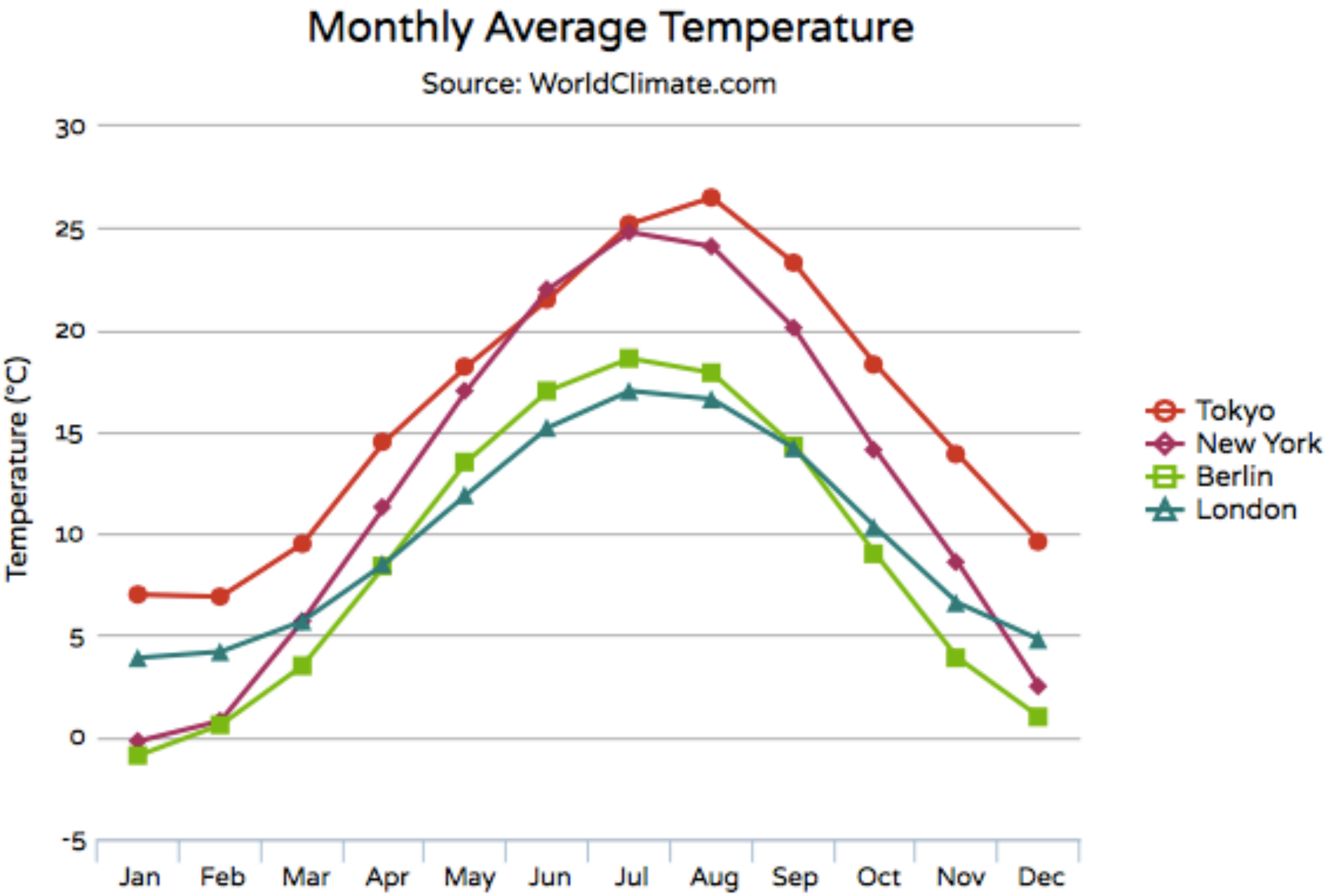
Introducing D3.js

D3 is Not a Charting Library

```
$('#container').highcharts({
  title: {  text: 'Monthly Average Temperature', x: -20 },
  subtitle: { text: 'Source: WorldClimate.com', x: -20 },
  xAxis: { categories: ['Jan', 'Feb', 'Mar', 'Apr', 'May', //…
  yAxis: { title: { text: 'Temperature (°C)' },
       plotLines: [{value: 0, width: 1, color: '#808080'}] },
  legend: { layout: 'vertical', align: 'right',
        verticalAlign: 'middle', borderWidth: 0 },
  series: [{ name: 'Tokyo',    data: [ 7.0, 6.9, 9.5, 14.5, //…
           { name: 'New York', data: [-0.2, 0.8, 5.7, 11.3, //…
           { name: 'Berlin',   data: [-0.9, 0.6, 3.5,  8.4, //…
           { name: 'London',   data: [ 3.9, 4.2, 5.7,  8.5, //…
});
```

Where One Statement = A Chart



Charting Library - Buying a Home

D3 - Buying a Home at Home Depot

D3 Philosophy

- D3 is not really a “visualization library”; it does not draw visualizations
- D3 = “Data Driven Documents”; it primarily associates data with DOM elements and manages the results
- D3 also provides tools that you can use to draw visualizations

D3 Components

- Core: selections, transitions, data, localization, colors,
- Scales: convert between data and visual encodings
- SVG: utilities for creating Scalable Vector Graphics
- Time: parse/format times, compute calendar intervals,
- Layouts: derive data for positioning elements
- Geography: project spherical coord., lat/long math
- Geometry: utilities for 2D geometry, e.g. Voronoi,
- Behaviors: reusable interaction behaviors

