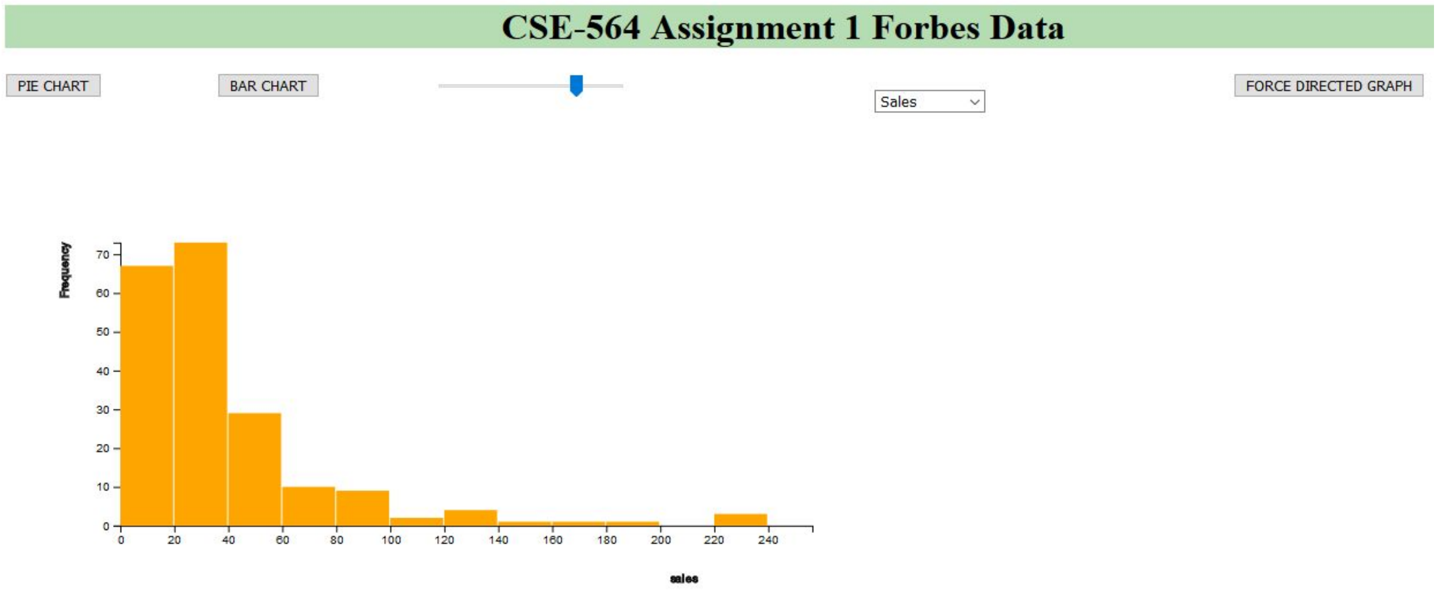
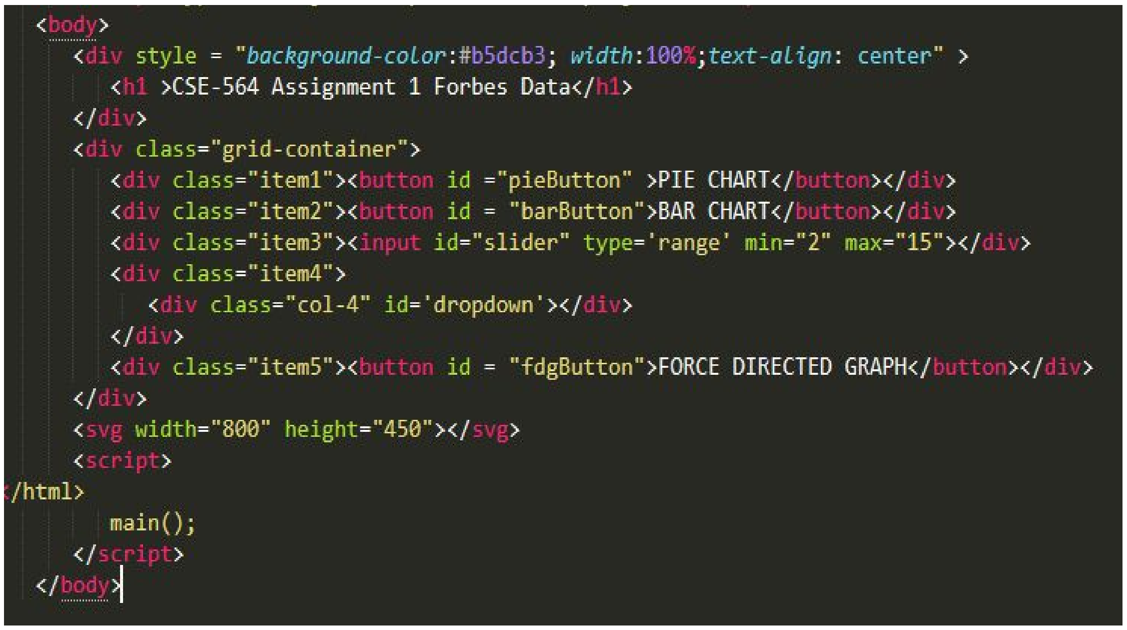
CSE 564 Assignment 1

