F21DV - Data Visualization and Analytics

Coursework Lab 2

Due on Friday 25 Feb 2022

Submitted By: Shashank Ayanikkatt Vengalapurath

Demonstrated to: Xue Shuangjiang

Demonstration on: 25 Feb 2022

School of Mathematical and Computer Sciences

Heriot-Watt University

Table of Contents

Overview…… 4

Part 1. CSS Effects /Animations…… 5

Exercise 1…… 5

Exercise 2……6

Part 2. Events……7

Exercise 3…… 7

Exercise 4…… 8

Exercise 5…… 9

Part 3. D3 Transitions……10

Exercise 6…… 10

Exercise 7…… 11

Exercise 8…… 12

Exercise 9…… 13

Exercise 10…… 14

Exercise 11…… 15

Exercise 12…… 16

Exercise 13…… 17

Exercise 14…… 18

Part 4. Animated Chart……19

Exercise 15…… 19

Exercise 16…… 20

Exercise 17…… 21

Part 5. Changing Data and Transitioning……22

Exercise 18…… 22

Exercise 19…… 23

Exercise 21…… 24

Part 6. Pie Chart……25

Exercise 24…… 25

Exercise 25…… 26

Exercise 26…… 27

Part 7. D3 Force Layout……28

Exercise 28…… 28

Exercise 31…… 29

Overview:

The purpose of this course work is getting an understanding of how interactive visualization works in web pages, get a more understanding of how animations and transition effects work, how one can dynamically update/change data and its visualization and to generate an interactive graphs/graphics using the d3 library.

Part 1. CSS Effects/Animations

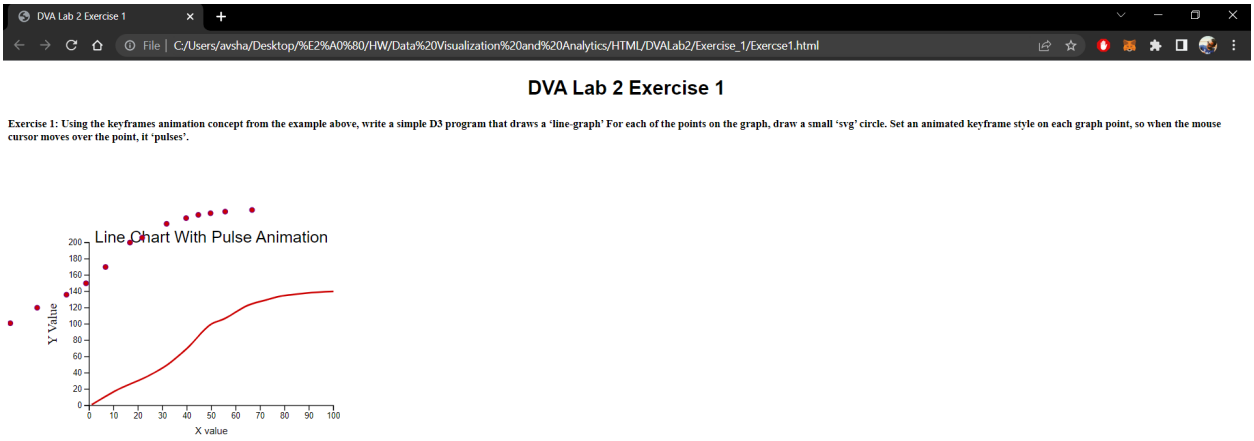
Exercise 1:

Using the keyframes animation concept from the example above, write a simple D3 program that draws a 'line-graph' For each of the points on the graph, draw a small 'svg' circle. Set an animated keyframe style on each graph point, so when the mouse cursor moves over the point, it pulses.

Snippet:

```
<style>
    .pulse {                          /* The style is used to pulse class    */
      border: 4px solid blue;         /* will have a border color of blue with 4px size */
      fill: lightblue;                /* with the colour blue */
      stroke: purple;
      -webkit-transform: scale(1);
      -webkit-transform-origin: 50% 50%; /* will postion the pusle */
      transform: scale(1);
      transform-origin: 10% 100%;
    }

    .pulse:hover              /* when the mouse hover over  */
    {
      -webkit-animation-name: pulsar;
      -webkit-animation-duration: 1s;
      -webkit-animation-iteration-count: infinite;
      -webkit-animation-direction: alternate;
      animation-name: pulsar;
      animation-duration: 1s;
      animation-iteration-count: infinite;
      animation-direction: alternate;
      -webkit-transform-origin: 10% 100%;
      transform-origin: 10% 100%;
    }

    @keyframes pulsar {
      from {
          fill: red;
      }
      to {
          fill: red;
          transform: scale(2,3);
          transform-origin: 50% 50%;
      }
    }
    </style>
```

Output:



Exercise 2:

Create a webpage using D3 (adds items dynamically), then set the styles for the items so they use CSS to display extra information when the mouse cursor moves over them.

Snippet:

```
<style>

    div{                    /* style for the div */
        position: relative;
        display: none;      /* so that the div doest show at start */
    }

    h2 {

      padding: 20px;        /* for givin padding */
    }

    h2:hover + div {        /* when mouse hovers over the h2 */
      background-color: blue;
      position: absolute;
      display: inline-block;
    }
    </style>
```

Output:

DVA Lab 2 Exercise 2

Exercise 2: Create a webpage using D3 (adds items dynamically), then set the styles for the items so they use CSS to display extra information when the mouse cursor moves over them.

12

55

68

29

34

75

Part 2. Events

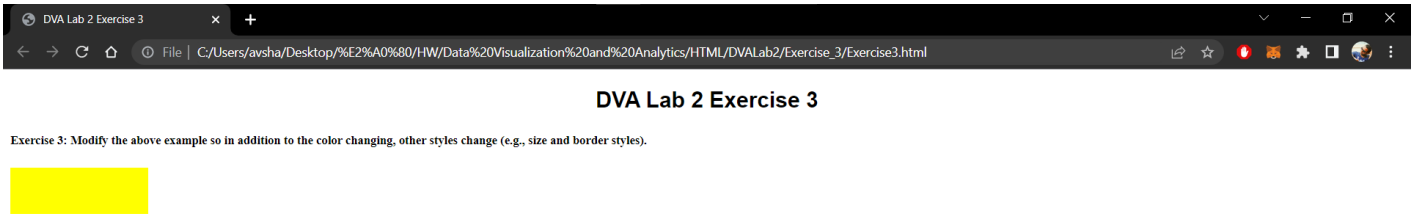
Exercise 3:

Modify the above example so in addition to the color changing, other styles change (e.g., size and border styles).