Let’s Build a Chart

1. Setup and Scaffolding: HTML, JSON, and AJAX
2. D3 Scales: Map data ⇒ DOM
3. Draw with SVG

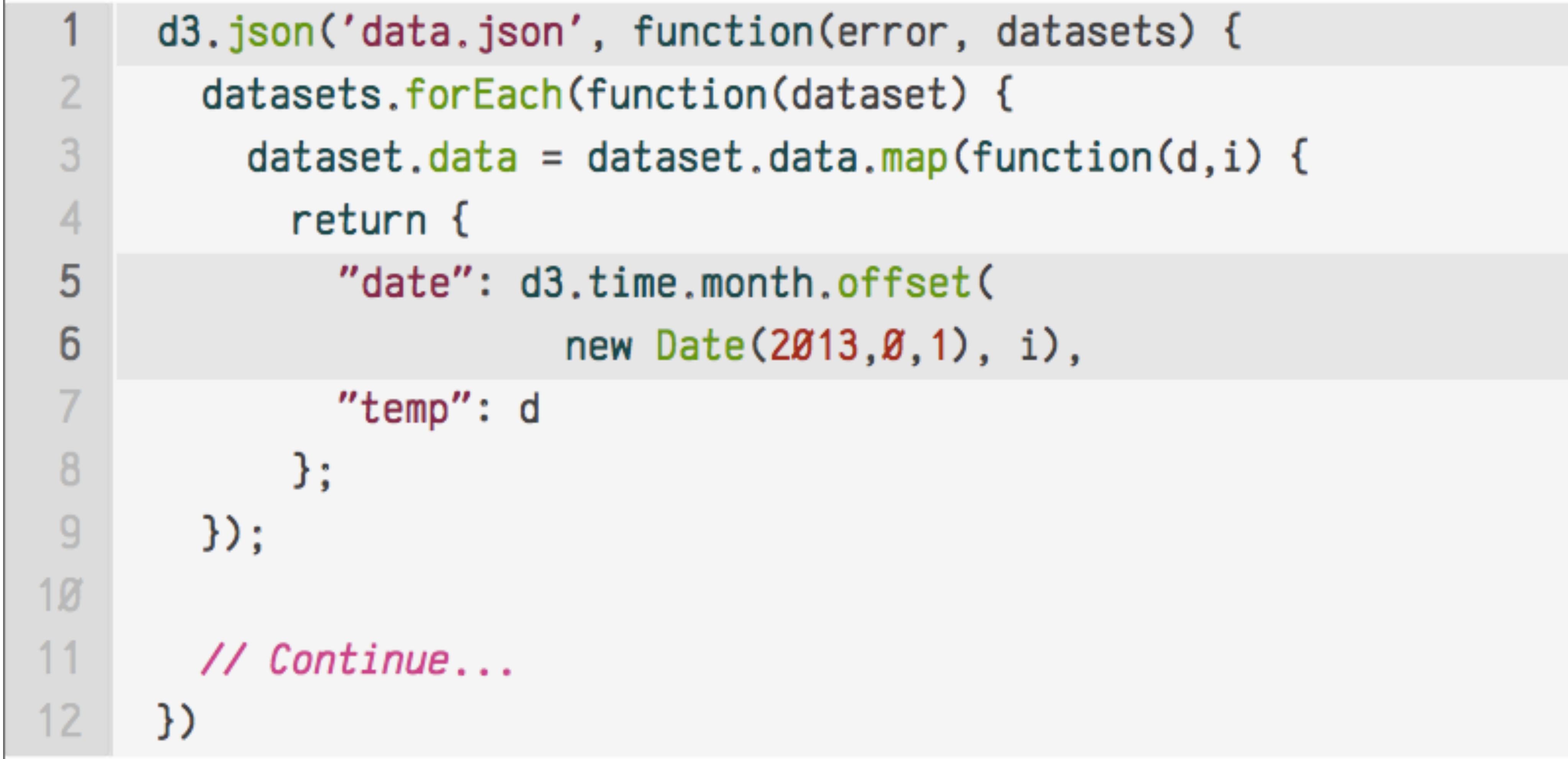
HTML Scaffolding

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=’utf-8’>
  <title>Basic line demo</title>
</head>
<body>
  <script src=’http://d3js.org/d3.v3.min.js’></script>
</body>
</html>
```

Data in JSON Format

```
[{
  ”name”: ”Tokyo”,
  ”data”: [ 7.0, 6.9, 9.5, 14.5, 18.2, 21.5, 25.2, 26.5, //…
 },{
  ”name”: ”New York”,
  ”data”: [-0.2, 0.8, 5.7, 11.3, 17.0, 22.0, 24.8, 24.1, //…
 },{
  ”name”: ”Berlin”,
  ”data”: [-0.9, 0.6, 3.5,  8.4, 13.5, 17.0, 18.6, 17.9, //…
 },{
  ”name”: ”London”,
  ”data”: [ 3.9, 4.2, 5.7,  8.5, 11.9, 15.2, 17.0, 16.6, //…
}]
```

Retrieve the Data



```
d3.json('data.json', function(error, datasets) {
  datasets.forEach(function(dataset) {
    dataset.data = dataset.data.map(function(d,i) {
      return {
        "date": d3.time.month.offset(
                  new Date(2013,0,1), i),
        "temp": d
      };
  });

  // Continue...
})
```

Use a Scale to Map Data ⇒ DOM

```
var y = d3.scale.linear()
          .range([height, 0])
          .domain(d3.extent(dataset, dataset.temp))
          .nice();
```

- Range is from `height` to `0` because SVG coordinates position y-value of `0` at top
- `d3.extent()` finds minimum and maximum of array with defined accessor
- `d3.nice()` rounds scale to “human-friendly” values

Scales Don't Have to be Linear

