# SVG Text Element

## The Goal

In this section, we will cover the SVG Text Element, why it is important and how we use it within our D3.js Data Visualizations.

First, we will cover what the SVG Text Element is and how we will use it.

Then, we will use the SVG Text Element to get a feel for how it works.

Finally, we will create and add SVG Text Elements to a Data Visualization using the D3.js.

## SVG Text Element

The SVG Text Element defines a graphics element consisting of text.

The attributes and properties of the the SVG Text Element indicate things like the font specification, writing direction, and attributes for how to exactly render and paint the characters.

Because SVG Text Elements are rendered using the same rendering methods as the rest of the SVG Graphical Elements, the same coordinate system, transformations, ... etc also apply.

The SVG Text Element renders the first character at the initial current text position.

This position is defined by the 'x' and 'y' attributes of the SVG Text Element.

This position is possible to massage with the 'text-anchor' property.

Given this definition, we can write the SVG Text Element as follows:

```
<text x="20" y="20" font-family="sans-serif" font-size="20px" fill="red">Hello!</text>
```

This tells us that the text will start to be drawn at point (20, 20), with the font-family sans-serif, with the font-size of "20px", and the fill color of "red".

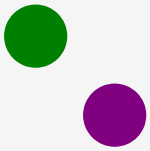
We will cover one major uses of the SVG Text Element:

1. **Text Labels** - Adding a Text Label to an existing SVG Element

## Adding an SVG Text Element

We start by manually creating some circles with SVG:

```
<svg width="200" height="200">
  <circle cx="20" cy="20" r="20" fill="green" />
  <circle cx="70" cy="70" r="20" fill="purple" />
</svg>
```



We would now like to add some text on top of the green circle.

We can use the SVG Text Element to add this text.

Because we know that the green circle is drawn around the point (20,20) [cx=20, cy=20], we setup our SVG Text Element to start being drawn from that point.

Given the definition above, we can write the SVG Text Element as follows:

```
<text x="20" y="20" font-family="sans-serif" font-size="20px" fill="red">Hello!</text>
```

Putting it into the SVG example we are manually building, we get the following:

```
<svg width="100" height="100">
  <circle cx="20" cy="20" r="20" fill="green" />
  <circle cx="70" cy="70" r="20" fill="purple" />
  <text x="20" y="20" font-family="sans-serif" font-size="20px" fill="red">Hello!</text>
</svg>
```


How about that?!

We were able to manually add text on top of the green circle.

Again, this is because the green circle is drawn around the point (cx=20, cy=20) while the text is drawn from point (x=20, y=20).

This is useful to note because if we can programmatically define the points to draw circles with D3.js, we should be able to programmatically define the points to write the text using D3.js.

We can the try to center the text on the circle.

Luckily, SVG has thought of that already for us.

We can add this to our SVG Text Element by adding the "text-anchor" attribute:

```
<text x="20" y="20"
      font-family="sans-serif"
      font-size="20px"
      text-anchor="middle"
      fill="red">Hello!</text>
```

The code is written over several lines for readability.

Notice we added text-anchor="middle".

Putting it into the SVG, we get:

```
<svg width="100" height="100">
  <circle cx="20" cy="20" r="20" fill="green" />
  <circle cx="70" cy="70" r="20" fill="purple" />
  <text x="20" y="20"
        font-family="sans-serif"
        font-size="20px"
        text-anchor="middle"
        fill="red">Hello!</text>
</svg>
```


**Great** - we were able to center the SVG Text Element around the point (20,20).

**Not so Great** - the Hello is cut off on the left-hand side.

It turns out that the SVG Viewport cuts off anything that is not in the viewport.

The element is still there, SVG just doesn't show it.

