Yishan Wang F20DV Lab 2

25 February 2022 demonstrated to

Xue, Shuangjiang

All the exercise can be viewed on this link.

https://jamesw99.github.io/

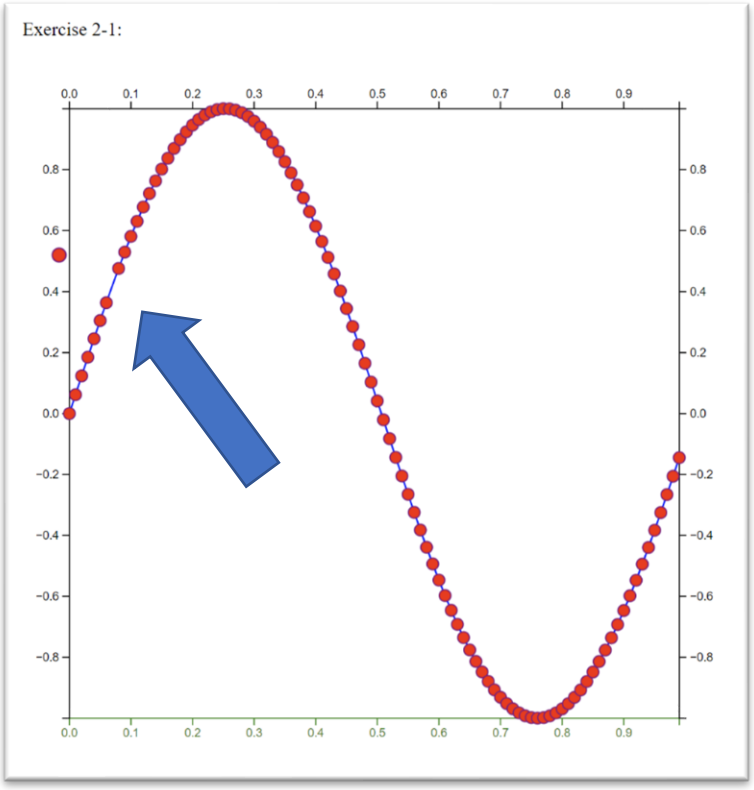
PLEASE LET ME KNOW IF ANY PAGE CANNOT BE OPEN

BECAUSE JSON AND CSV HOST ON GITHUB PRIVATE REPO, SO THE TOKEN MAY REFRESH.

yw2007@hw.ac.uk

Exercises 1:

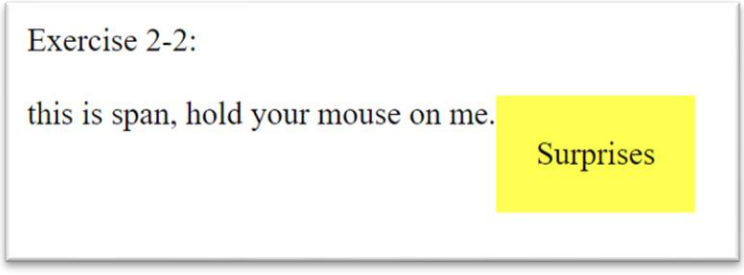
I combine e1-26 with the demo code, and add .attr("class", "pulse"); when select dot.



Exercises 2:

Based on the demo code, I create a script to using styles.

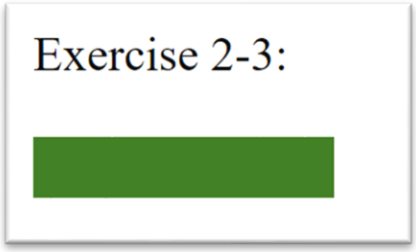
```
<script>

    d3.select("body")
        .append("span")
        .text("this is span,
hold your mouse on me.")

    d3.select("body")
        .append("div")
        .text("Surprises")
</script>
```

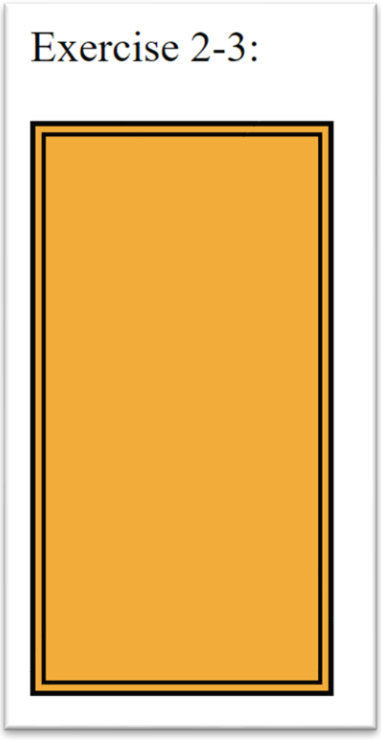


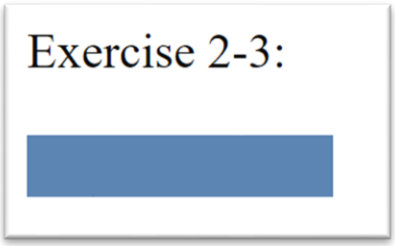
Exercises 3:

Based on the demo code. I add a few lines on mouse over and mouse out to make rectangle larger when mouse on it and recover the size when mouse out.

```
.style("width", "100px")

.style("height", "200px")

.style("border", "6px")
.style("border-style", "double")

……

.style("border", "none")
.style("width", "100px")
.style("height", "20px")
```



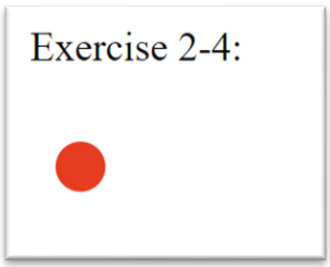


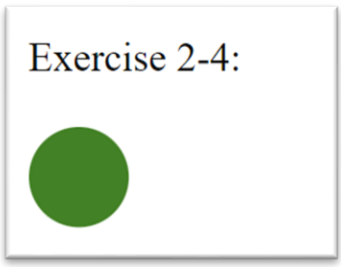


Exercises 4:

Based on least exercise, I change the div to svg, and append a circle in the svg. Other part are similar, select that circle and control the radius and color.

```
d3.select('body')
    .append("svg")
    .style('width', '100px')
    .style('heigh', '100px');

d3.select("svg")
    .append("circle")
    .attr("r", 10)
    .attr("cx", 20)
    .attr("cy", 20)
    .style("fill", "red")

d3.selectAll("circle")
    .on("mouseover", function(){
        // mouse on style
        d3.select(this)
            .attr("r", 20)
            .style("fill", "green")
    })

    // mouse out style
    .on("mouseout", function(){
        d3.select(this)
            .attr("r", 10)
            .style("fill", "red")
    });
```





Exercises 5:

Based on e4, I redefine the svg to 3840*2160 to make sure its in entire screen. I also add a variable call text, it can save current mouse position. All last, I add mouse move event, when the mouse move, function will update the mouse position (coords) and show on logs and svg.

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

var text = svg.append('text')
    .attr("x", 0)
    .attr("y", 0)
    .text("(0,0)")

d3.select('svg')
    .on('mousemove', function(event) {
        var coords = d3.pointer( event );
        var x = coords[0];
        var y = coords[1];
        console.log( coords[0], coords[1] ) // log the mouse x,y
position
        text.attr("x", x)
        text.attr("y", y)
        text.text("(" + x + ", " + y + ")")
    });
```

Exercises 6:

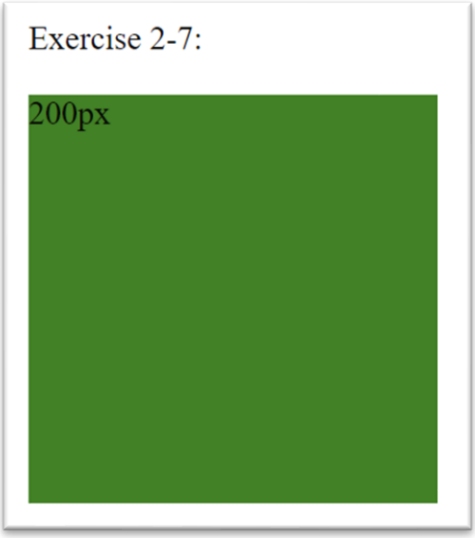
Based on the demo code, I add another's transition.

```
    .transition()
    .duration(2000)
    .style("background-color", "green");
```

Exercises 7:

Based on 5 e6, I add a style to control div size.

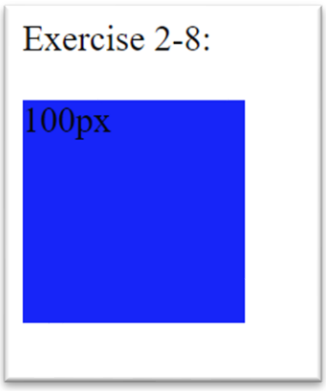
```
    .style('width', '200px')
    .style('height', '200px')
    .style("background-color", "green")
    .text('200px');
```

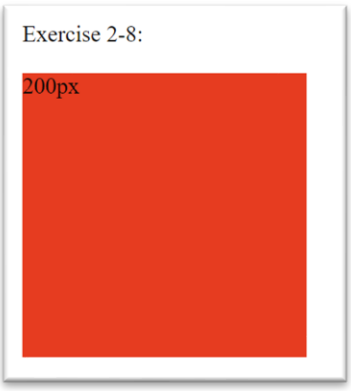


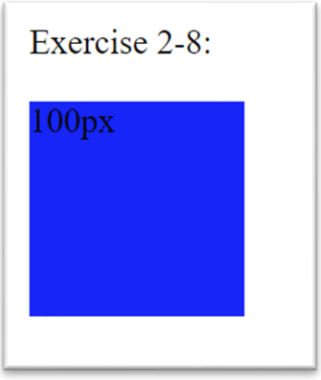
Exercises 8:

Based on e3 and e7, I combined mouse over event and transition()

```
// mouse on color

    .on("mouseover", function(event){
        d3.select(this)
            .transition()
            .duration(1000)
            .style('width', '200px')
            .style('height', '200px')
            .style("background-color", "red")
            .text('200px')
    })