```
var x = d3.time.scale()
          .range([0, width])
          .domain(
            d3.extent(dataset, dataset.date)
              .map(function(d, i) {
                d3.time.day.offset(d, i ? 15 : -16)
            })
        );
```

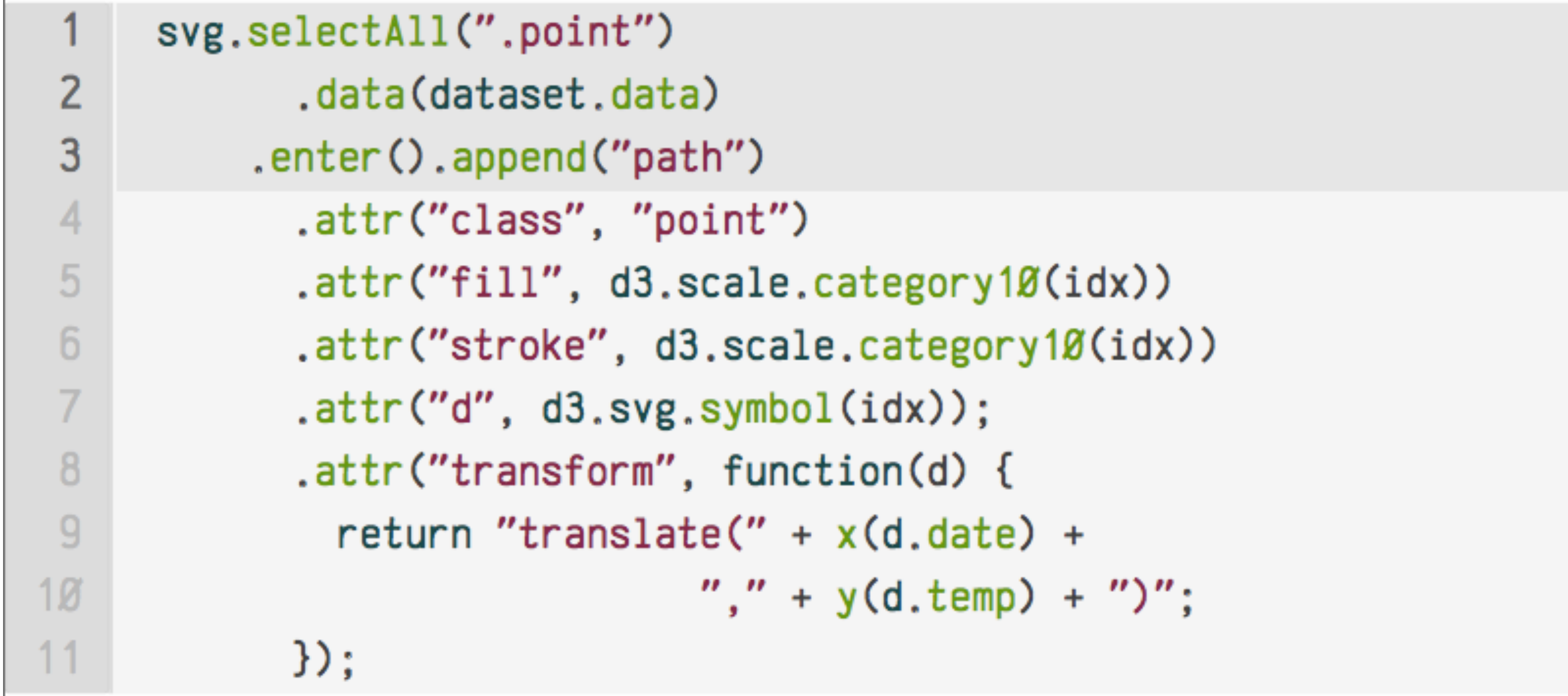
- Domain of x-axis extended 16 days before and 15 days after data values

Create the SVG Container

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

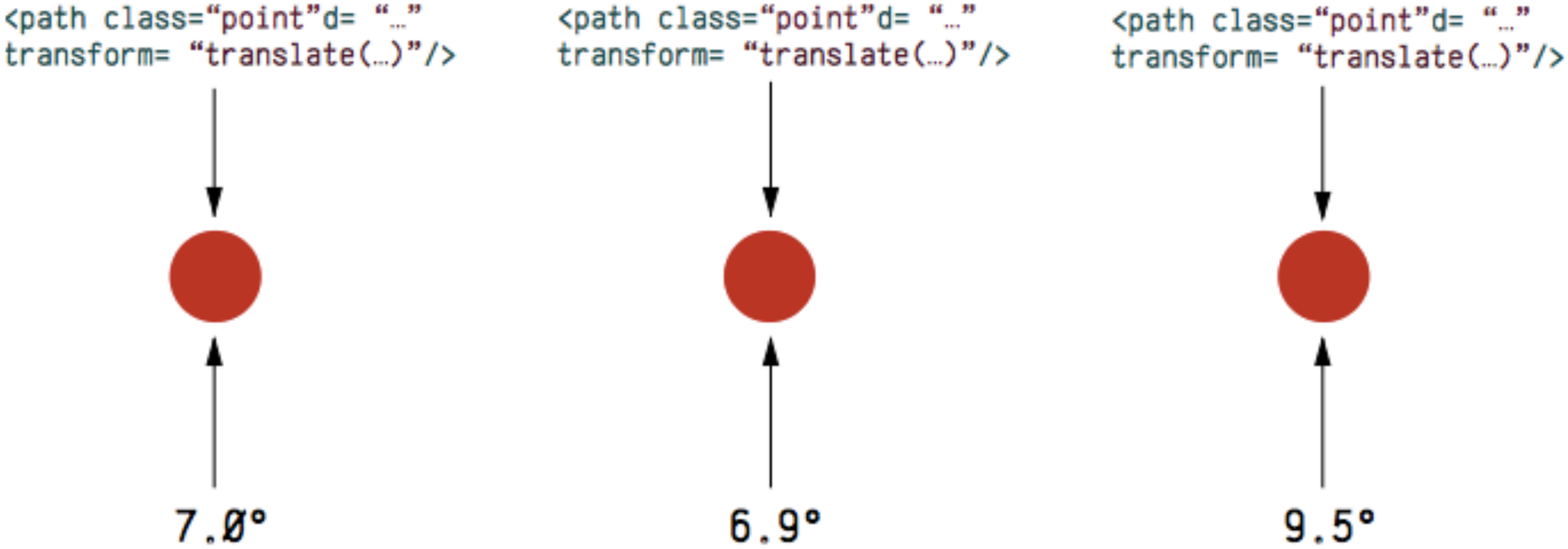
- Potential gotcha: With D3, unlike jQuery, the `append()` function returns the newly appended DOM element(s) instead of the original selection.

Graph the Data Points



```
svg.selectAll(”.point”)
      .data(dataset.data)
    .enter().append(”path”)
      .attr(”class”, ”point”)
      .attr(”fill”, d3.scale.category10(idx))
      .attr(”stroke”, d3.scale.category10(idx))
      .attr(”d”, d3.svg.symbol(idx));
      .attr(”transform”, function(d) {
        return ”translate(” + x(d.date) +
                                ”,” + y(d.temp) + ”)”;
      });
```

Associate DOM and Data



```
<path class="point"d= "…"
transform= "translate(…)"/>
```

```
<path class="point"d= "…"
transform= "translate(…)"/>
```

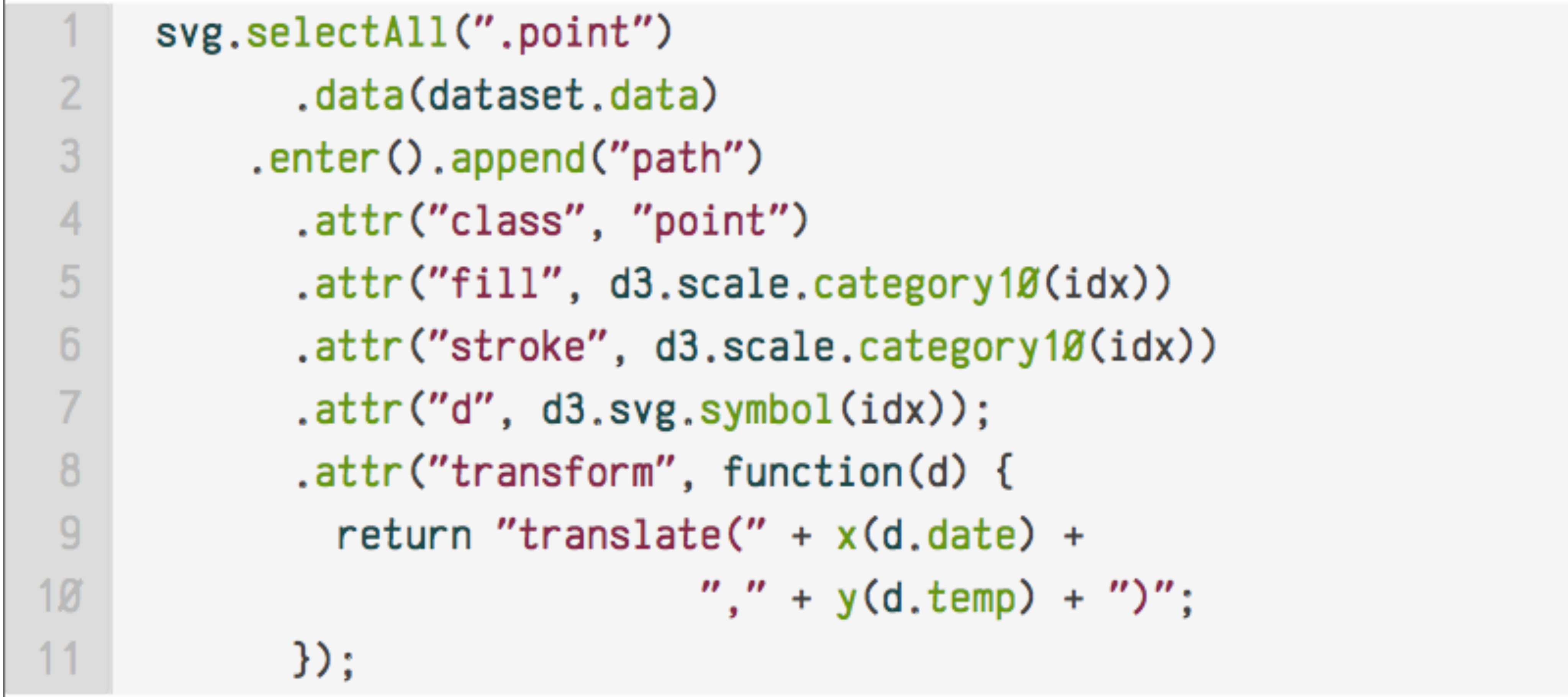
```
<path class="point"d= "…"
transform= "translate(…)"/>
```

7.0°

6.9°

9.5°

Graph the Data Points (cont’d)