***For now***, we will not worry about this.

To put a "Hello" on top of the purple circle, we can do the same thing:

```
<svg width="100" height="100">
  <circle cx="20" cy="20" r="20" fill="green" />
  <circle cx="70" cy="70" r="20" fill="purple" />
  <text x="20" y="20" font-family="sans-serif" font-size="20px" fill="red">Hello!</text>
  <text x="70" y="70" font-family="sans-serif" font-size="20px" fill="red">Hello!</text>
</svg>
```


Works great - we are able to add the SVG Text element specifically where we want it.

Again, notice that the right hand side got cut off because of the SVG Viewport.

At this point we are feeling good - we can add SVG Text Elements manually where we want them to our Data Visualization.

## SVG Text Element and D3.js

In both the green and red circle cases, the text that goes on top of them is related to them.

We used the center point of each circle to figure out where to place our SVG Text Element.

Which means that the data used to create the circles can be used to create the text elements!

Which further means that the way we create SVG Circles using D3.js can be used to create Text Elements.

First, let us create the two SVG Circles using D3.js:

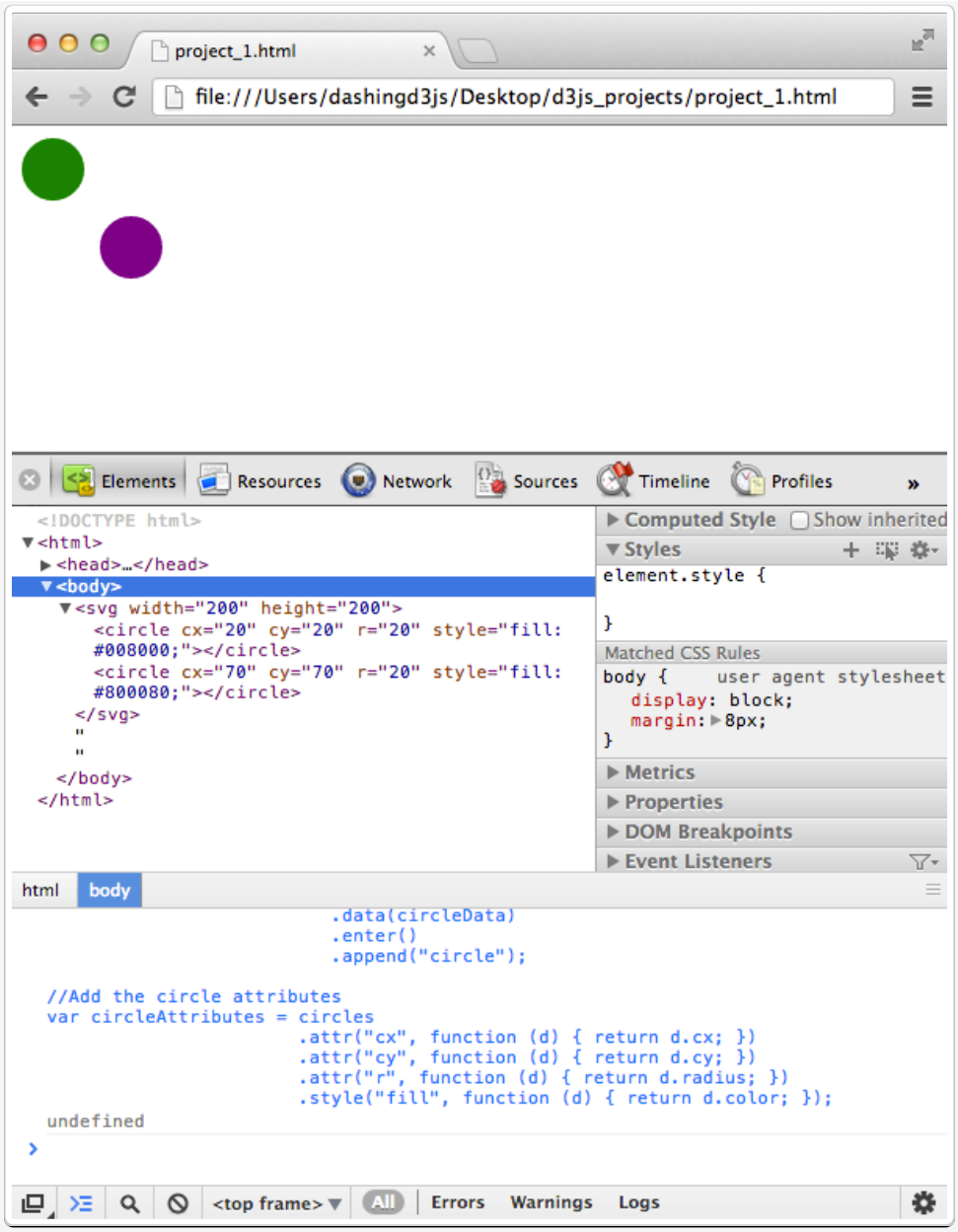
```
//Circle Data Set
var circleData = [
  { "cx": 20, "cy": 20, "radius": 20, "color" : "green" },
  { "cx": 70, "cy": 70, "radius": 20, "color" : "purple" }];

//Create the SVG Viewport
var svgContainer = d3.select("body").append("svg")
                                    .attr("width",200)
                                    .attr("height",200);

//Add circles to the svgContainer
var circles = svgContainer.selectAll("circle")
                          .data(circleData)
                          .enter()
                          .append("circle");

//Add the circle attributes
var circleAttributes = circles
                       .attr("cx", function (d) { return d.cx; })
                       .attr("cy", function (d) { return d.cy; })
                       .attr("r", function (d) { return d.radius; })
                       .style("fill", function (d) { return d.color; });
```

