SVG Basic Shapes and D3.js

The Goal

In this section, we will go over the basic SVG shapes again and how to create them using D3.js.

We will cover previous SVG basic shape examples and how set their attributes using D3.js.

Drawing an SVG Circle using D3.js

We previously saw how to construct an SVG Circle in the Basic Building Blocks section:

Circle Example & Code:

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

We have also constructed SVG Circles using D3.js in the previous section (Using JSON to Simplify Code).

```
var jsonCircles = [
  { "x_axis": 30, "y_axis": 30, "radius": 20, "color" : "green" },
  { "x_axis": 70, "y_axis": 70, "radius": 20, "color" : "purple"},
  { "x_axis": 110, "y_axis": 100, "radius": 20, "color" : "red"}];

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

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

var circleAttributes = circles
                       .attr("cx", function (d) { return d.x_axis; })
                       .attr("cy", function (d) { return d.y_axis; })
                       .attr("r", function (d) { return d.radius; })
                       .style("fill", function(d) { return d.color; });
```

Simplifying the example above, we can then make the process of drawing a circle as two step process:

1) Make an SVG container (the SVG Coordinate Space)

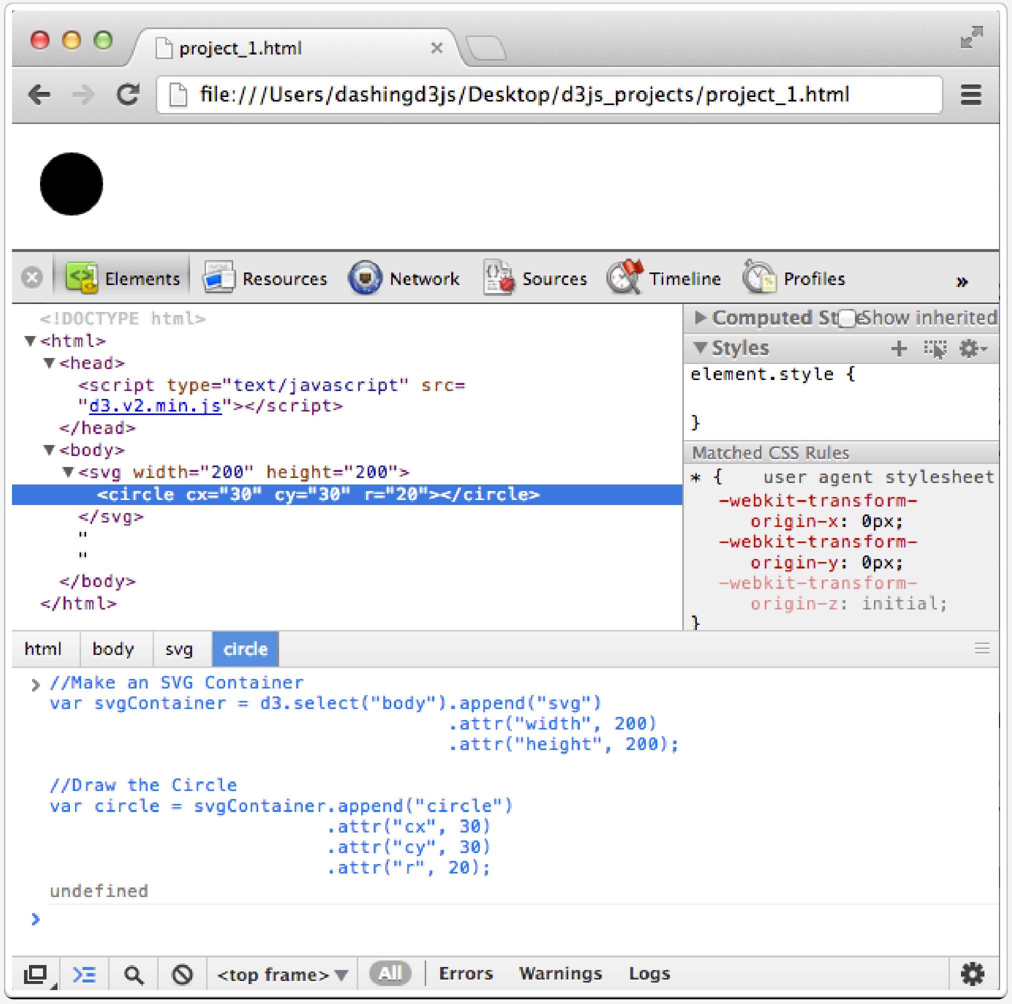
2) Draw the circle

So in the simplest case, our JavaScript code would look like:

```
//Make an SVG Container
var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 200)
                                    .attr("height", 200);

//Draw the Circle
var circle = svgContainer.append("circle")
                         .attr("cx", 30)
                         .attr("cy", 30)
                         .attr("r", 20);
```

