Web Application Programming

Introduction to d3.js

- D3 stands for **Data-Driven Documents**.
- D3.js is a JavaScript library for manipulating documents based on data.
- D3.js is a dynamic, interactive, online data visualizations framework used in a large number of websites.
- D3.js is written by **Mike Bostock**, created as a successor to an earlier visualization toolkit called **Protovis**.

Data visualization

- Data visualization is the presentation of data in a pictorial or graphical format.
- The primary goal of data visualization is to communicate information clearly and efficiently via statistical graphics, plots and information graphics.
- Data visualization helps us to communicate our insights quickly and effectively. Any type of data, which is represented by a visualization allows users to compare the data, generate analytic reports, understand patterns and thus helps them to take the decision.
- Data visualizations can be interactive, so that users analyze specific data in the chart.

D3.js

- D3.js is a JavaScript library used to create interactive visualizations in the browser.
- The D3.js library allows us to manipulate elements of a webpage in the context of a data set.These elements can be **HTML, SVG,** or **Canvas elements** and can be introduced, removed, or edited according to the contents of the data set.It is a library for manipulating the DOM objects.
- D3.js can be a valuable aid in data exploration, it gives user control over its data's representation and lets user add interactivity.

Features

- Extremely flexible.
- Easy to use and fast.
- Supports large datasets.
- Declarative programming.
- Code reusability.
- Has wide variety of curve generating functions.
- Associates data to an element or group of elements in the html page.

Advantages of d3js

- D3.js is an open source project and works without any plugin.
- It requires very less code.
- Great data visualization.
- It is modular. You can download a small piece of D3.js, which you want to use. No need to load the whole library every time.
- Easy to build a charting component.
- DOM manipulation.
- D3.js works on all the browsers except IE8 and lower.

Disadvantages of d3js

- Some functions in the D3.js is not supported in the older browsers.
- It has data-source limitations.
- Concealing original data is not easy with D3.js.
- The biggest disadvantage is it has a very steep learning curve.
- D3 **cannot easily conceal original data**.
- If you're using data that you don't want shared, it can be challenging to use D3.
- D3 doesn't generate predetermined visualizations for you.
- It can be time-inefficient to generate D3 visualizations where a quick chart from an alternative source would perform beautifully.

D3.js as Javascript Library

D3.js is an open source JavaScript library for :

- Data-driven manipulation of the Document Object Model (DOM).
- Working with data and shapes.
- Laying out visual elements for linear, hierarchical, network and geographic data.
- Enabling smooth transitions between user interface (UI) states.
- Enabling effective user interaction.

Connecting d3.js library to HTML page

- It can be done in two ways :

Include the D3.js library from your project's folder.

Include D3.js library from CDN (Content Delivery Network).

- The following code has to be included in the HTML document in order to connect the d3.js library
- http://d3js.org/d3.v4.min.js

D3.js-Drawing charts

D3.js is used to create a static SVG chart. It helps to draw the following charts −

Bar chart

Circle chart

Pie chart

Line chart

Bubble chart

BAR CHART

Bar charts are used to display and compare the number, frequency or other measure for discrete categories or groups.

The x-axis (i.e. horizontal axis)-represents the different categories it has no scale.

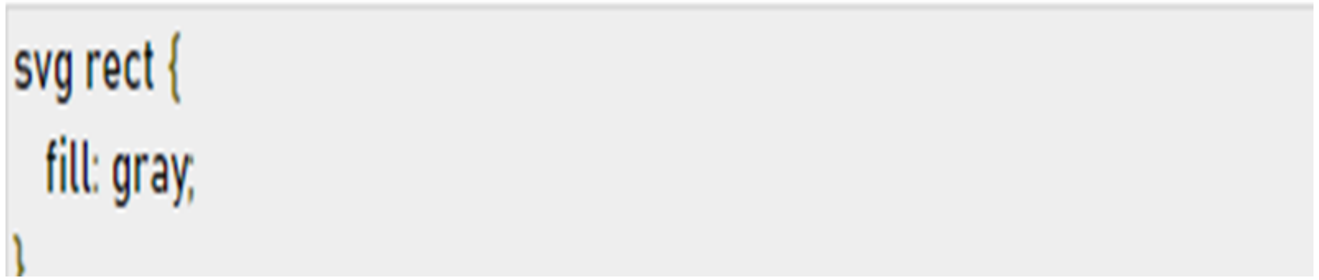
The y-axis (i.e. vertical axis)-does have a scale and this indicates the units of measurement.