Snippet:

```
// Note: d3.mouse was removed in d3v6, you should use d3.pointer(event)
console.log(d3.pointer(event));
})
.on("mouseout", function(){   // When the mouse moves out of the div
d3.select(this)
.transition()
.duration(500)
.style("background-color", "yellow")
.style('width','150px')
.style('height','50px')
.style("stroke-dasharray", ("10,3"))
.style('stroke','purple')
.style('stroke-width',2)
});
</script>
```

Output:



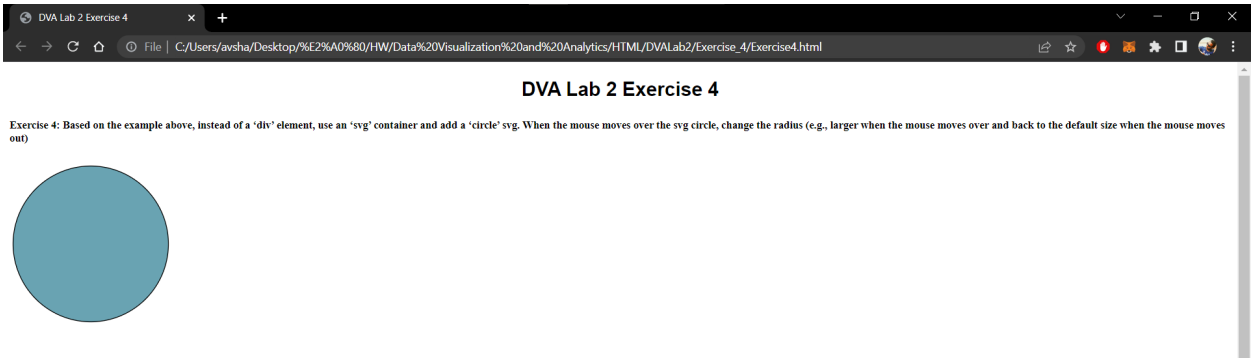
Exercise 4:

Based on the example above, instead of a ‘div’ element, use an ‘svg’ container and add a ‘circle’ svg. When the mouse moves over the svg circle, change the radius (e.g., larger when the mouse moves over and back to the default size when the mouse moves out).

Snippet:

```
var mycircle= svg.append('circle') // to create a circle with the dimensions
            .attr('cx', 100)
            .attr('cy', 100)
            .attr('r', 50)
            .attr('stroke', 'black')
            .attr('fill', '#69a3b2');



    mycircle            // the var in which the the circle is stored
        .on("mouseover",function(){     // when the mouse hovers over the circle
            d3.select(this)
                .transition()
                .duration(3000)     // will transition with duration of 3 sec
                .attr("r",100);     // will change the radius
        })

        .on("mouseout",function(){      // when the mouse hovers out
            d3.select(this)
                .transition()
                .duration(2000)
                .attr("r",50); // will chnage the radius to the original value
        })
```

Output:



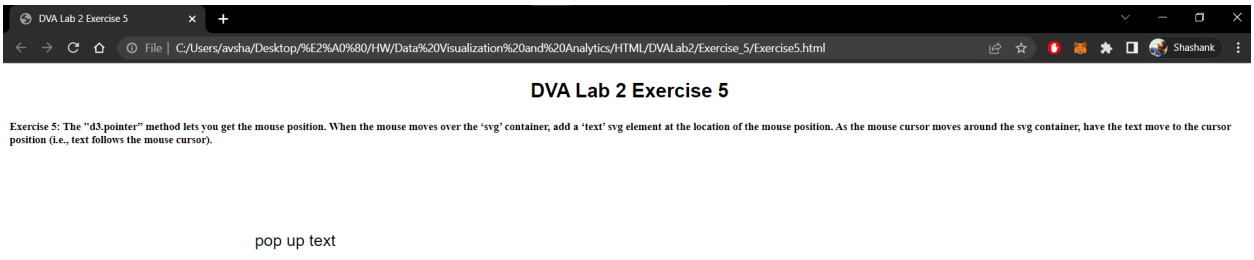
Exercise 5:

The "d3.pointer" method lets you get the mouse position. When the mouse moves over the 'svg' container, add a 'text' svg element at the location of the mouse position. As the mouse cursor moves around the svg container, have the text move to the cursor position (i.e., text follows the mouse cursor).

Snippet:

```
<script>

    var w="500";
    var h="500";

    var svg = d3.select("body")            // to create an svg element
                .append("svg")
                .attr("width",w)
                .attr("height",h);

   var text = svg.append('text')           // to append text
                .attr('x', w/2 + 100)
                .attr('y',100)
                .attr('text-anchor','middle')
                .style('font-family', 'Helvetica')
                .style('font-size',20)
                .text('pop up text')

                d3.selectAll("text")
                        .on("mouseover", function(event){   // when mouse comes over the text
                        d3.select(this)
                        console.log(d3.pointer(event));
                    })

</script>
```

Output:



Part 3. D3 Transitions

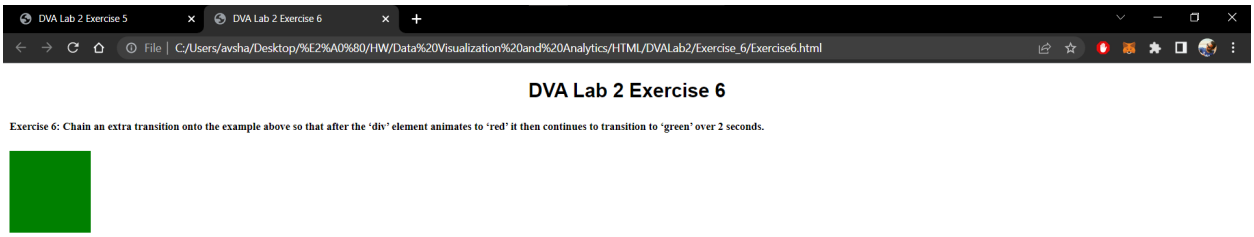
Exercise 6:

Chain an extra transition onto the example above so that after the ‘div’ element animates to ‘red’ it then continues to transition to ‘green’ over 2 seconds.

