16.2 Lesson Summary - Bar and Line Charts with D3

D3 allows users to generate visual representations of data by supporting a number of functions that compliment Scalable Vector Graphics (SVG).

Concept: D3 provides the functionality to **parse CSV** files into JavaScript objects. If your CSV file had customer data with first and last names you could list each of their names using the following code:

*d3.csv("./customer-data.csv").then(function(customerData) {*

*customerData.forEach(function(customer) {*

*console.log("Name:", customer.firstName, customer.lastName);*

*});*

*}).catch(function(error) {*

*console.log(error);*

*});*

* Activity: 02-Ins\_Loading\_Data, 03-Par\_BarChart\_From\_CSV

Concept: To use D3 to find the **maximum** value in an array you can use the following code:

*d3.min(myArray);*

To use D3 to find the **minimum** value in an array you can use the following code:

*d3.min(myArray);*

To use D3 to find the **minimum** and **maximum** values in an array you can use the following code:

*d3.min(myArray);*

* Activity: 04-Evr\_Scales

Concept: It's often necessary to **scale** your data in order to accurately and clearly create a visual representation. Some data visualization tools will automatically scale your data but D3 requires you specify how to scale your data. To scale your data, you must create a scaling function. To create a scaling function, you must pass the input bounds and output bounds into a D3 scale function. This will indicate how your original data should be scaled. If your original data ranges from 0 to 10 and you want to double it in a linear manner you could use the following code:

*var myScale = d3.scaleLinear()*

*.domain([0, 10])*

*.range([0, 20]);*

The ***scaleBand*** fuction allows you to convert categorical data into numerical data. To convert a number of colors to numbers displayed in a chart you could use the following code:

*var myScale = d3.scaleBand()*

*.domain(["Blue", "Red", "Green", "Yellow"])*

*.range([0, 1000]);*

* Activity: 04-Evr\_Scales

Concept: Once you have created the necessary scales to display your data you can integrate them with your SVG chart data like in the following code:

*chartGroup.selectAll(".bar")*

*.data(dataArray)*

*.enter()*

*.append("rect")*

*.classed("bar", true)*

*.attr("x", (d, i) => xScale(dataCategories[i]))*

*.attr("y", d => yScale(d))*

*.attr("width", xScale.bandwidth())*

*.attr("height", d => chartHeight - yScale(d));*

* Activity: 05-Ins\_Intro\_To\_Axes, 06-Stu\_Complete\_Bar\_Chart

Concept: D3 provides functionality to generate **SVG lines** from coordinates. For example:

*var coordinates = [*

*[100, 120],*

*[200, 250],*

*[300, 160]*

*];*

*var lineGenerator = d3.line();*

*var svg = d3.select("#my-path");*

*svg.append("path")*

*.attr("fill", "none")*

*.attr("stroke", "blue")*

*.attr("stroke-width", 5)*

*.attr("d", lineGenerator(coordinates));*

* Activity: 07-Ins\_Line\_Generators\_Intro, 08-Stu\_Generating\_Lines

Concept: You can add **lines** to an **SVG** visualization by combining D3's HTML element manipulation functionality to generate a line drawing function and then pass in the data for the function to draw. For example:

*var drawLine = d3*

*.line()*

*.x(data => data.x)*

*.y(data => data.y);*

*chartGroup.append("path")*

*.attr("d", drawLine(myData))*

*.classed("line", true);*

* Activity: 09-Ins\_LineChart, 10-Stu\_LineChart