// mouse out color
    .on("mouseout", function(){
        d3.select(this)
            .transition()
            .duration(1000)
            .style('width', '100px')
            .style('height', '100px')
            .style("background-color", "blue")
            .text('100px')
    });
```



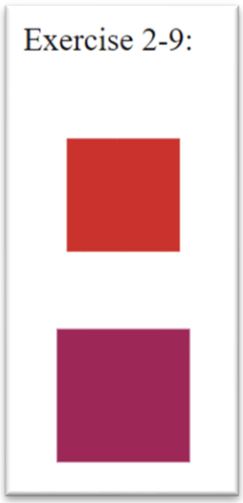




Exercises 9:

Based on the demo code, I add an extra div and replace easy to easeLinear.

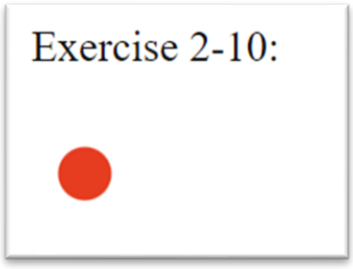
```
    .ease( d3.easeLinear )
```

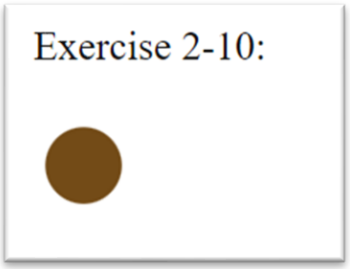


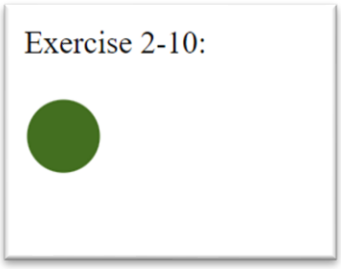
Exercises 10:

Based on e4, I add transition when mouse over and out.

```
            .transition()
            .ease( d3.easeBounce )
            .duration(1000)
```



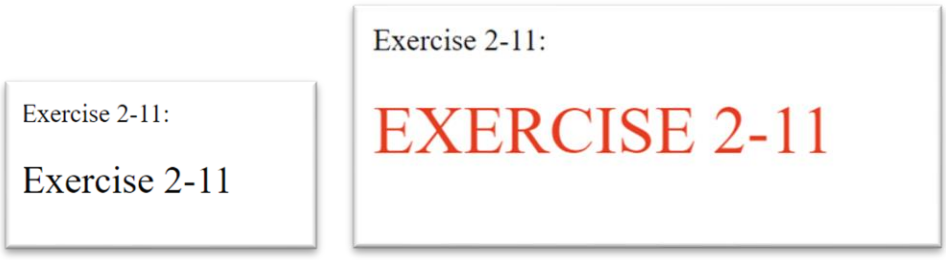




Exercises 11:

I define a div as begin and select is. Set the text to "EXERCISE 2-11", color to "red" and size to "36px" when mouse over. recover when mouse out.

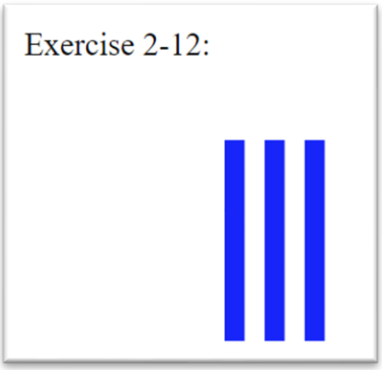
```
d3.select("body")
    .append("div")
    .text("Exercise 2-11")
    .style("font-size", "24px")

d3.selectAll("div")

    // mouse on style
    .on("mouseover", function(event){
        d3.select(this)
            .text("EXERCISE 2-11")
            .transition()
            .ease( d3.easeBounce )
            .duration(1000)
            .style("font-size", "36px")
            .style("color", "red")
    })

    // mouse out style
    .on("mouseout", function(){
        d3.select(this)
            .text("Exercise 2-11")
            .transition()
            .ease( d3.easeBounce )
            .duration(1000)
            .attr("r", 10)
            .style("font-size", "24px")
            .style("color", "black")
    });
```



Exercise 12:

Based on the demo code, I duplicate the bar 2 and change the name to bar 3, also duplicate in update function.

```
var bar3 = svg.append("rect")
    .attr("fill", "blue")
    .attr("x", 140)
    .attr("y", 20)
    .attr("height", 20)
    .attr("width", 10)

    ……


bar3.transition()
        .ease(d3.easeLinear)
        .duration(2000)
        .delay(4000)
        .attr("height",100)
```



Exercise 13:

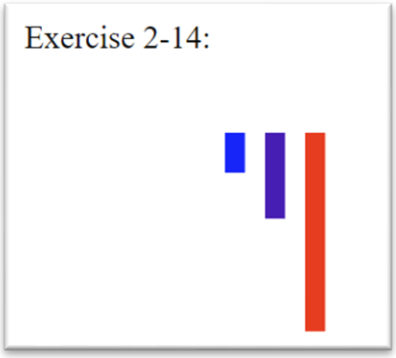
Based on the e13, I duplicate the content of update in same function and change the height to 20.

