("Data Visualisation Lab 1");

    // Selecting first paragraph by id
    d3.select("#p1").style("font-weight", "bold");

    // Add style to all paragraphs
    d3.selectAll("p").style("font-family", "Roboto");

    // Style paragraph 2 using id #p2
    d3.select("#p2").style("font-size", "50px");
    d3.select("#p2").text("Getting started with D3");

    // Add class to all "p"s on the fly so they can be referenced by class
    d3.selectAll("p").classed("paragraph", true);
```



Exercise 3.

For this exercise I created a for loop that would iterate 10 times. Inside this loop I created a placeholder variable that would take the value of the new div during each iteration. There is then an if condition that checks if the iterator “i” is below 6; if this is true the div is formatted accordingly. Different formatting is then applied if the condition (i < 6) is not true. I also give each div a class of either “firstFive” or “secondFive” depending on the iteration they are created in so that they can be referenced later if need be.

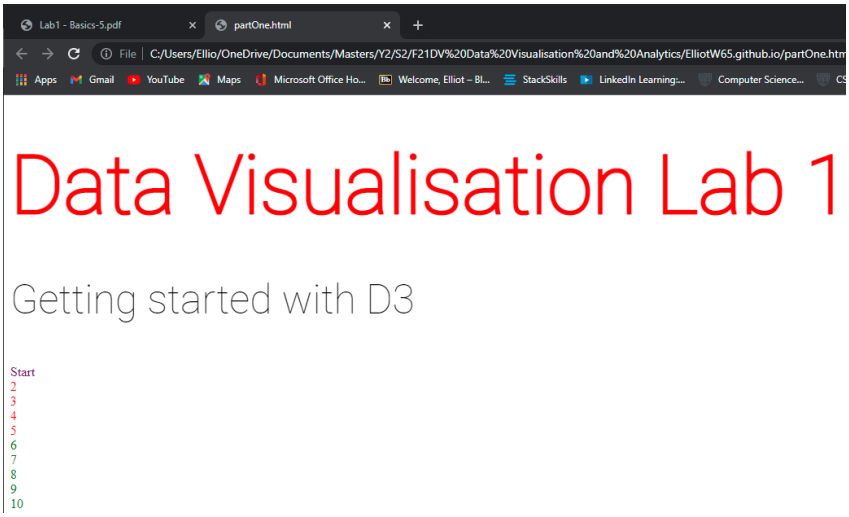
```
// Exercise 3
for (let i = 1; i < 11; i++) {
    let newDiv = d3.select("body").append("div")
    if(i < 6) {
        newDiv.classed("firstFive", true);
        newDiv.text(i);
        newDiv.style("color", "red");
    }
    else {
        newDiv.classed("secondFive", true);
        newDiv.text(i);
        newDiv.style("color", "green");
    }
}
```



Exercise 4.

For this exercise I opted to use just the select(“div”) selector as that will select the first div in the list as per the exercise instructions.

```
// Exercise 4
d3.select("div").text("Start");
d3.select("div").style("color", "purple");
```



Exercise 5.

```
// Exercise 5
d3.select("body").append("div").text("Hello World!").style("color", "green");
```

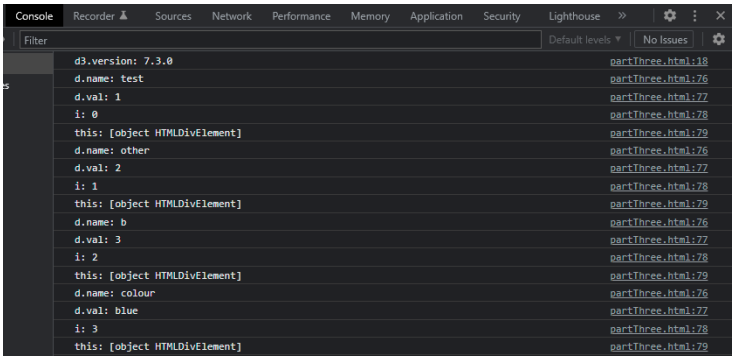
Start
2
3
4
5
6
7
8
9
10
Hello World!

Part 3. Data

Exercise 6.

I chose to make the value of the new 'colour' variable to be 'blue' and printed this value in the console along with the other values.

