A Developer Diary
{about:"code learn and share"}

February 20, 2016 By Abhisek Jana — Leave a Comment (Edit)

Create iWatch Activity Chart using d3.js

In this tutorial we will **Create iWatch Activity Chart using d3.js**. Here we will use almost all the features we have learnt so far on Pie Chart. We will create multiple functions to reuse the code as much as possible.

Lets look at the details in the chart.

UI Elements:

1. 3 Background Elements
2. 3 Foreground Elements
3. Multiple Shadows and Gradients
4. 3 PNG Images

We will first set the width and height. Then set a color scale. After that create the `svg` element.

```
var w=300,h=300;
var outerRadius=(w/2);
var width=30,gap=2;

var innerRadius=outerRadius-30;

var color=["#e90b3a", "#a0ff03", "#1ad5de"];

var svg=d3.select("#chart")
    .append("svg")
    .attr({
        width:w,
        height:h,
        class:'shadow'
    }).append('g')
    .attr({
        transform:'translate('+w/2+','+h/2+')'
    });
```

Now lets create a `JSON` object for the 3 sets of activities. We have name for each activity type, the initial value, color and icon.

```
var circles=[
        {name:'activity1',percent:70,color:'#e90b3a',icon:'stand.png'},
        {name:'activity2',percent:80,color:'#a0ff03',icon:'walk.png'},
        {name:'activity3',percent:65,color:'#1ad5de',icon:'run.png'}
];
```

Next create the gradent and drop shadow definitions. You can find the below functions later in this tutorial.

```
createGradient(svg,'#fe08b5','#ff1410','gradient');
createDropShadow(svg,'dropShadow',4,1,1);
```

The following section is very important. We will iterate through the JSON object, calculate the `innerRadius` and call `createCircle()` function to create both the background and foreground elements for each activity. Don't worry about the `createCircle()` function, we will create them in a min.

The outermost activity has a gradient, we will check if `i = 0` and call `addGradient()` function. Afterwards, we will call `addDropShadow()` and `addStartImage()`.

```
for(var i=0;i < circles.length;++i){
    if(i>0){
        outerRadius=innerRadius-gap;
    }
    innerRadius=outerRadius-width;


circles[i].chart=createCircle(svg,outerRadius,innerRadius,circles[i].color,circles[i].percent);

    if(i==0){
        addGradient(circles[i].chart.path,'gradient');
    }

    addDropShadow(circles[i].chart.path,'dropShadow');
    addStartImage(svg,'https://www.adeveloperdiary.com/wp-
content/uploads/2015/11/'+circles[i].icon,outerRadius);
}
```

Let’s create the `createCircle()` function. It accepts the svg element we already created, outer and inner radius, color and initial value.

We will calculate the ratio based on the percent. Then add the background arc and path element. Repeate the same for the foreground as well. Set the `endAngle` to `0` for the foreground. Add the `transition()` function to the foreground path element.

The `createCircle()` returns a JSON Object which contains both the path and arc object for the foreground element.

```
var createCircle=function(svg,outerRadius,innerRadius,color,percent){

  var ratio=percent/100;

  var arcBackground=d3.svg.arc()
          .innerRadius(innerRadius)
          .outerRadius(outerRadius)
          .startAngle(0)
          .endAngle(2*Math.PI);

  var pathBackground=svg.append('path')
          .attr({
              d:arcBackground
          })
          .style({
              fill:color,
              opacity:.2
          });

  var arcForeground=d3.svg.arc()
          .innerRadius(innerRadius)
          .outerRadius(outerRadius)
          .cornerRadius(20)
          .startAngle(-0.05);

  var pathForeground=svg.append('path')
          .datum({endAngle:0})
          .attr({
              d:arcForeground
          })
          .style({
              fill:color
```

```
        });

  pathForeground.transition()
          .duration(1500)
          .ease('elastic')
          .call(arcTween,((2*Math.PI))*ratio,arcForeground);

  var chart={path:pathForeground,arc:arcForeground};

  return chart;
};
```

We need use the `arcTween()` function for each activity since all of them will have seperate values and they need to animate independently. Here is the function. This is straightforward.

```
var arcTween=function(transition, newAngle,arc) {
    transition.attrTween("d", function (d) {
        var interpolate = d3.interpolate(d.endAngle, newAngle);

        return function (t) {
            d.endAngle = interpolate(t);
            return arc(d);
        };
    });
};
```

Find the `addGradient()` and `addDropShadow()` functions. Both the functions accepts the path object and an id. We already have created the gradient and drop shadow definitions, so we just need to set the fill and filter property.

```
var addGradient=function(path,id){
   path.style({
       fill:"url('#"+id+"')"
   });
};

var addDropShadow=function(path,id){
   path.style({
       filter:"url('#"+id+"')"
   });
};
```