```
    // recover
    bar1.transition()
        .ease(d3.easeLinear)
        .duration(2000)
        .delay(6000)
        .attr("height",20)

    bar2.transition()
        .ease(d3.easeLinear)
        .duration(2000)
        .delay(8000)
        .attr("height",20)

    bar3.transition()
        .ease(d3.easeLinear)
        .duration(2000)
        .delay(10000)
        .attr("height",20)
```

Exercise 14:

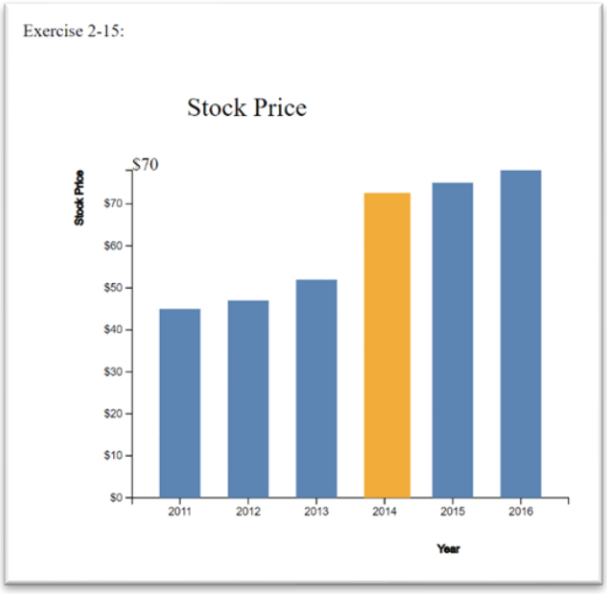
based on e14, I changed the bar color when grow and recovery.

```
    bar2.transition()
        .attr("fill", "red")

……

    bar2.transition()
        .attr("fill", "blue")
```



Exercise 15:

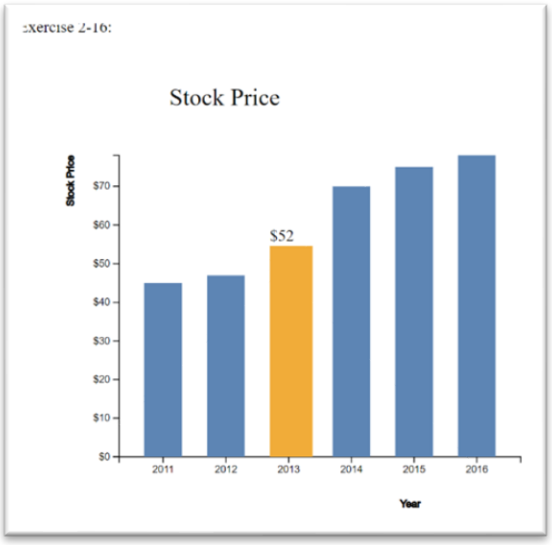
Combine two parts of demo code can achieve goal.



Exercise 16:

Based on e15, when append text, the return “d.year” be changed to" “i.year”. And “d.value” also be changed to “i.value”.

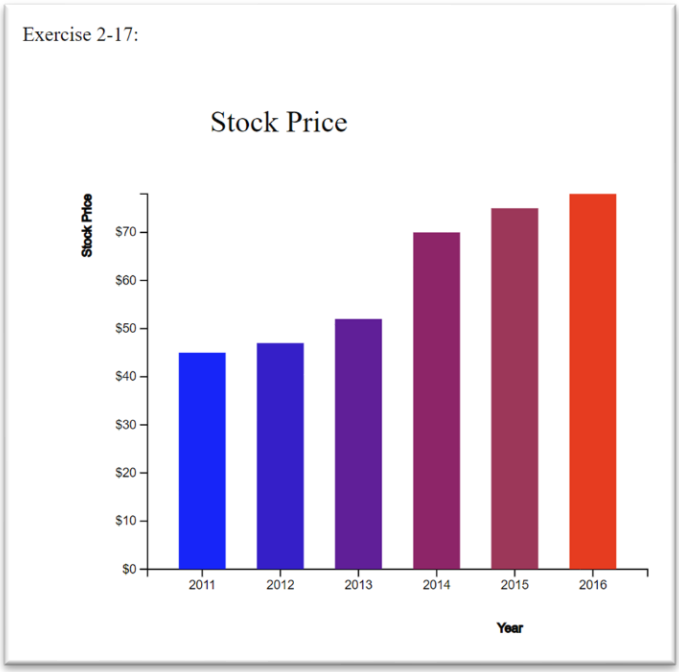
```
    g.append("text")
        .attr('class', 'val')
        .attr('x', function() {
            return x(i.year);
        })
        .attr('y', function() {
            return y(i.value) - 15;
        })
        .text( function(d) { return '$' + i.value; } );
```



Exercise 17:

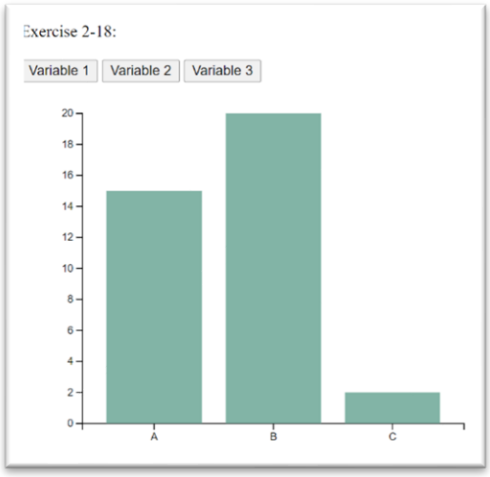
Based on the e16 and lab 1-28, I define a color domain and range. And I custom the bar color when append.