Which gives us:




Looking at the code and remembering back to the Binding Data to DOM Elements and Using Data Bound to DOM Elements, you'll know that the magic happens in the following two parts of the JavaScript code:

```
1  //Add circles to the svgContainer
2  var circles = svgContainer.selectAll("circle")
3                            .data(circleData)
4                            .enter()
5                            .append("circle");
6
7  //Add the circle attributes
8  var circleAttributes = circles
9                         .attr("cx", function (d) { return d.cx; })
10                        .attr("cy", function (d) { return d.cy; })
11                        .attr("r", function (d) { return d.radius; })
12                        .style("fill", function (d) { return d.color; });
```

The **//Add circles to the svgContainer** code takes the data and binds the data to the SVG Circle DOM Elements.

The **//Add the circle attributes** code takes the data bound to the SVG Circle DOM Elements and

uses it to define the attributes of the SVG Circle.

The SVG Text Element can be added in the same way.

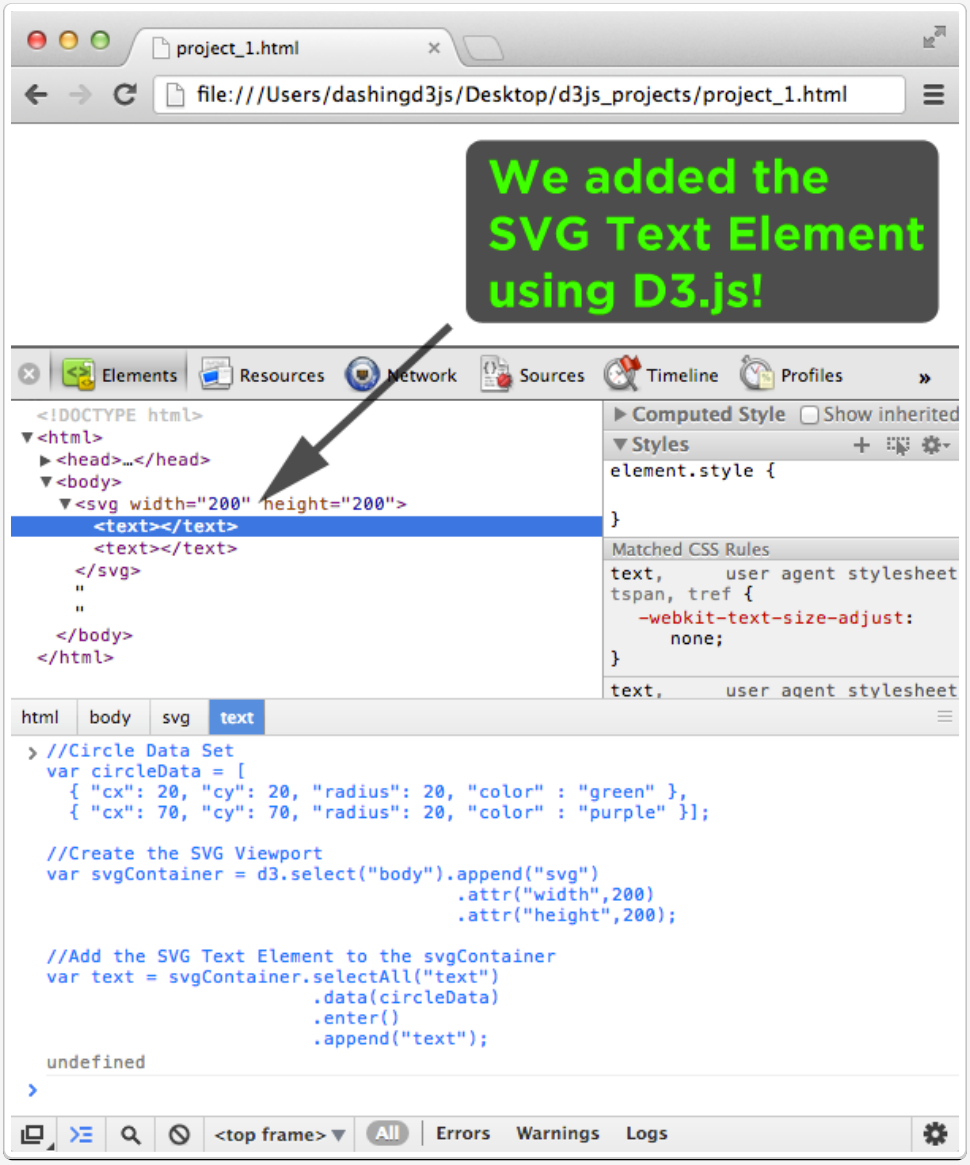
Let us do a simple example first (ignoring the circles and just adding the SVG Text Element):

```
//Circle Data Set
var circleData = [
  { "cx": 20, "cy": 20, "radius": 20, "color" : "green" },
  { "cx": 70, "cy": 70, "radius": 20, "color" : "purple" }];

//Create the SVG Viewport
var svgContainer = d3.select("body").append("svg")
                                    .attr("width",200)
                                    .attr("height",200);

//Add the SVG Text Element to the svgContainer
var text = svgContainer.selectAll("text")
                       .data(circleData)
                       .enter()
                       .append("text");
```

This gives us:

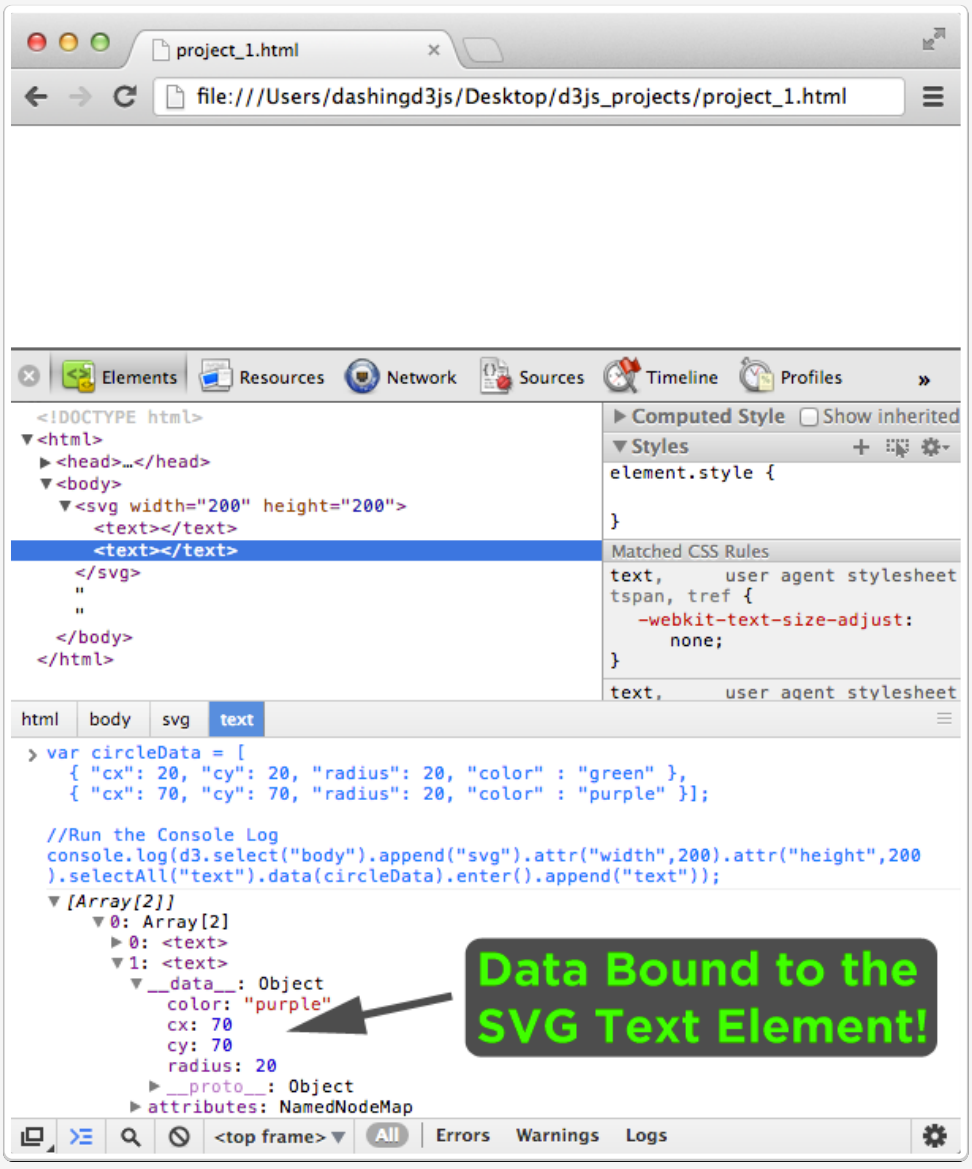


If we run the D3.js code in the console log:

```
var circleData = [
  { "cx": 20, "cy": 20, "radius": 20, "color" : "green" },
  { "cx": 70, "cy": 70, "radius": 20, "color" : "purple" }];

//Run the Console Log
console.log(d3.select("body").append("svg").attr("width",200).attr("height",200).selectAll("text").data(circleData).enter().append("text"));
```

We can see that the data was correctly bound to the SVG Text Element:




Now that we have bound the data and confirmed that the data is actually there, we can use D3.js to define the attributes of the SVG Text Element.

Much like we did with the circle, we can write the following:

```
//Add the text attributes
var textLabels = text
                 .attr("x", function(d) { return d.cx; })
                 .attr("y", function(d) { return d.cy; })
                 .text( function (d) { return "( " + d.cx + ", " + d.cy +" )"; })
                 .attr("font-family", "sans-serif")
```

```
                .attr("font-size", "20px")
                .attr("fill", "red");
```

Both the **.attr(.....)** for "x" and "y" are used to define the X and Y coordinates of where the Text should start.

The **.text(.....)** returns a string concatenation of the "cx" point and "cy" point of the circle.