```
let otherdata = [ {name:'test', val:1},
                  {name:'other', val:2},
                  {name:'b', val:3},
                  {name:'colour', val:'blue'} ];

let paragraph = d3.select("body")
                  .selectAll("div")
                  .data(otherdata)
                  .text(function (d, i) {
                      console.log("d.name: " + d.name);
                      console.log("d.val: " + d.val);
                      console.log("i: " + i);
                      console.log("this: " + this);
                      return 'cont:' + d.name; // return value is used to set the 'text'
                  });
```



Exercise 7.

I carried out this exercise on the divs created in Part 2, replacing the text in the divs with new values where appropriate.

```
let num = [10, 50, 100, 200];
let paragraph = d3.select("body")
                  .selectAll("div")
                  .data(num)
                  .text(function (d, i) {
                      return 'cont:' + d; // return value is used to set the 'text'
                  })
                  .style("color", function(d, i) {
                      if ( d >= 50 && d <=100 ) {
                          return "red";
                      }
                      else {
                          return "yellow";
                      }
                      return 'blue';
                  });
```

cont:10
cont:50
cont:100
cont:200
5
6
7
8
9
10
Hello World!

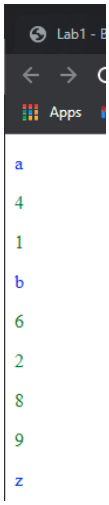
Elliot Whitehouse Lab 1: Demonstrated: 04/02/2022

Part 4. Data Binding

Exercise 8.

When starting Part 4 I chose to start a new file to keep each Part logically separate. I completed this exercise by using the typeof operator to check if a value is a number or string.

```
<script>
    // Exercise 8
    var myData = ['a', 4, 1, 'b', 6, 2, 8, 9, 'z'];

    var p = d3.select("body")
        .selectAll("span")
        .data(myData)
        .enter()
        .append('p')
        .text(function (d, i) {
            return d;
        })
        .style('color', function(d, i) {
            console.log(typeof d);
            if(typeof d == 'number') {
                return 'green';
            }
            else if(typeof d == 'string') {
                return 'blue';
            }
            else {
                return 'black';
            }
        });


</script>
```

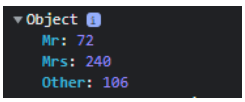


Part 5. Loading Data

Exercise 9.

For this exercise I loaded the data in using d3.csv and then passed the result to a .then(function(data)) to handle the array of objects returned. I then created an object to contain the counts of each kind of title (“Mr”, “Mrs” and “Other”), initialising them all to 0. I looped through the data and checked each data.Name and used String.includes(substring) to match the appropriate title. Each time there was a match (or none for “Other”) I incremented the value of the respective object property by 1. I then printed out the result to the console.

```
const titles = {
    Mr : 0,
    Mrs : 0,
    Other : 0
}

const sex = {
    male : 0,
    female : 0
}

for(i of data) {

    if(i.Name.includes("Mrs")) {
        titles.Mr ++;
    }
    else if(i.Name.includes("Mr")) {
        titles.Mrs ++;
    }
    else {
        titles.Other ++;
    }

    if(i.Sex == "female") {
        sex.female++;
    }
    else if(i.Sex == "male") {
        sex.male++;
    }
}

console.log(titles);
```



Exercise 10.

To print out this information I used the same method as in the above exercise however instead of printing the result to the console I appended “p”s to the body with text displaying the result.