```
svg.selectAll(”.point”)
      .data(dataset.data)
    .enter().append(”path”)
      .attr(”class”, ”point”)
      .attr(”fill”, d3.scale.category10(idx))
      .attr(”stroke”, d3.scale.category10(idx))
      .attr(”d”, d3.svg.symbol(idx));
      .attr(”transform”, function(d) {
        return ”translate(” + x(d.date) +
                        ”,” + y(d.temp) + ”)”;
      });
```

Add the Connecting Lines

```
svg.append(”path”)
    .datum(dataset.data)
    .attr(”fill”, ”none”)
    .attr(”stroke”, color(i))
    .attr(”stroke-width”, ”3”)
    .attr(”d”,
        d3.svg.line()
            .x(function(d) { return x(d.date); })
            .y(function(d) { return y(d.temp); })
    );
```

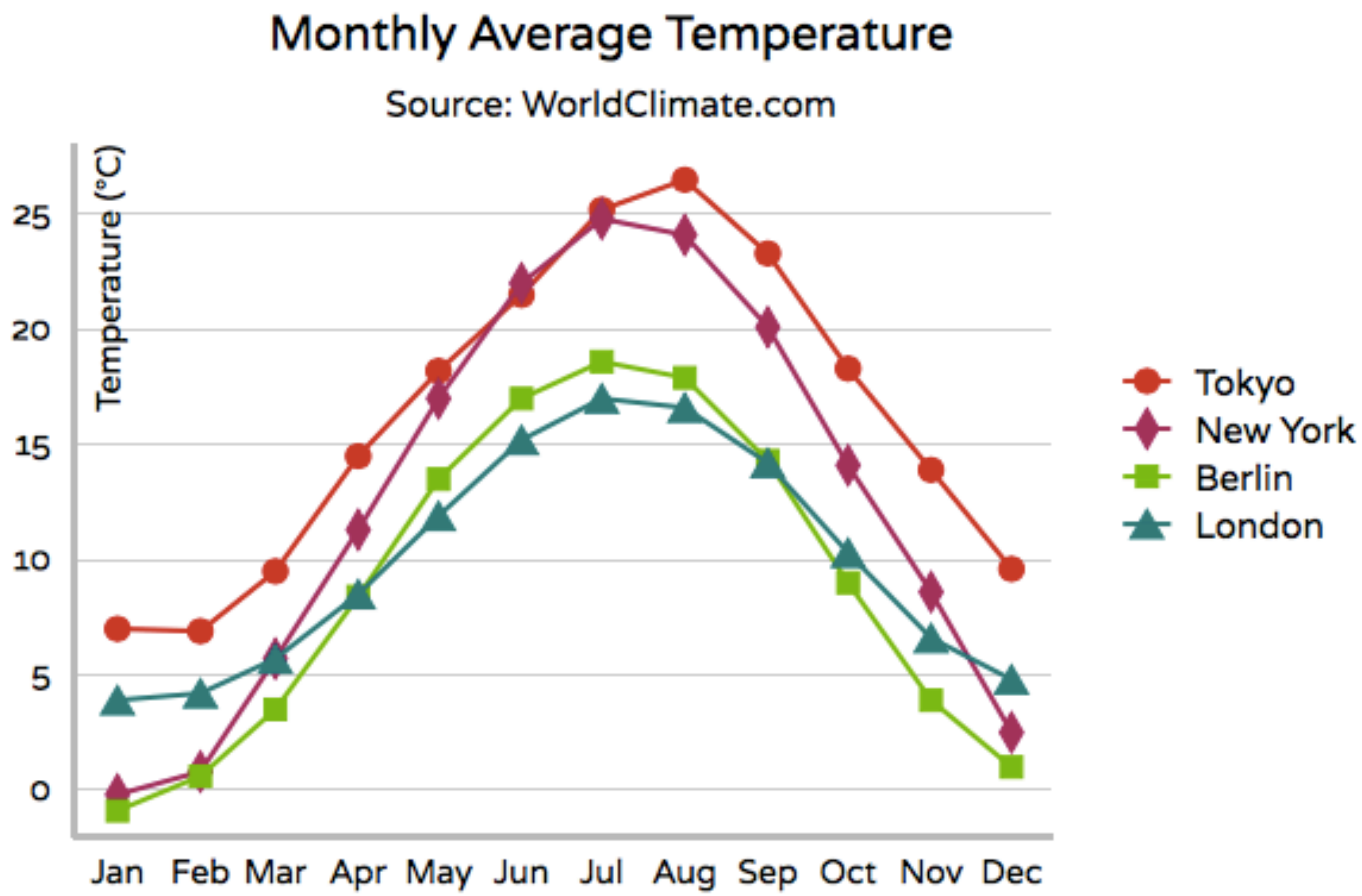
- `datum()`: maps the entire dataset array to a single element
- `d3.svg.line()`: SVG `d` attribute for the path

Add the Axes