Draw a bar chart

For example:

We can use the rect elements for the bars and text elements to display our data values corresponding to the bars.

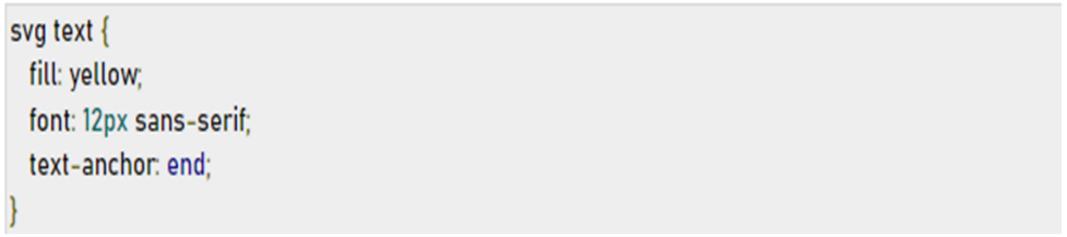
Step 1: Adding style in the rect element.



```
svg rect {
  fill: gray;
}
```

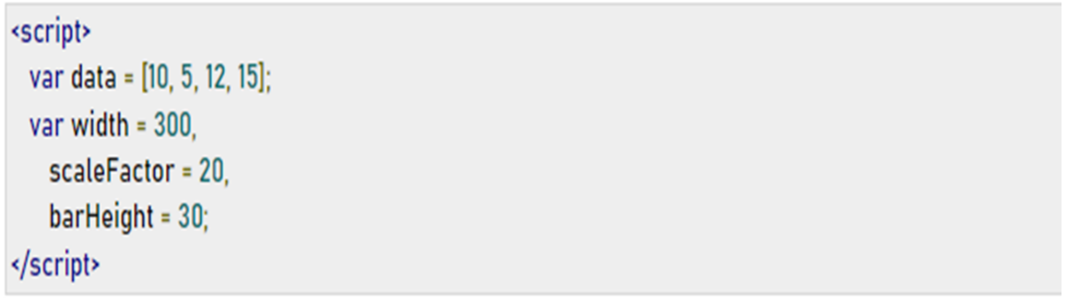
Continuation

<u>Step 2</u>: Add style in text elements.



```
svg text {
  fill: yellow;
  font: 12px sans-serif;
  text-anchor: end;
}
```

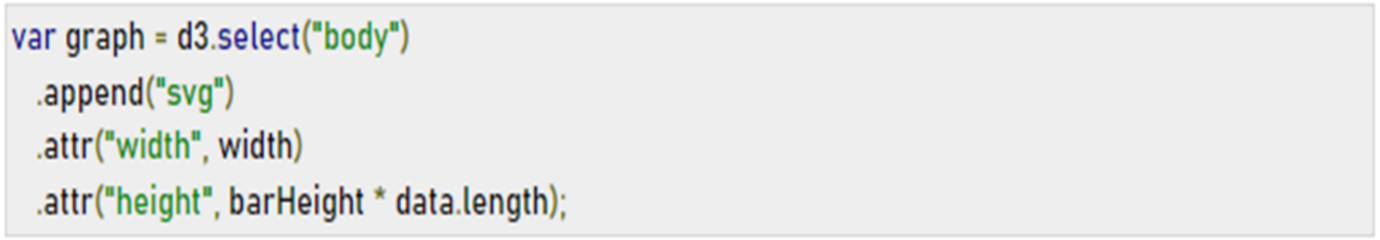
<u>Step 3</u>: Define variables.



```
<script>
  var data = [10, 5, 12, 15];
  var width = 300,
    scaleFactor = 20,
    barHeight = 30;
</script>
```