```
d3.select("body")
.selectAll("p")
.data(Object.keys(ages))
.enter()
.append("p")
.append("text")
.text(function(d,i) {
    return "In the age range with an upper limit of: "
    + Object.keys(ages)[i]
    + " there were " + ages[d] + " with heart failure."
})
```

In the age range with an upper limit of: 31 there were 0 with heart failure.

In the age range with an upper limit of: 41 there were 7 with heart failure.

In the age range with an upper limit of: 61 there were 164 with heart failure.

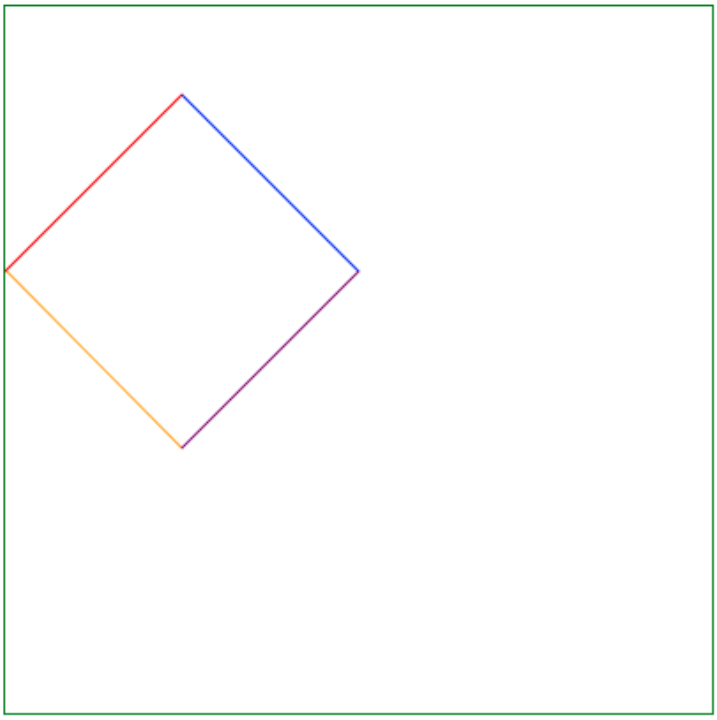
In the age range with an upper limit of: 101 there were 0 with heart failure.

Part 6. SVG

Exercise 11.

Initially I found it confusing to align the x1,x2 and y1 and y2 coordinates by after moving each around and c=seeing the result I understood how to position the coordinates in order to create the square.

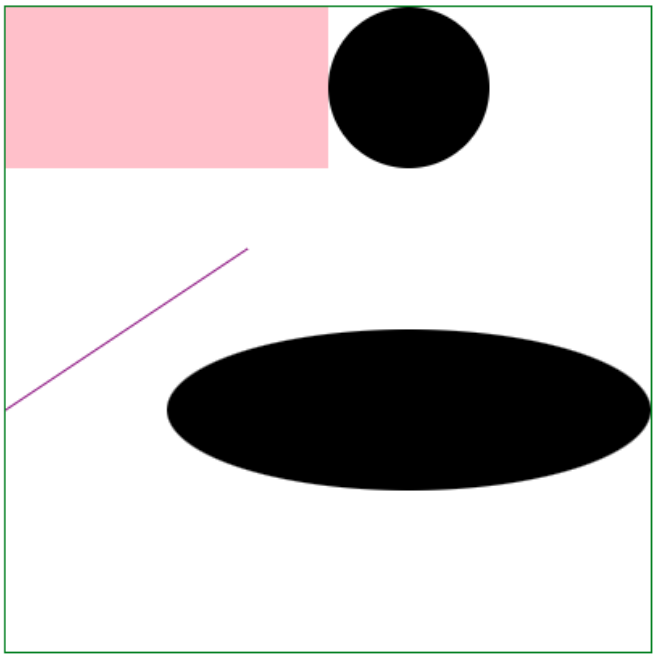
```
<script>
    // Exercise 11
    //Create SVG element
    var svg = d3.select("body")
        .append("svg")
        .attr("width", 400)
        .attr("height", 400)
        .style("border", '1px solid green');
    //Create line element inside SVG
    svg.append("line")
        .attr("x1", 100)
        .attr("x2", 200)
        .attr("y1", 50)
        .attr("y2", 150)
        .attr("stroke", "blue")
    svg.append("line")
        .attr("x1", 100)
        .attr("x2", 0)
        .attr("y1",50)
        .attr("y2",150)
        .attr("stroke", "red")
    svg.append("line")
        .attr("x1", 1)
        .attr("x2", 100)
        .attr("y1",150)
        .attr("y2",250)
        .attr("stroke", "orange")
    svg.append("line")
        .attr("x1", 100)
        .attr("x2", 200)
        .attr("y1",250)
        .attr("y2",150)
        .attr("stroke", "purple")
</script>
```



Exercise 12.

At first I had trouble getting d3 to read in the data from a csv locally as I was running the html files off my local machine, however after doing some research I found out that Chrome could not access these local files and so instead had to run a local server using Python. This allowed Chrome to view the local csv and subsequently read in the data.

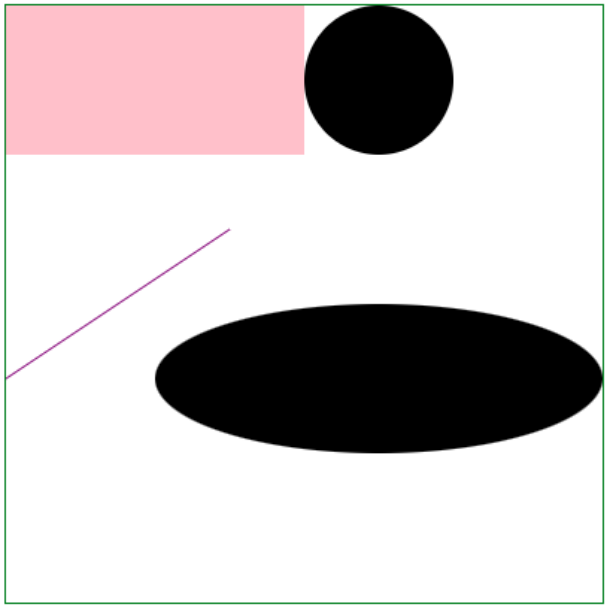
```
// Exercise 12
d3.csv(file, function(data) {

