14. D3JS Scales API.

D3.js provides scale functions to perform data transformations. These functions map an input domain to an output range.

14.1. Configuring API

You can configure the API directly using the script below.

Example:

```
<script src = "https://d3js.org/d3-array.v1.min.js"></script>
<script src = "https://d3js.org/d3-collection.v1.min.js"></script>
<script src = "https://d3js.org/d3-color.v1.min.js"></script>
<script src = "https://d3js.org/d3-format.v1.min.js"></script>
<script src = "https://d3js.org/d3-interpolate.v1.min.js"></script>
<script src = "https://d3js.org/d3-time.v1.min.js"></script>
<script src = "https://d3js.org/d3-time-format.v2.min.js"></script>
<script src = "https://d3js.org/d3-scale.v1.min.js"></script>
<body>
   <script>
   </script>
</body>
```

14.2. Scales API Methods

D3 provides the following important scaling methods for different types of charts. Let us understand then in detail..

- **d3.scaleLinear()** − Constructs a continuous linear scale where we can input data (domain) maps to the specified output range.

- **d3.scaleIdentity()** − Construct a linear scale where the input data is the same as the output.

- **d3.scaleTime()** − Construct a linear scale where the input data is in the dates and the output in numbers.

- **d3.scaleLog()** − Construct a logarithmic scale.

- **d3.scaleSqrt()** − Construct a square root scale.

- **d3.scalePow()** − Construct an exponential scale.

- **d3.scaleSequential()** − Construct a sequential scale where output range is fixed by interpolator function.

- **d3.scaleQuantize()** − Construct a quantize scale with discrete output range.

- **d3.scaleQuantile()** − Construct a quantile scale where the input sample data maps to the discrete output range.

- **d3.scaleThreshold()** − Construct a scale where the arbitrary input data maps to the discrete output range.

- **d3.scaleBand()** − Band scales are like ordinal scales except the output range is continuous and numeric.

- **d3.scalePoint()** − Construct a point scale.

- **d3.scaleOrdinal()** − Construct an ordinal scale where the input data includes alphabets and are mapped to the discrete numeric output range.

Before doing a working example, let us first understand the following two terms −

- **Domain** − The Domain denotes minimum and maximum values of your input data.
- **Range** − The Range is the output range, which we would like the input values to map to...

14.3. Working Example

Let us perform the d3.scaleLinear function in this example. To do this, you need to adhere to the following steps

Step 1: Define variables − Define SVG variables and data using the coding below.

```
var data = [100, 200, 300, 400, 800, 0]
   var width = 500,
      barHeight = 20,
      margin = 1;
```

Step 2: Create linear scale − Use the following code to create a linear scale.

```
var scale = d3.scaleLinear()
   .domain([d3.min(data), d3.max(data)])
   .range([100, 400]);
```

Here, for the minimum and maximum value for our domain manually, we can use the built-in d3.min() and d3.max() functions, which will return minimum and maximum values respectively from our data array.

Step 3: Append SVG attributes − Append the SVG elements using the code given below.

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

Step 4: Apply transformation − Apply the transformation using the code below.

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

Step 5: Append rect elements − Append the rect elements to scaling as shown below.

```
g.append("rect")
   .attr("width", function (d) {
      return scale(d);
   })
   .attr("height", barHeight - margin)
```

Step 6: Display data − Now display the data using the coding given below.

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

```
        .text(function (d) { return d; });
```

Step 7: Working Example − Now, let us create a bar chart using the d3.scaleLinear() function as follows. Create a webpage "scales.html" and add the following changes to it.

```
<!DOCTYPE html>
<html>
   <head>
      <script src = "https://d3js.org/d3.v4.min.js"></script>
   </head>

   <body>
      <script>
         var data = [100, 200, 300, 350, 400, 250]
         var width = 500, barHeight = 20, margin = 1;

         var scale = d3.scaleLinear()
            .domain([d3.min(data), d3.max(data)])
            .range([100, 400]);

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

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

         g.append("rect")
```

```
            .attr("width", function (d) {
               return scale(d);
            })

            .attr("height", barHeight - margin)
            g.append("text")
            .attr("x", function (d) { return (scale(d)); })
            .attr("y", barHeight / 2).attr("dy", ".35em")
            .text(function (d) { return d; });
        </script>
    </body>
</html>
```

Output: The above code will display the following result in the browser.