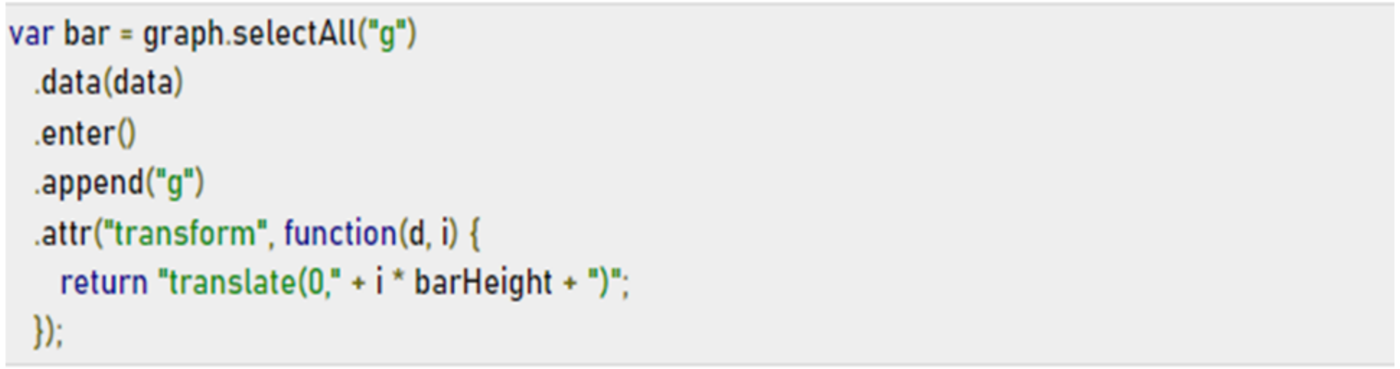
Continuation

Step 4: Append SVG elements.



```
var graph = d3.select("body")
  .append("svg")
  .attr("width", width)
  .attr("height", barHeight * data.length);
```

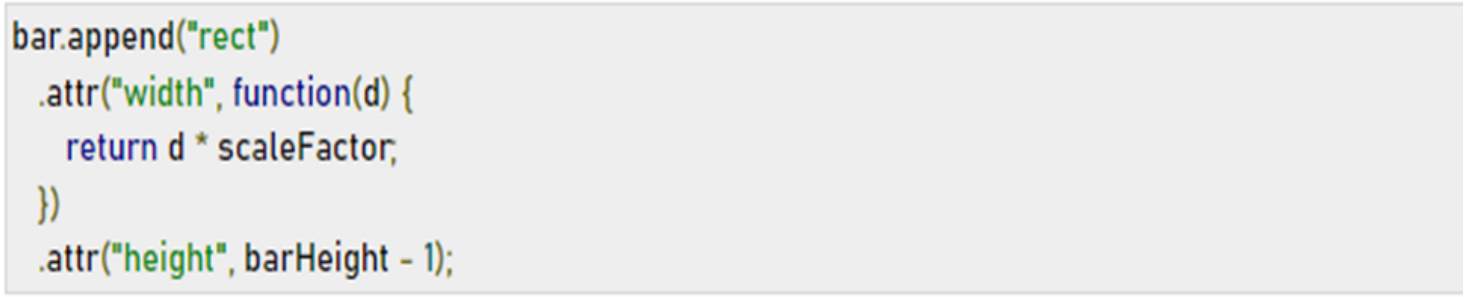
Step 5: Apply transformation



```
var bar = graph.selectAll("g")
  .data(data)
  .enter()
  .append("g")
  .attr("transform", function(d, i) {
    return "translate(0," + i * barHeight + ")";
  });
```