Which gives us:



```
<!DOCTYPE html>
▼<html>
  ▼<head>
     <script type="text/javascript" src=
     "d3.v2.min.js"></script>
   </head>
  ▼<body>
    ▼<svg width="200" height="200">
       <circle cx="30" cy="30" r="20"></circle>
     </svg>
     "
     "
   </body>
 </html>
```

```
element.style {

}

* {     user agent stylesheet
  -webkit-transform-
     origin-x: 0px;
  -webkit-transform-
     origin-y: 0px;
  -webkit-transform-
     origin-z: initial;
}
```

```
> //Make an SVG Container
  var svgContainer = d3.select("body").append("svg")
                                      .attr("width", 200)
                                      .attr("height", 200);

  //Draw the Circle
  var circle = svgContainer.append("circle")
                           .attr("cx", 30)
                           .attr("cy", 30)
                           .attr("r", 20);
  undefined
>
```

The necessary SVG attributes for drawing a circle are the "cx", "cy" and "r".

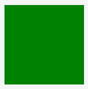
Note - If we leave off the style method, then we get a black circle. Which is fine, we care about making a circle first and foremost, then after that we can style it.

Notice that the important attributes we need to draw an SVG Circle in D3.js are - **cx**, **cy** and **r**.

Drawing an SVG Rectangle using D3.js

We previously saw how to construct an SVG Rectangle in the Basic Building Blocks section:

Rectangle Example & Code:



```
<svg width="50" height="50">
  <rect x="0" y="0" width="50" height="50" fill="green" />
</svg>
```

Using the Circle example, you can probably guess that to build a rectangle, you will need an "x", "y", "width" and "height".

You can then make the process of drawing a rectangle as two step process:

1) Make an SVG container (the SVG Coordinate Space)