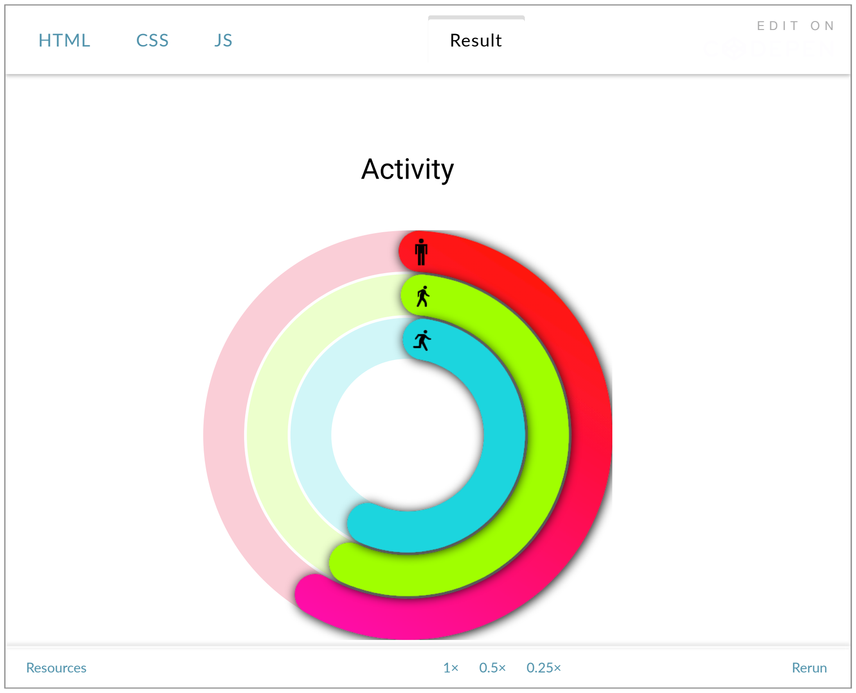
Next create the `addStartImage()` function to add the png image. We will add an image element with `xlink:href` as the image URL. Then we will will move the image to the correct location using the `outerRadius`.

```
var addStartImage= function (svg,url,outerRadius) {
    svg.append('image')
        .attr({
            'xlink:href':url,
            width:20,
            height:20,
            transform:'translate(0,'+ (-outerRadius+6) +')'
    });
};
```

Believe or not, we are done with the chart creation. Now we just need to animate it. We will animate all 3 foregrounds.

```
var updateData=function(){
    for(var i=0;i < circles.length;++i) {
        updateChart(circles[i].chart.path,circles[i].chart.arc,Math.floor((Math.random()
* 60) + 20));
    }
    setTimeout(updateData,1500);
};

setTimeout(updateData,1500);

var updateChart= function (path,arc,percent) {

    var ratio=percent/100;

    path.transition()
            .duration(1500)
            .ease('elastic')
            .call(arcTween,((2*Math.PI))*ratio,arc);
};
```

Lets look at the demo now:



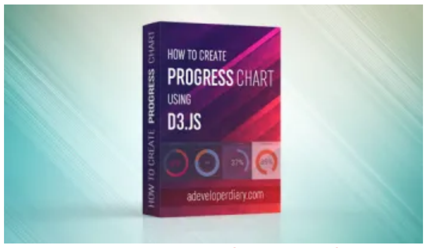
Here are the `createGradient()` and `createDropShadow()` function.

```
var createGradient=function(svg,color1,color2,id){

    var defs = svg.append("defs");

    var gradient=defs.append("linearGradient")
            .attr("id", id)
            .attr("x1", "0%")
            .attr("y1", "100%")
            .attr("x2", "50%")
            .attr("y2", "0%")
            .attr("spreadMethod", "pad");

    gradient.append("svg:stop")
            .attr("offset", "0%")
            .attr("stop-color", color1)
            .attr("stop-opacity", 1);

    gradient.append("svg:stop")
            .attr("offset", "100%")
            .attr("stop-color", color2)
            .attr("stop-opacity", 1);
};

var createDropShadow=function(svg,id,stdDeviation,dx,dy){

    var defs = svg.append("defs");

    var filter = defs.append("filter")
            .attr("id", id);

    filter.append("feGaussianBlur")
            .attr("in", "SourceAlpha")
            .attr("stdDeviation", stdDeviation)
            .attr("result", "blur");
    filter.append("feOffset")
            .attr("in", "blur")
            .attr("dx", dx)
            .attr("dy", dy)
            .attr("result", "offsetBlur");

    var feMerge = filter.append("feMerge");

    feMerge.append("feMergeNode")
            .attr("in", "offsetBlur");
    feMerge.append("feMergeNode")
            .attr("in", "SourceGraphic");
};
```

Find the full code in github.

Related

Create a Simple Pie Chart using D3.js
In "D3.js"



How to create Progress chart using d3.js
In "D3.js"

Create 3D Donut Chart using D3.js
In "D3.js"

Filed Under: D3.js, JavaScript | Tagged With: actity, chart, d3.js, Donut Charts, example, iwatch, JavaScript, Learn, Pie Chart, Programming, Progress Chart

Subscribe to stay in loop

* indicates required

Email Address *

Subscribe

Leave a Reply

Logged in as Abhisek Jana. Edit your profile. Log out? Required fields are marked *

Comment *

Post Comment