    if(data.Shape == "rect") {
        newsvg.append(data.Shape)
        .attr("x", data.x)
        .attr("y", data.y)
        .attr("width", data.Width)
        .attr("height", data.Height)
        .attr("fill", data.fill);
    }
    else if(data.Shape == "circle") {
        console.log("Circle");
        newsvg.append(data.Shape)
        .attr("cx", data.x)
        .attr("cy", data.y)
        .attr("r", data.rx);
    }
    else if(data.Shape == "ellipse") {
        newsvg.append(data.Shape)
        .attr("cx", data.x)
        .attr("cy", data.y)
        .attr("rx", data.rx)
        .attr("ry", data.ry);
    }
    else if(data.Shape == "line") {
        newsvg.append(data.Shape)
        .attr("x1", data.x)
        .attr("x2", data.y)
        .attr("y1",data.rx)
        .attr("y2",data.ry)
        .attr("stroke", data.fill)
    }


});
```



Exercise 13.

For this exercise I used the same code structure as before by adding an SVG to the screen and then using a foreach loop to loop through the data and appending each shape with the necessary attributes. However, this time I added the append and attribute function calls inside a join and then enter function. I also added an exit in the join which print and exit message to the console and removes the shape.

```
var exThirteen = d3.select("body")
    .append("svg")
    .attr("width", 400)
    .attr("height", 400)
    .style("border", '1px solid green');

// Exercise 13
d3.csv(file).then(function(data) {
    data.forEach(element => {
        console.log(element.Shape);
        if(element.Shape == "rect") {
            exThirteen.selectAll(element.Shape)
            .data(data)
            .join(
                enter => {
                    enter.append(element.Shape)
                    .attr("x", element.x)
                    .attr("y", element.y)
                    .attr("width", element.Width)
                    .attr("height", element.Height)
                    .attr("fill", element.fill);
                },
                exit => {
                    console.log("Exit called");
                    exit.remove();
                }
            )
        }
```



Part 7. Bar Chart

Exercise 14 & 15.

In order to create the bar chart to show the age distribution of people with heart failure I ran a function on the data to produce and object with the counts of instances of people who fell into each age range. To do this I used a foreach loop on the array of objects that is the initial dataset and then looped through an object with the age buckets in with all the ages initially set to 0. In each iteration of the loop there is a condition to check if the current persons age is less then the object key which would indicate it falls into that bucket. The value of that key in the object is then incremented by one. I then used this new “ages” object as the base data for the bar chart.