```
    var color = d3.scaleSequential()
        .domain([0,5])
        .range(["red", "blue"]);

……

        // return bar color
        .style("fill", function(d, i) {
            var index = data.sort(function(a, b) {
                    return a.index - b.index;
            })
            return color(5-index.indexOf(d))
        });
```



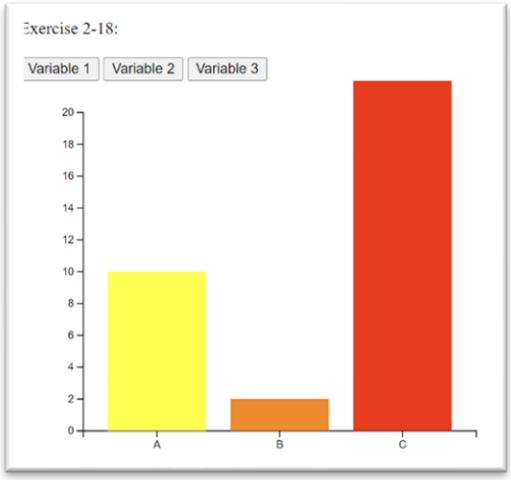
Exercise 18:

Duplicate and modify the demo code data 2 to data 3.



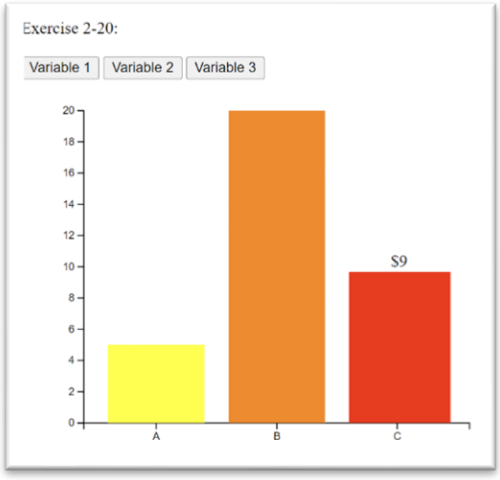
Exercise 19:

As same as e17 I define color and fill in in the bar. (Three color, red orange and yellow)



Exercise 20:

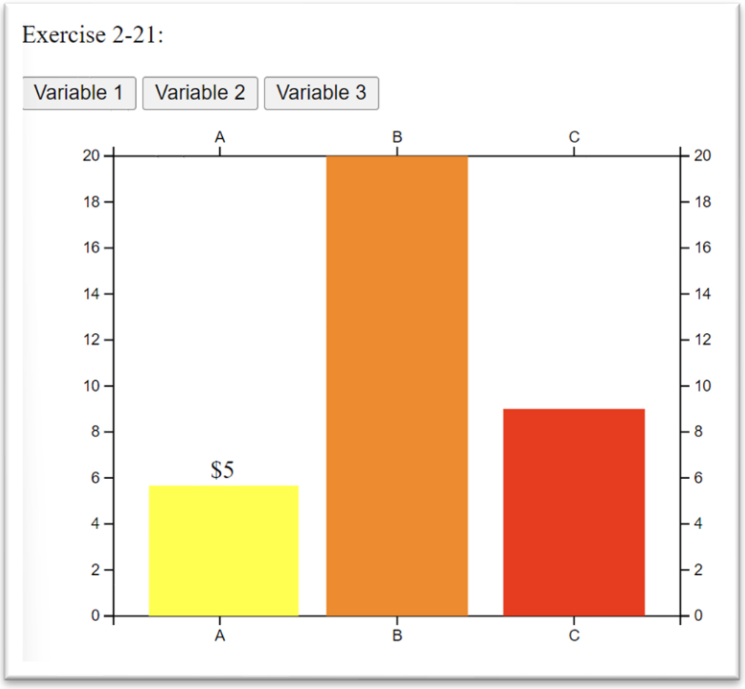
I paste mouse over and out function from e17 and add event between append("rect") and merge(u).



Exercise 21:

Add top axis and right axis between x axis and y axis.