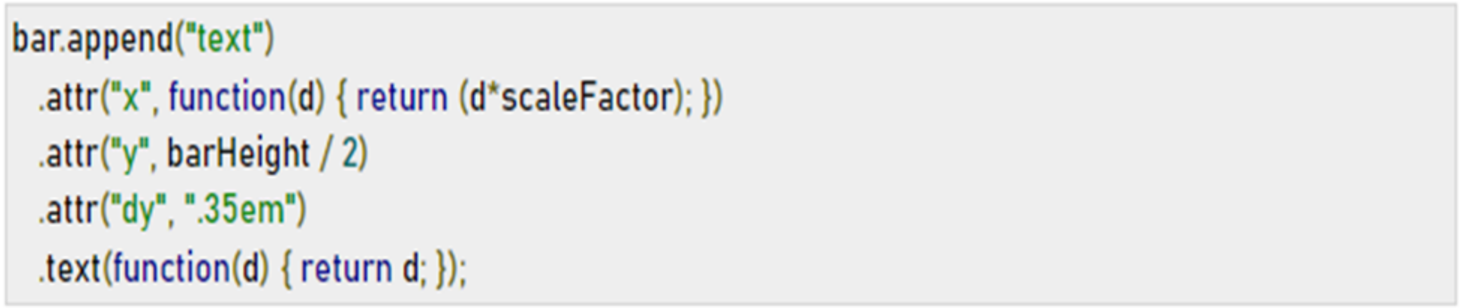
Continuation

Step 6: Append rect elements to the bar



```
bar.append("rect")
  .attr("width", function(d) {
    return d * scaleFactor;
  })
  .attr("height", barHeight - 1);
```

Step 7: Display data.



```
bar.append("text")
  .attr("x", function(d) { return (d*scaleFactor); })
  .attr("y", barHeight / 2)
  .attr("dy", ".35em")
  .text(function(d) { return d; });
```

Circle introduction

A circle chart is a circular statical graphic which is divided into slice to illustrate numerical proportion

It provide a real time output and are ideal heard copy of a data is nedded

Total value of circle is always 100%, to workout with % for a circle chart we follow the given bellow steps

1) categorize the data

2) calculate the total

3) divide the categories

4) convert into %

5) finally calculate the degrees

Circle

Step 1 − **Define variables** − Let us add the variables in the script. It is shown in the code block below.

```
<script>
   var width = 400;
   var height = 400;
   var data = [10, 20, 30];
   var colors = ['green', 'purple', 'yellow'];
</script>
```

Step 2 − **Append SVG elements** − Let us append SVG elements in D3 using the following code.

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

Step 3 − Apply transformation

```
var g = svg.selectAll("g")
   .data(data)
   .enter()
   .append("g")

   .attr("transform", function(d, i) {
      return "translate(0,0)";
   })
```

Step 4 − Append circle elements

```
g.append("circle")
.attr("cx", function(d, i) {
   return i*75 + 50;
})
.attr("cy", function(d, i) {
  return 75;
})
.att
.attr("fill", function(d, i){
  return colors[i];
})r("r", function(d) {
  return d*1.5;
})
```

Step 5 − Display data

```
g.append("text")
  .attr("x", function(d, i) {
    return i * 75 + 25;
  })
  .attr("y", 80)
  .attr("stroke", "teal")
  .attr("font-size", "10px")
  .attr("font-family", "sans-serif")
  .text(function(d) {
    return d;
  });
```

Pie Chart

A Pie chart is a circular statical graphic which is divided into slice to illustrate numerical proportion

It provide a real time output and are ideal heard copy of a data is nedded

Total value of circle is always 100%, to workout with % for a circle chart we follow the given bellow steps

1) categorize the data

2) calculate the total

3) divide the categories

4) convert into %

5) finally calculate the degrees

Continuation of Pie chart

Step1: Dataset

For this example, we will take an array of objects where each object has two attributes: `name` and `share`.

```
var data = [{name: "Alex", share: 20.70},
                    {name: "Shelly", share: 30.92},
                    {name: "Clark", share: 15.42},
                    {name: "Matt", share: 13.65},
                    {name: "Jolene", share: 19.31}];
```

The shares sum up to 100!

