Dynamic SVG Coordinate Space

The Goal

In this section, we will cover how to make the SVG Coordinate Space dynamic so that our Data Visualization is visible regardless of the data.

We will the make the SVG Coordinate Space scale up and/or down to fit our data.

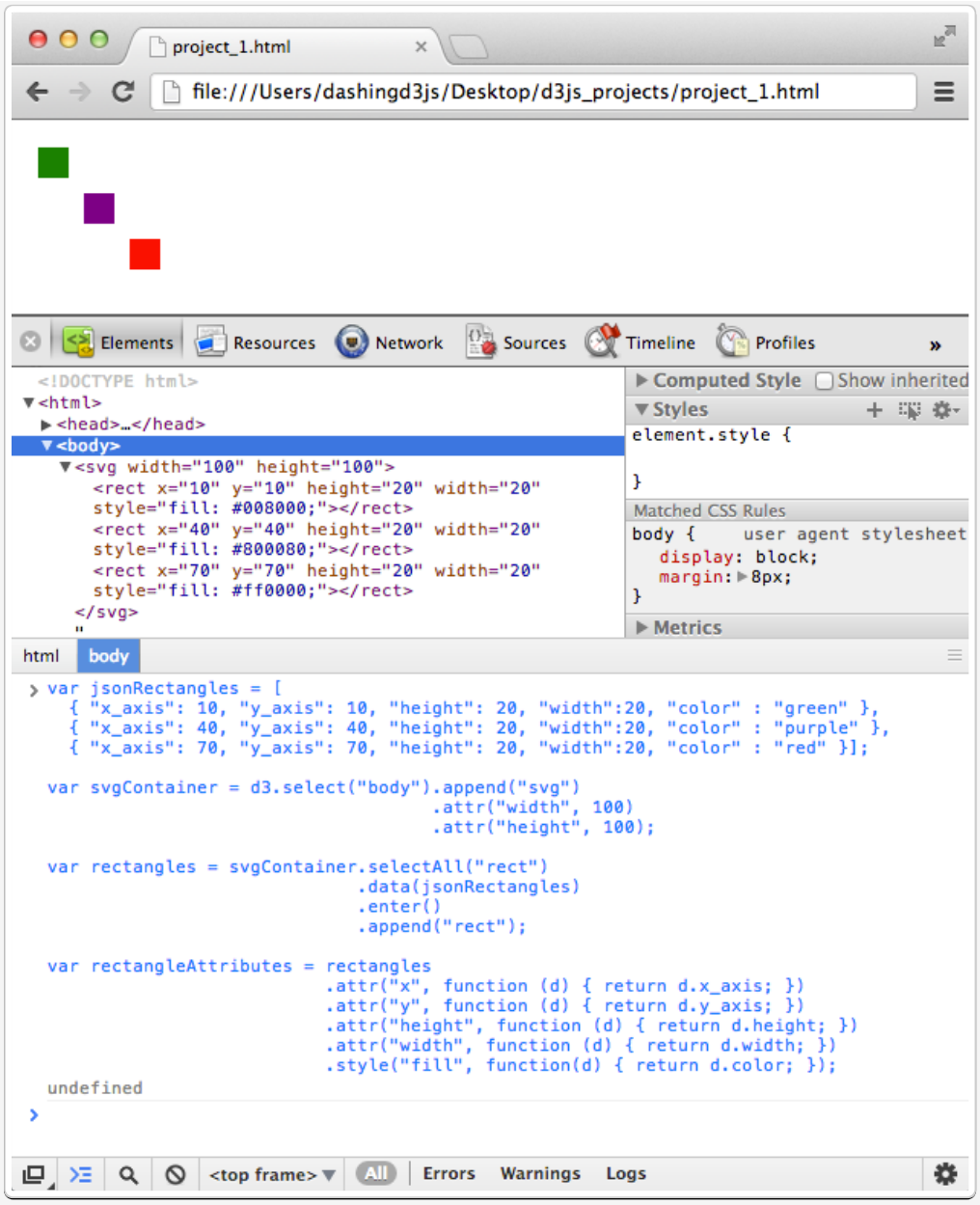
Three SVG Rectangle Example

We start with three rectangles

(similar to our example in the Using JSON to Simplify Code section).

```
var jsonRectangles = [
  { "x_axis": 10, "y_axis": 10, "height": 20, "width":20, "color" : "green" },
  { "x_axis": 40, "y_axis": 40, "height": 20, "width":20, "color" : "purple" },
  { "x_axis": 70, "y_axis": 70, "height": 20, "width":20, "color" : "red" }];

var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 100)
                                    .attr("height", 100);

var rectangles = svgContainer.selectAll("rect")
                             .data(jsonRectangles)
                             .enter()
                             .append("rect");

var rectangleAttributes = rectangles
                          .attr("x", function (d) { return d.x_axis; })
                          .attr("y", function (d) { return d.y_axis; })
                          .attr("height", function (d) { return d.height; })
                          .attr("width", function (d) { return d.width; })
                          .style("fill", function(d) { return d.color; });
```

Which gives us:



Which is great.