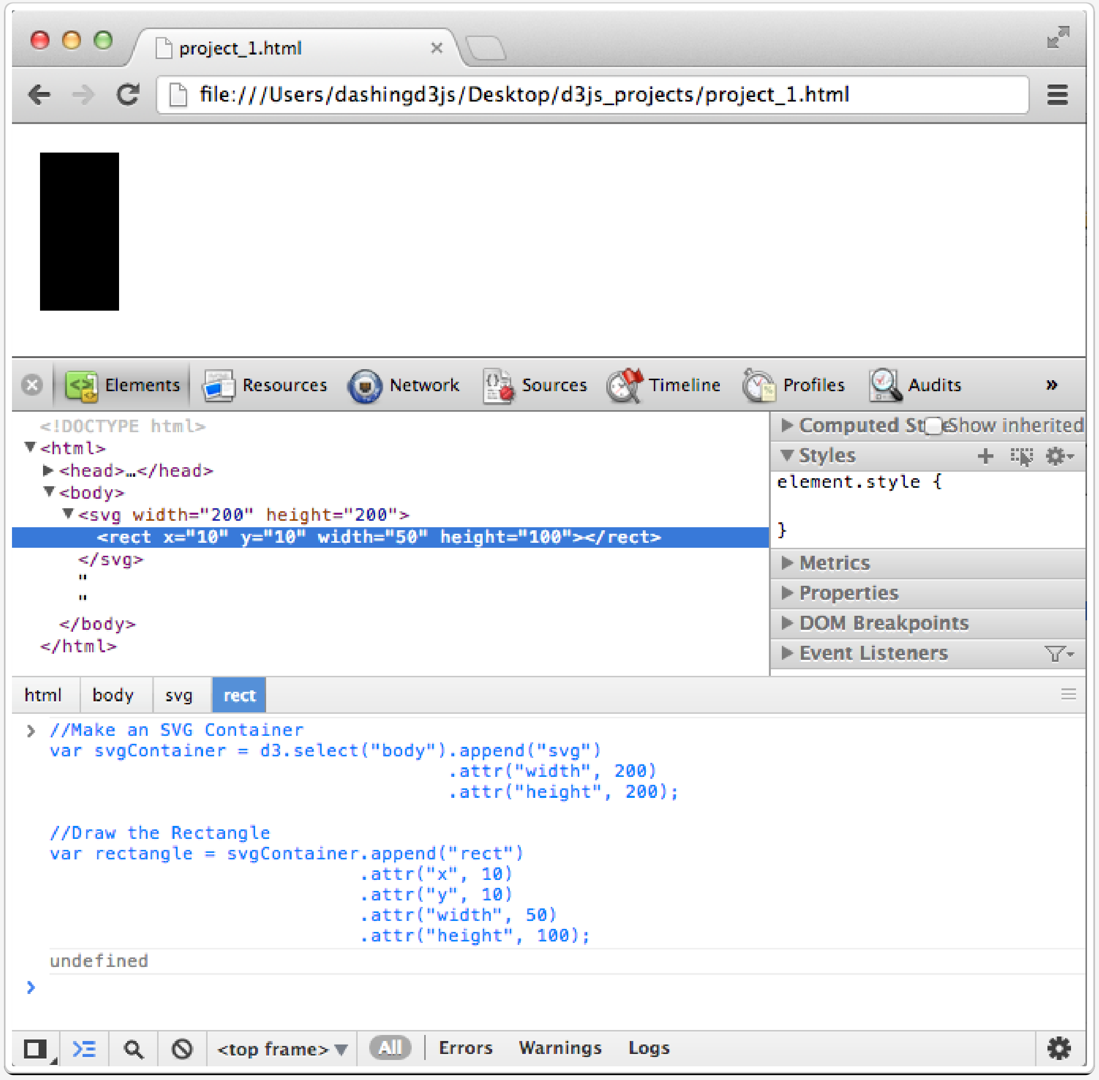
2) Draw the rectangle

So in the simplest case, our JavaScript code would look like:

```
//Make an SVG Container
var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 200)
                                    .attr("height", 200);

//Draw the Rectangle
var rectangle = svgContainer.append("rect")
                            .attr("x", 10)
                            .attr("y", 10)
                            .attr("width", 50)
                            .attr("height", 100);
```

Which gives us:



```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body>
    ▼<svg width="200" height="200">
        <rect x="10" y="10" width="50" height="100"></rect>
      </svg>
      "
      "
    </body>
</html>
```

```
element.style {

}
```

```
> //Make an SVG Container
  var svgContainer = d3.select("body").append("svg")
                                      .attr("width", 200)
                                      .attr("height", 200);

  //Draw the Rectangle
  var rectangle = svgContainer.append("rect")
                              .attr("x", 10)
                              .attr("y", 10)
                              .attr("width", 50)
                              .attr("height", 100);
  undefined
>
```

The necessary SVG attributes for drawing a rectangle are the "x", "y", "width" and "height".

Note - If we leave off the style method, then we get a black rectangle. Which is fine, we care about making a rectangle first and foremost, then after that we can style it.

Two important things to pay attention to

- The SVG word for rectangle is **rect**. Which makes the append operator **.append("rect")** not .append("rectangle")!
- The SVG Coordinate system is in place so the height starts from the **TOP** and moves **down**

Notice that the important attributes we need to draw an SVG Rectangle in D3.js are - **x**, **y**, **width** and **height.**

Drawing an SVG Ellipse using D3.js

We previously saw how to construct an SVG Ellipse in the Basic Building Blocks section:

Ellipse Example & Code:

```
<svg width="50" height="50">
```

```
  <ellipse cx="25" cy="25" rx="15" ry="10" fill="red" />
</svg>
```

Using the Circle and Rectangle examples, you can probably guess that to build an ellipse, you will need a "cx", "cy", "rx" and "ry".

