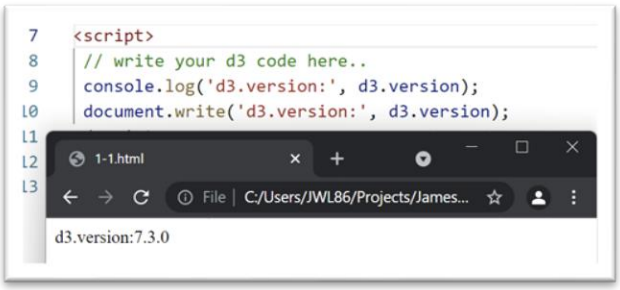
# Yishan Wang  F20DV Lab 1

4 February 2022 demonstrated to Professor Benjamin Kenwright

All the exercise can be viewed on this link.
https://jamesw99.github.io/

1:

The d3 version been print on console and page




2: Change other style properties of the paragraph tag (e.g., font-size, line-height, font-family, contents, …)

Changed, see source code

I use <p style> to modify the DOM()

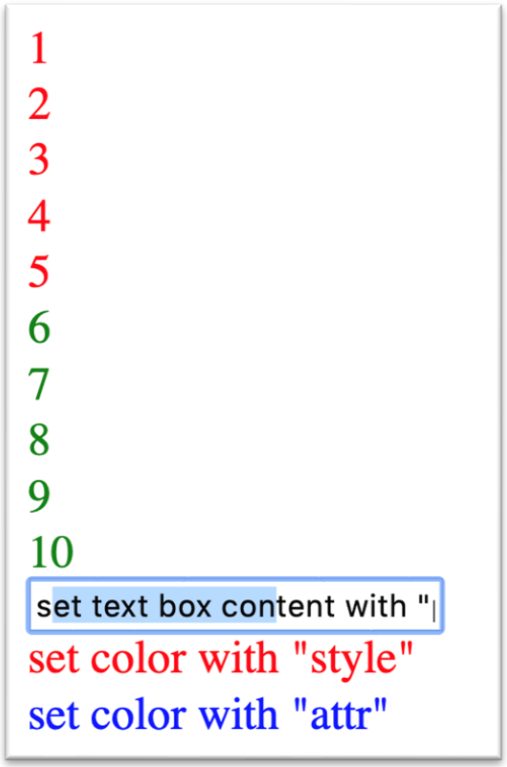
```
<p style="color:red;">red</p>
<p style="font-size:50px;color:red;">font-size:50px</p>
<p style="font-size:30px;color:red;line-height: 100px;">Red;font-size:30px;line-height: 100px</p>
<p style="font-size:30px;color:red;line-height: 100px;font-family:'Franklin Gothic Medium', 'Arial Narrow';"> ... & font-family:'Franklin Gothic Medium', 'Arial Narrow';</p>
<p style="font-size:30px;color:red;line-height: 100px;"> ... & contents</p>
<span style="color: aqua;">This is span.</span>
<script>
	let newspan = d3.select("body").append('span');
	newspan.text('this is new span.');
</script>
```

3:

In exercise 3, I use a for loop to create , the first 5  colour be set to red by style() and last 5  been set to green by attr()

At last, I try the property with a text box.

```
for (var i = 1; i <= 10; ++i) {
    var div = d3.select('body').append('div');
    div.text(i);
    if (i <= 5)
        div.style('color', 'red');
    else
        div.attr('style', 'color:green');
}

var textbox = d3.select('body').append('input');
textbox.property('value', 'set text box content with "property"');

var red = d3.select('body').append('div');
red.text('set color with "style"').style('color', 'red');

var blue = d3.select('body').append('div');
blue.text('set color with "attr"').attr('style', 'color:blue');
```



4:

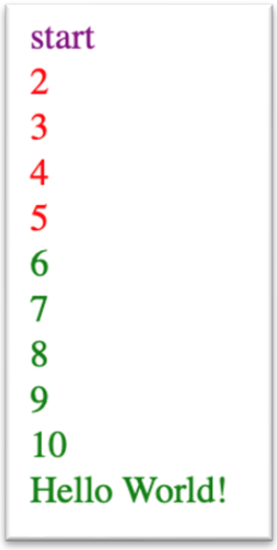
In the same script block, I use ‘d3.select’ to modify first  colour to purple and content to ‘start’.

```
d3.select('div').style('color', 'purple');
d3.select('div').text('start');
```


5:

When want to add something to main body, we can use d3 to select body block and append a text block, such as div, text and p etc. after that can set text and text style in block.

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



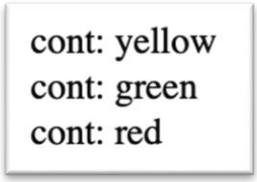

6:

I add a color as key for ‘otherdata’ list element, and add an extra line for print console:

“{name:'test', val:10, color:'yellow'}”

“console.log("color: " + d.color);”

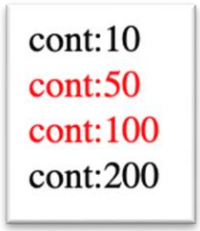
```
let otherdata = [
    {name:'test', val:10, color:'yellow'},
    {name:'other', val:50, color:'green'},
    {name:'full', val:200, color:'red'}
];

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("color: " + d.color);
        console.log("this: " + this);
        return 'cont: ' + d.color; // return value is used to set the 'text'
    });
```



7:

In the demo, all number will be set to red is larger or equal then 100, so I modify the condition to smaller or equal than 100 and if true, going to second judge, if larger or equal then 50, will return red, else will end of the judgment. The default color is black.

```
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 <= 100 ) {
            if (d >= 50){
                return "red";
            }
        }
    });
```



8:

Based on the demo code, I add an style function, if typeof d === "number" it will return green, if type of d equal string will return blue.

```
var myData = ['a', 4, 1, 'b', 6, 2, 8, 9, 'z' ];

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


9:

In this exercise, I write 2 loops, one for match title, other one for match gender

First loop supports all already known title, include Mr, Mrs, Miss, Ms, Master, Col, Dona, Dr, Rev.

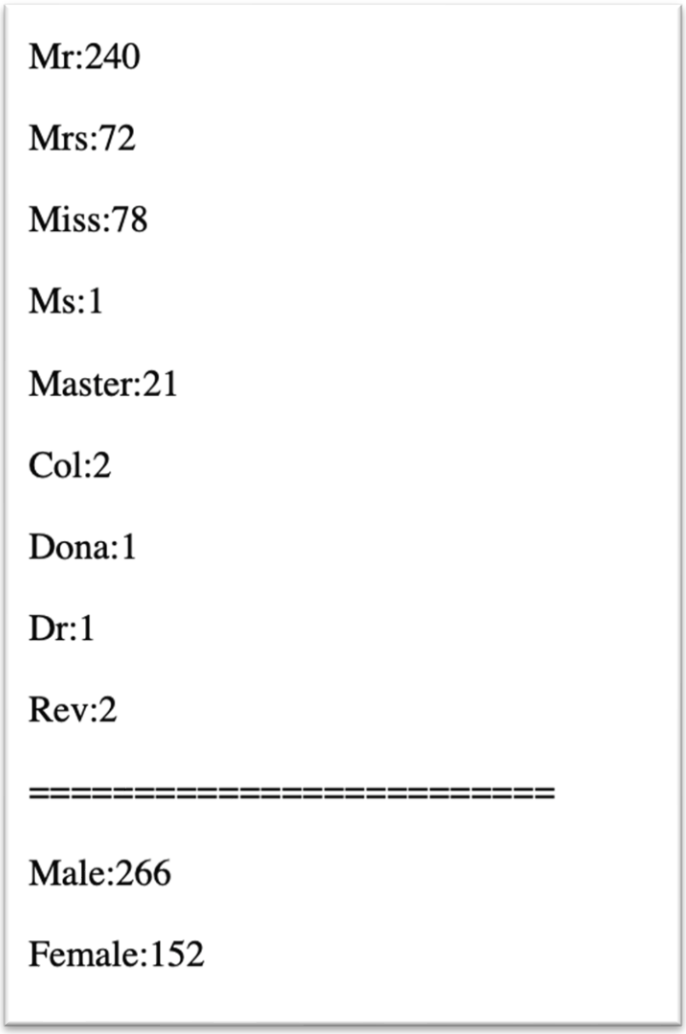
In second loop all gender is male and female.

The result show in console and screen both.