Snippet:

```
<script>
    d3.select('body')
      .append("div")    // will append a div with the following dimensions
      .style('width',  '100px')
      .style('height', '100px')
      .style('background-color', 'blue')  // with the color blue
      .transition()  // will add a transition effect
      .duration(1000)  // duration of 1 sec
      .style("background-color", "red") // to change the color red
      .transition()   // transition again
      .duration(2000)
      .style("background-color", "green");
    </script>
```

Output:



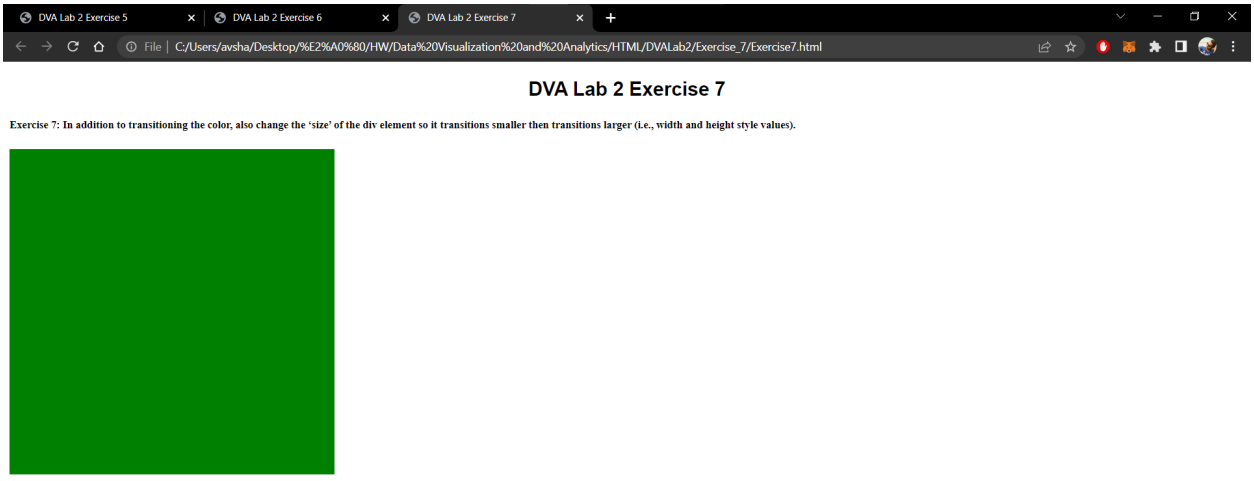
Exercise 7:

In addition to transitioning the color, also change the ‘size’ of the div element so it transitions smaller then transitions larger (i.e., width and height style values).

Snippet:

```
<script>

    d3.select('body')
      .append("div") // to append a div with the following dimensions
      .style('width',  '200px')
      .style('height', '200px')
      .style('background-color', 'blue') // with the color blue
      .transition()
      .duration(1000)
      .style('background-color', 'red')
      .style('width','50px') // this will chnage the width
      .style('height','50px') // will chnage the hight of the div
      .transition()
      .duration(2000) // the duration will be 2 sec
      .style('background-color', 'green') // will chnage the color
      .style('width','400px') // will chnage the width
      .style('height','400px');// will chnage the height
    </script>
```

Output:



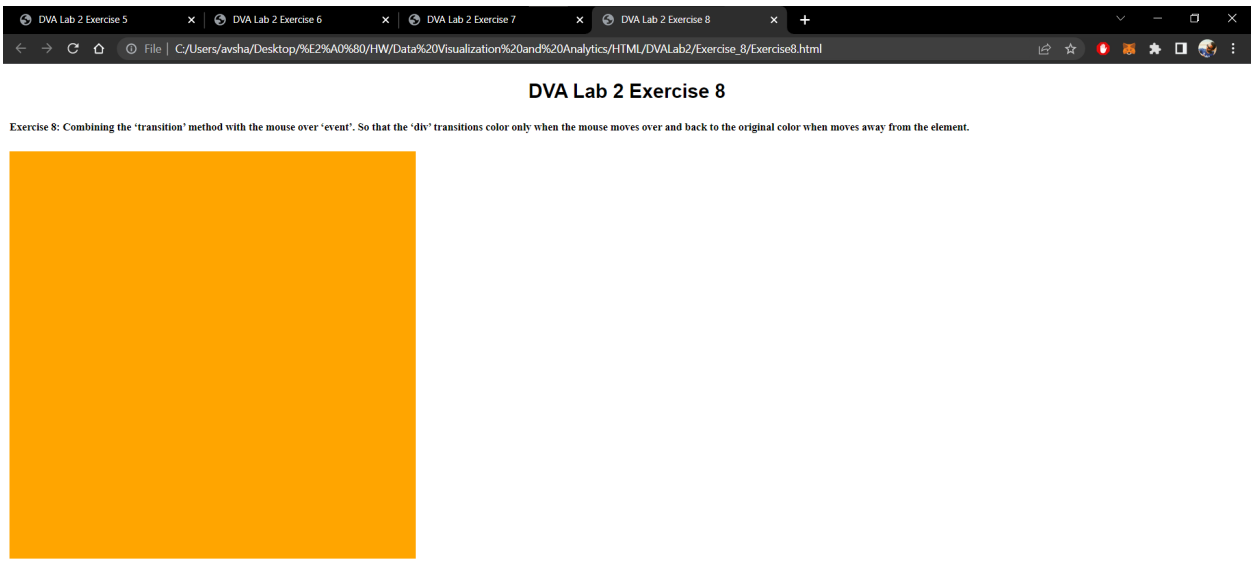
Exercise 8:

Combining the ‘transition’ method with the mouse over ‘event’. So that the ‘div’ transitions color only when the mouse moves over and back to the original color when moves away from the element.

