CSE 564 Mini Project 1 Manisha Mishra

Aim: To parse the data from a CSV file and implement the following using d3.js:

1. Binning the variables in the data into a fixed range.
2. Creating a bar chart of the variables picked above.
3. Allowing users to select another variable from the menu to update the chart.
4. Display the values on top of the bar only on mouse-over.
5. Making the selected bar wider and higher on mouse-over.
6. On mouse-click, transforming the bar graph into a pie chart and back.
7. Increasing/Decreasing bar graph width when mouse moves right/left.

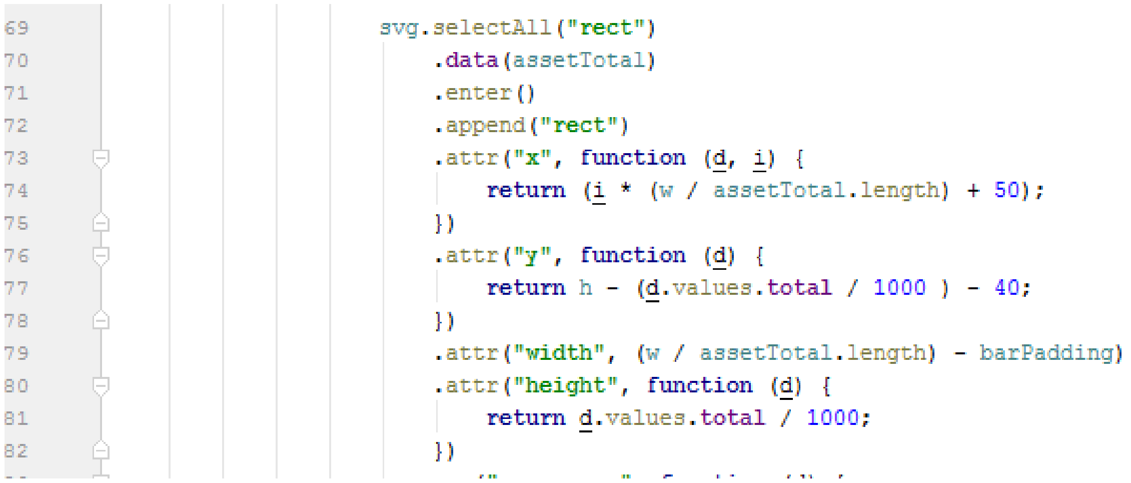
CSV File chosen: The file has data on the 248 largest Canadian firms with publicly available information in the mid-1970s. Information is given about the assets owned by the companies (in millions of USD), the industrial sector it falls into, country of ownership and the number of interlocking director and executive positions shared with other major firms.

`https://vincentarelbundock.github.io/Rdatasets/doc/car/Ornstein.html`
`https://vincentarelbundock.github.io/Rdatasets/datasets.html`

Implementation:

1. **Binning Variables:**
   Assets and Sectors – Total assets owned by Companies in each industrial sector.
   Nations vs Interlocks – Number of interlocks in companies owned by different countries.

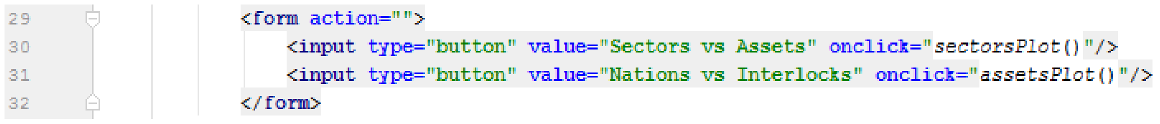
2. **Creating the Bar Chart:**
   Data from the file is mapped to the respective variables, after which an SVG element is created. The following snippet implements the bar chart function.



```
svg.selectAll("rect")
    .data(assetTotal)
    .enter()
    .append("rect")
    .attr("x", function (d, i) {
        return (i * (w / assetTotal.length) + 50);
    })
    .attr("y", function (d) {
        return h - (d.values.total / 1000 ) - 40;
    })
    .attr("width", (w / assetTotal.length) - barPadding)
    .attr("height", function (d) {
        return d.values.total / 1000;
    })
```