```
// Top axis
var t = d3.scaleBand()
    .range([ 0, width ])
    .domain(data1.map(function(d) { return d.group; }))
    .padding(0.2);
svg.append("g")
    .attr("transform", "translate(0, 0)")
    .call(d3.axisTop(t))

// Add Right axis
var r = d3.scaleLinear()
    .domain([0, 20])
    .range([ height, 0]);
svg.append("g")
    .attr("transform", "translate(" + width + ", 0)")
    .call(d3.axisRight(r))
```



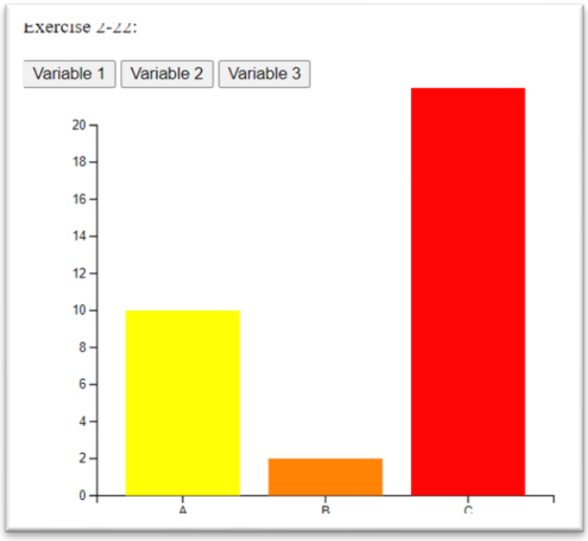
Exercise 22:

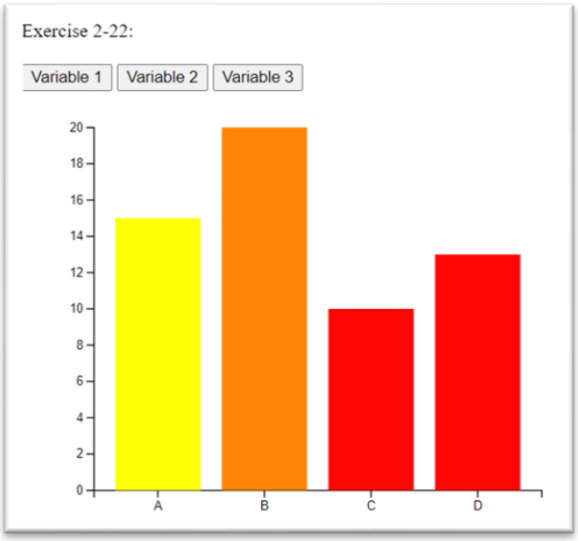
In this exercise, the key point is ".domain(data.map(function(d) { return d.group; }))" domain should be defined by data instead of data1. And update function should cover the create svg.

```
const data3 = [
    {group: "A", value: 15},
    {group: "B", value: 20},
    {group: "C", value: 10},
    {group: "D", value: 13}
];
```

```
function update(data) {
    d3.selectAll('svg').remove()
// append the svg object to the body of the page
var svg = d3.select('body')
    .append('div')
```

```
// X axis
var x = d3.scaleBand()
    .range([ 0, width ])
    .domain(data.map(function(d) { return d.group; }))
```

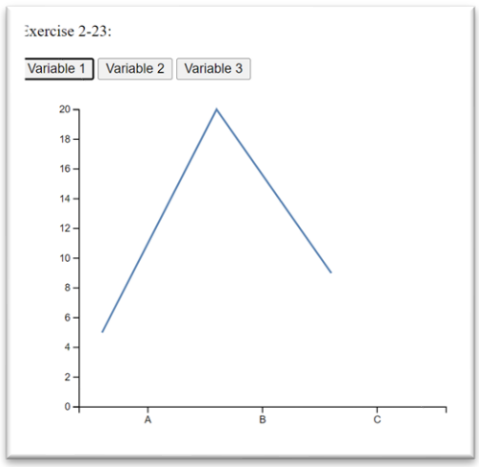


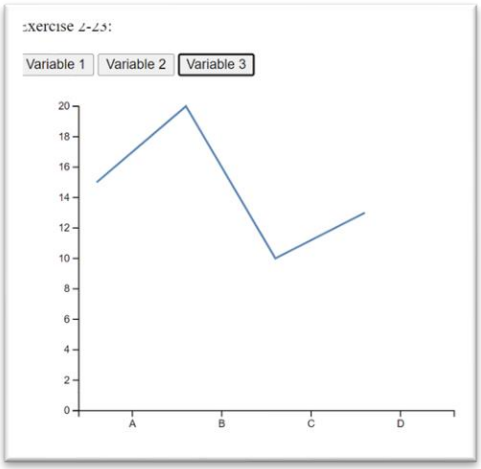


Exercise 23:

Based on e22, I replaced draw bar part to a draw line part.