```
Mr = 240 Mrs = 72 Miss = 78 Ms = 1 Master = 21 Col = 2 Dona = 1  1-9.html:62
Dr = 1 Rev = 2
Male = 266 female = 152                                          1-9.html:78
```

```
function match(data) {
    var mr = 0;
    var mrs = 0;
    var miss = 0;
    var ms = 0;
    var master = 0;
    var col = 0;
    var dona = 0;
    var dr = 0;
    var rev = 0;

    // match all known title
    data.forEach( function(d)  {
        // console.log(d.Name) ;
        if (d.Name.match(/Mrs./)){
            mrs +=1;
        } else if (d.Name.match(/Mr./)) {
            mr += 1;
        } else if (d.Name.match(/Miss./)){
            miss += 1;
        } else if (d.Name.match(/Ms./)){
            ms += 1;
        } else if (d.Name.match(/Master./)){
            master += 1;
        } else if (d.Name.match(/Col./)){
            col += 1;
        } else if (d.Name.match(/Dona./)){
            dona += 1;
        } else if (d.Name.match(/Dr./)){
            dr += 1;
        } else if (d.Name.match(/Rev./)){
            rev +=1;
        } else{
            console.log('Unknown title!')
        }

    })
```

```
    // match gender
    var male = 0;
    var female = 0;
    data.forEach( function(d)  {
        // console.log(d.Name) ;
        if (d.Sex === 'male'){
            male +=1;
        } else if (d.Sex === 'female') {
            female += 1;
        }  else{
            console.log('Unknown gender!')
        }

    })

    d3.select("body").append("p").text("Mr:" + mr);
    d3.select("body").append("p").text("Mrs:" + mrs);
    d3.select("body").append("p").text("Miss:" + miss);
    d3.select("body").append("p").text("Ms:" + ms);
    d3.select("body").append("p").text("Master:" + master);
    d3.select("body").append("p").text("Col:" + col);
    d3.select("body").append("p").text("Dona:" + dona);
    d3.select("body").append("p").text("Dr:" + dr);
    d3.select("body").append("p").text("Rev:" + rev);

    d3.select("body").append("p").text("========================");

    d3.select("body").append("p").text("Male:" + male);
    d3.select("body").append("p").text("Female:" + female);


}

d3.csv(titaniccsv)
.then(match);
```

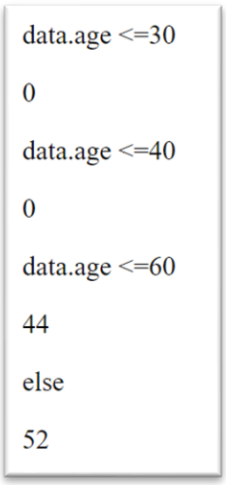



10:

This part is similar with exercise 8, the extra part is added two judge if age and death event.

The function will iterate the heartfailurecsv, if this example DEATH_EVENT equal 1,  it will going to further judge to judge the example age range. The age has 4 ranges, all range in a list call ageRange. Finally, the result will be print on screen.

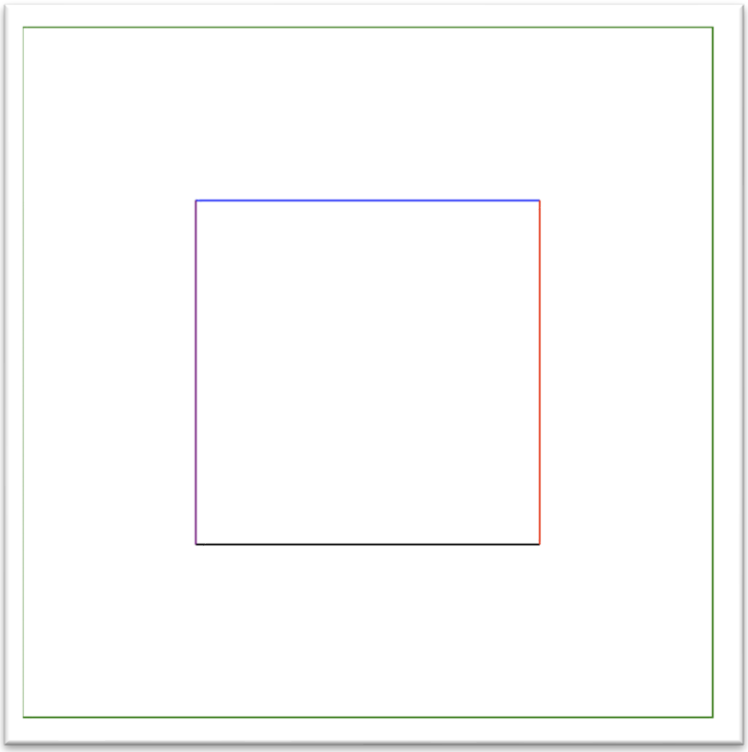
```
let heartfailurecsv =
'https://raw.githubusercontent.com/akmand/datasets/master/heart_failure.csv';

var num = [0, 0, 0, 0];
d3.csv(heartfailurecsv,
function(data) {
    //console.log( data.age);
    if (data.DEATH_EVENT == 1) {
        if (data.age <=30){
            num[0] += 1;
        } else if(data.age <=40){
            num[1] += 1;
        } else if(data.age <=60){
            num[2] += 1;
        } else{
            num[3] += 1;
        }
    }

    d3.select("body")
        .selectAll("div")
        .data(num)
        // .enter()
        // .append("div")
        .text(function (d, i) {
            return d;
        })
});
```



11.

I draw 4 lines, in the SVG element, north is blue, south is black, west is purple, and east is red. The corner location is (100,100), (100,300), (300,100), (300,300)

```
//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
//north
svg.append("line")
.attr("x1", 100)
.attr("x2", 300)
.attr("y1", 100)
.attr("y2", 100)
.attr("stroke", "blue")

//south
svg.append("line")
.attr("x1", 100)
.attr("x2", 300)
.attr("y1", 300)
.attr("y2", 300)
.attr("stroke", "black")

//west
svg.append("line")
.attr("x1", 100)
.attr("x2", 100)
.attr("y1", 100)
.attr("y2", 300)
.attr("stroke", "purple")

//east
svg.append("line")
.attr("x1", 300)
.attr("x2", 300)
.attr("y1", 100)
.attr("y2", 300)
.attr("stroke", "red")
```



12 & 13.

In this exercise I define 2 areas, one call csvbox for input shapes information. Another is canvas, that show the shape.

Have a redraw function, this function will run when page be open, and also will run when csvbox “onchange”.

This function will parser the test in csvbox, if text in csv file is well formatted, the function will draw the shape, otherwise will print the error message on the screen, such as not support share and lack of arguments etc.

This page also supports ‘enter’ and ‘exit’ concepts, when csvbox has changed, all content in canvas will be removed, and append new shape.