```
svg.append(”g”)
    .attr(”transform”, ”translate(0,” + height + ”)”)
    .call(d3.svg.axis()
              .scale(x)
              .tickFormat(d3.time.format(”%b”))
              .orient(”bottom”));
```

- `d3.svg.axis()` constructs a complete SVG axis, including tick marks, grid lines, labels, etc.
- The `scale()` method defines the values for the axis

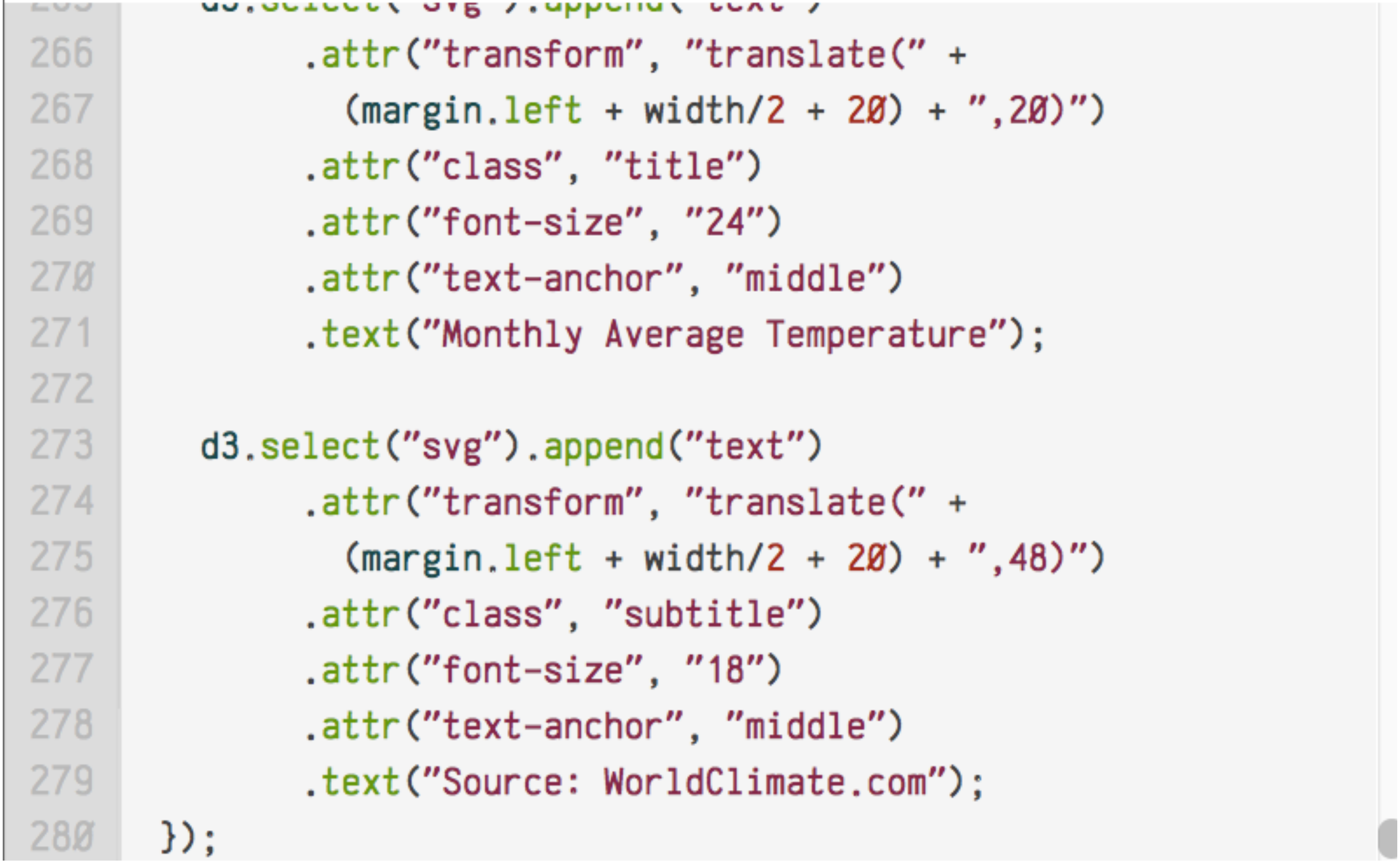
The D3 Version



Monthly Average Temperature

Source: WorldClimate.com

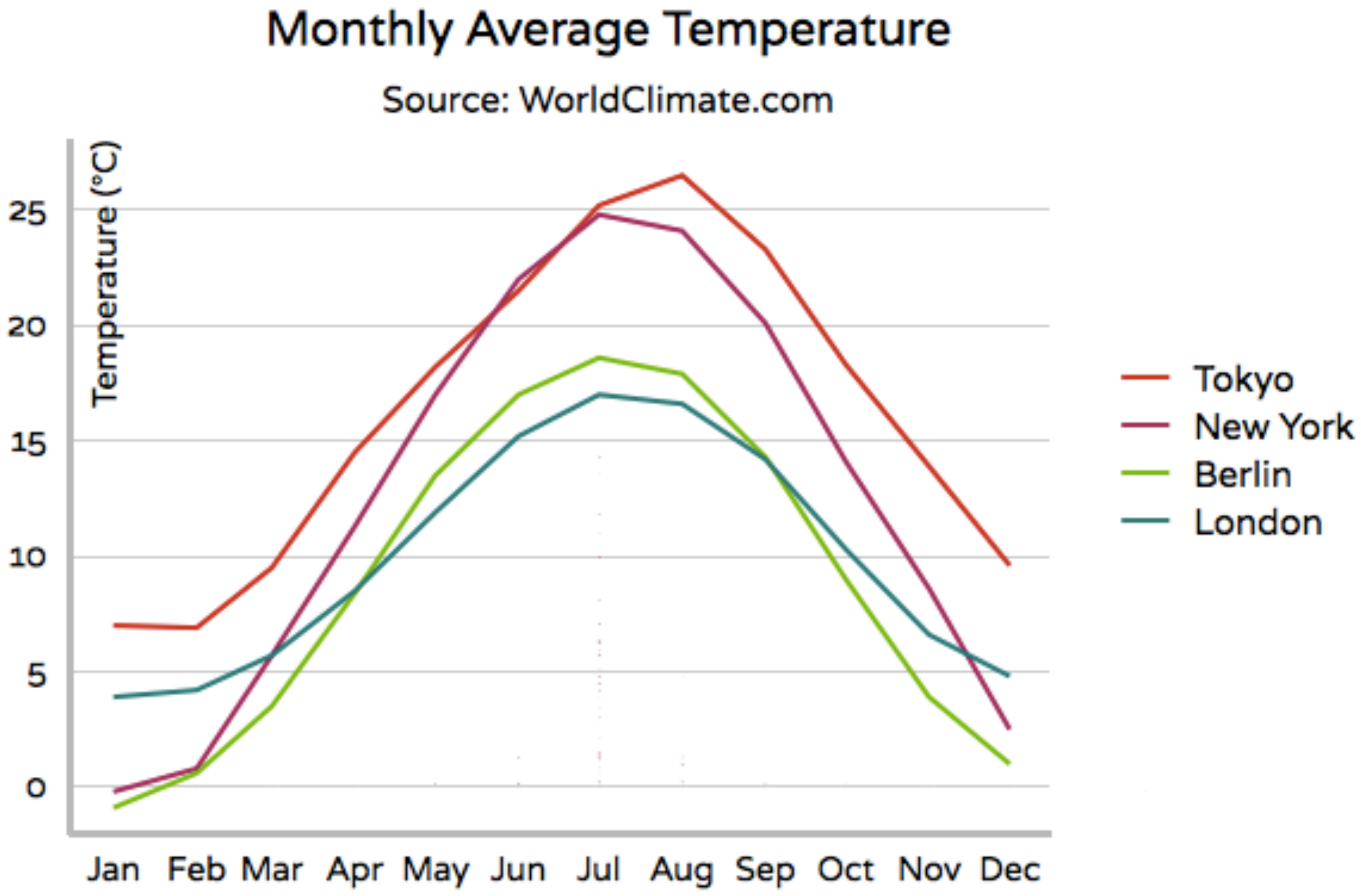
That’s a Lot of Code