Sayak Ghosh (112025780)




```
  <body>
      <div style = "background-color:#b5dcb3; width:100%;text-align: center" >
          <h1 >CSE-564 Assignment 1 Forbes Data</h1>
      </div>
      <div class="grid-container">
          <div class="item1"><button id ="pieButton" >PIE CHART</button></div>
          <div class="item2"><button id = "barButton">BAR CHART</button></div>
          <div class="item3"><input id="slider" type='range' min="2" max="15"></div>
          <div class="item4">
              <div class="col-4" id='dropdown'></div>
          </div>
          <div class="item5"><button id = "fdgButton">FORCE DIRECTED GRAPH</button></div>
      </div>
      <svg width="800" height="450"></svg>
      <script>
</html>
          main();
      </script>
  </body>
```

The above html code snippet defines the ui layout illustrated above. The main function is defined in the js file.

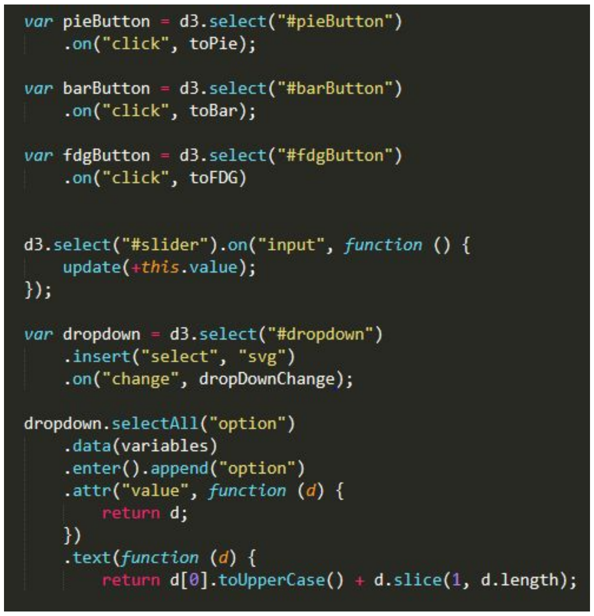
Workflow:

Initialize svg container with svg elements.

UI Elements:

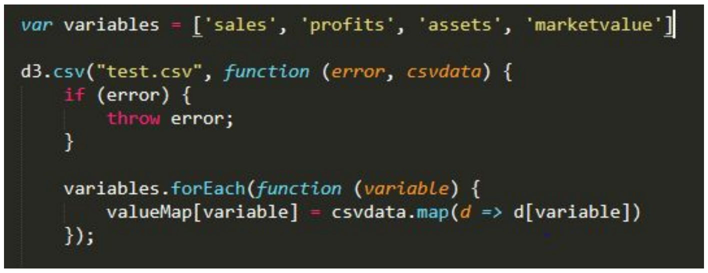
- Pie Chart (Button) - Renders the pie chart of the selected data
- Bar Chart (Button) - Renders the histogram of the selected data
- Slider - Adjust the bin size of the data
- Drop down list - List of available fields to filter the data
- Force Directed Graph (Button) - Renders the force directed graph

  The onClick functions *toPie,toBar,toFDG* renders the pie chart, bie chart and the force directed graph respectively. The functions has been explained in details later on.



```
var pieButton = d3.select("#pieButton")
    .on("click", toPie);

var barButton = d3.select("#barButton")
    .on("click", toBar);

var fdgButton = d3.select("#fdgButton")
    .on("click", toFDG)


d3.select("#slider").on("input", function () {
    update(+this.value);
});

var dropdown = d3.select("#dropdown")
    .insert("select", "svg")
    .on("change", dropDownChange);

dropdown.selectAll("option")
    .data(variables)
    .enter().append("option")
    .attr("value", function (d) {
        return d;
    })
    .text(function (d) {
        return d[0].toUpperCase() + d.slice(1, d.length);
```