```
<body>

    <pre>
CSV format example:
#     type, color, x, y, ...
# rect:                       width, height
# circle:                     radius
# ellipse:                    rx, ry
# line:                       x1, y1
    </pre>

    <!-- initial shape -->
    <textarea id="csvbox" style="width: 800px; height: 100px;"
onchange="redraw();">
rect,    green,  10, 20, 20, 30
circle,   red,    40, 40, 20
ellipse, yellow, 40, 80, 20, 30
line,    purple, 60, 20, 80, 80
</textarea>

    <br>

    Enter new code in the text box and click the box below to update the shape.
<br>
    <div id="log">
    </div>
    <svg id="canvas" width=800px height="600px" style="border: 1px solid
green">
</body>
```

```
<script>
// Supported shapes
const MAPPING = {
    'rect': ['fill', 'x', 'y', 'width', 'height'],
    'circle': ['fill', 'cx', 'cy', 'r'],
    'ellipse': ['fill', 'cx', 'cy', 'rx', 'ry'],
    'line': ['stroke', 'x1', 'y1', 'x2', 'y2'],
};

function redraw() {
    var svg = d3.select("#canvas");
    svg.selectAll("*").remove();

    var logdiv = d3.select('#log');
    logdiv.selectAll('p').remove();

    // read csv box
    var csv = d3.select("#csvbox").property('value');
    var lineno = 0;
    for (var line of csv.split('\n')) {
        lineno = lineno + 1;
        // read single line
        var cells = line.split(",").map(x => x.trim()).filter(x => x.length != 0);
        if (cells.length == 0)
            continue;

        function log(x) {
            logdiv.append('p').text(`Line ${lineno}: ${x}`);
        }

        try {
            var kind = cells[0];
            var attrs = MAPPING[kind];
            if (attrs == undefined) {
                log(`unknown shape ${kind}`);
                break;
            }

            var values = cells.slice(1);
            if (attrs.length != values.length) {
                log(`csv length mismatch (expected ${attrs.length}, got ${values.length}`);
                break;
            }
```

```
            var shape = svg.append(kind);

            // draw
            for (var i = 0; i < attrs.length; ++i) {
                shape.attr(attrs[i], values[i]);
            }
        } catch (e) {
            log(`error handling: ${e}`)
        }
    }
}
redraw();
</script>
```

```
CSV format example:
#     type, color, x, y, ...
# rect:                       width, height
# circle:                     radius
# ellipse:                    rx, ry
# line:                       x1, y1
```

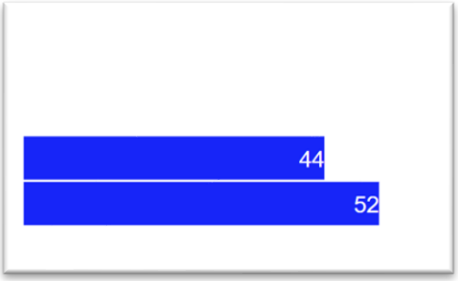
```
rect,    green,  10, 20, 20, 30
circle,   red,    40, 40, 20
ellipse, yellow, 40, 80, 20, 30
line,    purple, 60, 20, 80, 80
line,    purple, 60, 20, 80, 50
```

Enter new code in the text box and click the box below to update the shape.

14.

Base on exercise 10 I replaced the output string to the bar chart. I also use then keywords to make sure the output function only be run once at the end of the script.

```
<style>

    svg rect {
    fill: blue;
    }
    svg text {
    fill:red;
    font: 10px sans-serif;
    text-anchor: end;
    }
</style>
```

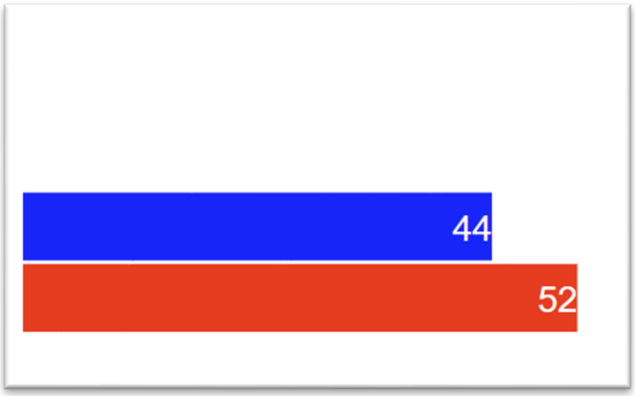



```
}).then(function bar(){
    var width = 800;
    var scaleFactor = 3;
    var barHeight = 20;
    var graph = d3.select("body")
        .append("svg")
        .attr("width", width)
        .attr("height", barHeight * 4);
    var bar = graph.selectAll("g")
        .data(num)
        .enter()
        .append("g")
        .attr("transform", function(d, i) {
            return "translate(0," + i * barHeight + ")";
        });
        bar.append("rect")
        .attr("width", function(d) {
            return d * scaleFactor;
        })
        .attr("height", barHeight - 1);
    bar.append("text")
        .attr("x", function(d) {
            return (d*scaleFactor); })
        .attr("y", barHeight / 2)
        .attr("dy", ".35em")
        .text(function(d) { return d; });
});
```

15.

Compare with 14, I remove the SVG color in <style>, and add a fill in bar variable, due to only 96 people's dead for heart failure, so if number larger then 50, the bar will be set to red.

```
}).then(function bar(){

    var width = 200;
    var scaleFactor = 3;
    var barHeight = 20;
    var graph = d3.select("body")
        .append("svg")
        .attr("width", width)
        .attr("height", barHeight * 4);
    var bar = graph.selectAll("g")
        .data(num)
        .enter()
        .append("g")
        .attr("transform", function(d, i) {
            return "translate(0," + i * barHeight + ")";
        });
        bar.append("rect")
        .attr("width", function(d) {
            return d * scaleFactor;
        })
        .attr("height", barHeight - 1)
        .attr("fill", function(d, i) {
            if(d > 50){
                return "red";
            } else{
                return "blue";
            }
        });
    bar.append("text")
        .attr("x", function(d) { return (d*scaleFactor); })
        .attr("y", barHeight / 2)
        .attr("dy", ".35em")
        .text(function(d) { return d; });
```



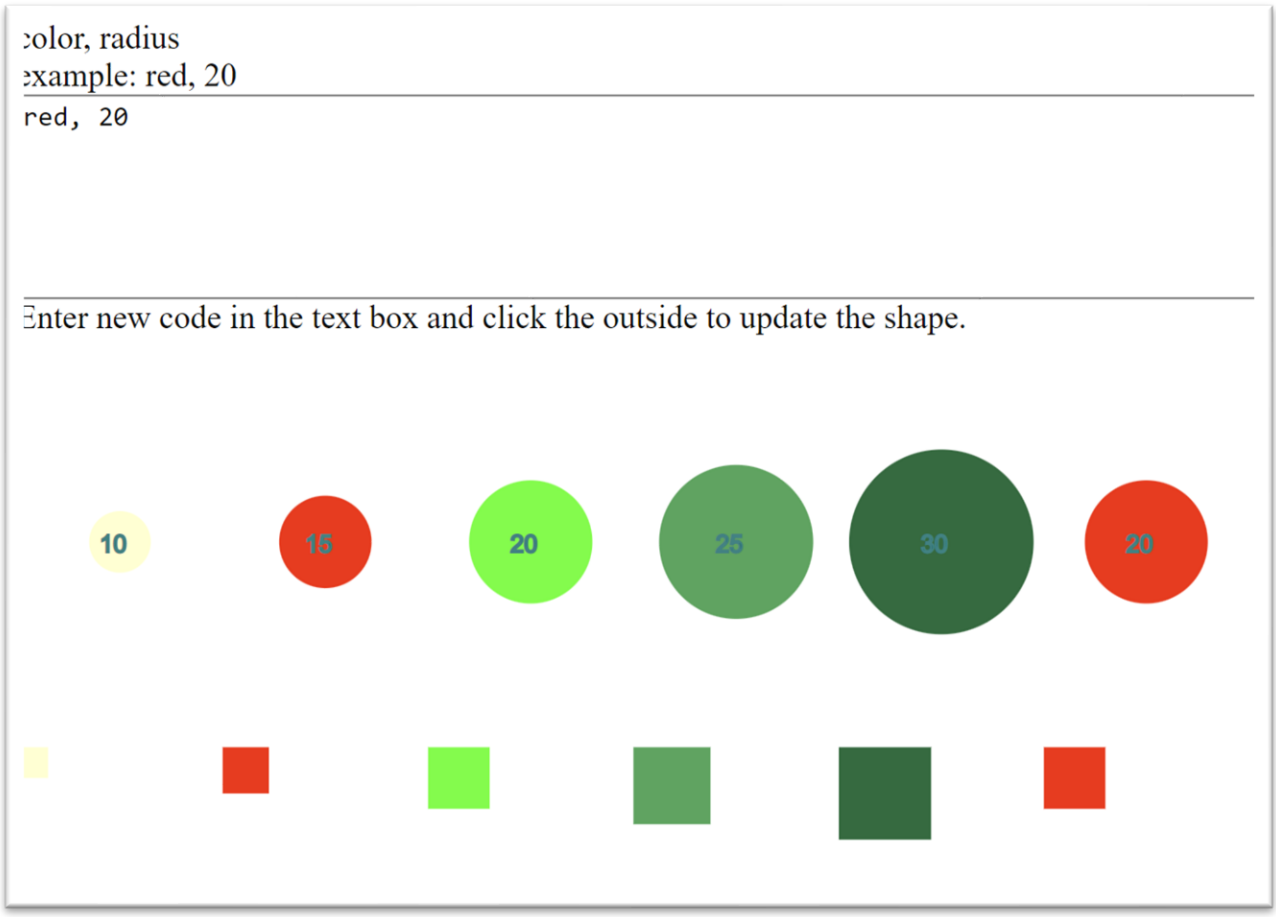
16.

I define 2 function, put demo draw code in to draw() , when the page be open, draw function will run. User and add shape in csvbox, when CSV on change, redraw function will run, redraw() will remove current shape and draw a new one.