```
  d3.select("svg").append("text")
       .attr("transform", "translate(" +
         (margin.left + width/2 + 20) + ",20)")
       .attr("class", "title")
       .attr("font-size", "24")
       .attr("text-anchor", "middle")
       .text("Monthly Average Temperature");

  d3.select("svg").append("text")
       .attr("transform", "translate(" +
         (margin.left + width/2 + 20) + ",48)")
       .attr("class", "subtitle")
       .attr("font-size", "18")
       .attr("text-anchor", "middle")
       .text("Source: WorldClimate.com");
});
```

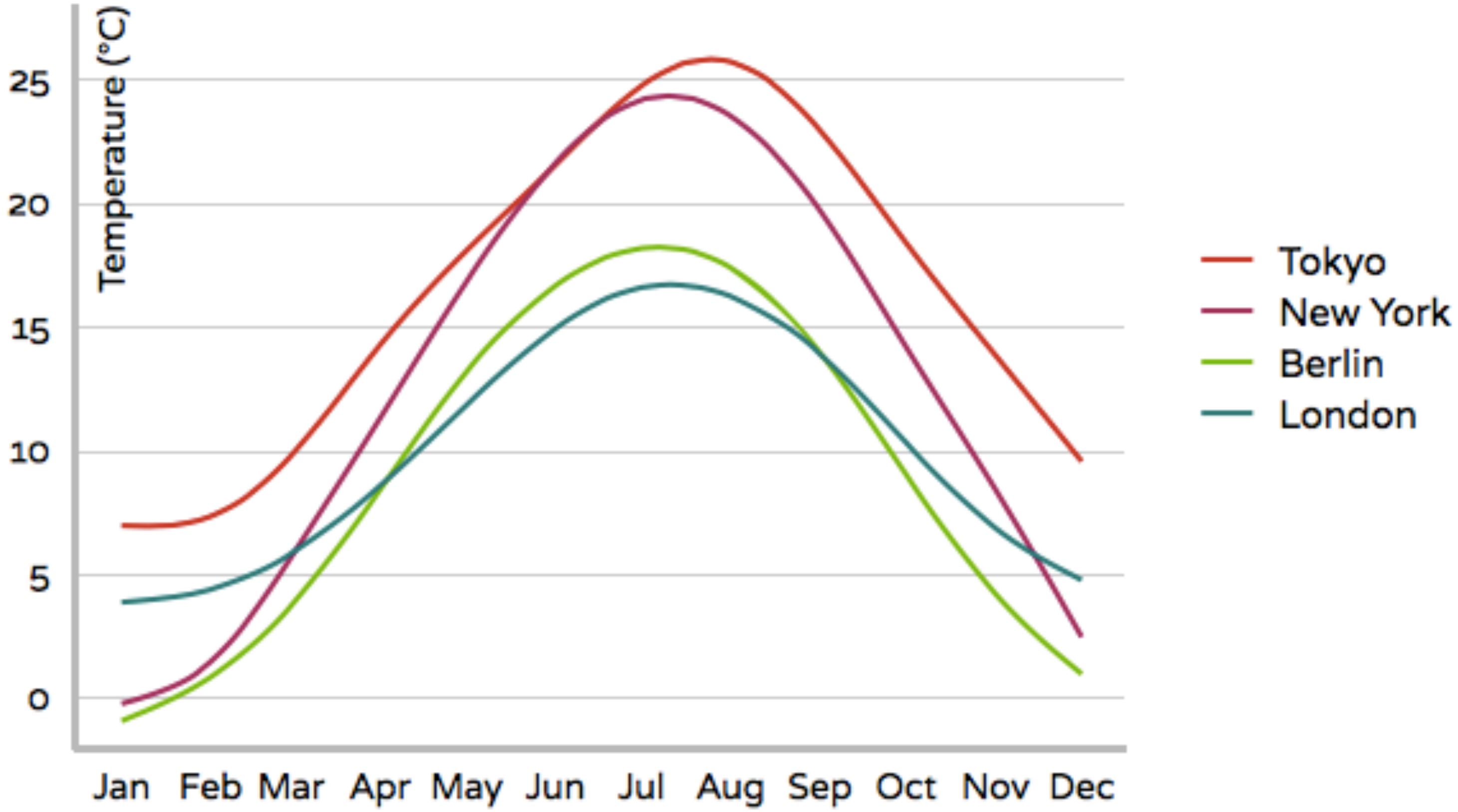
But with D3 We Can Do More



(That Animation was D3)

