Using the SVG Coordinate Space

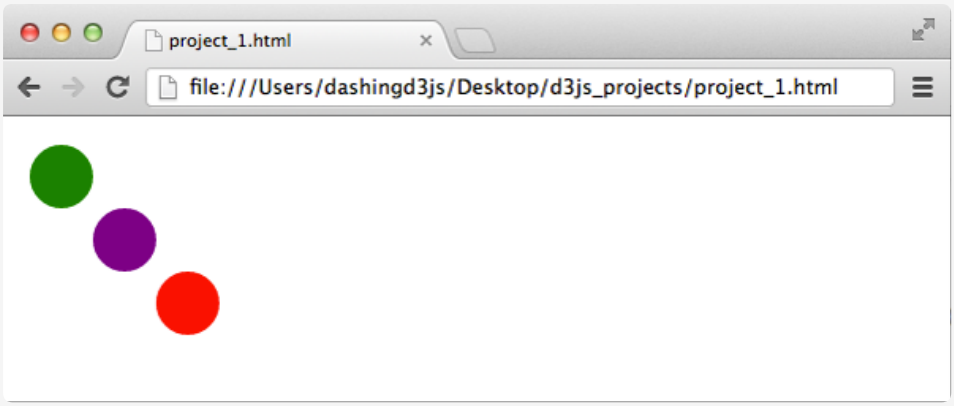
The Goal

In this section, you will use D3.js to add SVG elements to specific coordinates in a graph based on data.

Our Goal is to take the following data set:

```
var spaceCircles = [30, 70, 110];
```

and transform it to this data visualization using D3.js:



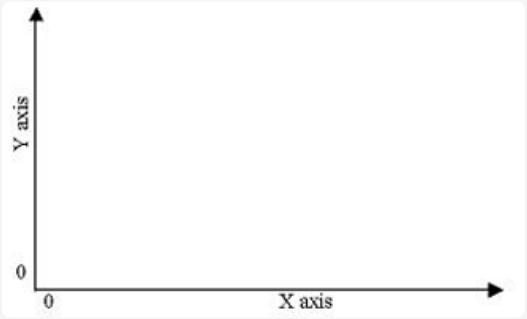
Mathematical / Graph Coordinate Space

Before we get into using D3.js to add SVG elements to specific coordinates in a graph based on data, let us talk about the coordinate space.

We learned about basic mathematical graphs in basic mathematics.

These graphs are a 2-dimensional flat space represented as a rectangle.

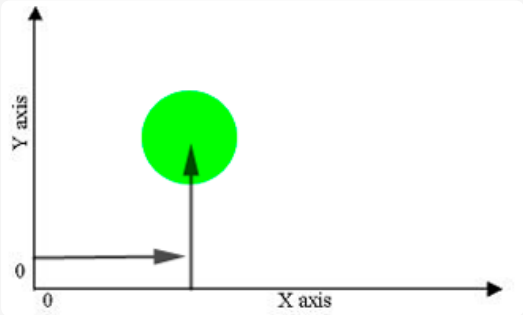
These graphs have a coordinate space where x=0 and y=0 coordinates fall on the bottom left.



These graphs have the **X** coordinate growing from *left to right*.

These graphs also have the **Y** coordinate growing from *bottom to top*.

Which means, when we talk about drawing a circle with x=30 and y=30 coordinates, we go 30 units from the bottom left to the right and then we go 30 units up.

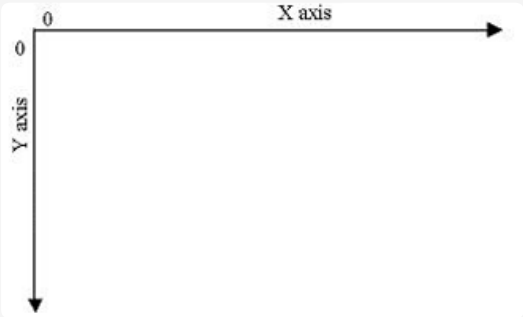


SVG Coordinate Space

SVG Coordinate Space works in the same way that mathematical graph coordinate space works except for two important features:

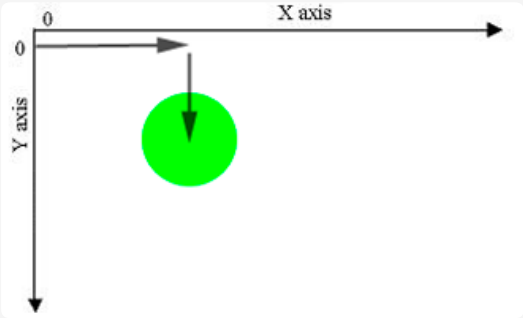
1 - SVG Coordinate space has x=0 and y=0 coordinates fall on the top left.

2 - SVG Coordinate space has the **Y** coordinate growing from *top to bottom*.



Which means as **Y** increases, the coordinates move down, not up.

So when we talk about drawing a circle with x=30 and y=30 coordinates in SVG Coordinate Space, we go 30 units from the top left to the right and then we go **down** 30 units up.



.append('svg') as a Coordinate Space

When we have created a D3.js visualization, so far we have started with the following code

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

Which gives us this the following HTML SVG element:

Until now, we had glossed over how to think about this SVG element. No longer!

We can think of this SVG element as a graph 200 units wide and 200 units tall.

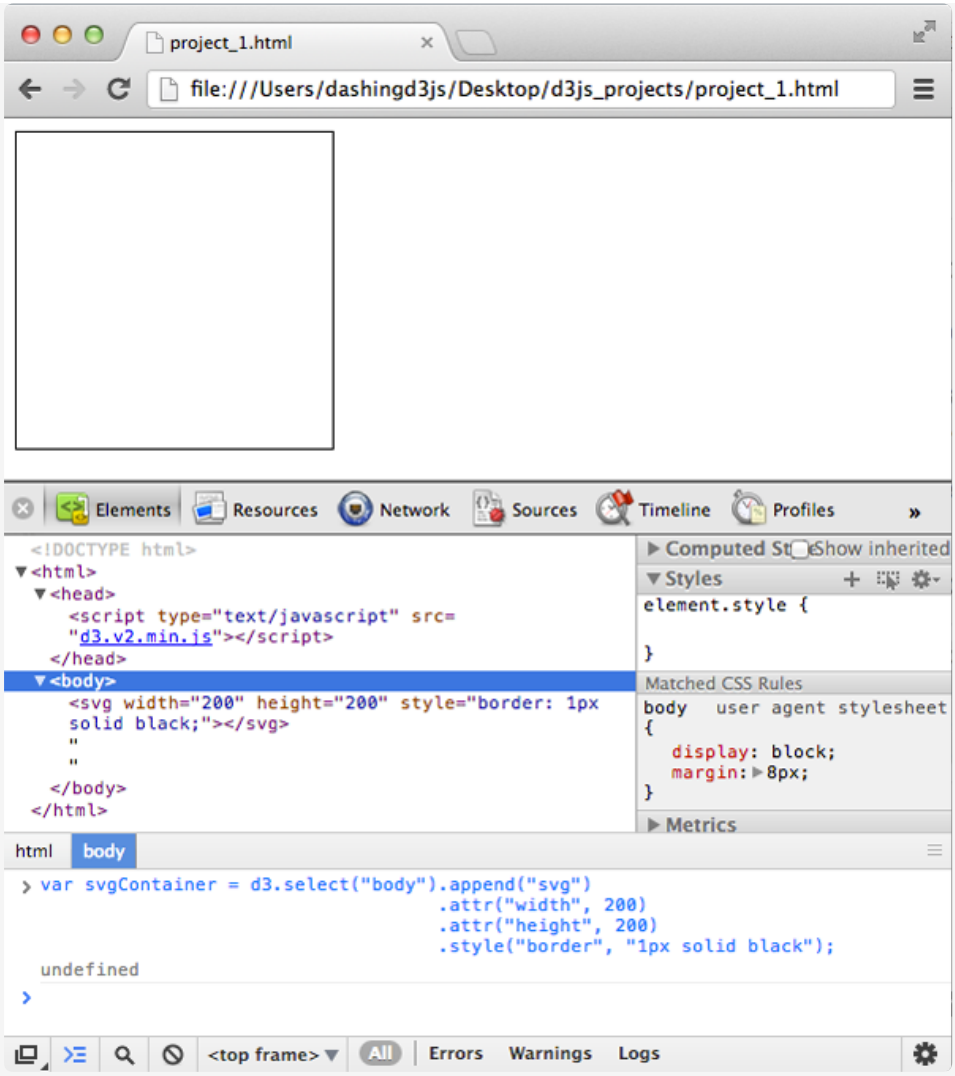
And with what we learned from the above section (SVG Coordinate Space), we now know that the X and Y zero coordinates are at the top left.

We also now know that as the Y coordinate grows, it will move from the top to the bottom of our graph.

To clarify what this looks like, we can style our SVG element by adding some style to it like this:

```
1  var svgContainer = d3.select("body").append("svg")
2                                      .attr("width", 200)
3                                      .attr("height", 200)
4                                      .style("border", "1px solid black");
```

Which gives us this:



This is useful for thinking about how the data visualization takes shape given the inverted Y coordinate system.

Position SVG Elements in the SVG Coordinate Space

When we create an SVG Circle element in the following fashion:

```
<svg width="50" height="50">
  <circle cx="25" cy="25" r="25" fill="purple" />
</svg>
```

The **cx** tells us to move 25 units from the left to the right of the SVG element.

The **cy** tells us to move 25 units from the top to the bottom of the SVG element.

Recall how we created an SVG circle with D3.js before:

```
var circleSelection = svgSelection.append("circle")
                                  .attr("cx", 25)
                                  .attr("cy", 25)
```

```
                                    .attr("r", 25)
                                    .style("fill", "purple");
```

The **.attr("cx", 25)** and **.attr("cy", 25)** allow us to set the attributes of the SVG circle.

And as we saw later in the Creating SVG Elements Based on Data section, we can programmatically set these attributes:

```
var circleAttributes = circles
                       .attr("cx", 50)
                       .attr("cy", 50)
                       .attr("r", function (d) { return d; });
```

Except, this time, instead of using a function in the "radius" **r** attribute, we will use a function in the **cx** and **cy** attributes.

Create an SVG Element to Hold SVG Elements

Start with a basic webpage:

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" src="d3.v2.min.js"></script>
  </head>
  <body>
  </body>
</html>
```

Open the JavaScript Console with the HTML elements visible.

Then type the following into the JavaScript Console:

```
var spaceCircles = [30, 70, 110];

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

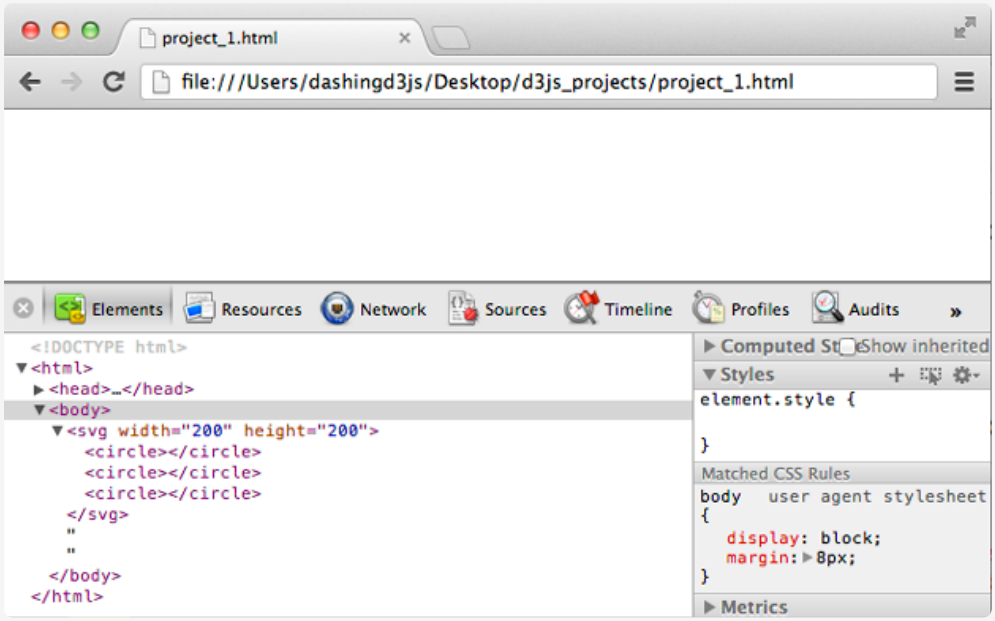
This gives us the data set and an SVG element to hold the SVG Circles we will add shortly:

Bind Data to SVG Circles

The next step will be to take the data and bind it to our SVG Circle elements.

```
var spaceCircles = [30, 70, 110];

var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 200)
                                    .attr("height", 200);

var circles = svgContainer.selectAll("circle")
                          .data(spaceCircles)
                          .enter()
                          .append("circle");
```

This gives us the three SVG circle elements in our webpage:



Which is great - though the circles don't appear. This is because have not specified the attributes for each circle.

Use Bound Data to Alter SVG Circle Coordinates

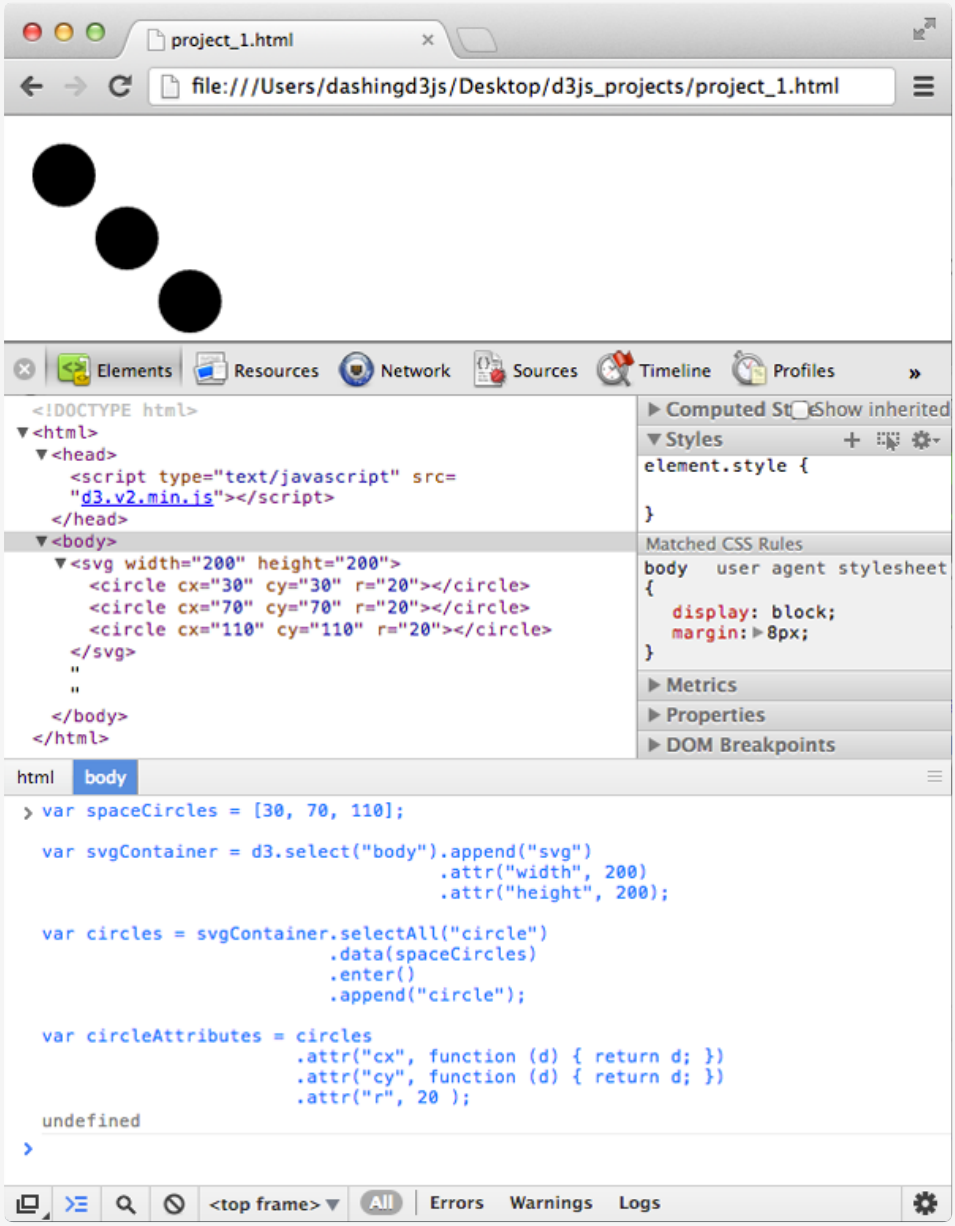
Now that the data is bound to the SVG Circle Elements, we can use it to specify the attributes for the x-coordinate and y-coordinate.

D3.js allows us to use a function in the "cx" and "cy" attributes:

```
var spaceCircles = [30, 70, 110];

var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 200)
                                    .attr("height", 200);

var circles = svgContainer.selectAll("circle")
                          .data(spaceCircles)
                          .enter()
                          .append("circle");

var circleAttributes = circles
                       .attr("cx", function (d) { return d; })
                       .attr("cy", function (d) { return d; })
                       .attr("r", 20 );
```

As you can see from the above code, we added the attributes to our circle selection. The data set specifies the circle center's x-coordinate and y-coordinate, so we use D3.js to update the "cx" and "cy" SVG Circle Element attribute for each SVG Circle.

If we run this in the JavaScript Console we get:



We are getting close. We now have three SVG Circle Elements and their coordinates correspond to the coordinates given in the the data set. The last thing we have to do is to color the SVG Circle based on the data.

Styling SVG Elements Based on Data Revisited

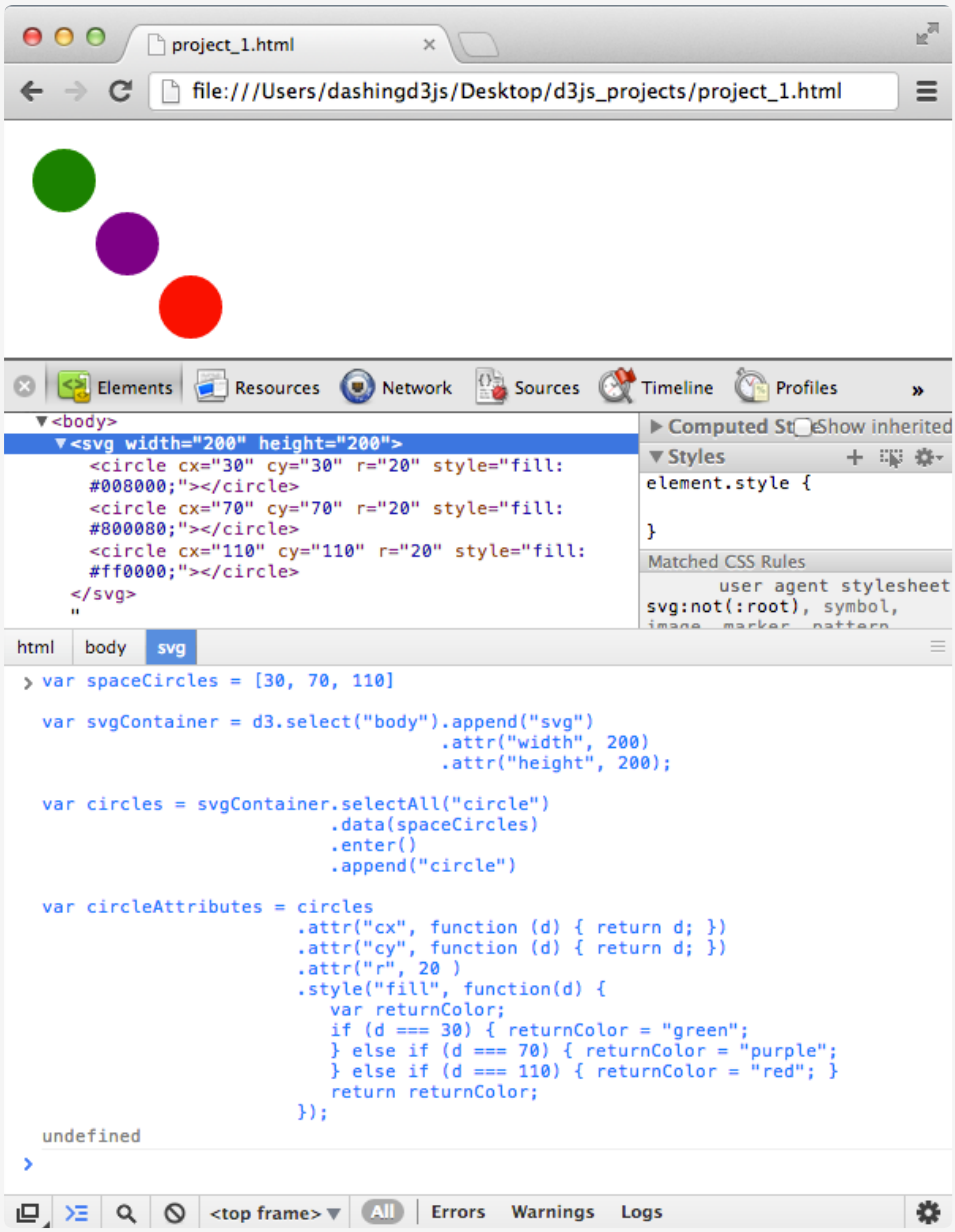
If you recall the Creating SVG Elements Based on Data section, we used the bound data to alter the SVG element's style.

As we saw before, D3.js lets us use a function inside of the .style() operator - which means we can write the following:

```
1  var spaceCircles = [30, 70, 110];
2
```

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

var circles = svgContainer.selectAll("circle")
                          .data(spaceCircles)
                          .enter()
                          .append("circle");

var circleAttributes = circles
                       .attr("cx", function (d) { return d; })
                       .attr("cy", function (d) { return d; })
                       .attr("r", 20 )
                       .style("fill", function(d) {
                         var returnColor;
                         if (d === 30) { returnColor = "green";
                         } else if (d === 70) { returnColor = "purple";
                         } else if (d === 110) { returnColor = "red"; }
                         return returnColor;
                       });
```

Running the above JavaScript code in the JavaScript Console gives us this:



Congratulations - You created, styled, and placed SVG elements in the SVG Coordinate Space based on data from a data set!

Want to better understand this topic? Check out this step-by-step course => Introductory D3 Training

← Creating SVG Elements Based on Data

Data Structures D3.js Accepts →