You can then make the process of drawing a ellipse as two step process:

1) Make an SVG container (the SVG Coordinate Space)

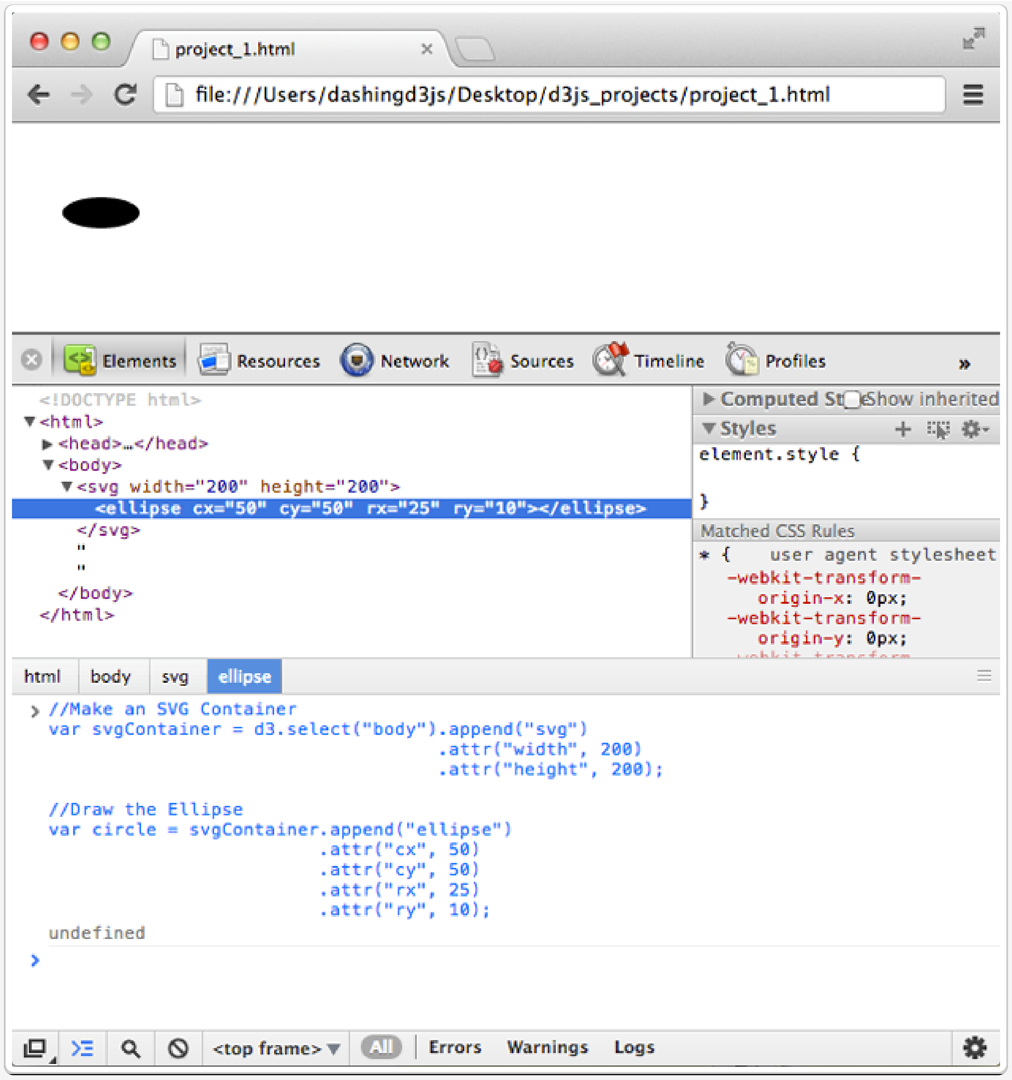
2) Draw the ellipse

So in the simplest case, our JavaScript code would look like:

```
//Make an SVG Container
var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 200)
                                    .attr("height", 200);

//Draw the Ellipse
var circle = svgContainer.append("ellipse")
                         .attr("cx", 50)
                         .attr("cy", 50)
                         .attr("rx", 25)
                         .attr("ry", 10);
```

Which gives us:



```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body>
    ▼<svg width="200" height="200">
        <ellipse cx="50" cy="50" rx="25" ry="10"></ellipse>
      </svg>
      "
      "
    </body>
  </html>
```

```
element.style {
}
* {    user agent stylesheet
  -webkit-transform-
     origin-x: 0px;
  -webkit-transform-
     origin-y: 0px;
```

```
> //Make an SVG Container
  var svgContainer = d3.select("body").append("svg")
                                      .attr("width", 200)
                                      .attr("height", 200);

  //Draw the Ellipse
  var circle = svgContainer.append("ellipse")
                           .attr("cx", 50)
                           .attr("cy", 50)
                           .attr("rx", 25)
                           .attr("ry", 10);
  undefined
>
```

The necessary SVG attributes for drawing an ellipse are the "cx", "cy", "rx" and "ry".