```
d3.selectAll(”.point”)
  .transition()
  .duration(2000)
  .ease(”bounce”)
  .attr(”transform”, function(d) {
    return ”translate(” + x(d.date) + ”,” +
      (height - margin.top - margin.bottom - 10) + ”)”;
  })
  .remove();
```

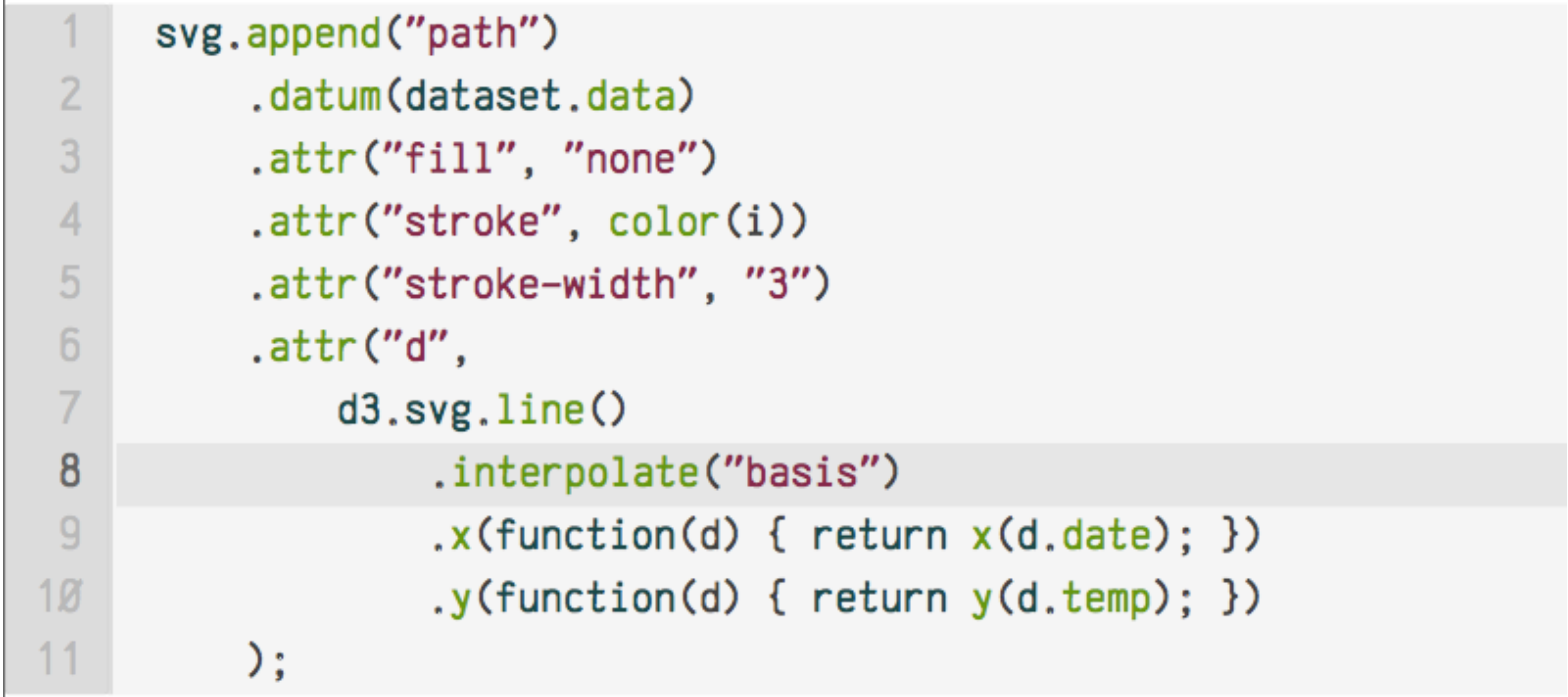
But with D3 We Can Do More



Monthly Average Temperature

Source: WorldClimate.com

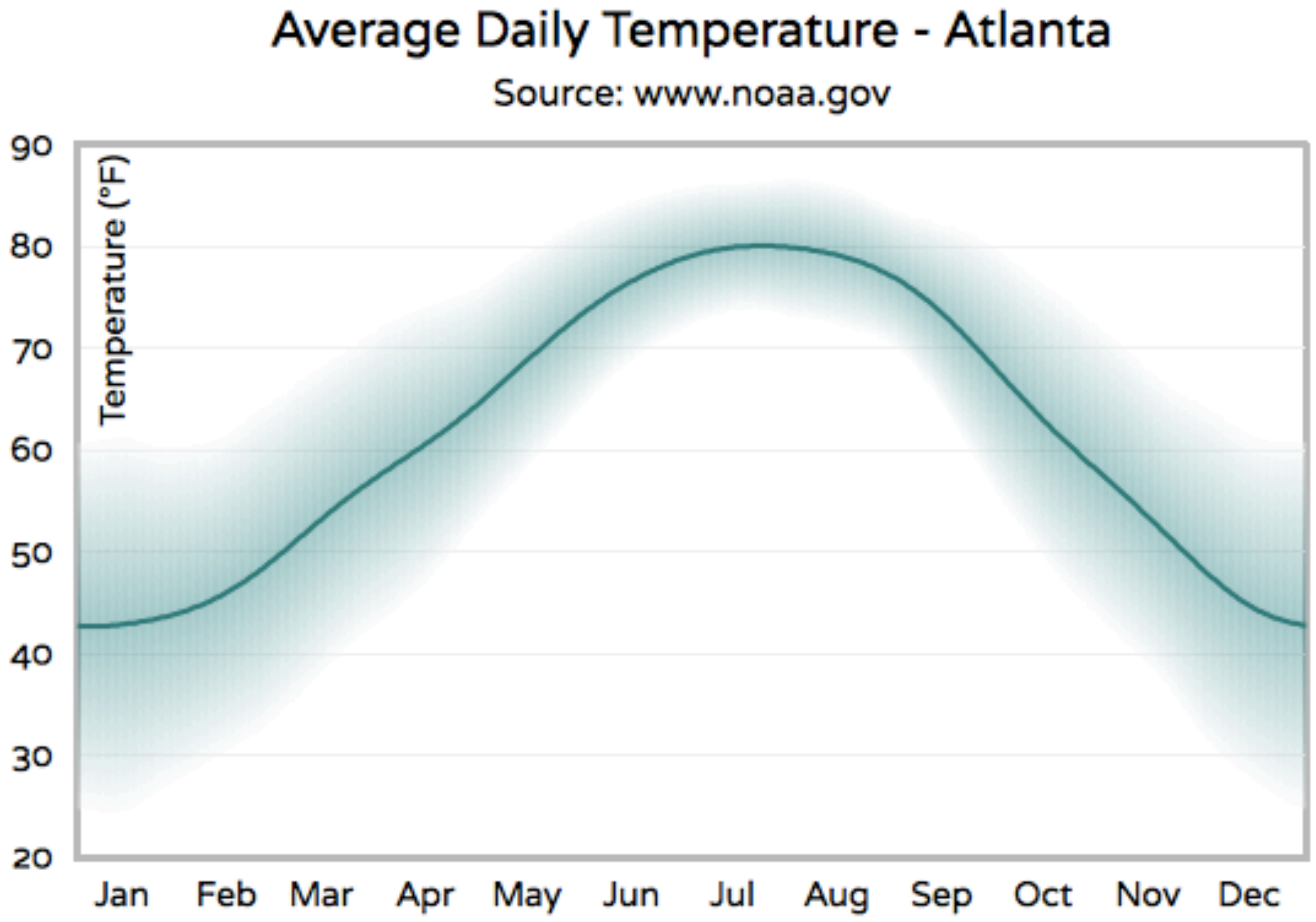
Small Addition to the Line Function



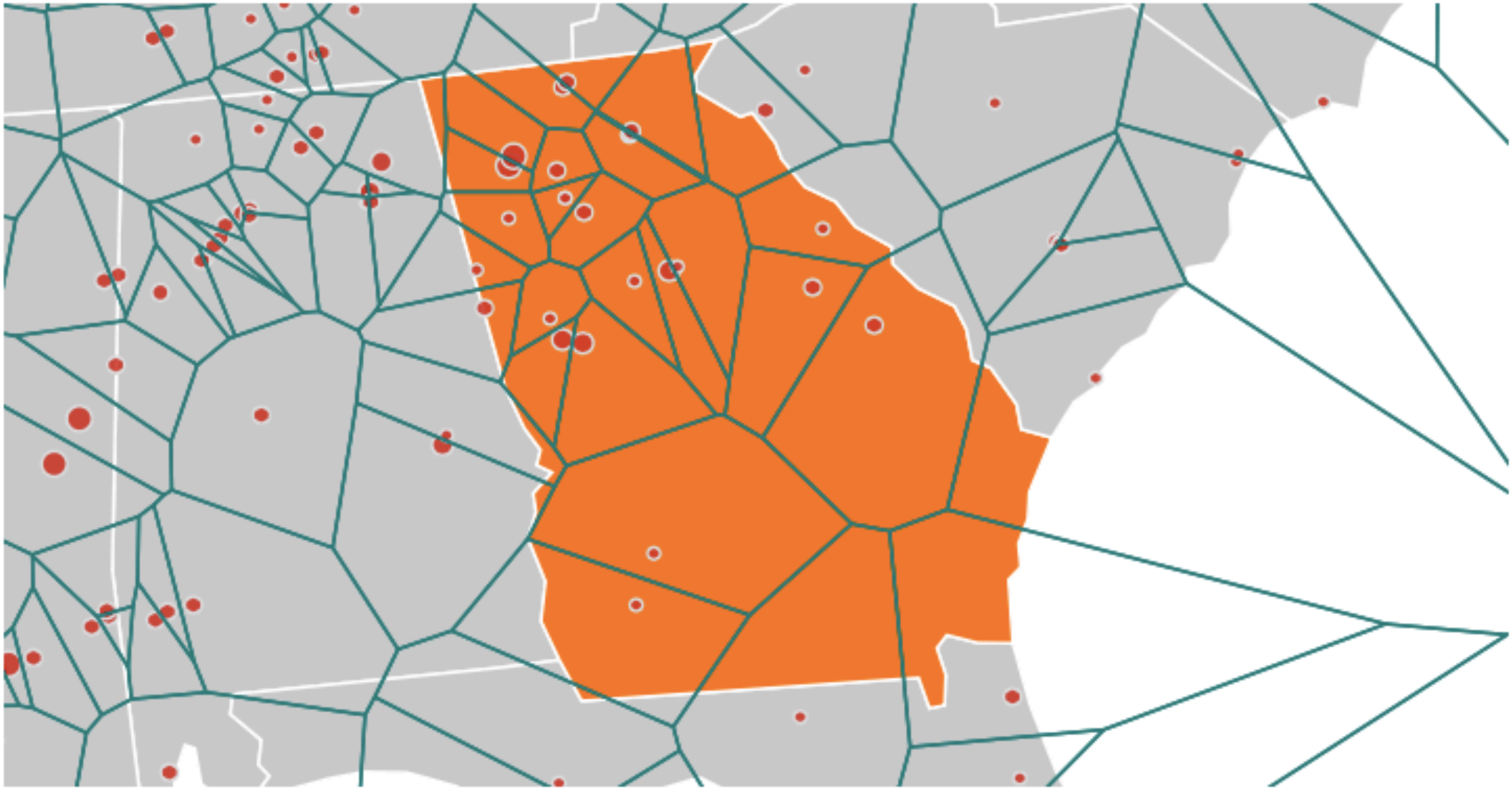
```
svg.append(”path”)
    .datum(dataset.data)
    .attr(”fill”, ”none”)
    .attr(”stroke”, color(i))
    .attr(”stroke-width”, ”3”)