Snippet:

```
mydiv =  d3.select('body')
            .append('div')      // will append a div with the follwing dimensions and style
            .style("width","300px")
            .style("height","300px")
            .style("background-color","blue");

 mydiv
    .on("mouseover",function(){     // when eht mouse is over the div
        d3.select(this)
            .transition()       // will have a transtion effect
            .duration(1000)     // of duration 1 sec and the following styles
            .style("width","500px")
            .style("height","500px")
            .style("background-color","orange");
    })

    .on("mouseout",function(){      // when the mouse comes out of the div it will have the following attributes
        d3.select(this)
            .transition()
            .duration(1000)
            .style("width","300px")
            .style("height","300px")
            .style("background-color","blue");
    })
```

Output:



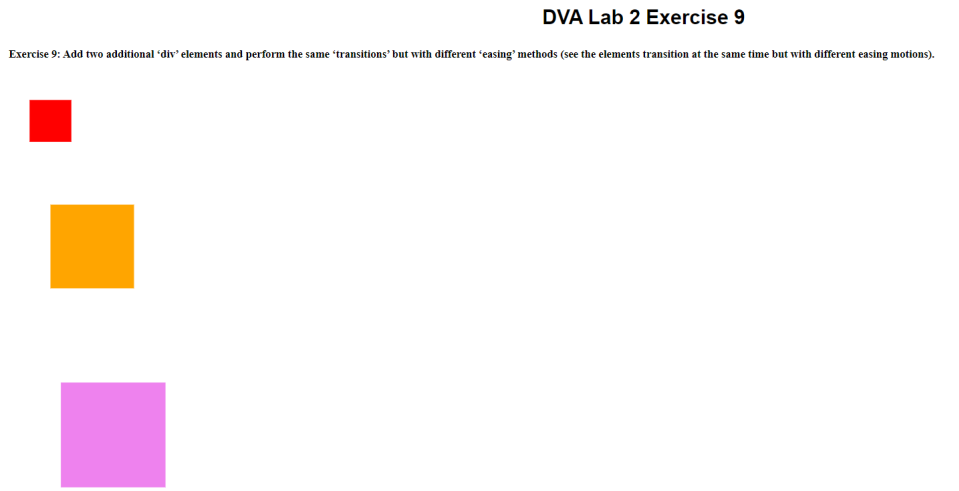
Exercise 9:

Add two additional ‘div’ elements and perform the same ‘transitions’ but with different ‘easing’ methods (see the elements transition at the same time but with different easing motions).

Snippet:

```
var check=  d3.select('body')
                .append("div")      // to append a div with the follwing attributes for the 1st box
                .style('width',  '100px')
                .style('height', '100px')
                .style('background-color', 'blue')
                .style('transform', 'scale(1.0)')
                .transition()
                .ease( d3.easeBounce ) // will have a transtion with ease bounce effect
                .duration(1000)
                .style("background-color", "red")
                .style('transform', 'scale(0.5)')

 var design=  d3.select('body') // to append a div with the follwing attributes for the 2nd box
                .append("div")
                .style('width',  '200px')
                .style('height', '200px')
                .style('background-color', 'yellow')
                .style('transform', 'scale(1.0)')
                .transition()
                .ease( d3.easeElastic ) // will have a ease elastic effect
                .duration(1000) //duration of 1 sec
                .style("background-color", "orange")
                .style('transform', 'scale(0.5)') // to scale the size

 var box=  d3.select('body') // to append a div with the follwing attributes for the 3rd box
                .append("div")
                .style('width',  '250px')
                .style('height', '250px')
                .style('background-color', 'green')
                .style('transform', 'scale(1.0)')
                .transition()
                .ease( d3.easeCubic ) // will have ease cubix effect
                .duration(1000)
                .style("background-color", "violet")
                .style('transform', 'scale(0.5)')
```

Output:



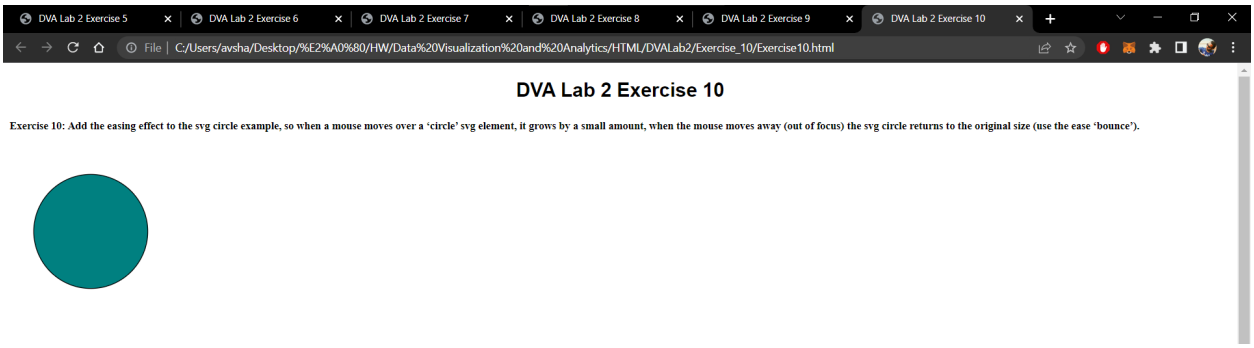
Exercise 10:

Add the easing effect to the svg circle example, so when a mouse moves over a ‘circle’ svg element, it grows by a small amount, when the mouse moves away (out of focus) the svg circle returns to the original size (use the ease ‘bounce’).

Snippet:

```
mycircle
    .on("mouseover",function(){     // when the mouse hovers over the circle
        d3.select(this)
            .transition()
            .duration(2000)
            .ease( d3.easeBounce ) // will have ease bounce effect
            .attr("r",75);      // with radius given
    })

    .on("mouseout",function(){      // when the mouse hovers out the circle
        d3.select(this)
            .transition()
            .duration(2000)
            .ease( d3.easeBounce ) // with ease bounce effect
            .attr("r",50);      // with the radius
    })
```

Output:



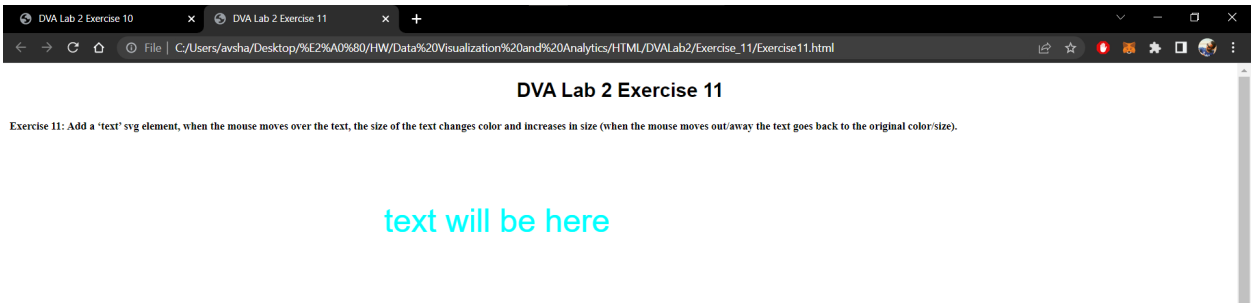
Exercise 11:

Add a ‘text’ svg element, when the mouse moves over the text, the size of the text changes color and increases in size (when the mouse moves out/away the text goes back to the original color/size).