```
d3.csv(heartfailurecsv).then(function(data) {
    console.log(data);

    const ages = {
        31: 0,
        41: 0,
        61: 0,
        101: 0
    };

    data.forEach(element => {
        for(const i in ages) {
            if(element.age < i) {
                ages[i] += 1;
            }
        };
    });

    var graph = d3.select("body")
        .append("svg")
        .attr("width", 1000)
        .attr("height", barHeight * 4);

    var bar = graph.selectAll("g")
        .data(Object.keys(ages))
        .enter()
        .append("g")
        .attr("transform", function(d, i) {
            return "translate(0," + i * barHeight + ")";
    });
    bar.append("rect")
        .attr("width", function(d,i) {
            console.log(ages[d] * scaleFactor);
            return ages[d] * scaleFactor;
        })
        .attr("fill", function(d) {
            if (ages[d] > 99) {
                return "red";
            } else if (ages[d] > 50) {
                return "orange";
            }
                return "green";

        })
        .attr("height", barHeight - 1);

    bar.append("text")
        .attr("x", function(d) { return (ages[d] * scaleFactor)})
        .attr("y", barHeight / 2)
        .attr("dy", ".35em")
        .attr("text-anchor", "middle")
        .text(function(d) { console.log(ages[d]); return ages[d]; });

});
```

Exercise 15

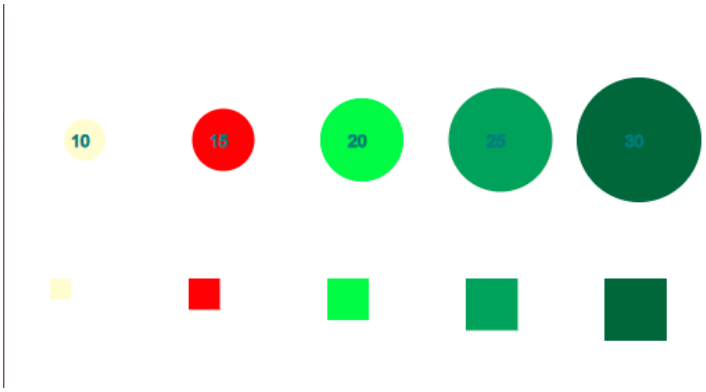
Result

Part 8. Circle Chart

Exercise 16.

In order to complete this exercise I appended rectangles to the “g”s created for the circles, however I had an issue adding text to the rectangles which I believe is because the text is only added once to the “g” which meant it went to the circle rather than both the circle and the rectangle. I struggled to find a solution for this.

```
<script>
    const width = 500;
    const height = 500;
    const data = [10, 15, 20, 25, 30];
    // Note different valid ways of specifying color
    const colors = ['#ffffcc','red','rgb(0,255,0)','#31a354','#006837'];
    const svg = d3.select("body")
    .append("svg")
    .attr("width", width)
    .attr("height", height);
    const g = svg.selectAll("g")
    .data(data)
    .enter()
    .append("g")
    .attr("transform", function(d, i) {
        return "translate(0,0)";
    })
    g.append("circle")
    .attr("cx", function(d, i) {
        return i*100 + 50;
    })
    .attr("cy", function(d, i) {
        return 100;
    })
    .attr("r", function(d) {
        return d*1.5;
    })
    .attr("fill", function(d, i){
        return colors[i];
    })

    g.append("rect")
    .attr("x", function(d, i) {
        return i*100 + 25;
    })
    .attr("y", function(d, i) {
        return 200;
    })
    .attr("width", function(d) {
        return d*1.5;
    })
    .attr("height", function(d) {
        return d*1.5;
    })
    .attr("fill", function(d, i){
        return colors[i];
    })