Fig. 1

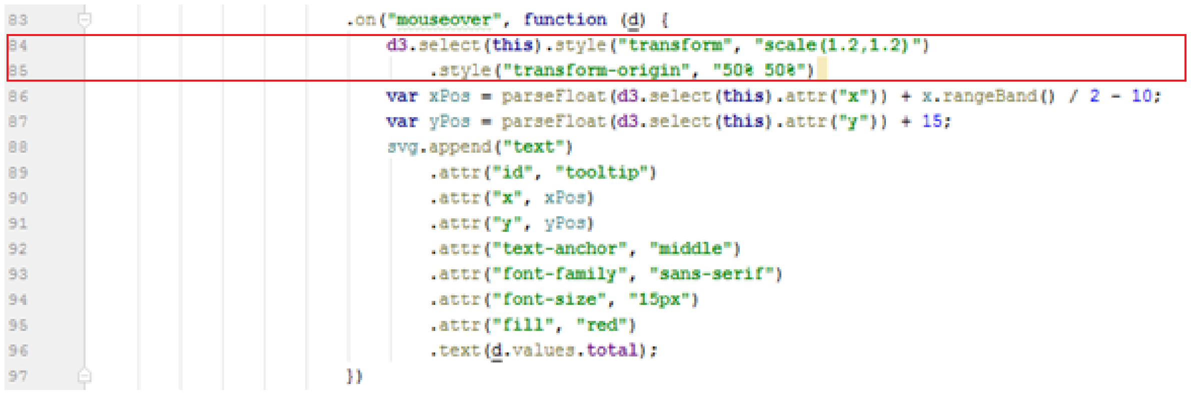
3. **Changing variables:**
   The code shown below implements the button functionality. On click, the functions responsible for implementing the bar charts of the respective variables are called.



```
<form action="">
    <input type="button" value="Sectors vs Assets" onclick="sectorsPlot()"/>
    <input type="button" value="Nations vs Interlocks" onclick="assetsPlot()"/>
</form>
```

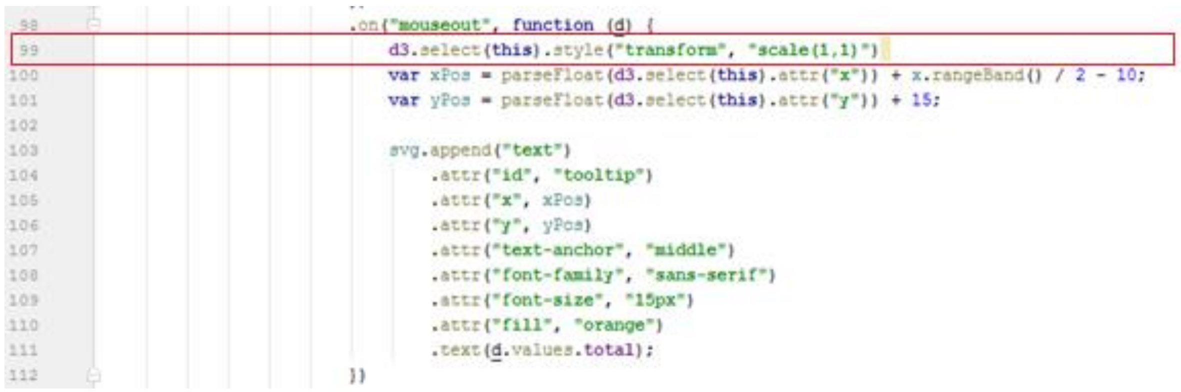
Fig. 2

4. **Displaying values only on mouse-over:**



```
83                .on("mouseover", function (d) {
84                    d3.select(this).style("transform", "scale(1.2,1.2)")
85                        .style("transform-origin", "50% 50%")
86                    var xPos = parseFloat(d3.select(this).attr("x")) + x.rangeBand() / 2 - 10;
87                    var yPos = parseFloat(d3.select(this).attr("y")) + 15;
88                    svg.append("text")
89                        .attr("id", "tooltip")
90                        .attr("x", xPos)
91                        .attr("y", yPos)
92                        .attr("text-anchor", "middle")
93                        .attr("font-family", "sans-serif")
94                        .attr("font-size", "15px")
95                        .attr("fill", "red")
96                        .text(d.values.total);
97                })
```

