16. D3JS Shapes API.

This chapter discusses the different shape generators in D3.js.

16.1. Configuring API

You can configure the API directly using the script below.

Example:

```
<script src = "https://d3js.org/d3-path.v1.min.js"></script>
<script src = "https://d3js.org/d3-shape.v1.min.js"></script>
<body>
   <script>
   </script>
</body>
```

16.2. Shapes Generators

D3.js supports different shapes. Let us go through the prominent shapes in detail.

Arcs API

The arc generator produces a circle or annulus shape. We have used these API methods in the previous pie charts chapter. Let us understand the various Arcs API methods in detail.

- **d3.arc()** − This method is used to create a new arc generator.

- **arc(args)** − It is used to generate an arc with the specified given arguments. Default settings with an object radii and angles is defined below.

Example:

```
<script>
   var arc = d3.arc();
   arc({
      innerRadius: 0,
      outerRadius: 100,
      startAngle: 0,
      endAngle: Math.PI / 2
   });
</script>
```

- **arc.centroid(args)** − This method is used to compute the midpoint [x, y] of the centerline of the arc with the specified arguments.

- **arc.innerRadius([radius])** − This method is used to set the inner radius from the given radius and return an arc generator. It is defined below −

Example:

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

- **arc.outerRadius([radius])** − This method is used to set the outer radius from the given radius and return an arc generator. It is defined as follows.

Example:

```
function outerRadius(d) {
```

```
        return d.outerRadius;
    }
```

- **arc.cornerRadius([radius])** − This method is used to set the corner radius from the given radius and return an arc generator. It is defined as follows.

Example:

```
function cornerRadius() {
    return 0;
}
```

If the corner radius is greater than zero, the corners of the arc are rounded using the circles of the given radius. The corner radius may not be larger than (outerRadius - innerRadius) / 2.

- **arc.startAngle([angle])** − This method is used to set the start angle to the function from the given angle. It is defined as follows:

Example:

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

- **arc.endAngle([angle])** − This method is used to set the end angle to the function from the given angle. It is defined as follows.

Example:

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

- ***arc.padAngle([angle])*** − This method is used to set the pad angle to the function from the given angle. It is defined as follows.

Example:

```
function padAngle() {
   return d && d.padAngle;
}
```

- **arc.padRadius([radius])** − This method is used to set the pad radius to the specified function from the given radius. The pad radius determines the fixed linear distance separating adjacent arcs, defined as padRadius *padAngle.

- **arc.context([context])** − This method is used to set the context and return an arc generator.

16.3. Pies API

This API is used to create a Pie generator. We have performed these API methods in the previous chapter. We will discuss all those methods in detail.

- **d3.pie()** − Constructs a new pie generator with the default settings.

- **pie(data[, arguments])** − This method is used to generate a pie for the given array values. It returns an array of objects. Objects are datum's arc angles. Each object has the following properties −
  - **data** − the input datum; the corresponding element in the input data array.
  - **value** − the numeric value of the arc.
  - **index** − index of the arc.
  - **startAngle** − the start angle of the arc.
  - **endAngle** − the end angle of the arc.
  - **padAngle** − the pad angle of the arc.

- **pie.value([value])** − This method is used to set the value to the specified function and generates a pie. It is defined as follows:

Example:

```
function value(d) {
   return d;
}
```

- **pie.sort([compare])** − This method is used to sort the data to the specified function and generates pie. The comparator function is defined as follows.

Example:

```
pie.sort(function(a, b)
   { return a.name.localeCompare(b.name); }
);
```

Here, the compare function takes two arguments 'a' and 'b', each element from the input data array. If the arc for 'a' should be before the arc for 'b', then the comparator must return a number less than zero. If the arc for 'a' should be after the arc for 'b', then the comparator must return a number greater than zero.

- **pie.sortValues([compare])** − This method is used to compare the value from the given function and generates a pie. The function is defined as follows.

Example:

```
function compare(a, b) {
   return b - a;
}
```

- **pie.startAngle([angle])** − This method is used to set the start angle of the pie to the specified function. If the angle is not specified, it returns the current start angle. It is defined as follows.

Example:

```
function startAngle() {
   return 0;
}
```

- **pie.endAngle([angle])** − This method is used to set the end angle of the pie to the specified function. If angle is not specified, it returns the current end angle. It is defined as follows.

Example:

```
function endAngle() {
   return 2 * Math.PI;
}
```

- **pie.padAngle([angle])** − This method is used to set the pad angle to the specified function and generates the pie. The function is defined as follows.

Example:

```
function padAngle() {
   return 0;
}
```

16.4. Lines API

Lines API is used to generate a line. We have used these API methods in the Graphs chapter. Let us go through each methods in detail.

- **d3.line()** − This method is used to create a new line generator.
- **line(data)** − This method is used to generate a line for the given array of data.
- **line.x([x])** − This method is used to set the x accessor to the specified function and generates a line. The function is defined below,

Example:

```
function x(d) {
   return d[0];
}
```

- **line.y([y])** − This method is used to set the 'y'' accessor to the specified function and generates a line. The function is defined as follows.

Example:

```
function y(d) {
   return d[1];
}
```

- **line.defined([defined])** − This method is used to set the defined accessor to the specified function. It is defined as follows.

Example:

```
function defined() {
  return true;
}
```

- **line.curve([curve])** − It is used to set the curve and generates the line.

- **line.context([context])** − This method is used to set the context and generates a line. If the context is not specified, it returns null.

- **d3.lineRadial()** − This method is used to create new radial line; it is equivalent to the Cartesian line generator.

- **lineRadial.radius([radius])** − This method is used to draw a radial line and the accessor returns the radius. It takes distance from the origin(0,0).

In the next chapter, we will learn about the Colors API in D3.js.