Snippet:

```
mytext
    .on("mouseover",function(){     // when mouse hovers over the text
        d3.select(this)
          .transition()       // will have a transtion
          .duration(500)      // duration of 0.5 sec
          .style('font-size','40px')  // with font size
          .style("fill", "#00FFFF")   // color
          console.log(event);     // will show the log event when the mouse is over the circle

    })

mytext
    .on("mouseout",function(){      // when mouse hovers out the text
        d3.select(this)
          .transition()
          .duration(500)      // duration of 0.5 sec
          .style('font-size','30px')      // will have a size
          .style("fill", "#004669");      // the following color

    })
```

Output:



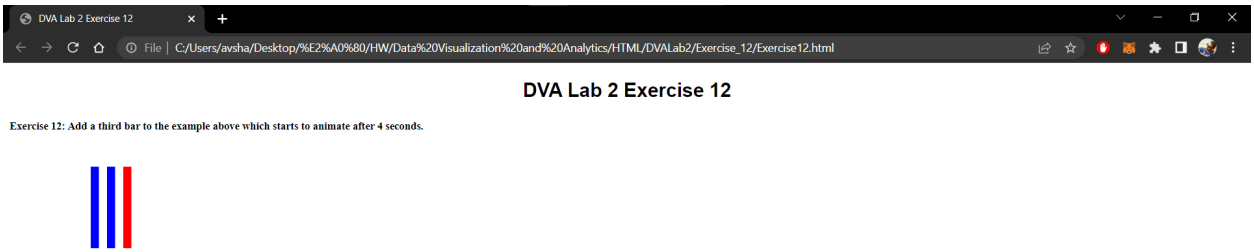
Exercise 12:

Add a third bar to the example above which starts to animate after 4 seconds.

Snippet:

```
function update() {
    bar1.transition()  // bar 1 will have the below transition
        .ease(d3.easeLinear)    // with ease Linear  effect
        .duration(2000)     // duration of 2 sec
    .attr("height",100)

    bar2.transition()  // bar 2 will have the below transition
        .ease(d3.easeLinear) // with ease Linear  effect
        .duration(2000)
        .delay(2000)    // delay with 2 sec
        .attr("height",100)


    bar3.transition() // bar 3 will have the below transition
        .ease(d3.easeLinear) // with ease Linear  effect
        .duration(2000)
        .delay(4000)        // delay 4 sec
        .attr("height",100)
            }
```

Output:



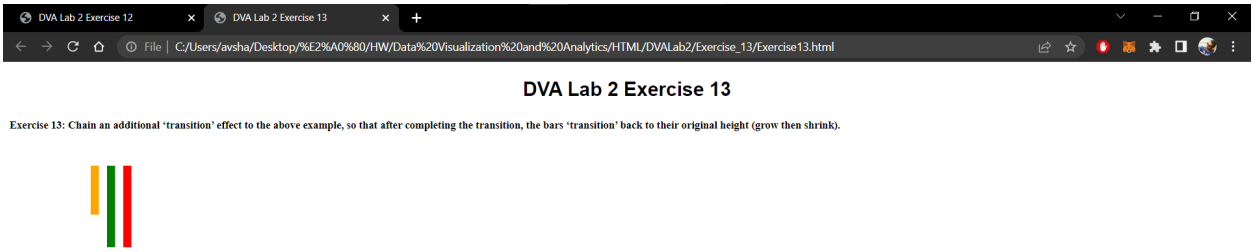
Exercise 13:

Chain an additional ‘transition’ effect to the above example, so that after completing the transition, the bars ‘transition’ back to their original height (grow then shrink).

Snippet:

```
bar3.transition() // bar 3 will have the following transitoin
    .ease(d3.easeLinear)
    .duration(2000)
    .delay(4000) // delay of 4 sec
    .attr("height",100)


bar1.transition()
    .ease(d3.easeLinear)
    .duration(2000)
    .delay(6000)  // delay of 6 sec
    .attr("height",20)
    console.log("hjiiii")

bar2.transition()
    .ease(d3.easeLinear)
    .duration(2000)
    .delay(8000)        // delay of 8 sec
    .attr("height",20)

bar3.transition()
    .ease(d3.easeLinear)
    .duration(2000)
    .delay(10000)       // dealy of 10 sec
    .attr("height",20)
```

Output:



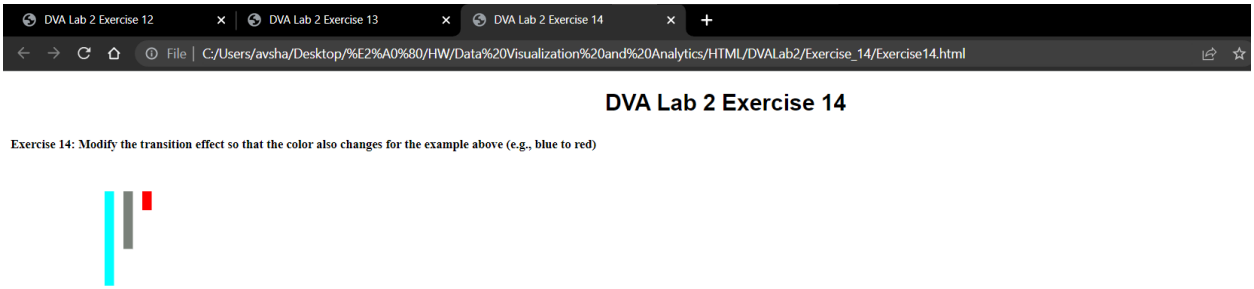
Exercise 14:

Modify the transition effect so that the color also changes for the example above (e.g., blue to red).