Continuation

Step 2: D3 and SVG container

First, we need to import the D3 script using the `src` attribute and then initialize our SVG container with the appropriate width and height:

```
<script src="https://d3js.org/d3.v4.min.js"></script>
<svg width="500" height="400"></svg>
```

Continuation

Step 3: Set dimensions

```
var svg = d3.select("svg"),
            width = svg.attr("width"),
            height = svg.attr("height"),
            radius = 200;
```

Step 4: Set scale

```
var ordScale = d3.scaleOrdinal()
                            .domain(data)
                            .range(['#ffd384','#94ebcd','#fbaccc'
,'#d3e0ea','#fa7f72']);
```

Continuation

Step 5: Pie generator

```
var pie = d3.pie().value(function(d) {
                return d.share;
            });

        var arc = g.selectAll("arc")
                   .data(pie(data))
                   .enter();
```

Step 6: Fill chart

```
var path = d3.arc()
                     .outerRadius(radius)
                     .innerRadius(0);

        arc.append("path")
           .attr("d", path)
           .attr("fill", function(d) { return ordScale(d.data.name); });
```

Continuation

Step 7: Add labels

```
var label = d3.arc()
                      .outerRadius(radius)
                      .innerRadius(0);

        arc.append("text")
           .attr("transform", function(d) {
                    return "translate(" + label.centroid(d) + ")"
;
            })
           .text(function(d) { return d.data.name; })
           .style("font-family", "arial")
           .style("font-size", 15);
```

Step 5: Pie generator

```
var pie = d3.pie().value(function(d) {
                return d.share;
            });

        var arc = g.selectAll("arc")
                   .data(pie(data))
                   .enter();
```

Step 6: Fill chart

Line chart

Step 1: Dataset

◦Before starting we need data set to base our chart on.

◦So we have taken a basic 2D array of random numbers

```
var dataset1 = [
            [1,1], [12,20], [24,36],
            [32, 50], [40, 70], [50, 100],
            [55, 106], [65, 123], [73, 130],
            [78, 134], [83, 136], [89, 138],
            [100, 140]
        ];
```

Continuation of line chart

Step 2: D3 and SVG container

Import the D3 script using src attribute

Then initialize SVG container with appropriate width and height

```
<script src="https://d3js.org/d3.v4.min.js"></script>
 <svg width="500" height="400"></svg>
```

Continuation

Step 3: Set margin

Set margin for SVG to make the chart look more centered and clear

Four variables: svg, margin, width, height.svg initialized with 500px to 400px.

These

```
var svg = d3.select("svg"),
        margin = 200,
        width = svg.attr("width") - margin,
        height = svg.attr("height") - margin
```

Continuation

Step 4: Set scale

For discrete data visualization on the x-axis, we construct a linear scale or scaleLinear().scaleLinear() uses the linear equation (y=mx+c) to interpolate domain and range across the axis.

```
var xScale = d3.scaleLinear().domain([0, 100]).range([0, width]),
        yScale = d3.scaleLinear().domain([0, 200]).range([height, 0]);
```

Continuation

Step 5: Add text

we need to add title of the x-axis label, y-axis label to our plot. First append text to our svg, then we set the position, style, and actual text attribute.

To rotate our text for the y-axis, we use transform and specify the angle of rotation

Step 6: Add axis

Add both axis. For x-axis d3.axisBottom because we need to align it at the bottom of the canvas.

For y axis, we call d3.axisLeft because we need to align it to the left of the canvas.

```
g.append("g")
        .attr("transform", "translate(0," + height + ")")
        .call(d3.axisBottom(xScale));

       g.append("g")
        .call(d3.axisLeft(yScale));
```

Step 7: Scatter dots

We should add a dot for every coordinate of the dataset.