Fig. 3

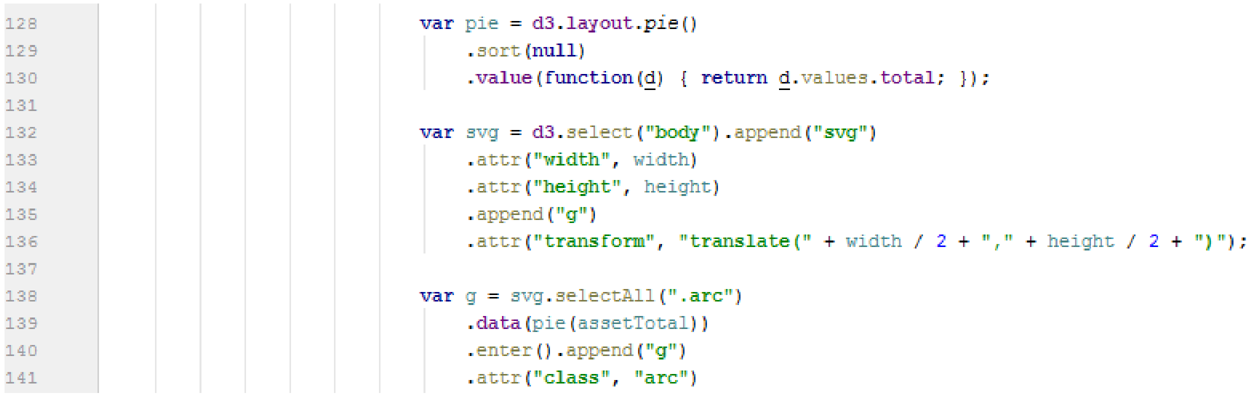


```
98                .on("mouseout", function (d) {
99                    d3.select(this).style("transform", "scale(1,1)")
100                   var xPos = parseFloat(d3.select(this).attr("x")) + x.rangeBand() / 2 - 10;
101                   var yPos = parseFloat(d3.select(this).attr("y")) + 15;
102
103                   svg.append("text")
104                       .attr("id", "tooltip")
105                       .attr("x", xPos)
106                       .attr("y", yPos)
107                       .attr("text-anchor", "middle")
108                       .attr("font-family", "sans-serif")
109                       .attr("font-size", "15px")
110                       .attr("fill", "orange")
111                       .text(d.values.total);
112               })
```

Fig. 4

5. **Making the bar higher and wider on mouse-over:**
   The boxes highlighted in red in Fig. 3 & 4 implement this function. Upon mouse-over, the selected bar increases by 20% (Line 84) and reduces to its original size upon mouse-out (Line99).

6. **Transforming into a pie-chart and back upon mouse-click:**
   Fig. 5 is implemented using the ***.on("click")*** function in the bar graph. The size of the pie chart, radius etc. are also calculated in this function. When the pie-chart is clicked, the function to display the bar chart of the selected variable is called.