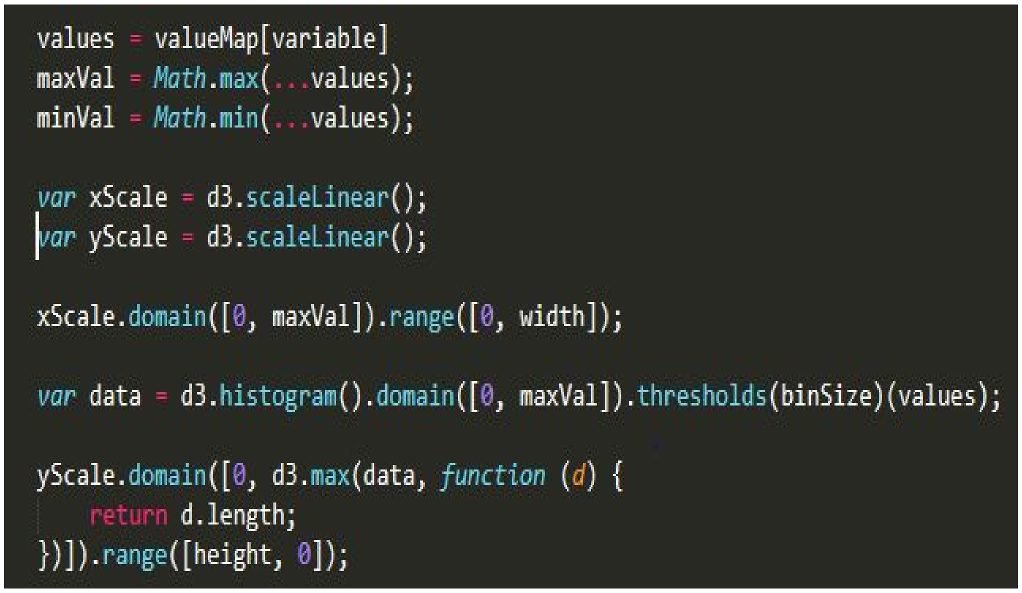
Load Data:



```
var variables = ['sales', 'profits', 'assets', 'marketvalue']

d3.csv("test.csv", function (error, csvdata) {
    if (error) {
        throw error;
    }

    variables.forEach(function (variable) {
        valueMap[variable] = csvdata.map(d => d[variable])
    });
```

Loads the data only the first time the page is loaded and stores the values of corresponding variables in a map.

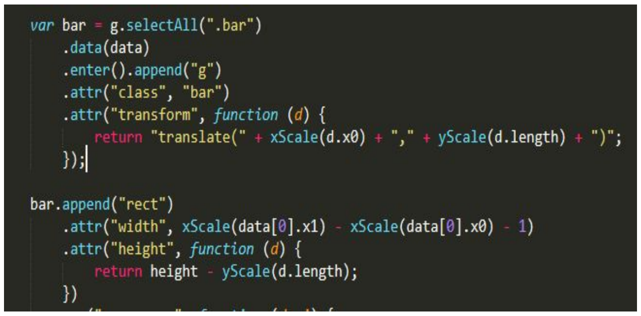
Features:



```
values = valueMap[variable]
maxVal = Math.max(...values);
minVal = Math.min(...values);

var xScale = d3.scaleLinear();
var yScale = d3.scaleLinear();

xScale.domain([0, maxVal]).range([0, width]);

var data = d3.histogram().domain([0, maxVal]).thresholds(binSize)(values);

yScale.domain([0, d3.max(data, function (d) {
    return d.length;
})]).range([height, 0]);
```

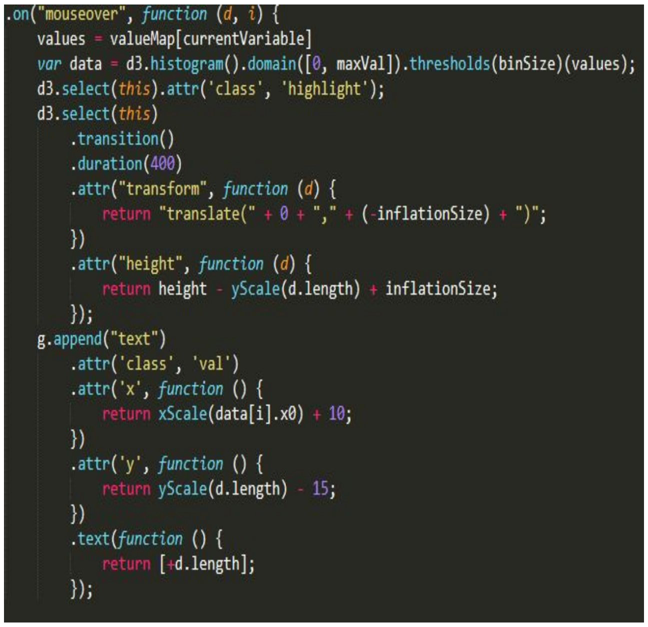
Bar Chart:

- The x-axis has been declared as a Linear scale with its range being 0- maximum value of the selected variable.
- The y-axis has also been declared as a Linear Scale with its range being the maximum possible frequency of any bin.



```
var bar = g.selectAll(".bar")
    .data(data)
    .enter().append("g")
    .attr("class", "bar")
    .attr("transform", function (d) {
        return "translate(" + xScale(d.x0) + "," + yScale(d.length) + ")";
    });

bar.append("rect")
    .attr("width", xScale(data[0].x1) - xScale(data[0].x0) - 1)
    .attr("height", function (d) {
        return height - yScale(d.length);
    })
```

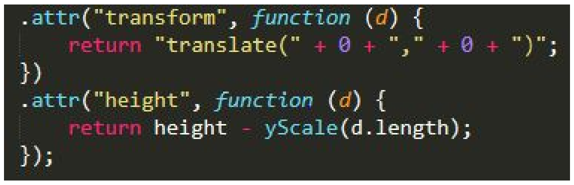
The bar elements of the bar chart are translated along the x-axis as the enter method iterates over the data and appended to g. Every bar is a rectangular element defined by its width , ie. the width of each element in the xscale and the height, defined by the value of the data which in this case is d.length.



```
.on("mouseover", function (d, i) {
    values = valueMap[currentVariable]
    var data = d3.histogram().domain([0, maxVal]).thresholds(binSize)(values);
    d3.select(this).attr('class', 'highlight');
    d3.select(this)
        .transition()
        .duration(400)
        .attr("transform", function (d) {
            return "translate(" + 0 + "," + (-inflationSize) + ")";
        })
        .attr("height", function (d) {
            return height - yScale(d.length) + inflationSize;
        });
    g.append("text")
        .attr('class', 'val')
        .attr('x', function () {
            return xScale(data[i].x0) + 10;
        })
        .attr('y', function () {
            return yScale(d.length) - 15;
        })
        .text(function () {
            return [+d.length];
        });
```

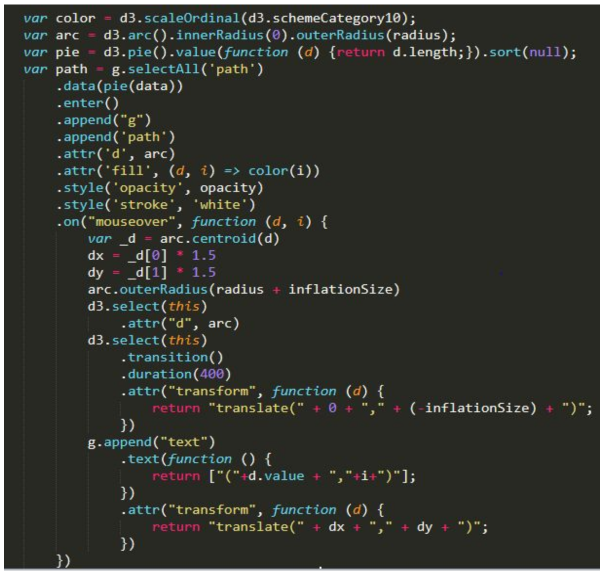
The mouse over action is defined over the bar element. On invocation it would increase the height of the bar by the inflation size . It also had to be offset by the same amount to keep the bar aligned with the x-axis.
A text element is also appended to g over the selected bar. It is translated to the right position as defined by the xscale and yscale.

The mouse out action reverses the changes made in mouse over action.



```
.attr("transform", function (d) {
    return "translate(" + 0 + "," + 0 + ")";
})
.attr("height", function (d) {
    return height - yScale(d.length);
});
```

Pie Chart:

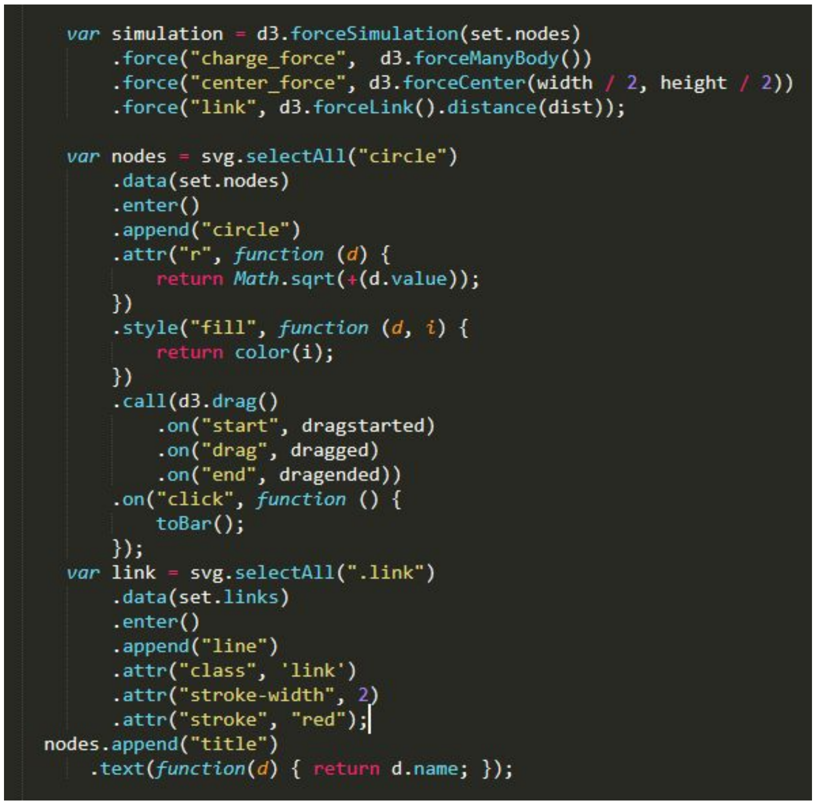


```
var color = d3.scaleOrdinal(d3.schemeCategory10);
var arc = d3.arc().innerRadius(0).outerRadius(radius);
var pie = d3.pie().value(function (d) {return d.length;}).sort(null);
var path = g.selectAll('path')
    .data(pie(data))
    .enter()
    .append("g")
    .append('path')
    .attr('d', arc)
    .attr('fill', (d, i) => color(i))
    .style('opacity', opacity)
    .style('stroke', 'white')
    .on("mouseover", function (d, i) {
        var _d = arc.centroid(d)
        dx = _d[0] * 1.5
        dy = _d[1] * 1.5
        arc.outerRadius(radius + inflationSize)
        d3.select(this)
            .attr("d", arc)
        d3.select(this)
            .transition()
            .duration(400)
            .attr("transform", function (d) {
                return "translate(" + 0 + "," + (-inflationSize) + ")";
            })
        g.append("text")
            .text(function () {
                return ["("+d.value + ","+i+")"];
            })
            .attr("transform", function (d) {
                return "translate(" + dx + "," + dy + ")";
            })
    })
```

Each arc is appended to the path as the enter method iterates over the data . Every arc is assigned a color as defined by the color template.
The mouseover action increased the arc radius by the inflation size. A new text element is added to g and translated to its corresponding position as defined by the centroid of the arc.
The mouseout function reverses the changes from the mouseover by resetting the radius of the arc and removing the text element.

Force Directed Graph:

The force directed graph visualizes the similarity between all the companies in the dataset. A company is defined by the features assets, profit, market value and sales. A company is deemed to be similar to be if the cosine similarity between their features is less than .995. In this scenario an edge is drawn between the two nodes representing the respective companies.



```
    var simulation = d3.forceSimulation(set.nodes)
        .force("charge_force",  d3.forceManyBody())
        .force("center_force", d3.forceCenter(width / 2, height / 2))
        .force("link", d3.forceLink().distance(dist));

    var nodes = svg.selectAll("circle")
        .data(set.nodes)
        .enter()
        .append("circle")
        .attr("r", function (d) {
            return Math.sqrt(+(d.value));
        })
        .style("fill", function (d, i) {
            return color(i);
        })
        .call(d3.drag()
            .on("start", dragstarted)
            .on("drag", dragged)
            .on("end", dragended))
        .on("click", function () {
            toBar();
        });
    var link = svg.selectAll(".link")
        .data(set.links)
        .enter()
        .append("line")
        .attr("class", 'link')
        .attr("stroke-width", 2)
        .attr("stroke", "red");
  nodes.append("title")
      .text(function(d) { return d.name; });
```

The nodes are represented as circles with its radius being defined as the value of the selected variable in the node object. For example if the selected variable is market value then the radius of the circle is defined by the market value of the company represented by the node.

The links are then drawn to complete the graph representation.

The simulation forces are defined along with the action for dragging the nodes.