1. After opening a HTML template file, in the head tag are written these files which helps us do important work.

**<script src="http://d3js.org/d3.v3.min.js"></script>**

**<script src="http://d3js.org/topojson.v1.js"></script>**

**<script src="http://d3js.org/queue.v1.min.js"></script>**

The first will let us map data into svg graphics (d3.js),

the second helps us interpret the instructions in india\_state\_2014\_simplified.topojson as an image in the browser (topojson),

And the third lets us work with multiple input files for a javascript script (queue).

2. Next I established the CSS portion.

**.state {**

**fill: #ffffff;**

**stroke: #bdbdbd;**

**stroke-width: 0.5px;}**

“State” refers to lines drawn as instructed by our topojson file (india\_state\_2014\_simplified.topojson).

**body {**

**font-family: Arial, sans-serif;}**

“Body” helps define text included in the area of the html page defined as “body”.

**div.tooltip {**

**position: absolute;**

**text-align: center;**

**padding: 0.5em;**

**font-size: 10px;**

**color: #222;**

**background: #FFF;**

**border-radius: 2px;**

**pointer-events: none;**

**box-shadow: 0px 0px 2px 0px #a6a6a6;**

**text-shadow: #f5f5f5 0 1px 0;}**

**.legend .legend-title {**

**text-align: left;**

**margin-bottom: 5px;**

**font-weight: bold;**

**font-size: 90%;}**

**.legend .legend-scale ul {**

**margin: 0;**

**margin-bottom: 5px;**

**padding: 0;**

**float: left;**

**list-style: none;}**

Notice that .legend and .tooltip refer to objects we’ll designate with our javascript, but we can still set what they’ll look like here in the CSS.

3. The next section is the body of the page, in which javascript will be embedded.

Some “var=” are used, which is setting up the variables for the code.

Margins, Axis, x-y coordinates are set.

**var color\_domain = ['0','10','20','30'];**

**var legend\_labels = ['<10','10-20','>20'];**

**var color = d3.scale.threshold()**

**.domain(color\_domain)**

**.range(["#000000","#ffffff", "#09C34D","#D4110C"]);**

Colors are coded by RGB HEX value.

**var svg = d3.select("body").append("svg")**

**.attr("width", width + margin.left + margin.right)**

**.attr("height", height + margin.top + margin.bottom)**

**.append("g")**

**.attr("transform",**

**"translate(" + margin.left + "," + margin.top + ")");**

The “svg” variable is a designation for  joining of a to-be-specified svg graphic with the body of the html page. D3 will let us map data from our files onto this svg designation.

**var projection = d3.geo.mercator()**

**.center([78, 27])**

**.scale(1200);**

**var path = d3.geo.path()**

**.projection(projection);**

Also note that the “path” variable is calling on a capability of D3 to draw lines based on geospatial information fed to it by projection variable.

**4.**

**queue()**

**.defer(d3.json, "india\_state\_2014\_simplified.topojson")**

**.defer(d3.csv, "air\_quality\_2011\_states.csv") //sample data**

**.await(ready);**

Queue() prepares files to be read by D3.

**5.**

**var pairResultWithId = {};**

**var pairNameWithId = {};**

**var pairNoWithId = {};**

**data.forEach(function(d) {**

**pairResultWithId[d.ST\_NAME] = d.annavgSO2;**

**pairNameWithId[d.ST\_NAME] = d.State;**

**pairNoWithId[d.ST\_NAME] = +d.annavgNO2; });**

This is where data is set to be visualized.

d.annavgSO2 and d.State refer to the column headers of our .csv.  There’s a “d” before them because it’s a default way of referencing data you’ve mapped into D3 with javascript.

**6.**

**var state\_geojson = topojson.feature(state, state.objects.india\_state\_2014);**

**svg.selectAll(".state")**

**.data(state\_geojson.features)**

**.enter().append("path")**

**.attr("class", "state")**

**.attr("d", path)**

**.style ( "fill" , function (d) {**

**var result = pairResultWithId[d.properties.ST\_NAME];**

**if (result!='') {return color(result);} })**

And now I select the svg objects I’ve created but not specified, and map our data onto them.

This will draw each state as an object, each with its own values. Also, the “.data” line associates information from our **india\_state\_2014\_simplified.topojson** file with the state objects (the stuff in parentheses refers to the way the topojson hierarchizes information and points the script to the right container in the hierarchy).

Also important is that “color” refers to the function set above in the code. “Color” expects a number as input, but instead of a specific number, I give it container filled with pairs of ID numbers and annavgSO2 values.

**7.**

**.style("opacity", 0.8)**

**.on("mouseover", function(d) {**

**d3.select(this).transition().duration(300).style("opacity", 1);**

**div.transition().duration(300)**

**.style("opacity", 1)**

**div.text(pairNameWithId[d.properties.ST\_NAME ] + " : " + pairResultWithId[d.properties.ST\_NAME])**

**.style("left", (d3.event.pageX) + "px")**

**.style("top", (d3.event.pageY -30) + "px");**

**x.domain(state\_geojson.features.map(function(d) { return pairNameWithId[d.properties.ST\_NAME]; }));**

**y.domain([0, d3.max(state\_geojson.features, function(d) { return pairNoWithId[d.properties.ST\_NAME]; })]);**

**})**

The “div.text” will behave according to our “div.tooltip” CSS style that was established at the top.  The duration of the transition (which in this case transitions from less to more opacity, creating a highlight effect) is listed in milliseconds.

**8.**

**var legend = d3.select('.legend-scale')**

**.append('ul')**

**.attr('class', 'legend-labels');**

**var keys = legend.selectAll('li')**

**.data(color\_domain);**

**keys.enter().append('li')**

**.text(function(d, i){ return legend\_labels[i];})**

**.append('span')**

**.style('background', function(d) { return color(d); });**

Last, the html that explains what each color means.