    g.append("text")
    .attr("x", function(d, i) {
        console.log(d);
        return i * 100 + 40;
    })
    .attr("y", 105)
    .attr("stroke", "teal")
    .attr("font-size", "12px")
    .attr("font-family", "sans-serif")
    .text(function(d) {
        return d;
    });
```

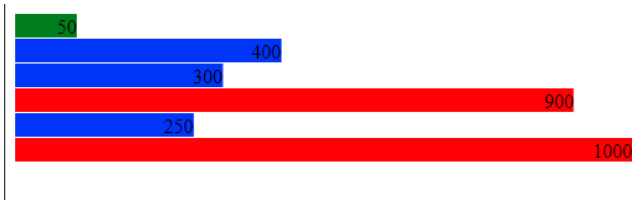


Part 9. Scales, Domain, Range

Exercise 17.

For this exercise I created a function in the “fill” attribute when appending the “rect” to the group. Within the function I created an if condition that checked the value of the parameter d passed to the function; if it is above 500 the function returned “red”, else if it was below 100 it would return “green” else it would return “blue”.

```
.attr("fill", function(d) {

    if (d > 500) {
        return "red";
    } else if (d < 100) {
        return "green";
    }
    else {
        return "blue"
    }

})
```



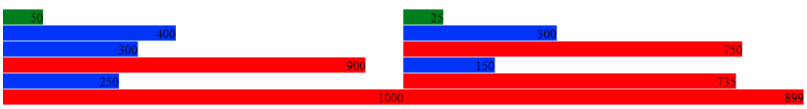
Exercise 18.

For this exercise I wrapped the code in a .then() function following a d3.csv function. This allowed me to specify an external file in the d3.csv function which would subsequently pass the parsed data into the .then() function where the code to create the bar chart is located.

Exercise 19.

After putting the code into a d3.csv function, I put that code into a function that takes a file name as a parameter which is passed to the d3.csv function which can then retrieve and process the data. This enabled bar charts to be shown multiple times on screen when run multiple times.

```
function barChart(fileName) {

    const data = [50, 400, 300, 900, 250, 1000]
    const width = 500;
    const barHeight = 20;
    const margin = 1;

    // Exercise 18
    d3.csv(fileName).then(function(data) {
        console.log(data);
```



Part 10. Axis

Exercise 20.

To create the axes on the top and bottom, I created 2 new variables using d3.axisTop() and d3.axisRight() which I initiated with the same “xscale” and “yscale” attributes as the existing x and y axes. I then appended these to the “g” element and adjusted the “translate” parameters in order to move them so a rectangle was created by the axes. To colour these axes, I added a class “altAxis” to them when appending them to the “g” so that I could references them using a selectAll(“.altAxis”) and change the stroke colour to blue.

Exercise 21.

I did this exercise in the “partNine.html” file with the bar chart exercises. To do this I changed the range of the x axis to [0, d3.max(dataArr)] which is the largest value in the data and changed the domain to [0, scale(d3.max(dataArr)] so that the axis was scaled by the same value as the data and so they aligned perfectly.

Part 12. Line Chart

Exercise 22.

In this exercise I put the section of code that appended the “path” to the SVG which in turn created the line into a function that accepted data and a line colour as parameters. Initially I had the code that created the axes in the function too, however when adding the cosine wave this created a second axis on top of the first and so I moved the axis code outside of the function so it was only added once.

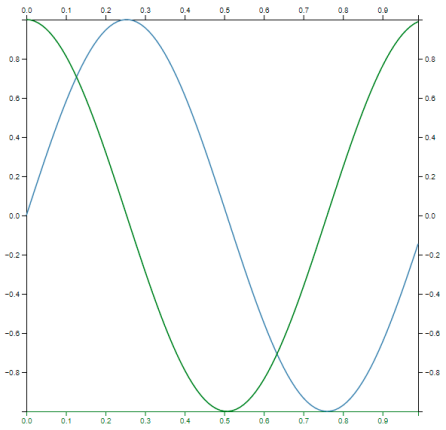
```
// Exercise 22
function waveGenerator(data, colour) {

    // Add the line
    svg.append("path")
    .datum(data)
    .attr("fill", "none")
    .attr("stroke", colour)
    .attr("stroke-width", 1.5)
    .attr("d", d3.line()
    .x(function(d) { return x(d.x) })
    .y(function(d) { return y(d.y) })
    );

}
```

Exercise 23.

To add the cosine wave I created a csv in the same format at the sine wave data but instead with cosine values for a wave of the same scale. I then added a d3.csv function to read in this data and called the “waveGenerator” function with the resulting data that was parsed in.

Exercise 24.

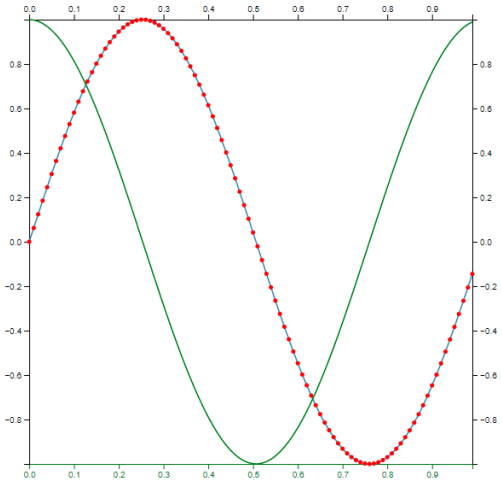
I found that after the data was parsed from an external file I needed to do a type conversion as the numbers in the csv were read in as strings. To do this I iterated through the array of objects returned by the d3.csv function and applied the + operator to each instance which worked.



Part 13. Markers

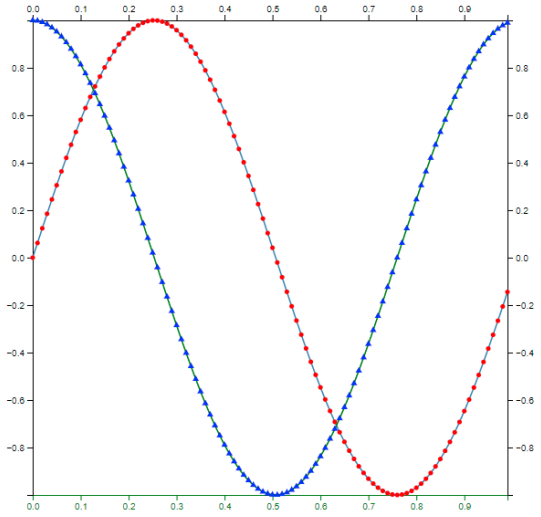
Exercise 25.

To add the circle points to the sine wave I used svg.selectAll(“dot”) to select the datapoint and then append “circle” after binding it to the data and calling enter(). I then set the cx and cy parameters to place the circle by calling the “x” function which was made from the d3.scaleLinear() function and then set the radius and filled the circle red.



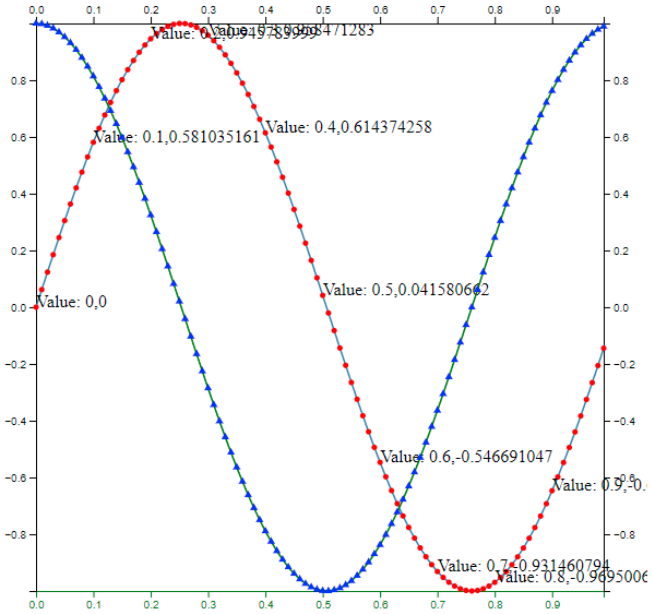
Exercise 26.

It took me a while to find a solution as to how to add triangles for the data points however I found a solution by attribute “d”, d3.symbol().type(d3.symbolTraingle).size(), to the path after selecting svg.selectAll(“dot”). I then set the transform-translate parameters to the centre of the data points using the “x” and “y” scaleLinear() functions created earlier.



Exercise 27.

When adding text to the sine wave points I used the .filter() function when appending “text” after calling d3.selectAll(“dot”). Within the .filter, I used an function to return true if the function iterator i is divisible by 10, which is determined by using the modulo operator i % 10 == 0. This added the value as text of every 10th data point.

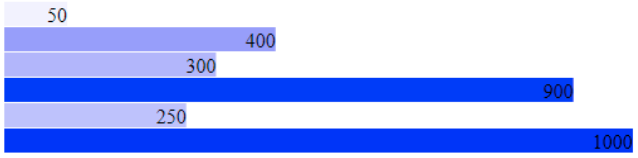


Part 14. Colours

Exercise 28.

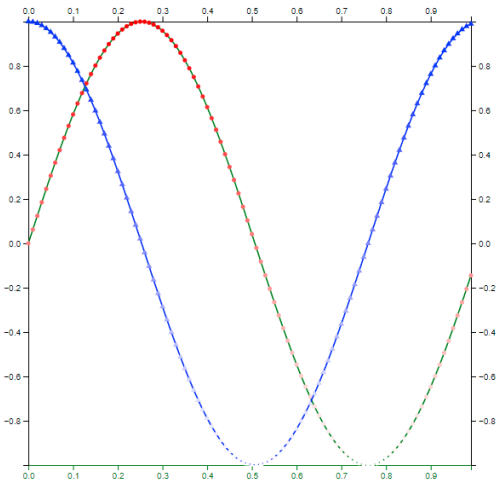
To create a dynamic colour range for the bar chart I used the scaleLinear().domain().range([“white”,”blue”]) function which I called inside a function inside the attr.(“fill”) function when adding the “g”s to the svg.

```
.attr("fill", function(d) {
    console.log(colour(d));
    return colour(d);
})
```



Exercise 29.

To add colour to the wave graph I used the same function as the exercise above, however I added it to the datapoint symbols using the same method as above and setting the domain to [-1, 1].



Part 15. Pie Chart

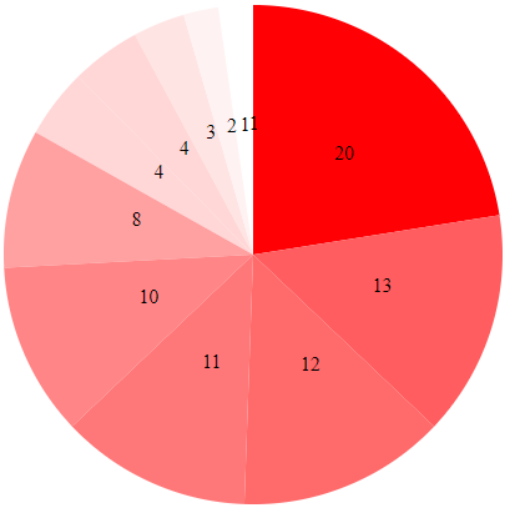
Exercise 30.

For this exercise I just extended the data array with some more numbers to make the total number of values 12.

Exercise 31.

To add the data label text to each arc I used the arc.select("g") method. I used this instead of arc.selectAll("g") as I found when I did this it was adding the data label for each arc for each "g", which meant there were 12x12 text labels written on top of each other. So by using only select("g") it only added the labels to the "g" that contained the arcs and therefore added them only once. To centre the label in the arc, I used arc.centroid(d,i) inside a transform-translate function which placed the label in the centre of the arc.

```
.attr("transform", function(d, i) { return "translate(" + arc.centroid(d, i) + ")"; })
```



Part 16. SVG Graphics

Exercise 32

For this exercise I added the image to the bar chart in the "partFourteen.html" file using svg.append("svg:image") and setting the height and width to that of the SVG.

```
// Exercise 32
svg.append("svg:image")
.attr("xlink:href", "fruit-g6eb75822c_1920.png")
.attr("width", 500)
.attr("height", 120)
.attr("x", 0)
.attr("y",0);
```