Supply dataset to the data and attribute and make circle for each coordinator.

Here cx specifies the horizontal position and cy specifies the vertical position.

r specifies the radius.

Translate all the points to match the translation of our axis.

Finally color the data points.

Step 8: Plot line

Connect all the dots, to make a line we use line generator of d3 by invoking d3.line().

This line generator can be used to compute the d attributeof SVG path element.

Here we append path to our SVG path element.

We append path to our SVG and then specify the class as line.

Then we transform it like we transformed our axes and datapoints earlier, and lastly, we style the line accordingly.

Bubble Chart

- A bubble chart is used to display three dimensions of the data.
- It is a variation of scatter chart, in which the data points are replaced with bubbles.
- The bubble sizes are represented with respect to the data dimension.
- It uses horizontal and vertical axes as value axes.

Steps to create Bubble Chart

•Step 1 : **Dataset**

Before even starting to code, we need a data set to base our chart on.

•Step 2 :**D3 and SVG container**

We need to import the D3 script using the **src** attribute and then initialize our SVG container with the appropriate width and height.

Ex: <script src="http://d3js.org/d3.v4.min.js"></script>

 <svg width="100" height="100"></svg>

•Step 3 : **Set dimensions**

Make an **svg** variable and define its constraints keeping in mind the height and width we defined for our **svg** container in step 2.

Ex : var svg = d3.select(“svg”)

  .attr(“width”,100)

  .attr(“height”,100);

•Step 4 : **Make bubbles**

Now, we will make the bubbles. To do that, we first need to append circle to our svg and then give it the data to plot. Then, we need to specify the position for each bubble (cx and cy). After that, we need to specify the radius for each circle.

In a bubble chart, the circle area should be representative of its value, not its radius. Therefore, we first take the value’s square root and then divide it by *pie* (π) to get the corresponding radius. Finally, we color each circle according to the color specified in our data set

•Step 5 : **Add labels**

Finally, we will add the labels to our bubble chart sections. But, first, we need to define the position where we need to write the labels, and for that, we will set the x and y coordinates accordingly. After that, we will append this text to our svg variable and style it accordingly.

For the better understanding of Bubble Chart we have the following demonstration of program.