The SVG Viewport (container)

```
var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 100)
                                    .attr("height", 100);
```

has a width of 100 units and a height of 100 units.

Which means that the lower right most point of the red rectangle lands at (90,90), which is still inside of our view port.

What if our purple rectangle x-coordinate, suddenly quadrupled from 40 to 160?

```
//Going from
{ "x_axis": 40, "y_axis": 40, "height": 20, "width":20, "color" : "purple" }

//to
{ "x_axis": 160, "y_axis": 40, "height": 20, "width":20, "color" : "purple" }

```

```
7  //so that our jsonRectangles becomes
8  var jsonRectangles = [
9  { "x_axis": 10, "y_axis": 10, "height": 20, "width":20, "color" : "green" },
10 { "x_axis": 160, "y_axis": 40, "height": 20, "width":20, "color" : "purple" },
11 { "x_axis": 70, "y_axis": 70, "height": 20, "width":20, "color" : "red" }];
```

This would mean that the purple rectangle would have x,y coordinates of **(160,40)**.

This coordinate is out of our viewport which has a width of 100 units and a height of 100 units.

Suddenly our Data Visualization would look like this:

As you can imagine, **This is not good!**.

Manually Adjusting SVG Container Space

To get our new data to fit inside of our SVG container, we would have to increase the width of the container to accommodate the new x-coordinate for the purple triangle.

Since the new purple rectangle is defined as follows:

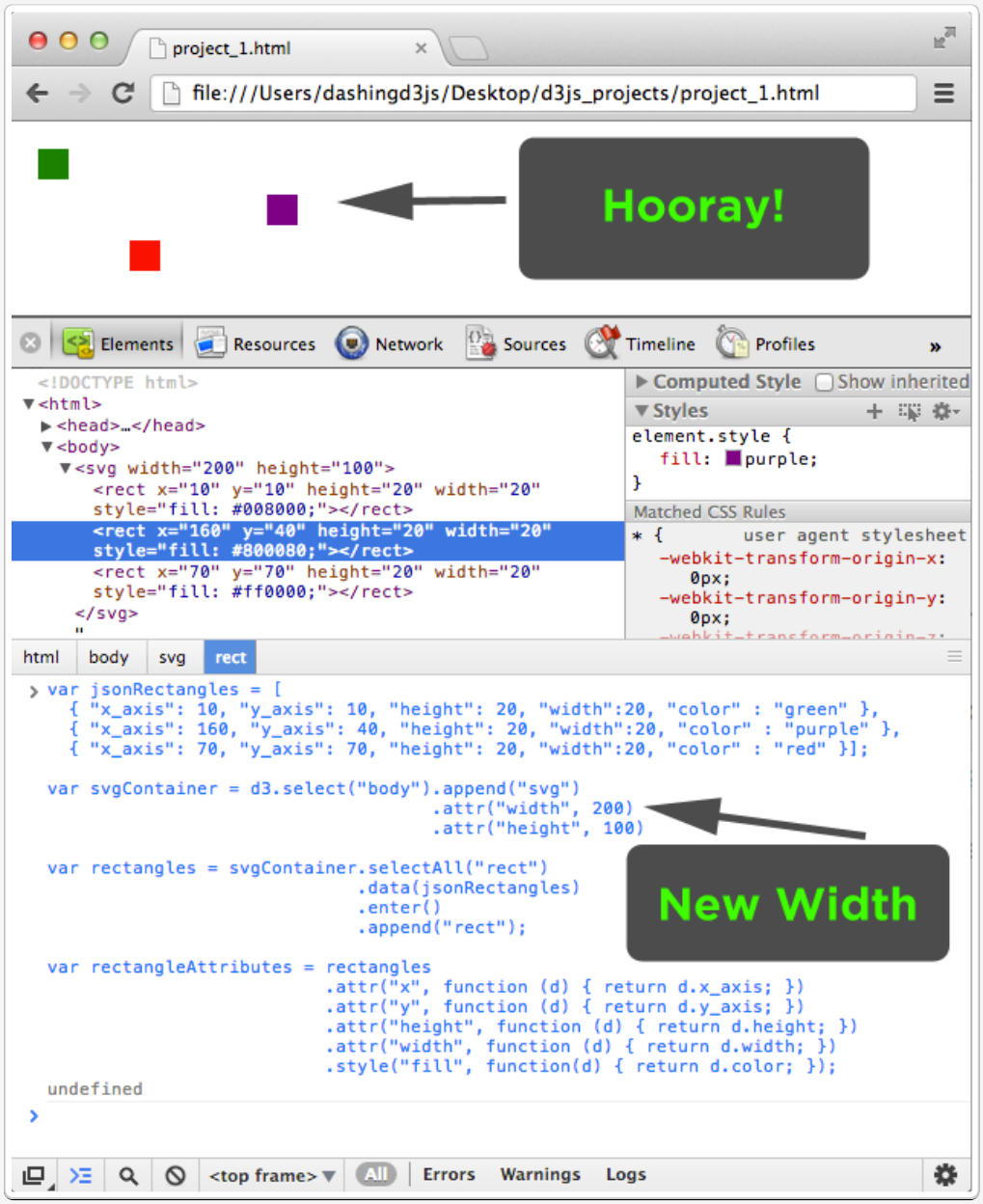
```
{ "x_axis": 160, "y_axis": 40, "height": 20, "width":20, "color" : "purple" }
```

It will have coordinates (160,40) at the top left and (180,60) at the bottom right.

So we could just re-write the SVG Viewport (container) to have a width that was much bigger than 180, say 200...

```
var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 200)
                                    .attr("height", 100);
```

Suddenly our Data Visualization would look like this:



Which is great, until the x-coordinate increases again. Or, the y-coordinate increases, or ... etc.

Dynamically Adjusting SVG Container Space

As you can guess, **what we really want to do** is to dynamically change the width and height attributes of the SVG Container/Viewport according to our data.

We are going to use some basic JavaScript to loop through our array of JSON objects to get the max x-coordinate and the max y-coordinate.

The max x-coordinate and max y-coordinate will be the bottom right hand point of the rectangle.

```
//New jsonRectangles Data (with purple rectangle x-coordinate now 160)
var jsonRectangles = [
  { "x_axis": 10, "y_axis": 10, "height": 20, "width":20, "color" : "green" },
  { "x_axis": 160, "y_axis": 40, "height": 20, "width":20, "color" : "purple" },
  { "x_axis": 70, "y_axis": 70, "height": 20, "width":20, "color" : "red" }];

var max_x = 0; //This will be updated to be the max x-coordinate
var max_y = 0; //This will be updated to be the max y-coordinate

//We loop through our jsonRectangles array
for (var i = 0; i < jsonRectangles.length; i++) {

  var temp_x, temp_y;

  // To get the farthest right hand point, we need to add the x-coordinate and the width
  var temp_x = jsonRectangles[i].x_axis + jsonRectangles[i].width;

  // To get the farthest bottom point, we need to add the y-coordinate and the height
  var temp_y = jsonRectangles[i].y_axis + jsonRectangles[i].height;

  /**
  * If the temporary x-coordinate is bigger than the max_x,
  * make the max_x equal to the temp_x
  * otherwise, do nothing.
  */
  if ( temp_x >= max_x ) {
    max_x = temp_x;
  }

  /**
  * If the temporary y-coordinate is bigger than the max_y,
  * make the max_y equal to the temp_y
  * otherwise, do nothing.
  */
  if ( temp_y >= max_y ) {
    max_y = temp_y;
  }

}//End of the loop

max_x;
//returns 180

max_y;
//returns 90
```

If you run this in the JavaScript Console, you will get that the max_x is 180 and the max_y is 90.

If the data changes, this max_x and max_y will always have maximum values of our data.

Now we have to update our SVG Container Viewport:

```
//From
```

```
var svgContainer = d3.select("body").append("svg")
                                    .attr("width", 200)
                                    .attr("height", 100);

//to (using the new max_x and max_y variables)
var svgContainer = d3.select("body").append("svg")
                                    .attr("width", max_x + 20)
                                    .attr("height", max_y + 20);
//Note - we add 20 units to the max_x and max_y to give the elements some stylistic room
```

Notice the note - we add some space to the max_x and max_y to give the elements some stylistic room.

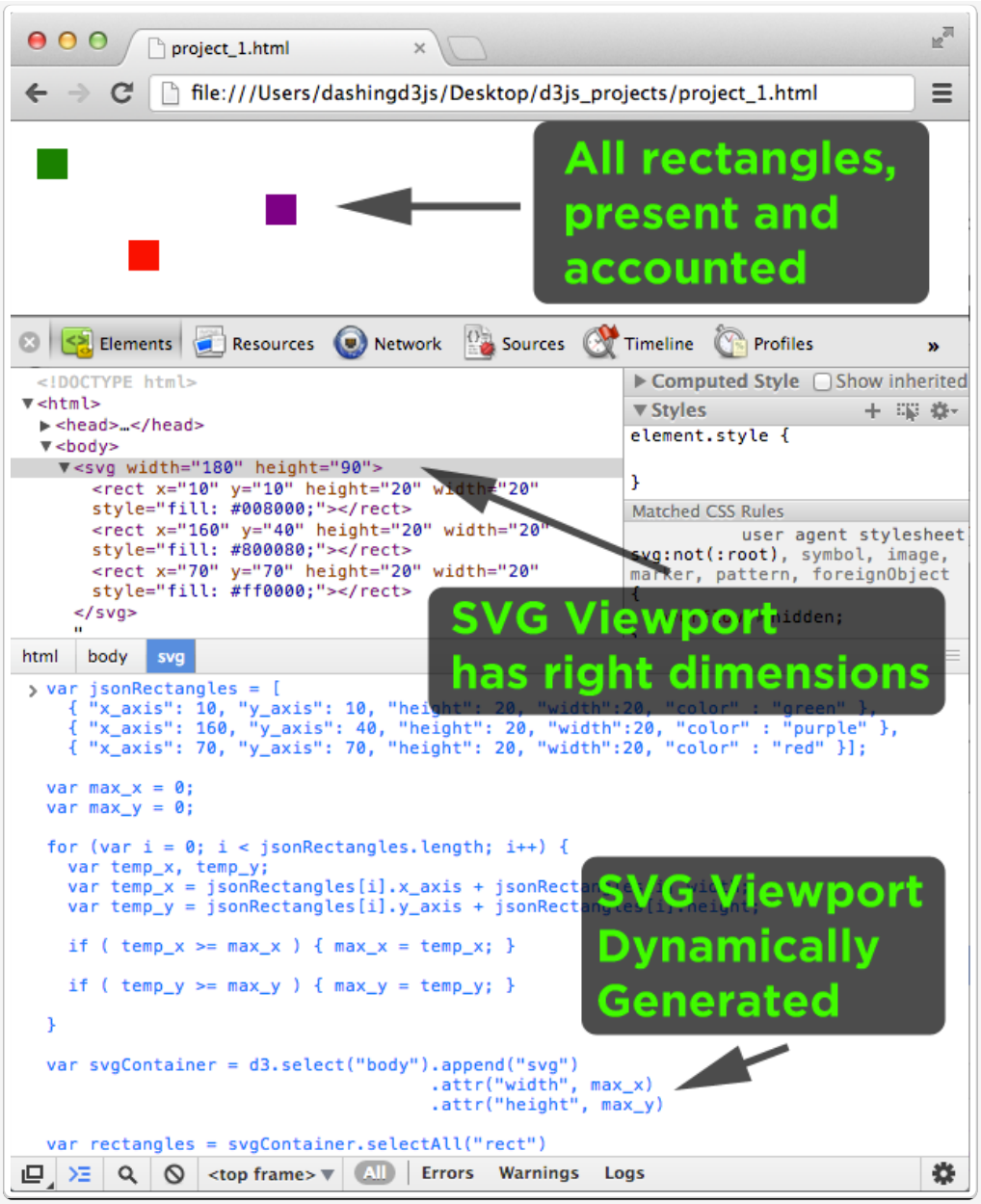
In this way, our SVG Container will now always display the right dimensions so that the data fits correctly inside of it.

The Finished Product

Now that we've figured that out, the full code with the updated jsonRectangles data now reads (javascript comments removed):

```
var jsonRectangles = [
  { "x_axis": 10, "y_axis": 10, "height": 20, "width":20, "color" : "green" },
  { "x_axis": 160, "y_axis": 40, "height": 20, "width":20, "color" : "purple" },
  { "x_axis": 70, "y_axis": 70, "height": 20, "width":20, "color" : "red" }];

var max_x = 0;
var max_y = 0;

for (var i = 0; i < jsonRectangles.length; i++) {
  var temp_x, temp_y;
  var temp_x = jsonRectangles[i].x_axis + jsonRectangles[i].width;
  var temp_y = jsonRectangles[i].y_axis + jsonRectangles[i].height;

  if ( temp_x >= max_x ) { max_x = temp_x; }

  if ( temp_y >= max_y ) { max_y = temp_y; }
}

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

var rectangles = svgContainer.selectAll("rect")
                             .data(jsonRectangles)
                             .enter()
                             .append("rect");

var rectangleAttributes = rectangles
                          .attr("x", function (d) { return d.x_axis; })
                          .attr("y", function (d) { return d.y_axis; })
                          .attr("height", function (d) { return d.height; })
                          .attr("width", function (d) { return d.width; })
                          .style("fill", function(d) { return d.color; });
```

Which gives us:



There we go!

All of the rectangles are there.

The SVG Viewport has the right dimensions (max_x + 20, max_y + 20).

And the SVG Viewport was generated dynamically so we didn't have to manually update the width and height.

Using a JavaScript FOR loop we were able to dynamically resize our SVG Viewport Container to fit our data.

So if/when the data changes again, this viewport will be ready to contain all of the Data Visualization.

← SVG Paths and D3.js

D3.js Scales →

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? :)