    .attr(”d”,
        d3.svg.line()
            .interpolate(”basis”)
            .x(function(d) { return x(d.date); })
            .y(function(d) { return y(d.temp); })
    );
```

- D3 has many interpolations: linear, step, b-spline, Cardinal spline, cubic, …

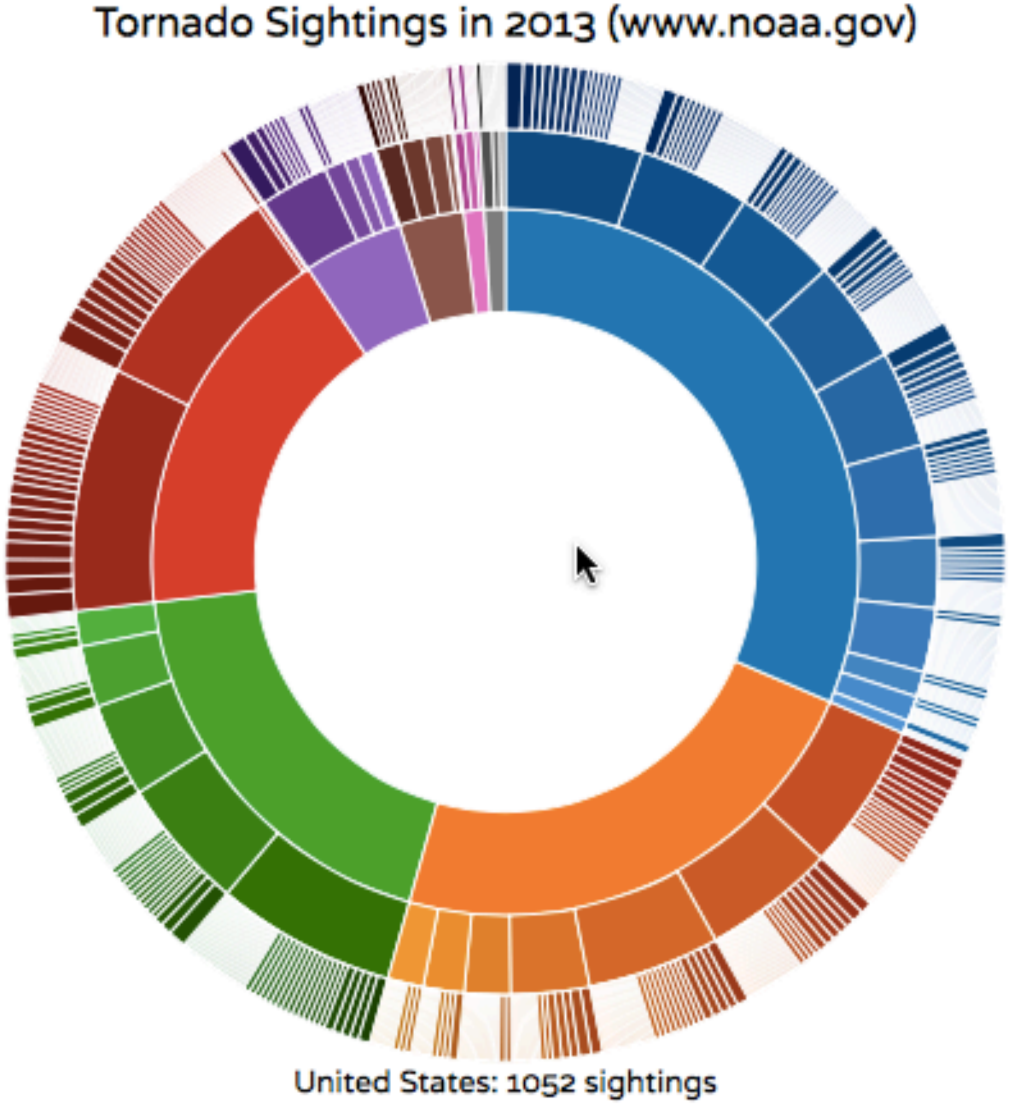
Why Bother with Monthly Values?



Maps are Useful Also



Conventions are not Constraints



More Information

- My personal web site http://jsDataV.is
- All examples from this presentation http://bl.ocks.org/sathomas
- Book from No Starch Press http://www.nostarch.com/datavisualization