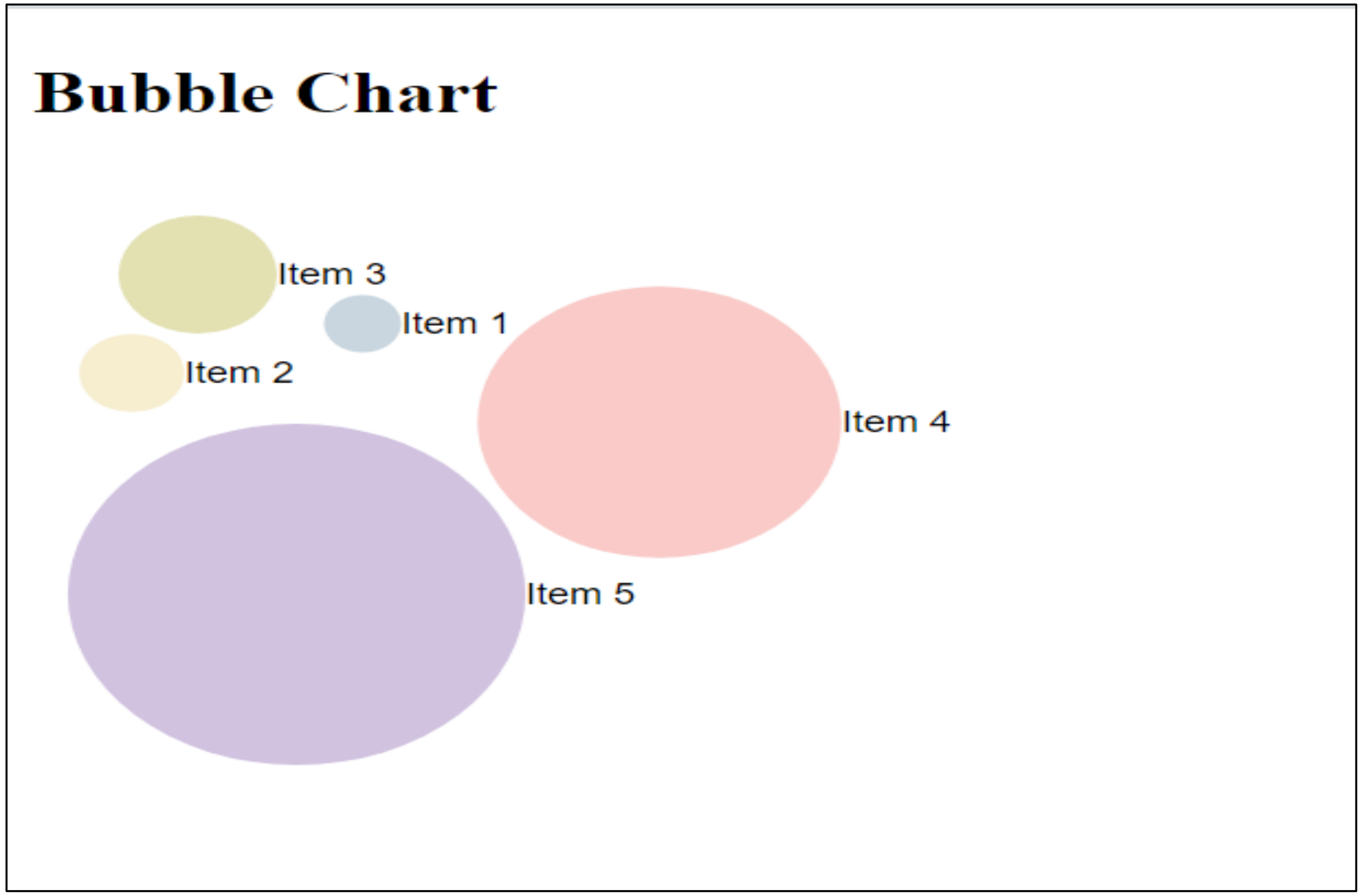
Program

```
<!DOCTYPE html>
<html lang="en">
<head>
        <script type="text/javascript" src="https://d3js.org/d3.v4.min.js"></script>
</head>
<body>
  <h2>Bubble Chart</h2>
  <svg width="500" height="500"></svg> <!--Step 2-->
    <script type="text/javascript">
// Step 1-Dataset
    var data = [
      {source:"Item 1", x: 100, y: 60, val: 1350, color: "#C9D6DF"},
      {source:"Item 2", x: 30, y: 80, val: 2500, color: "#F7EECF"},
      {source:"Item 3", x: 50, y: 40, val: 5700, color: "#E3E1B2"},
      {source:"Item 4", x: 190, y: 100, val: 30000, color: "#F9CAC8"},
      {source:"Item 5", x: 80, y: 170, val: 47500, color: "#D1C2E0"}
    ]
```

```
// Step 3
  var svg = d3.select("svg")
          .attr("width", 500)
          .attr("height", 500);
  // Step 4
  svg.selectAll("circle")
    .data(data).enter()
    .append("circle")
    .attr("cx", function(d) {return d.x})
    .attr("cy", function(d) {return d.y})
    .attr("r", function(d) {
      return Math.sqrt(d.val)/Math.PI
    })
    .attr("fill", function(d) {
      return d.color;
    });
```

```
// Step 5
   svg.selectAll("text")
     .data(data).enter()
     .append("text")
     .attr("x", function(d) {return d.x+
(Math.sqrt(d.val)/Math.PI)})
     .attr("y", function(d) {return d.y+4})
     .text(function(d) {return d.source})
     .style("font-family", "arial")
     .style("font-size", "12px")
   </script>
</body>
</html>
```

OUTPUT


Bubble Chart
Item 3
Item 1
Item 2
Item 4
Item 5