The rest of the **.attr(.....)** operators add the "font-family", "font-size" and "fill" for our SVG Text Elements.

Continuing the example without the circle elements yet, we can write the following:

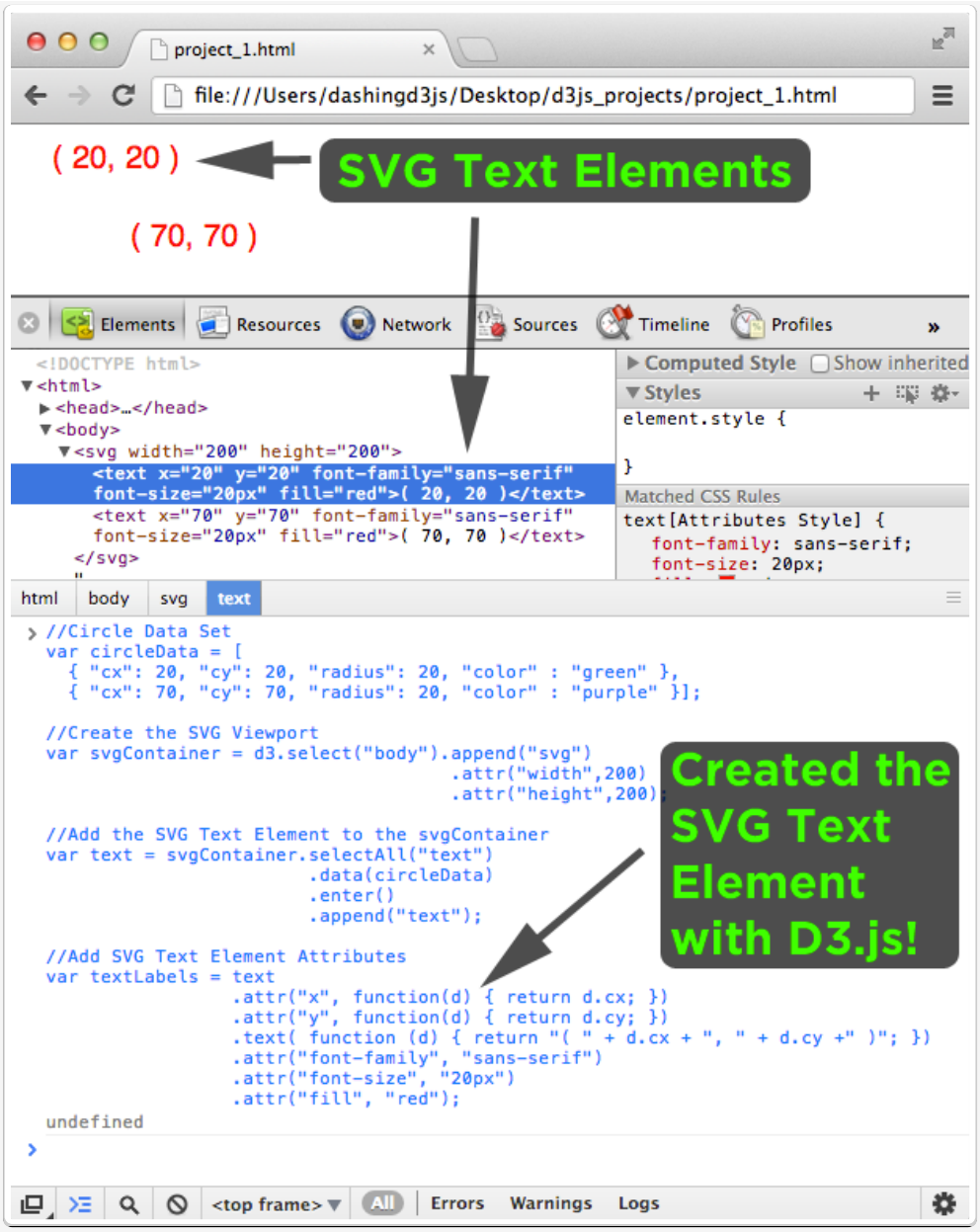
```
//Circle Data Set
var circleData = [
  { "cx": 20, "cy": 20, "radius": 20, "color" : "green" },
  { "cx": 70, "cy": 70, "radius": 20, "color" : "purple" }];

//Create the SVG Viewport
var svgContainer = d3.select("body").append("svg")
                                    .attr("width",200)
                                    .attr("height",200);

//Add the SVG Text Element to the svgContainer
var text = svgContainer.selectAll("text")
                       .data(circleData)
                       .enter()
                       .append("text");

//Add SVG Text Element Attributes
var textLabels = text
                 .attr("x", function(d) { return d.cx; })
                 .attr("y", function(d) { return d.cy; })
                 .text( function (d) { return "( " + d.cx + ", " + d.cy +" )"; })
                 .attr("font-family", "sans-serif")
                 .attr("font-size", "20px")
                 .attr("fill", "red");
```

Which gives us:




Good stuff - we created the SVG Text Elements with D3.js programmatically using the Circle Data Set.

We were able to place the SVG Text Elements where we wanted as well as write out the text that we wanted to write out.

In this case, we added the coordinates of where the text would start (or the center of the circle according to the circle data set).

**Finishing the example**, we add in the SVG Circle Elements to see the full data visualization:

```
1  //Circle Data Set
2  var circleData = [
3    { "cx": 20, "cy": 20, "radius": 20, "color" : "green" },
4    { "cx": 70, "cy": 70, "radius": 20, "color" : "purple" }];
5
6  //Create the SVG Viewport
7  var svgContainer = d3.select("body").append("svg")
8                                       .attr("width",200)
9                                       .attr("height",200);
10
```

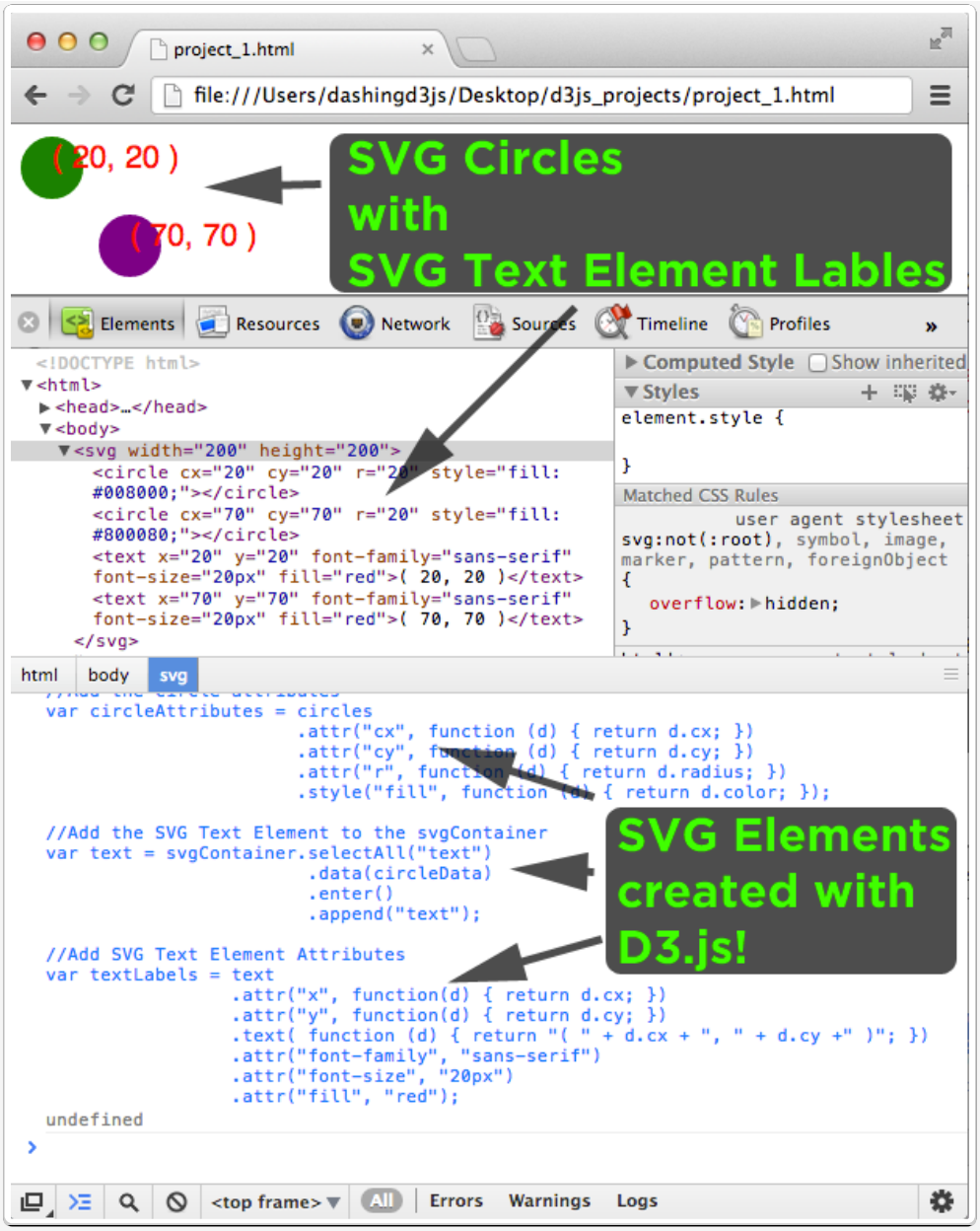
```
//Add circles to the svgContainer
var circles = svgContainer.selectAll("circle")
                          .data(circleData)
                          .enter()
                          .append("circle");

//Add the circle attributes
var circleAttributes = circles
                       .attr("cx", function (d) { return d.cx; })
                       .attr("cy", function (d) { return d.cy; })
                       .attr("r", function (d) { return d.radius; })
                       .style("fill", function (d) { return d.color; });

//Add the SVG Text Element to the svgContainer
var text = svgContainer.selectAll("text")
                        .data(circleData)
                        .enter()
                        .append("text");

//Add SVG Text Element Attributes
var textLabels = text
                 .attr("x", function(d) { return d.cx; })
                 .attr("y", function(d) { return d.cy; })
                 .text( function (d) { return "( " + d.cx + ", " + d.cy +" )"; })
                 .attr("font-family", "sans-serif")
                 .attr("font-size", "20px")
                 .attr("fill", "red");
```

Which gives us:




Bingo!

We created SVG Circle Elements and SVG Text Elements with D3.js based off of one data set.

This allowed us to add Text Labels to the SVG Circle Elements to make our data visualization much more readable.

Being able to programmatically add Text to our visualization using D3.js will allow us to add many wonderful things later on.

And with that, we have succeeded!

First, we covered what the SVG Text Element is and how we can use it.

Then, we used the SVG Text Element to get a feel for how it works.

Finally, we created and added SVG Text Elements to a Data Visualization using the D3.js.

← SVG Group Element and D3.js

D3.js Axes →

**Learn D3.js**

D3 Tutorial
D3 Screencasts
D3 Mapping Training
D3 Introductory Training
D3 Intermediate Training
D3 Advanced Training
D3 Corporate Training

**DashingD3js.com**

Blog
About
Hire Me
D3 Examples
D3 Resources
D3 & Data Viz Newsletter Archive

**Data Visualization & D3.js Weekly Newsletter**

Get D3.js and Data Visualization news, articles, jobs and more delivered to your inbox every Tuesday:

E-mail

Get the Newsletter

Did you sign up for the newsletter? :)