Snippet:

```
bar1.transition()
    .ease(d3.easeLinear)    // bar 1 will have a ease linaer transiton
    .duration(2000)
    .delay(6000)        // delay of 6 sec
    .attr("height",20) // height
    .attr("fill","black")   // color
    console.log("hjiiii")

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

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

Output:



Part 4. Animated Chart

Exercise 15:

Take the above example, and add in the following mouse events below:

e.g., .on("mouseover", onMouseOver) //Add listener for the mouseover event

.on("mouseout", onMouseOut) //Add listener for the mouseout event

Snippet:

```
function onMouseOver(d, i) {

d3.select(this).attr('class', 'highlight');
d3.select(this)
.transition()     // adds animation
          .duration(400)
          .attr('width', x.bandwidth() + 5)
          .attr("y", function(d) { return y(d.value) - 10; })
          .attr("height", function(d) { return height - y(d.value) + 10; });

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

//mouseout event handler function
function onMouseOut(d, i) {
        // use the text label class to remove label on mouseout
        d3.select(this).attr('class', 'bar');
        d3.select(this)
          .transition()     // adds animation
          .duration(400)
          .attr('width', x.bandwidth())
          .attr("y", function(d) { return y(i.value); })
          .attr("height", function(d) { return height - y(i.value); });

        d3.selectAll('.val')
          .remove()
}
```

Output:

DVA Lab 2 Exercise 15

Exercise 15: Take the above example, and add in the following mouse events below:
/*
e.g., .on("mouseover", onMouseOver) //Add listener for the mouseover event
.on("mouseout", onMouseOut) //Add listener for the mouseout event
***/**
//mouseover event handler function
function onMouseOver(d, i) {
d3.select(this).attr('class', 'highlight');
d3.select(this)

Exercise 16:

Modify the example above so the popup text that is displayed when the mouse cursor moves over each bar is positioned ‘above’ the bar instead of the top left.

Snippet:

```
g.append("text")
.attr('class', 'val')
.attr('x', function() {
return x(d.year);
})
.attr('y', function() {
return y(d.value) - 15;
})
.text( function(d) { return '$' + i.value; } ); // Value of the text
}
//mouseout event handler function
function onMouseOut(d, i) {
// use the text label class to remove label on mouseout
d3.select(this).attr('class', 'bar');

d3.select(this)
.transition() // adds animation
.duration(400)
.attr('width', x.bandwidth())
.attr("y", function(d) { return y(i.value); })
.attr("height", function(d) { return height - y(i.value); });
d3.selectAll('.val')
.remove()
}
```

Output:

DVA Lab 2 Exercise 15

Exercise 15: Take the above example, and add in the following mouse events below:
/*
e.g., .on("mouseover", onMouseOver) //Add listener for the mouseover event
.on("mouseout", onMouseOut) //Add listener for the mouseout event
***/**
//mouseover event handler function
function onMouseOver(d, i) {
d3.select(this).attr('class', 'highlight');
d3.select(this)

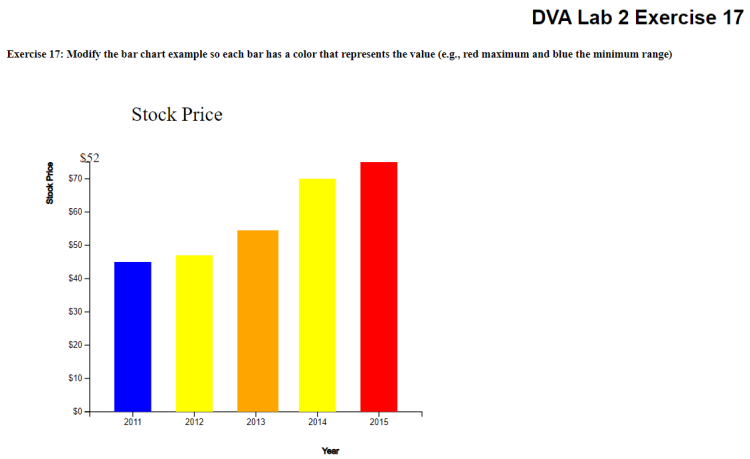
Exercise 17:

Modify the bar chart example so each bar has a color that represents the value (e.g., red maximum and blue the minimum range)

Snippet:

```
g.selectAll(".bar")
.data(data)
.enter().append("rect")
.attr("class", "bar")
// .on(….. ) - call mouse events here…
.on("mouseover",onMouseOver )
.on("mouseout",onMouseOut)
.attr("x", function(d) { return x(d.year); })
.attr("y", function(d) { return y(d.value); })
.attr("width", x.bandwidth())
.transition()
.ease(d3.easeLinear)
.duration(400)
.delay(function (d, i) {
return i * 50;
})
.attr("fill", function(d){ // to change the bar color according to value
    if(d.value > max){ return "red" ;}
    else if(d.value< min) { return "blue"}
    else { return "yellow"}

})
.attr("height", function(d) { return height - y(d.value); });
});
```

Output:



Part 5. Changing Data and Transitioning

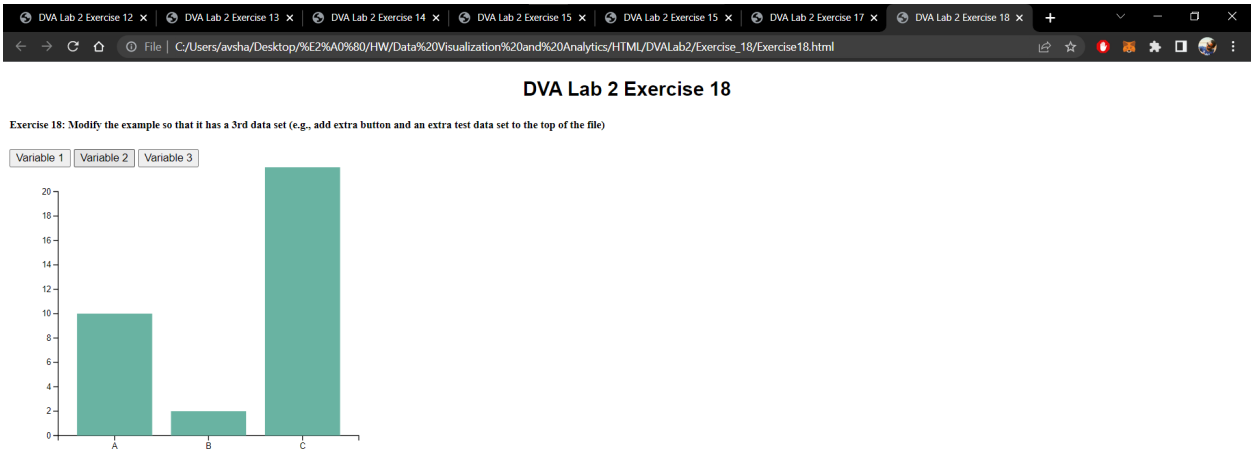
Exercise 18:

Modify the example so that it has a 3rd data set (e.g., add extra button and an extra test data set to the top of the file)

Snippet:

```
    <!-- Add buttons -->
<button onclick="update(data1)">Variable 1</button>
<button onclick="update(data2)">Variable 2</button>
<button onclick="update(data3)">Variable 3</button>

<script>
// create 2 data_set
const data1 = [
    {group: "A", value: 5},
    {group: "B", value: 20},
    {group: "C", value: 9}
];

const data2 = [
    {group: "A", value: 10},
    {group: "B", value: 2},
    {group: "C", value: 22}
];

const data3 = [
    {group: "A", value: 20},
    {group: "B", value: 15},
    {group: "C", value: 3}
];
```

Output:



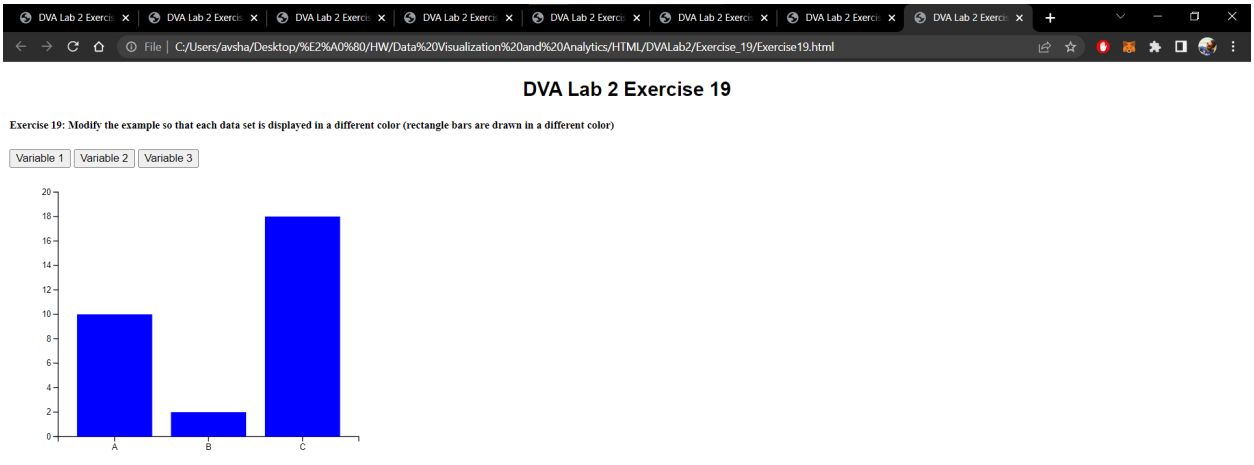
Exercise 19:

Modify the example so that each data set is displayed in a different color (rectangle bars are drawn in a different color)

Snippet:

```
u .enter()
    .append("rect")
    .merge(u)
    .transition()
    .duration(1000)
     .attr("x", function(d) { return x(d.group); })
     .attr("y", function(d) { return y(d.value); })
     .attr("width", x.bandwidth())
     .attr("height", function(d) { return height - y(d.value); })
     .attr("fill",function(){      // to change the color of the bars
        if(called == 1){return "red";}
          else if(called == 2){return "blue";}
              else {return "green";}

     })
}
```

Output:



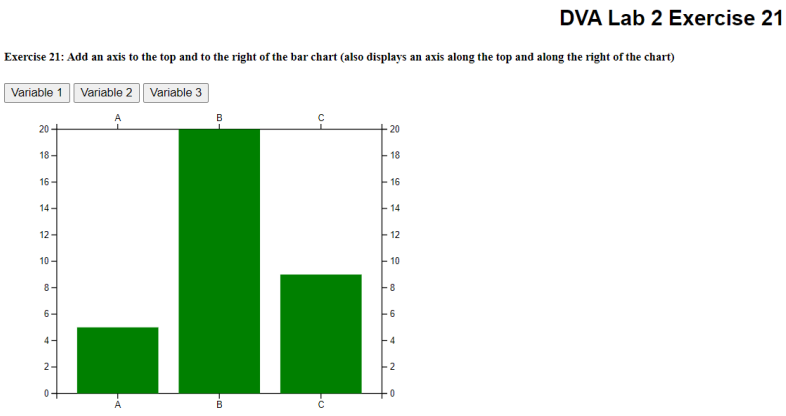
Exercise 21:

Add an axis to the top and to the right of the bar chart (also displays an axis along the top and along the right of the chart)

Snippet:

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

  var xTop = 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(xTop))

// Add Y axis
var y = d3.scaleLinear()
.domain([0, 20])
  .range([ height, 0]);
svg.append("g")
  .attr("class", "myYaxis")
  .call(d3.axisLeft(y));

  var yRight = d3.scaleLinear()
.domain([0, 20])
  .range([ height, 0]);
svg.append("g")
  .attr("class", "myYaxis")
  .attr("transform", "translate(" + width + "," + 0 + ")")
  .call(d3.axisRight(yRight));
```

Output:



Part 6. Pie Chart

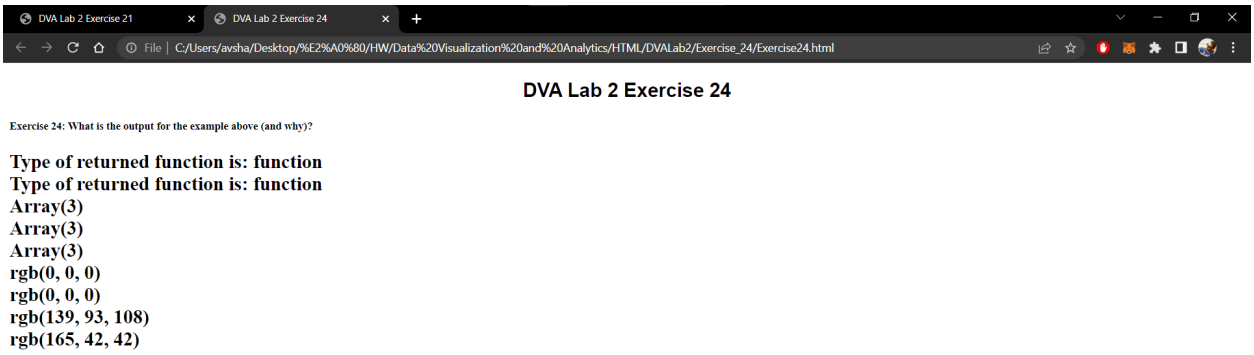
Exercise 24:

What is the output for the example above (and why)?