```
    svg.append("path")
        .datum(data)
        .attr("fill", "none")
        .attr("stroke", "steelblue")
        .attr("stroke-width", 2)
        .attr("d", d3.line()
            .x(function(d) { return x(d.group) })
            .y(function(d) { return y(d.value) })
        )
```





Exercise 24:

Result is [16.2, 34.4, 5.2]

because have a equation f(x) = a * (1 - t) + b * t

in this example 20 is a, 1 is b and 0.2 is t.

Exercise 25:

Result is rgb(128, 64, 0)

because cc method can find the averange between two color and print as rgb.

such as rgb of red is(255,0,0) and rgb of green is (0,128,0). cc(0.5) means completely average, so the result is rgb(128, 64, 0)

Exercise 26:

Exercise 27:

As same as e 23, but I remove x, y axis and build line part.

Reformatted data to key value format to streamline the code.

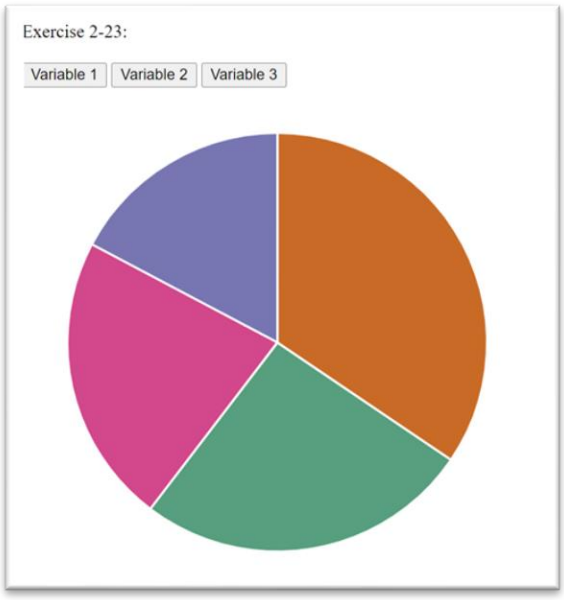
I defined a new radius const, to calculate each part radius of pie.

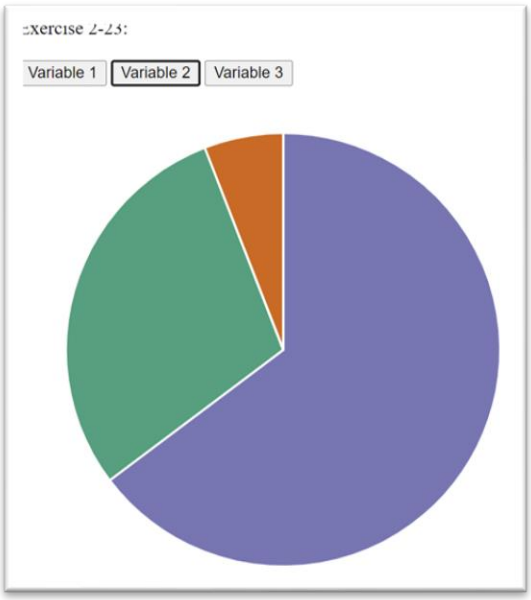
Color const is back to control each part color.

The key function of this exercise is update, first of all computes the position of each group on the pie, and then build the pie chart. Basically, each part of the pie is a path that I build using the arc function.

```
// create 2 data_set
const data1 = {a: 5, b: 20, c:9}
const data2 = {a: 10, b: 2, c:22}
const data3 = {a: 15, b: 20, c:10, d: 13}

// set the dimensions and margins of the graph
const width = 450,
    height = 450,
    margin = 40;

const radius = Math.min(width, height) / 2 - margin;

// set the color scale
const color = d3.scaleOrdinal()
    .domain(["a", "b", "c", "d"])
    .range(d3.schemeDark2);
```

```
function update(data) {
    // Compute the position of each group on the pie:
    const pie = d3.pie()
    .value(function(d) {return d[1]; })
    const data_ready = pie(Object.entries(data))

    // map to data
    const u = svg.selectAll("path")
    .data(data_ready)

    // Build the pie chart
    u.join('path')
    .transition()
    .duration(1000)
    .attr('d', d3.arc()
        .innerRadius(0)
        .outerRadius(radius)
    )
    .attr('fill', function(d){ return(color(d.data[0])) })
    .attr("stroke", "white")
    .style("stroke-width", "2px")
    .style("opacity", 1)
}
```





PLEASE LET ME KNOW IF ANY PAGE CANNOT BE OPEN

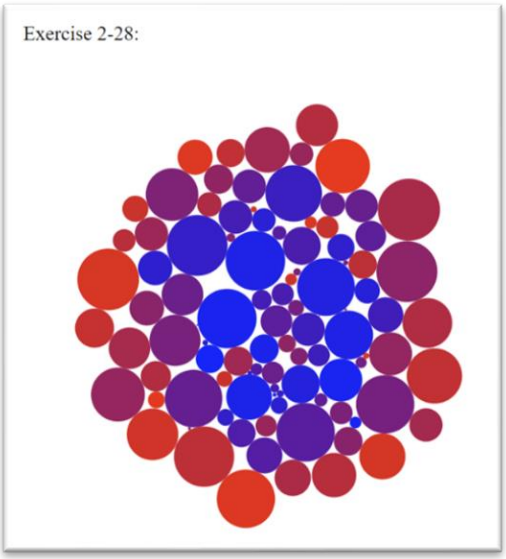
BECAUSE JSON AND CSV HOST ON GITHUB PRIVATE REPO, SO THE TOKEN MAY REFRESH.

yw2007@hw.ac.uk

Exercise 28:

As same as the e27, I define a color variable and use it in function ticked when fill circle.

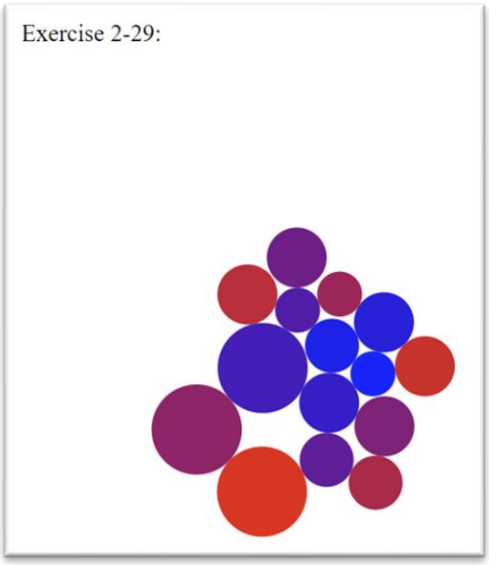
```
var color = d3.scaleLinear()
        .domain([0,numNodes])
        .range(["blue", "red"]);
……

        .join('circle')
        .attr('fill', function(d,i) {
        return color(i)
        })
        .attr('r', function(d) {
```



Exercise 29:

I create a json file to save the radius, and load it by d3.json.

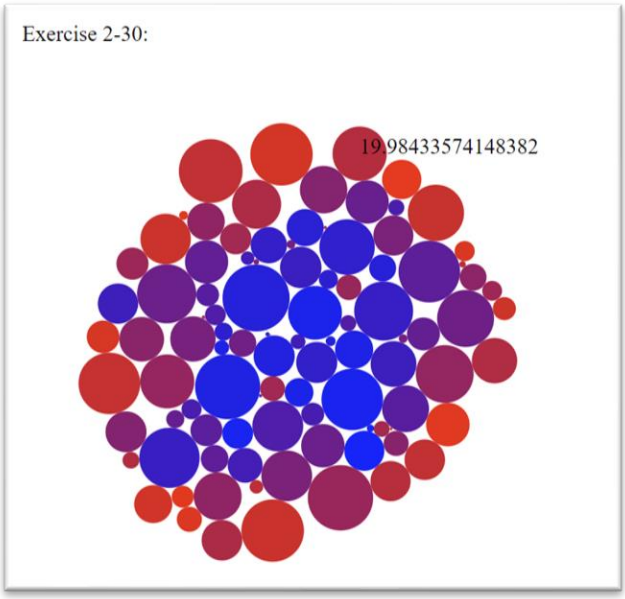
```
var nodes = []
json =
"https://raw.githubusercontent.com/JamesW99/JamesW99.github.io/main/l
ab2/json2-1.json?token=GHSAT0AAAAAABN4OYQU6ALIOSG353MPNQMGYRBYJUA"
var node = d3.json(json, {}).then((nodes) => {
    console.log("callback", nodes)


var numNodes = nodes.length;
```



Exercise 30:

As same as e15 , I add the mouse over and out event, mouseover will print radius according to circle position and mouseout will remove text.

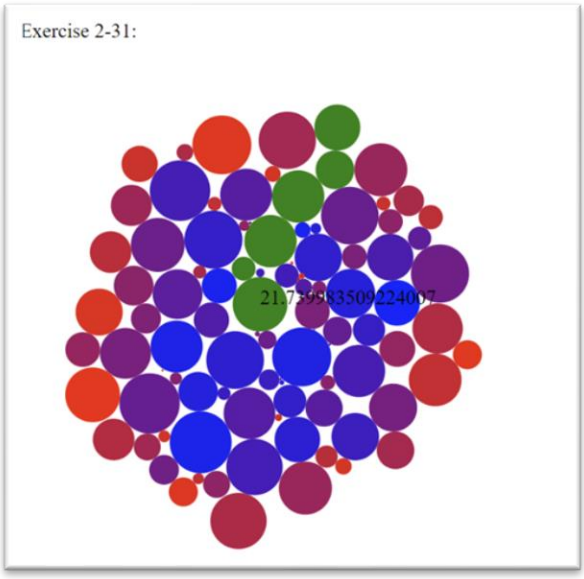
```
function onMouseOver(event) {
    var circle = d3.select(this)
    d3.select('svg')
        .append("text")
        .attr('class', 'val')
        .text( function(d) {    // radius
            return circle.attr("r")
        })
        .attr('x', function(d) { // x axis
            return circle.attr("cx")
        })
        .attr('y', function(d) { // y axis
            return circle.attr("cy")
        })
}

function onMouseOut(event) {
    d3.select('.val').remove()
}
```



Exercise 31:

Fill green on mouse over.

```
var circle = d3.select(this)
    circle.attr("fill", "green")
    d3.select('svg')
```



Exercise 32:

According to D3 documentation, I call d3 drag in ticked function. And define dragstarted dragged dragended to control when drag

```
    .attr('cy', function(d) {
    return d.y
    })
    .call(d3.drag()
        .on("start", dragstarted)
        .on("drag", dragged)
        .on("end", dragended));
}

function dragstarted(event, d) {
    if (!event.active) simulation.alphaTarget(.03).restart();
        d.fx = d.x;
        d.fy = d.y;
}
function dragged(event, d) {
    d.fx = event.x;
    d.fy = event.y;
}
function dragended(event, d) {
    if (!event.active) simulation.alphaTarget(0);
    d.fx = null;
    d.fy = null;
}
```