```
<body>

    <textarea id="csvbox" style="width: 800px; height: 100px;"
onchange="redraw();">
</textarea>

    <br>
    Enter new code in the text box and click the outside to update the shape. <br>
</body>

<script>
var counter = 0;
// Supported shapes

const width = 5000;
const height = 500;

// circle r
var data = [10, 15, 20, 25, 30];

// Note different valid ways of specifying color
var colors = ['#ffffcc','red','rgb(0,255,0)','#31a354','#006837'];

function redraw(){
    d3.selectAll("svg").remove();
    var csv = d3.select("#csvbox").property('value');
    for (var line of csv.split('\n')) {
        var cells = line.split(",").map(x => x.trim()).filter(x => x.length != 0);
        if (cells.length == 0)
            continue;

        colors.push(cells[0]);
        data.push(cells[1]);

    }
    draw();
}

function draw(){
```

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

draw();
</script>
```

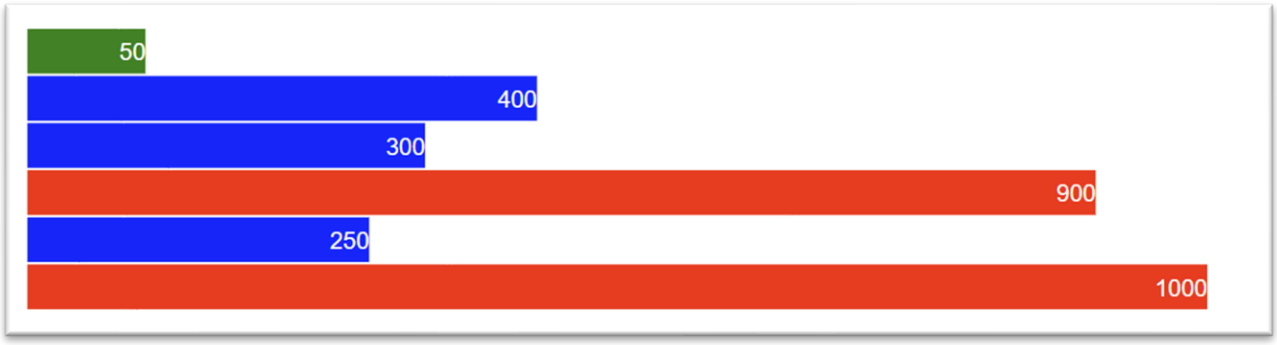


17.

Like e15, I extend the .attr("fill") to a function is below 100 will return green , if below then  500 will return blue, else will return red. I also modify the style, set the text color to white for easier to read.

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

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

```
<style>

    svg text {
    fill:white;
    font: 10px sans-serif;
    text-anchor: end;
    }
</style>
```



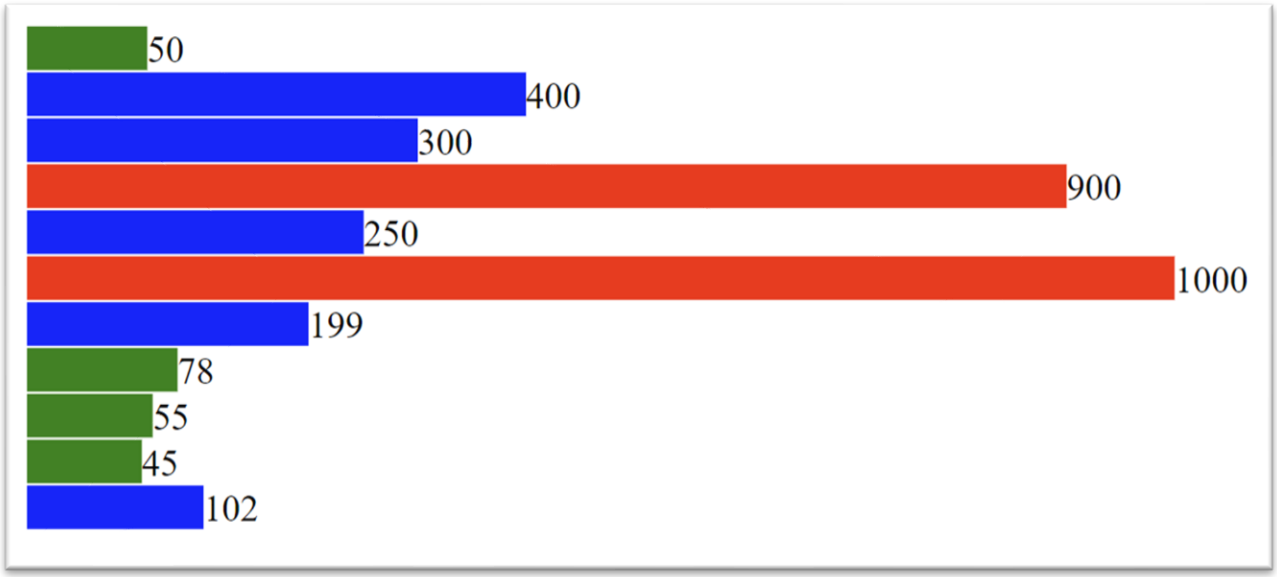
18.

Based on the e17, I add read csv from URL feature, the program will load a csv file from my GitHub repo.