Note - If we leave off the style method, then we get a black ellipse. Which is fine, we care about making a ellipse first and foremost, then after that we can style it.

Notice that the important attributes we need to draw an SVG Ellipse in D3.js are - **cx**, **cy**, **rx** and **ry**.

Drawing an SVG Straight Line using D3.js

We previously saw how to construct an SVG Straight Line in the Basic Building Blocks section:

Straight Line Example & Code:

```
<svg width="50" height="50">
  <line x1="5" y1="5" x2="40" y2="40" stroke="gray" stroke-width="5"  />
</svg>
```

Using the Circle, Rectangle, and Ellipse examples, you can probably guess that to build a straight line, you will need a "x1", "y1", "x2" and "y2".

You can then make the process of drawing a straight line as two step process:

1) Make an SVG container (the SVG Coordinate Space)

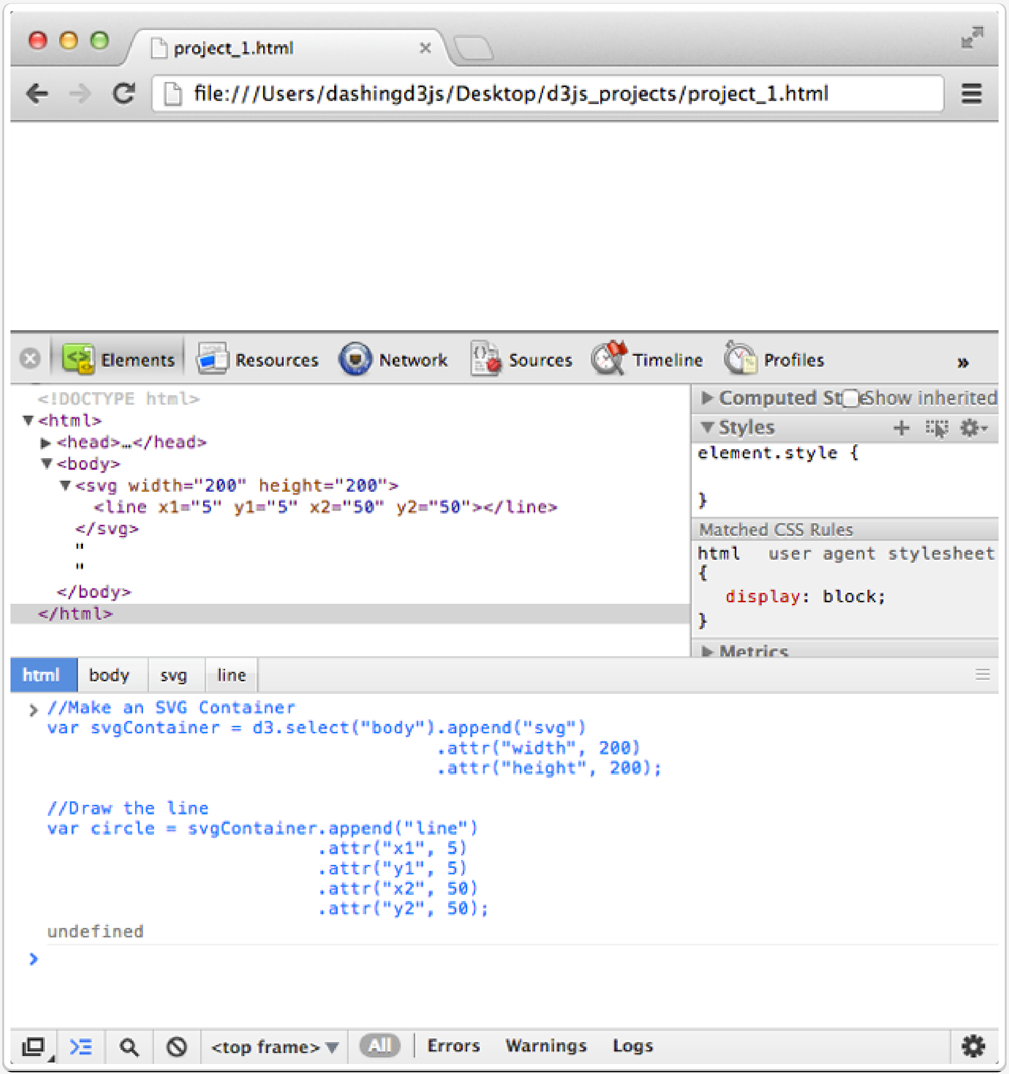
2) Draw the straight line

So in the simplest case, our JavaScript code would look like:

```
//Make an SVG Container
var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 200)
                                    .attr("height", 200);

//Draw the line
var circle = svgContainer.append("line")
                         .attr("x1", 5)
                         .attr("y1", 5)
                         .attr("x2", 50)
                         .attr("y2", 50);
```

Which gives us:



```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body>
    ▼<svg width="200" height="200">
        <line x1="5" y1="5" x2="50" y2="50"></line>
      </svg>
      "
      "
    </body>
</html>
```

```
element.style {

}
html    user agent stylesheet
{
   display: block;
}
```

```
> //Make an SVG Container
  var svgContainer = d3.select("body").append("svg")
                                      .attr("width", 200)
                                      .attr("height", 200);

  //Draw the line
  var circle = svgContainer.append("line")
                           .attr("x1", 5)
                           .attr("y1", 5)
                           .attr("x2", 50)
                           .attr("y2", 50);
  undefined
>
```

Wait a second?! Where is the line????? The SVG element is there, however we can't see it in our browser.

It turns out because the SVG 'line' elements are single lines and thus are geometrically one-dimensional, that is - they have no interior.

Which means that ‘line’ elements are never filled (see the ‘fill’ property).

Which means that our line does not take up space - so we don't actually see anything.

To fix this, make sure to give the line:

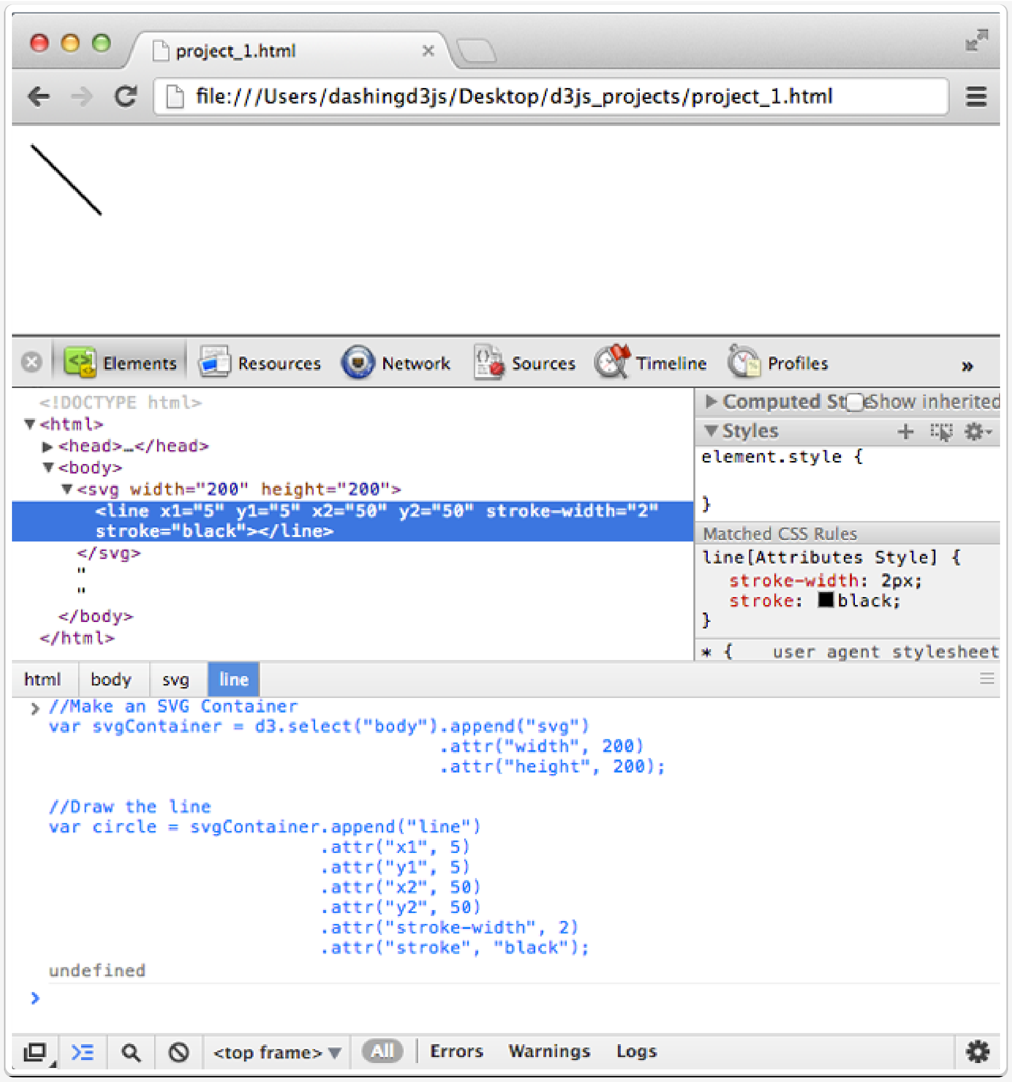
- **.attribute("stroke-width", NUMBER)**, where NUMBER is how wide the line is in units
- **.attribute("stroke", "COLOR")**, where COLOR is a color to used to color the line

So fixing our simplest case, our JavaScript code would look like:

```
        //Make an SVG Container
var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 200)
                                    .attr("height", 200);

//Draw the line
var circle = svgContainer.append("line")
                          .attr("x1", 5)
                          .attr("y1", 5)
                          .attr("x2", 50)
                          .attr("y2", 50)
                          .attr("stroke-width", 2)
                          .attr("stroke", "black");
```

Which gives us:



```
<!DOCTYPE html>
▼<html>
  ▶<head>…</head>
  ▼<body>
    ▼<svg width="200" height="200">
        <line x1="5" y1="5" x2="50" y2="50" stroke-width="2"
        stroke="black"></line>
      </svg>
      "
      "
    </body>
</html>
```

```
element.style {

}
line[Attributes Style] {
   stroke-width: 2px;
   stroke: ■black;
}
* {     user agent stylesheet
```

```
> //Make an SVG Container
  var svgContainer = d3.select("body").append("svg")
                                       .attr("width", 200)
                                       .attr("height", 200);

  //Draw the line
  var circle = svgContainer.append("line")
                           .attr("x1", 5)
                           .attr("y1", 5)
                           .attr("x2", 50)
                           .attr("y2", 50)
                           .attr("stroke-width", 2)
                           .attr("stroke", "black");
  undefined
>
```

Fantastic - now the line is visible!

The necessary SVG attributes for drawing a straight line are the "x1", "y1", "x2", "y2", "stroke" and "stroke-width".

Note - We don't use a style method with the line. Because ‘line’ elements are single lines and thus are geometrically one-dimensional, they have no interior. Which is why to style them we need to deal with the "stroke" color and "stroke-width".

Notice that the important attributes we need to draw an SVG Straight Line in D3.js are - **x1**, **y1**, **x2**, **y2**, **stroke-width** and **stroke**.

Drawing Polyline & Polygon SVG Basic Shapes using D3.js

The basic shapes we still have to cover are "Polyline" and "Polygon".

We previously saw how to construct an SVG Polyline and Polygon in the Basic Building Blocks section:

Polyline Example & Code:

```
<svg width="50" height="50">
  <polyline fill="none" stroke="blue" stroke-width="2"
    points="05,30
            15,30
            15,20
            25,20
            25,10
            35,10" />
</svg>
```

Polygon Example & Code:

```
<svg width="50" height="50">
  <polygon fill="yellow" stroke="blue" stroke-width="2"
    points="05,30
            15,10
            25,30" />
</svg>
```

Using the Circle, Rectangle, Ellipse, and Straight Line examples, you can probably guess that to build a Polyline and Polygon, you will need a "stroke", "stroke-width" and "points". And "fill" for the Polygon.

However, as you can see from the examples above - the **points** attribute contains a list of points where the x and y are separated by comas and different x and y coordinates are separated by spaces.

Building this out easily in D3.js is not pretty. Because D3.js loves data visualization and pretty things, the D3.js convention is to use the D3.svg.line() generator for polylines and polygons.

To construct the Polyline and Polygon SVG Basic Shapes using D3.js, we will have to learn about **SVG Paths**.

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

← Using JSON to Simplify Code

SVG Paths and D3.js →

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? :)