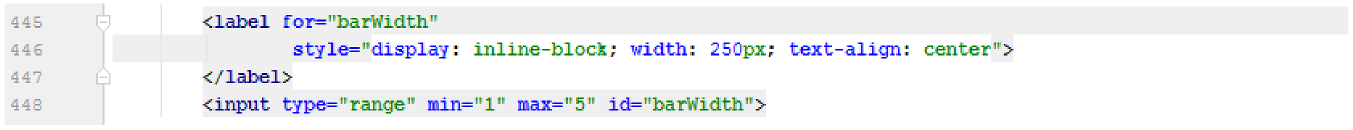
```
var pie = d3.layout.pie()
    .sort(null)
    .value(function(d) { return d.values.total; });

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height)
    .append("g")
    .attr("transform", "translate(" + width / 2 + "," + height / 2 + ")");

var g = svg.selectAll(".arc")
    .data(pie(assetTotal))
    .enter().append("g")
    .attr("class", "arc")
```

Fig. 5

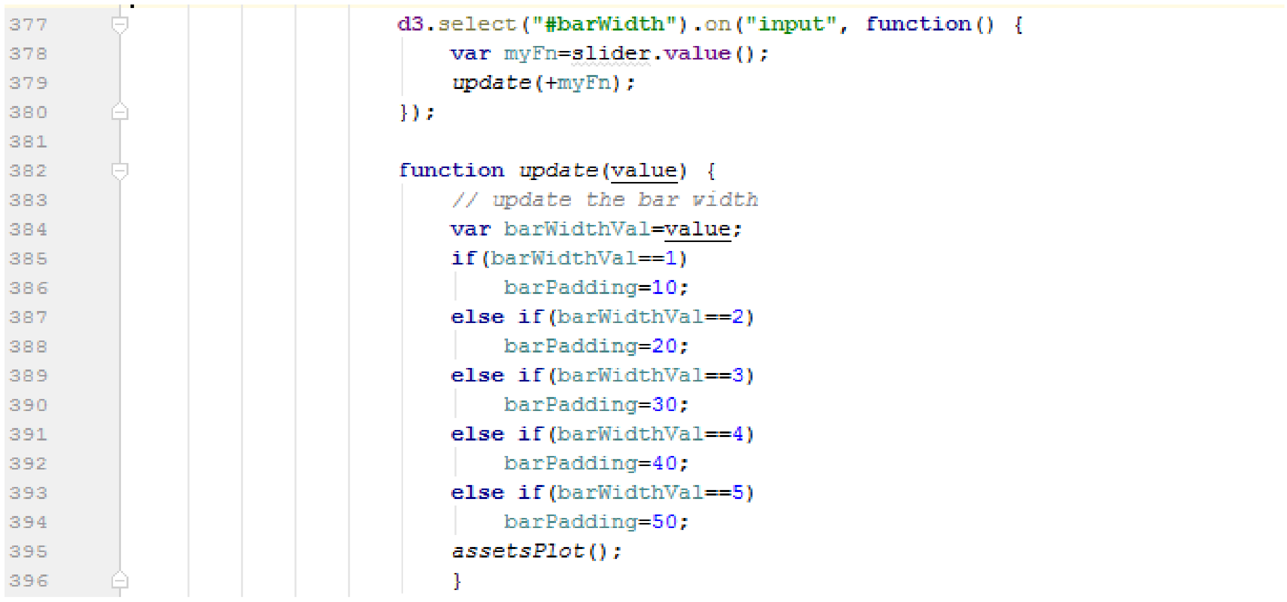
7. **Increasing/Decreasing bar graph width when mouse moves right/left**
   Implemented using a slider as shown in Fig. 6.



```
<label for="barWidth"
        style="display: inline-block; width: 250px; text-align: center">
</label>
<input type="range" min="1" max="5" id="barWidth">
```

Fig. 6

As the slider moves, the value of the variable *barWidth* is updated. With this updated value, the bar graph is displayed.



```
d3.select("#barWidth").on("input", function() {
    var myFn=slider.value();
    update(+myFn);
});

function update(value) {
    // update the bar width
    var barWidthVal=value;
    if(barWidthVal==1)
        barPadding=10;
    else if(barWidthVal==2)
        barPadding=20;
    else if(barWidthVal==3)
        barPadding=30;
    else if(barWidthVal==4)
        barPadding=40;
    else if(barWidthVal==5)
        barPadding=50;
    assetsPlot();
    }
```

Fig. 7