The csv file shows below, the program only uses the score features.

```
name,score
James,199
Bob,78
Daniel,55
Ziyad,45
Joshua,102
```

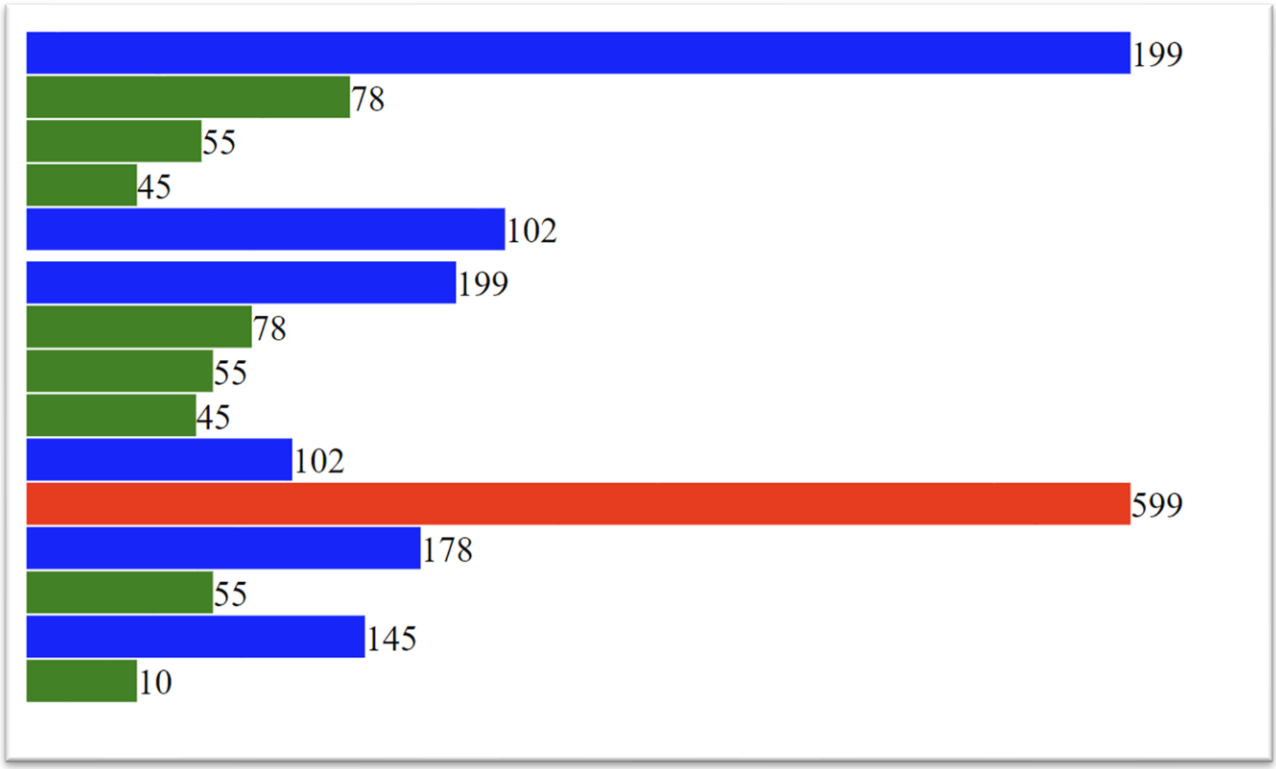
```
const link =
"https://raw.githubusercontent.com/JamesW99/JamesW99.github.
io/main/lab1/csv.csv?token=GHSAT0AAAAAABN4OYQURMC3NRRIOXJ2PP
7WYQDVJYA"


const data = [50, 400, 300, 900, 250, 1000]

d3.csv(link, function(d) {
    data.push(parseInt(d.score));
    // console.log(data)


}).then(function draw(){
    console.log(data)
    const width = 600;
    const barHeight = 20;
```



19.

Based on e18, I put all code in to run() function. And I create a new csv file include different data with same formats. At the end of script, I call run twice by cvs.csv and csv2.csv.

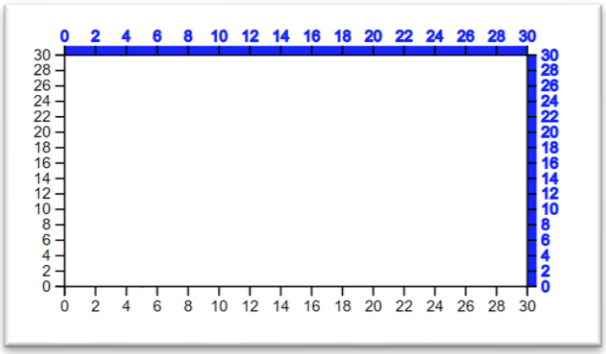
```
const link =
"https://raw.githubusercontent.com/JamesW99/JamesW99.github.io/main/lab1/csv
.csv?token=GHSAT0AAAAAABN4OYQURMC3NRRIOXJ2PP7WYQDVJYA"

const link2 =
"https://raw.githubusercontent.com/JamesW99/JamesW99.github.io/main/lab1/csv
2.csv?token=GHSAT0AAAAAABN4OYQVKVTWBNFRSLBEE5PMYQD3NOA"

const data = []
function run(url){
    d3.csv(url, function(d) {

    ..........

}
run(link);
run(link2);
```



20.

Based on the demo code, I duplicate two axis and move it to the right place, at least, I use fill and stroke make the new axis to blue.

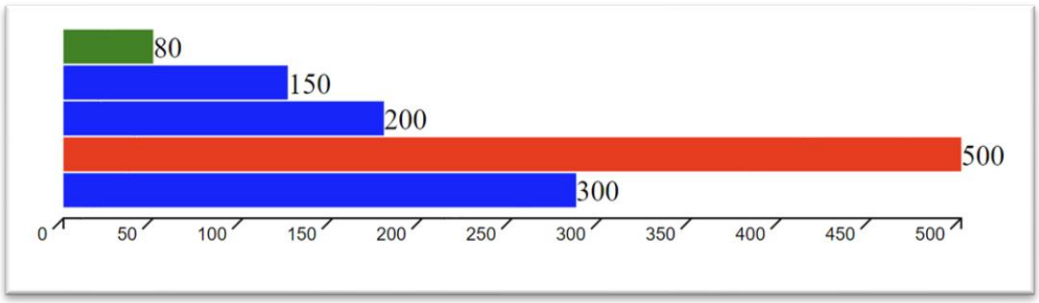
```
<script>

……

var topscale = d3.scaleLinear()
    .domain([0, d3.max(data)])
    .range([0, width - 100]);
var rightscale = d3.scaleLinear()
    .domain([0, d3.max(data)])
    .range([height/2, 0]);

……

var top_axis = d3.axisTop()
    .scale(topscale);
var right_axis = d3.axisRight()
    .scale(rightscale);

……
svg.append("g")
    .attr("transform", "translate(350, 30)" )
    .call(right_axis)
    .attr("fill", "blue")
    .attr("stroke", "blue");
var xAxisTranslate = height/2 + 30;

……

var xAxisTranslate = height/2 -120;
svg.append("g")
    .attr("transform", "translate(50, " + xAxisTranslate +")")
    .call(top_axis)
    .attr("fill", "blue")
    .attr("stroke", "blue")
</script>
```



21.

Based on the bar code and axis code, I combined two function and move the axis to the right place.

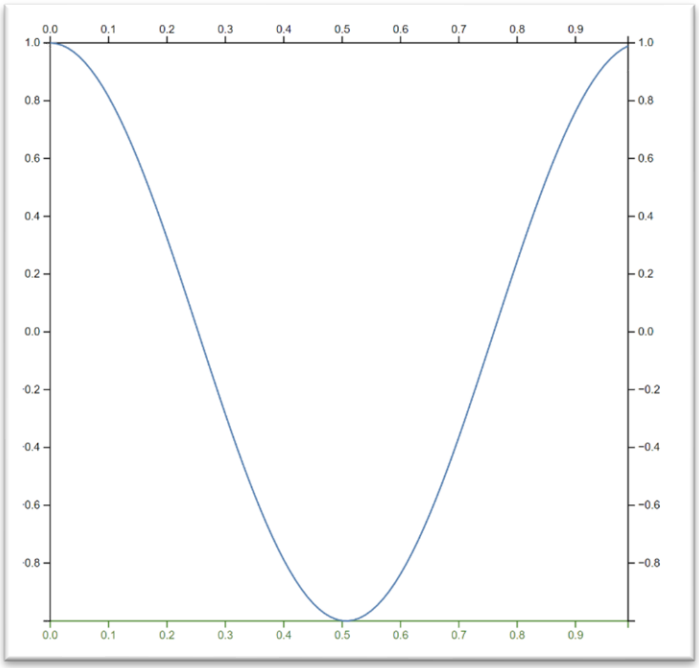
```
……
        var svg = d3.select("body")
            .append("svg")
            .attr("width", width)
            .attr("height", height);

        var xscale = d3.scaleLinear()
            .domain([0, d3.max(data)])
            .range([0, width - 100]);
        var yscale = d3.scaleLinear()
            .domain([0, d3.max(data)])
            .range([height/2, 0]);

        var x_axis = d3.axisBottom()
            .scale(xscale);
        var y_axis = d3.axisLeft()
            .scale(yscale);

        var xAxisTranslate = height/2 + 10;
        svg.append("g")
            // .attr("transform", "translate(50, 10)")
            .attr("transform", "translate(50, -150)")
            .call(y_axis)

        // svg.append("g")
            .attr("transform", "translate(50, " + 1 +")")
            .call(x_axis)
    })
}
run(link);
```



22.