Snippet:

```
<script>
        let intr = d3.interpolate( [20, 40, 4], [1, 12, 10])

        const i = d3.interpolateRgb({colors: ["red", "blue"]}, {colors: ["White", "black"]});

        var check = d3.interpolateLab("steelblue", "brown");

        console.log("Type of returned function is: ", typeof (intr) );

        console.log("Type of returned function is: ", typeof (i) );

        console.log( intr(0.2) )

        console.log( intr(0.6) )

        console.log( intr(1) )

        console.log( i(0.5) )

        console.log( i(1.0) )

        console.log( check(0.5) )

        console.log( check(1) )
</script>
```

Output:



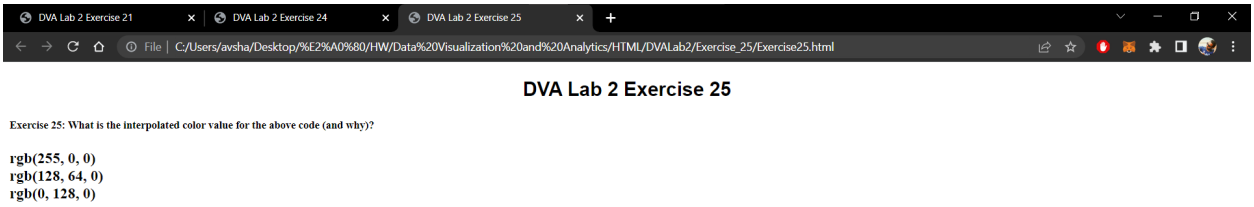
Exercise 25:

What is the interpolated color value for the above code (and why)?

Snippet:

```
<script>
    let cc = d3.interpolate('red', 'green')
    console.log( cc(0)  );
    console.log( cc(0.5)  );
    console.log( cc(1)  );

</script>
```

Output:



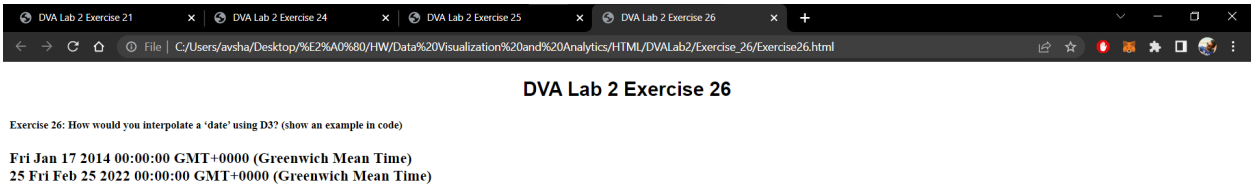
Exercise 26:

How would you interpolate a ‘date’ using D3? (Show an example in code)

Snippet:

```
<script>
        let  date = d3.interpolateDate(new Date("1995-02-17"), new Date("2022-02-025"))

        console.log( date(0.3)  );

        console.log( date(1.0)  );


</script>
```

Output:



Part 8. D3 Force Layout

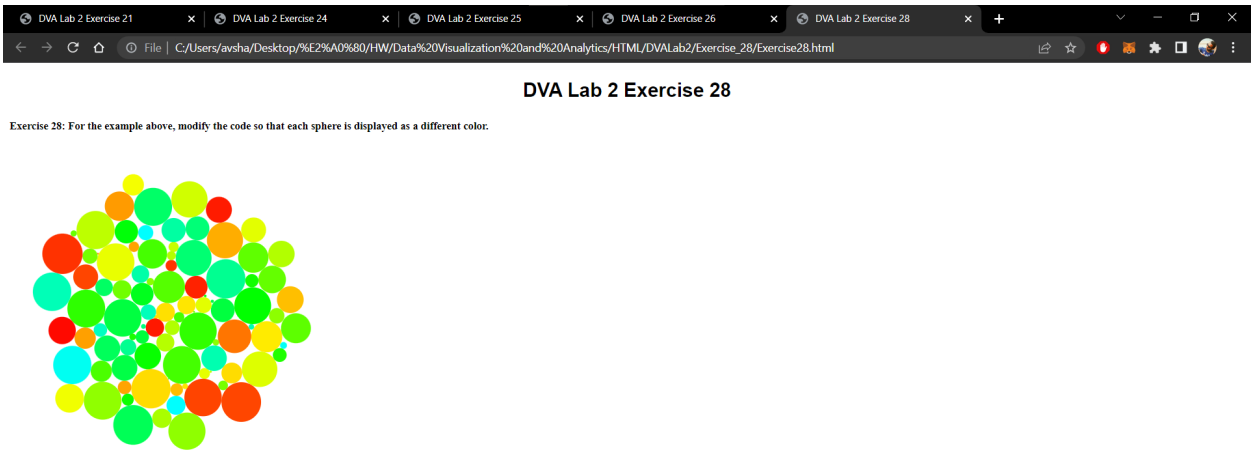
Exercise 28:

For the example above, modify the code so that each sphere is displayed as a different color.

Snippet:

```
function ticked() {
  var u = d3.select('svg')
    .selectAll('circle')
    .data(nodes)
    .join('circle')
    .attr("fill", function () {
        return "hsl(" + Math.random() * 180 + ",100%,50%)"; // giving color to circles
            })
    .attr('r', function(d) {
      return d.radius
    })
    .attr('cx', function(d) {
      return d.x
    })
    .attr('cy', function(d) {
      return d.y
    })
}
```

Output:



Exercise 31:

Modify the example, so the spheres change color when the mouse moves over them

Snippet:

```
function ticked() {
  var u = d3.select('svg')
    .selectAll('circle')
    .data(nodes)
    .join('circle')
    .attr("fill","yellow")
    .attr('r', function(d) {
      return d.radius
    })
    .attr('cx', function(d) {
      return d.x
    })
    .attr('cy', function(d) {
      return d.y
    })

    .on("mouseover", function () {  // on mouse on fucntion
    d3.select(this).attr("fill", "red");
    })

    .on("mouseout", function () { // on mouse out fucntion
    d3.select(this).attr("fill", "blue");
    })
}
```

Output: