# Using JSON to Simplify Code

## The Goal

In this section, you will understand how using JSON can simplify your D3.js code as well as your JavaScript code.

We will cover previous examples, how to bind JSON objects to the __data__ attribute and how we can use JSON to clean up the code for easier comprehension.

## Previous Example of Three Circles

In the Using the SVG Coordinate Space section, we created and styled three circles using D3.js

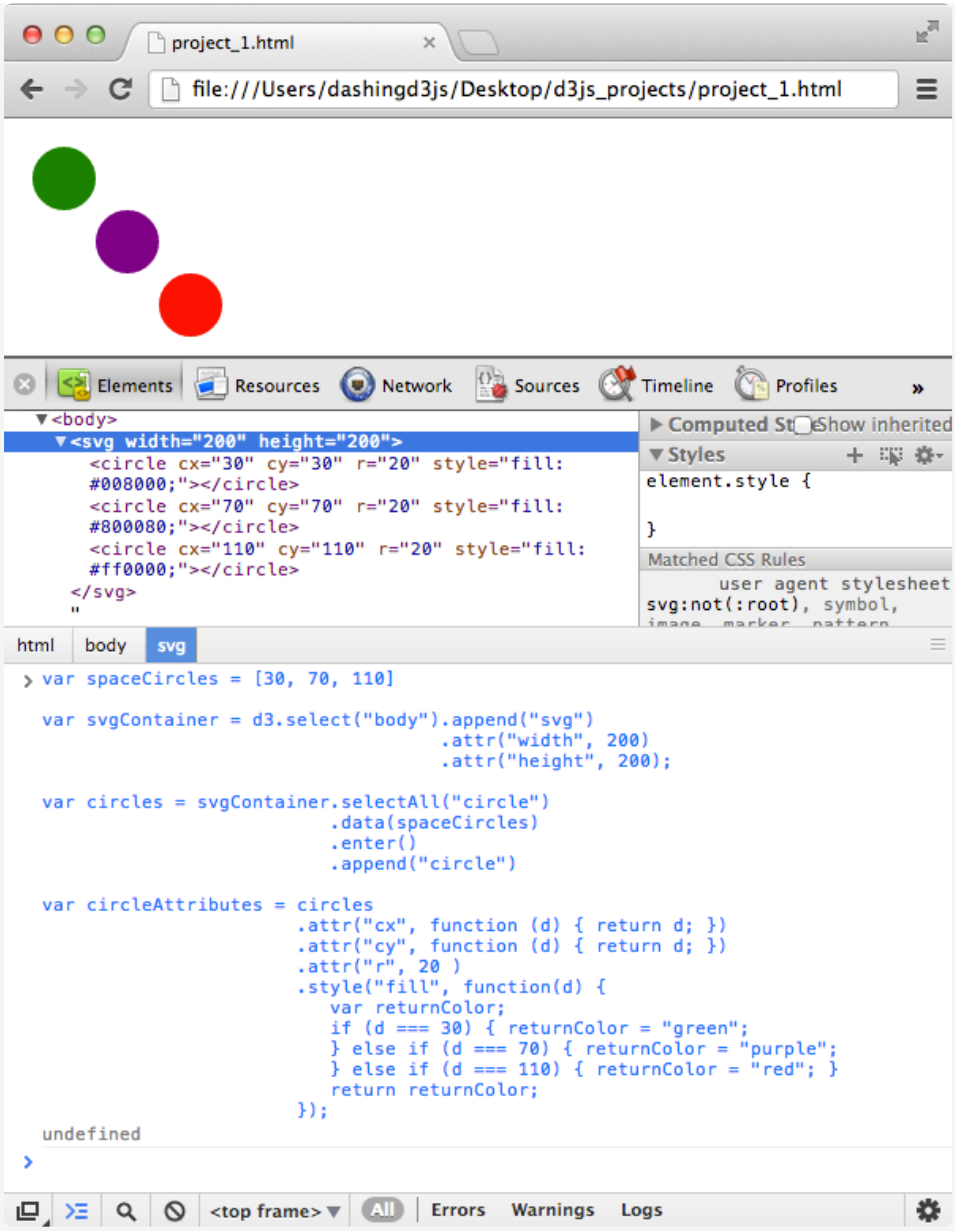
```
var spaceCircles = [30, 70, 110];

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 )
                       .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 gaves us this:




This example used the data to set the **cx**, **cy** and **style fill** for each circle.

However, as you probably noticed, the cx and cy units were the same.

Also - perhaps more dangerous, was that the ***if*** statements had hard coded values.

Hard coded values are never a good idea, because if our data changes, then we would have to change our code in several places.

## Binding JSON Objects to the __data__ Attribute

In the Data Structures D3.js Accepts section, we covered how we could use JavaScript JSON Objects in the Data Array we pass to D3.js.

Using our example above (the three circles) and the JSON notation, we could write our data array as follows:

```
1  var jsonCircles = [
2    {
3     "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"
}];
```

Note that we have moved our color from the ***if ... then*** functions in the D3.js code to our data.

This will not always be possible. For now, we will assume it is possible and deal with other cases later.

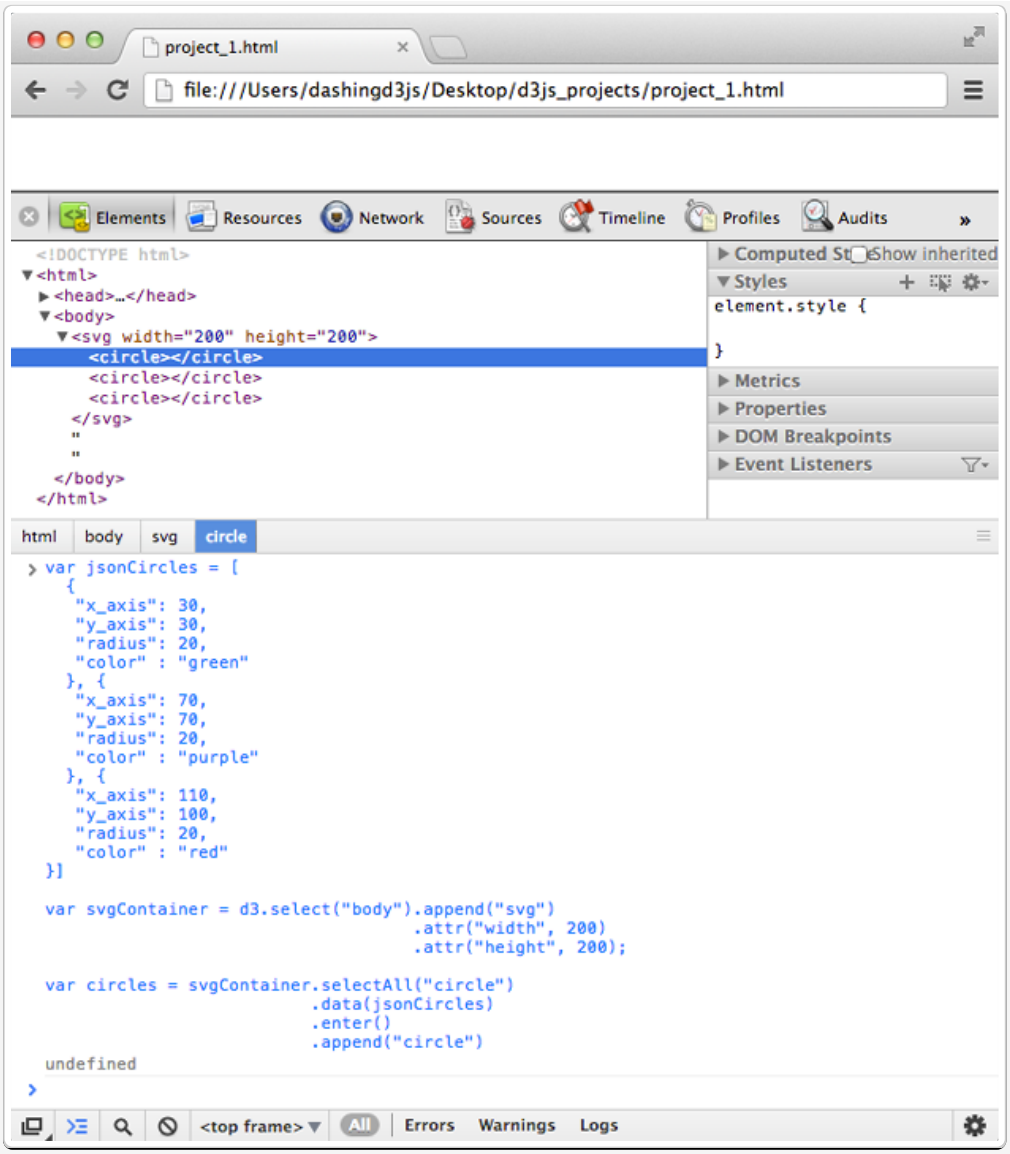
In doing this, we can now type the following into our JavaScript Console:

```
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");
```

Which gives us this:




Which we can check in the console.log to see what data was bound to each of the three circles:

```
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"
}];

 console.log(d3.select("body").append("svg").attr("width", 200).attr("height", 200).selectAll("circle").data(jsonCircles).enter().append("circle"));
```

This shows us that an **object** was bound to the __**data**__ attribute of the circle:


Which is fantastic!

Before we had been able to bind a single number to the __**data**__ attribute. Now we have bound a JSON object to the __**data**__ attribute.

And because the object is bound to the __data__ attribute, it means that D3.js can use it within its operator and methods!

## Use Bound JSON Objects to Alter SVG Elements

We will now use the bound JSON Objects in the __data__ attribute to alter the SVG Circles.

If you recall the Creating SVG Elements Based on Data section, we discussed the key to using the bound data, being the JavaScript function:

```
function (d) { return d; }
```

If you recall the previous section Data Structures D3.js Accepts, we covered how JSON objects allow us to use their name/value pair to call specific data from them:

```
var secretAgent = {
  "name":"James Bond",
  "drink":"dry martini - shaken, not stirred",
  "number":"007"
};

secretAgent.number;
// "007" is returned.
```

**Combining** these two pieces of knowledge, allows us to write the following in D3.js:

```
function (d) { return d.name; }
```

Where **name** is the name associated with the specific name/value pair in the JSON Object.

In our case, using our jsonCircles JSON data array:

```
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"
}];
```

We can write functions like the following:

```
function (d) { return d.x_axis; }

function (d) { return d.y_axis; }

function (d) { return d.radius; }

function (d) { return d.color; }
```

This leads to code with easier comprehension.

Which means we can write our earlier example as follows:

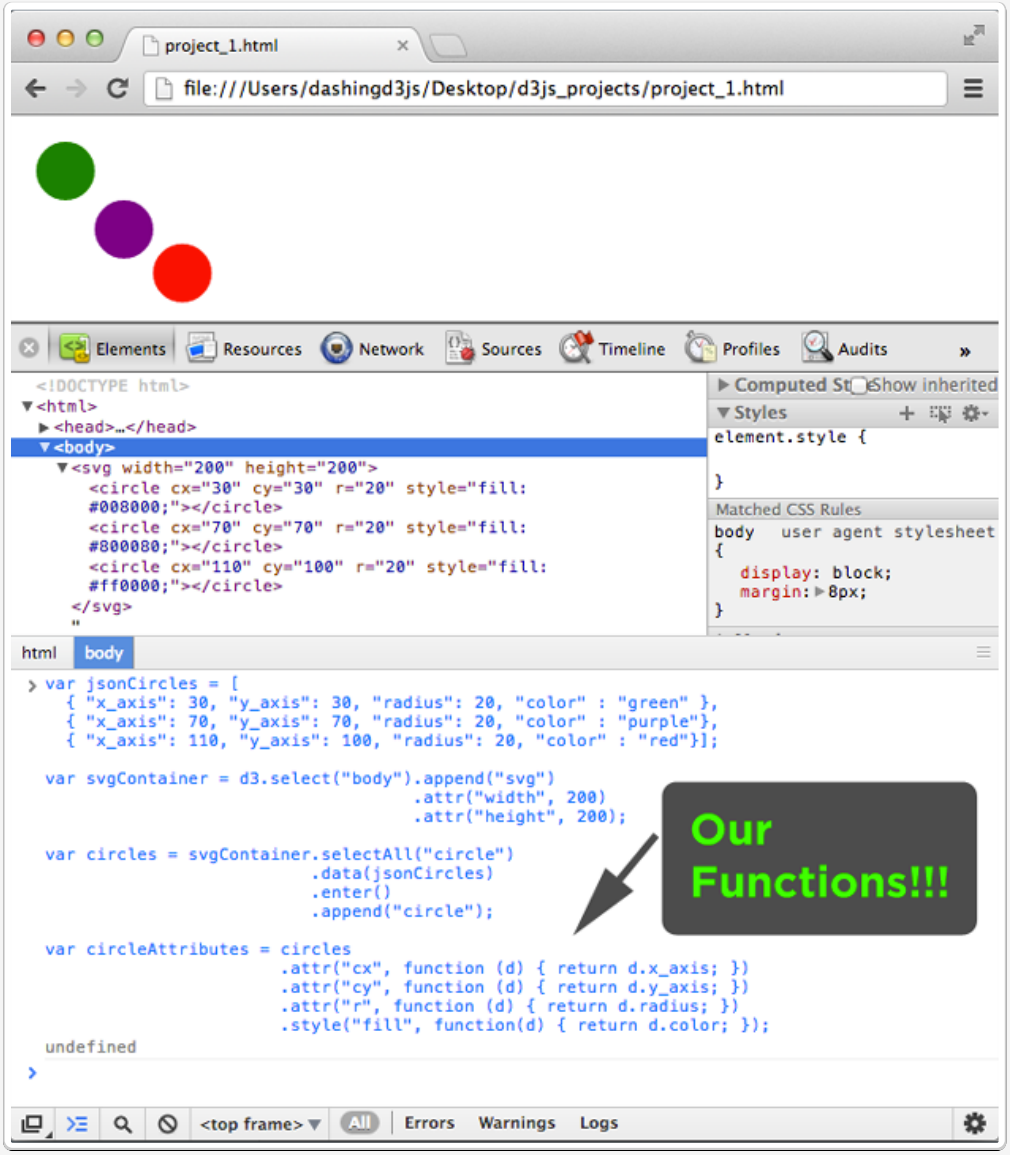
```
var jsonCircles = [
  { "x_axis": 30, "y_axis": 30, "radius": 20, "color" : "green" },
  { "x_axis": 70, "y_axis": 70, "radius": 20, "color" : "purple"},
  { "x_axis": 110, "y_axis": 100, "radius": 20, "color" : "red"}];

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; });
```

Which gives us:




Bravo!

We cleaned up our code by removing the hardcoded if/then statements from our .style() operator.

We also made it easier to understand what the **return d** was actually returning.

We have now successfully used JSON to create D3.js Data Visualizations as well as used functions in the operators and methods to style our SVG elements.

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

← Data Structures D3.js Accepts

SVG Basic Shapes and D3.js →

D3 Mapping Training
D3 Introductory Training
D3 Intermediate Training
D3 Advanced Training
D3 Corporate Training

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