Based on the demo code, I make it combined in a function, this function has a string parameter, if argument is Sin the function will draw a sine wave. It judged by if. And this function also supports tan and cos.

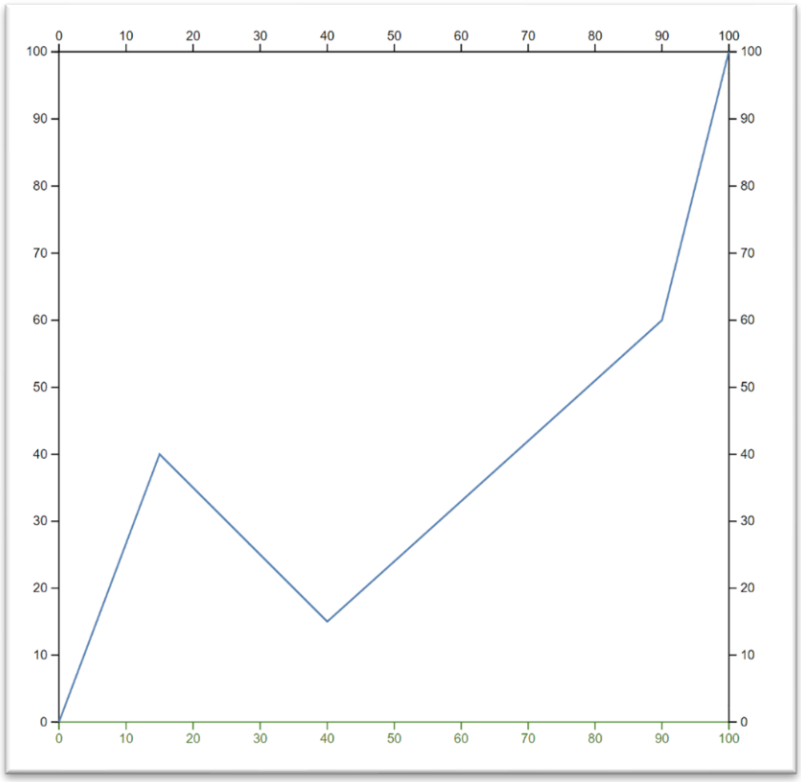
```
function fun(wavetype){

    for (let i = 0; i < numPoints; i++) {
        if (wavetype === 'sin'){
            data.push( { x: i/100, y: Math.sin( 6.2 * i/100 ) } );
        }
        if (wavetype === 'cos'){
            data.push( { x: i/100, y: Math.cos( 6.2 * i/100 ) } );
        }
        if (wavetype === 'tan'){
            data.push( { x: i/100, y: Math.tan( 6.2 * i/100 ) } );
        }
    }

……



svg.append("g")
        .call(d3.axisLeft(y));
    // right y axis
    svg.append("g")
        .attr("transform", `translate(${yMax},0)`)
        .call(d3.axisRight(y));

    // Add the line
    svg.append("path")
        .datum(data)
        .attr("fill", "none")
        .attr("stroke", "steelblue")
        .attr("stroke-width", 1.5)
        .attr("d", d3.line()
        .x(function(d) { return x(d.x) })
        .y(function(d) { return y(d.y) })
    );
}
// fun('sin');
fun('cos');
// fun('tan');
</script>
```




23.

Based on the e22 I remove judge sin cos tan part and add a function to read csv file. The csv file has the same format with e22. Then, run the run() function to draw the line.

```
const csvlink =
'https://raw.githubusercontent.com/JamesW99/JamesW99.github.io/main/lab1/line.
csv?token=GHSAT0AAAAAABN4OYQVEGHD7ZX4IUBJSV5AYQFEEFA'

……

const numPoints = 5;
var data = [];
d3.csv(csvlink, function(d){
    data.push( { x: parseInt(d.x), y: parseInt(d.y) } );
}).then(

function run(wavetype){


……
```

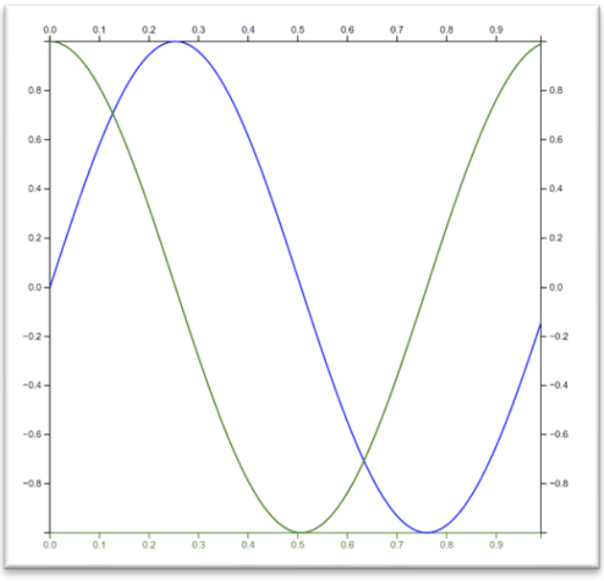



24.

Based on the e22, I extend the for loop, add another array call data2, each loop will calculate sin and cos both. At the end of the function, I add a extra append() the data2 be added in to csv.

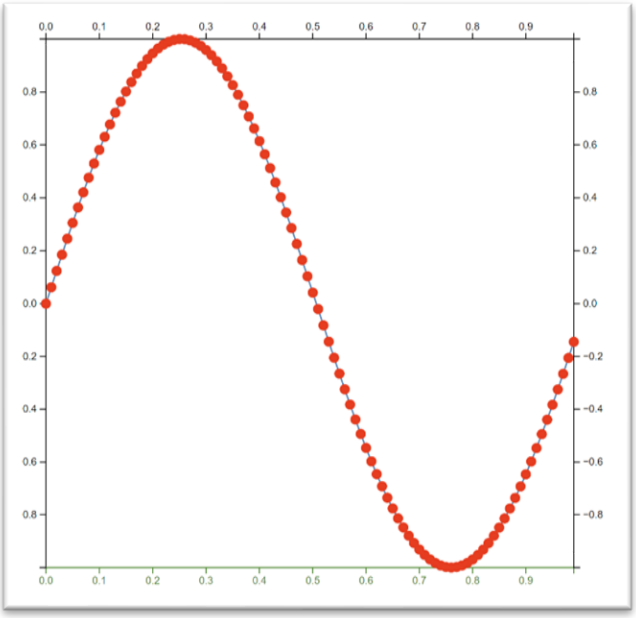
```
const data = [];

const data2 = [];
……
```

```
    // Add the line
    svg.append("path")
        .datum(data)
        // .datum(data2)
        .attr("fill", "none")
        .attr("stroke", "blue")
        .attr("stroke-width", 1.5)
        .attr("d", d3.line()
        .x(function(d) { return x(d.x) })
        .y(function(d) { return y(d.y) })
        );
    svg.append("path")
        .datum(data2)
        .attr("fill", "none")
        .attr("stroke", "green")
        .attr("stroke-width", 1.5)
        .attr("d", d3.line()
        .x(function(d) { return x(d.x) })
        .y(function(d) { return y(d.y) })
        );
}
// fun('sin');
fun();
// fun('tan');
</script>
```



25.

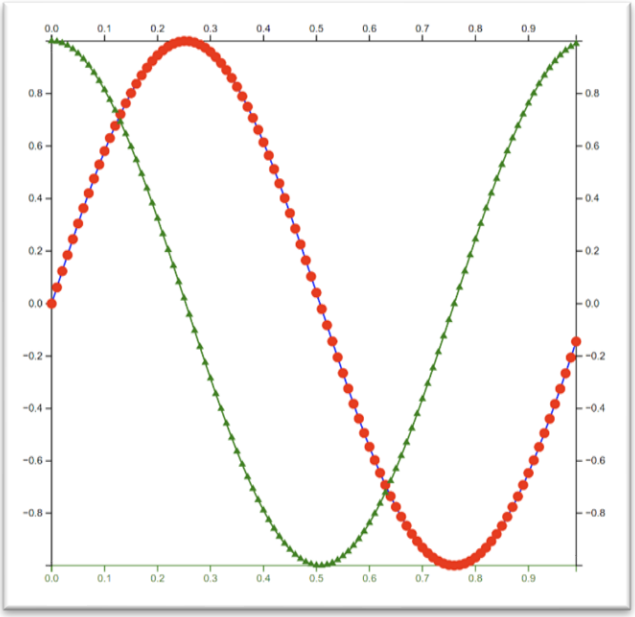
The requirements can be achieved using the example code.




26.

Based on the e24 and e25, due to append triangle is unexpected, so I define a variable call triangle, this variable type is “d3.symbolTriangle”. After that I add this var into data by attribute. This step will add one single triangle at the begin of the cosine wave, so I need a loop to make sure the triangle on the hole  line.

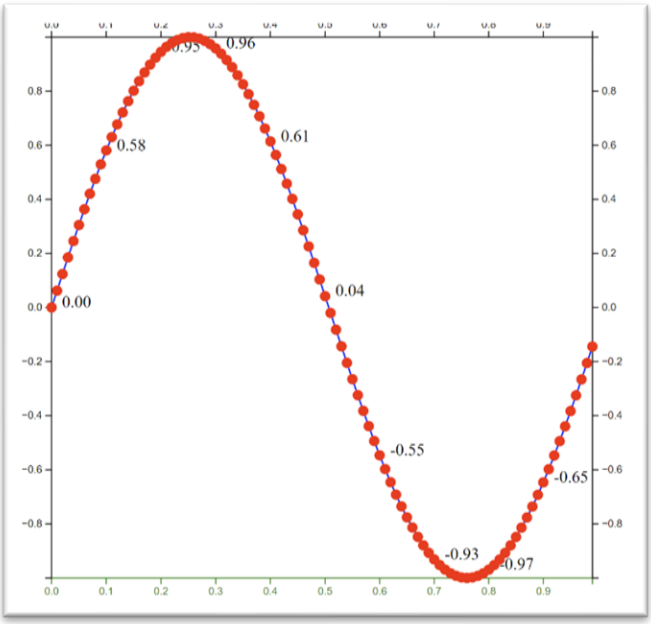
```
// add the triangles to line
var triangleSize = 25;

var triangle = d3.symbol()
        .type(d3.symbolTriangle)
        .size(triangleSize);
svg.selectAll("dot")
    .data(data2)
    .enter()
    .append("path")
    .attr("d", triangle)
    .attr("transform", function(d) {
        return "translate(" + x(d.x) + "," + y(d.y) + ")";
    })
    .style("fill", "green");
```




27.

Due to reduce cos wave to 10 prints will reduced accuracy, so I need to reduce data output. Based on the e25, I add a dots filter, if x % 10 == 0 will show the data.

```
var dots = svg.selectAll("dot")

    .data(data)
    .enter()
    .append("g")
    .attr('transform', d => `translate(${x(d.x)}, ${y(d.y)})`);

dots.append('circle')
    .attr("cx", 0)
    .attr("cy", 0)
    .attr("r", 5)
    .style("fill", "red");
dots.filter((_, i) => i % 10 == 0)
    .append('text')
    .attr('transform', 'translate(10, 0)')
    .text(x => x.y.toFixed(2));
```




28.

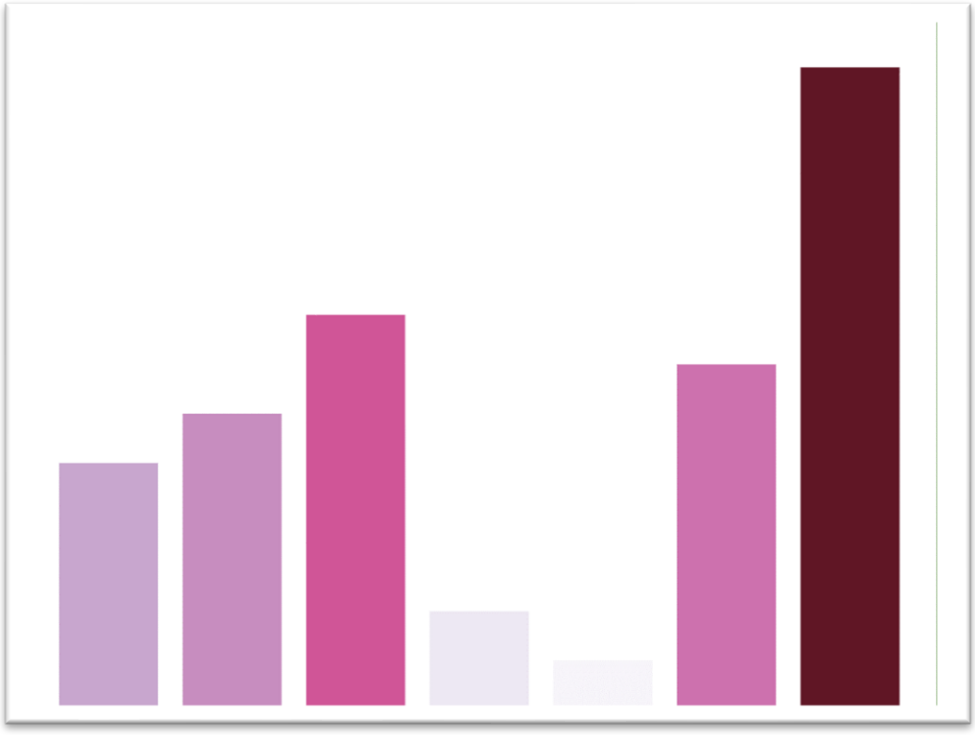
In part 15, I define a new array call weather, it includes week, temperature and weather.

Im use Math.max(min) to zoom and draw the axis. After that, draw the bar and use scaleSequential() to colorised the bar.

in interpolatePuRd, the color closer to black, its means more hotter, I also try the interpolateRainbow, bur the rainbow color make reader hard to realize which color mean hoter.

```
var weather = [

    ['Mon', 2, 'partly_cloudy'],
    ['Tue', 3, 'sunny'],
    ['Wed', 5, 'sunny'],
    ['Thu', -1, 'snow_light'],
    ['Fri', -2, 'snow_light'],
    ['Sat', 4, 'sunny'],
    ['Sun', 10, 'sunny'],
];
```

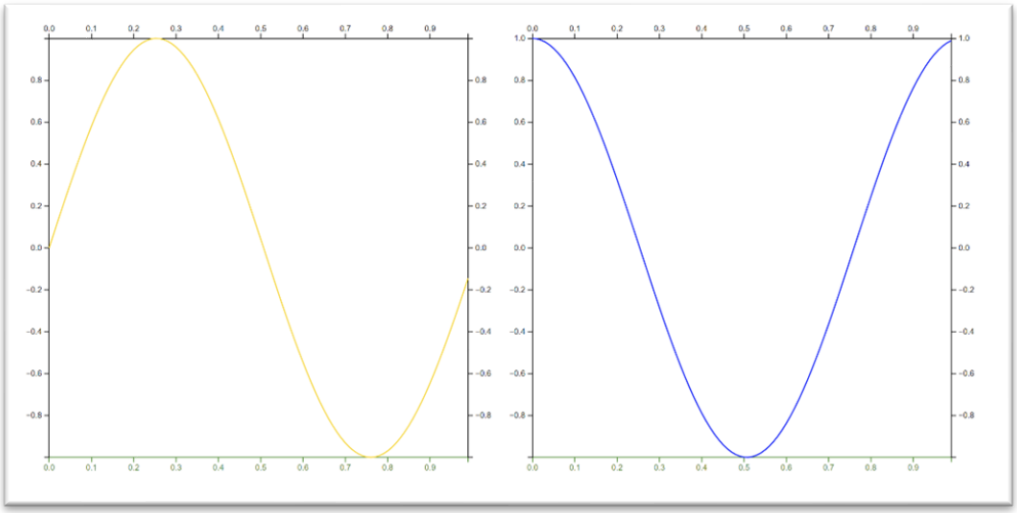
```
var temps = weather.map(x => x[1]);

var maxTemp = Math.max.apply(Math, temps),
    minTemp = Math.min.apply(Math, temps);

var graph = d3
    .select("svg")
    .attr('height', height)
    .attr('width', width);
var x = d3.scaleBand() //刻度
    .range([10, width - 10])
    .domain(weather.map(x => x[0]))
    .padding(0.2);
var y = d3.scaleLinear()
    .range([40, height - 40])
    .domain([minTemp, maxTemp]);
var bars = graph.selectAll('bars')
    .data(weather)
    .enter()
    .append('g');
var color = d3.scaleSequential()
    .domain([minTemp, maxTemp])
    .interpolator(d3.interpolatePuRd);
```



29.

Based on the e22, I add an extra parameter to control “linecolor”.

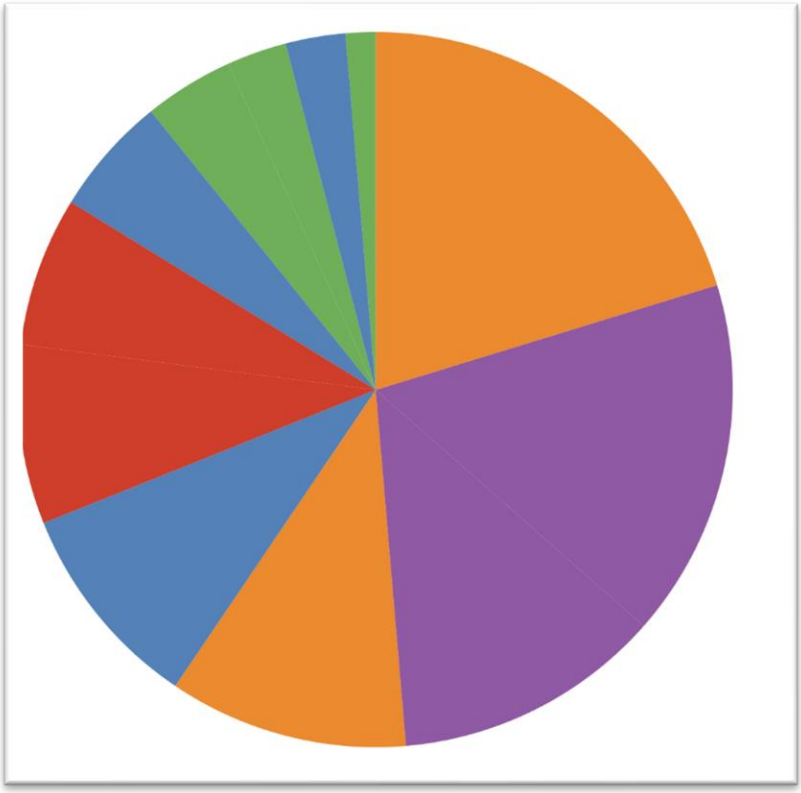
```
function fun(wavetype, linecolor) {

    var data = [];


……


    // Add the line
    svg.append("path")
        .datum(data)
        .attr("fill", "none")
        .attr("stroke", linecolor)
        .attr("stroke-width", 1.5)
        .attr("d", d3.line()
            .x(d => x(d.x))
            .y(d => y(d.y))
        );
}


var color = d3.scaleOrdinal()
    .domain(['sin', 'cos'])
    .range(["gold", "blue", "green", "yellow"]
);

fun('sin', color('sin'));
fun('cos', color('cos'));
```



30.

Add more numbers to data array.

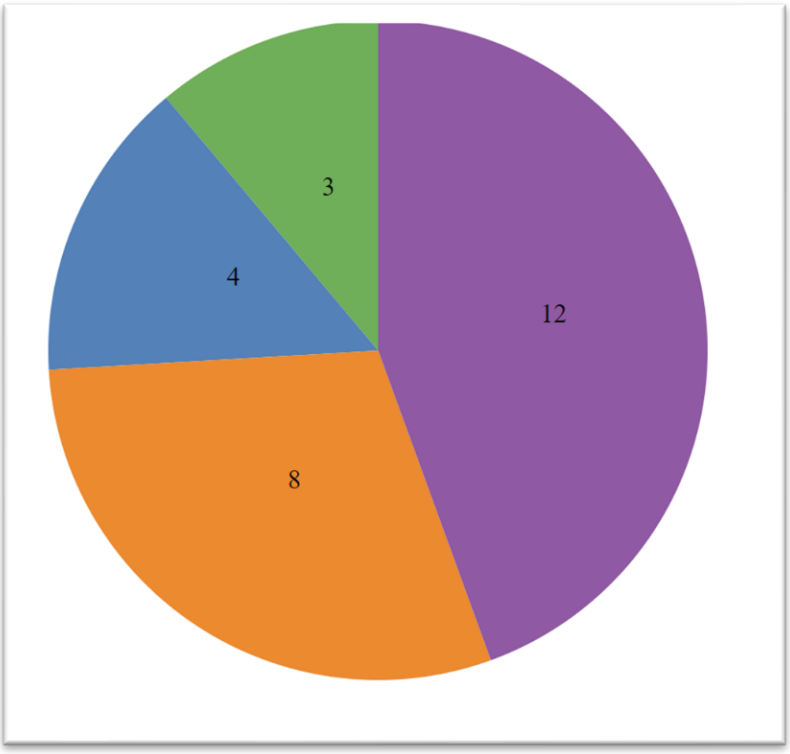
```
var data = [3, 4, 8, 12, 6, 2, 7, 15, 9, 5, 1, 2];
```



31.

Based on the demo code, I append text to arcs at the end of script.

```
<body>
    <svg id="svg"></svg>
</body>

<script>
……
arcs.append('text')
   .text(x => x.value)
   .attr("transform", x => `translate(${arc.centroid(x)})`);
</script>
```



32.

Based on the e28, I append extra PNG to variable bars. The image get from gstatic.com.

```
bars.append('svg:image')

    .attr("xlink:href", v =>
`https://ssl.gstatic.com/onebox/weather/48/${v[2]}.png`)
    .attr("width", 40)
    .attr("height", 40)
    .attr("x", v => x(v[0]) + x.bandwidth() / 2 - 40 / 2)
    .attr("y", v => height - y(v[1]) - 40);
```