16.1 Lesson Summary - Intro to D3 Graphing

The D3 JavaScript library has a number of functions that allow you to manipulate the contents of your webpage. D3 can create elements according to data sets. Scalable Vector Graphics (SVG) provide a way of creating shapes and images on your webpage according to parameters associated with each of the shapes. You can combine the functionality of D3 and SVG to produce dynamic charts for your webpage.

Concept: The D3 JavaScript library allows you to select and manipulate the HTML elements on your webpage. Once you have selected a group of elements you can run code on each element by **chaining** the ***each*** method to the *selectAll* method. You could reference the data and index of each element as parameters in your code as follows:

*d3.select("ul").selectAll("li")*

*.each(function(element\_data, element\_index) {*

*console.log("element", this);*

*console.log("data", element\_data);*

*console.log("index", element\_index);*

*});*

We can bind data to our DOM elements using the ***data*** method. For example:

*d3.select("ul").selectAll("li")*

*.data([12, 34, 22])*

You can set the text of your DOM elements equal to the data you have specified using the ***text*** method. For example:

*d3.select("ul").selectAll("li")*

*.data([12, 34, 22])*

*.text(function(d) {*

*return d;*

*});*

You can add new elements inside existing elements using the ***append*** method. For example:

*d3.select("ul").selectAll("li")*

*.append("p");*

If you want to add new elements based on a set of data you have defined you must utilize the ***enter*** method and apply an append after it. For example:

*d3.select("ul").selectAll("li")*

*.data([12, 34, 22])*

*.enter()*

*.append("li")*

*.text(function(d) {*

*return d;*

*});*

If you want to remove items based on their data you can use the ***exit*** and ***remove*** methods. For example:

*d3.select("ul").selectAll("li")*

*.data([12, 34, 22])*

*.exit()*

*.remove();*

* Activity: 01-Evr\_Binding\_Data

Concept: You can use more complex data objects to add data to your webpage. For example, if you have an array of customer objects you wanted to list by first name you could use the following code:

*var customerData = [{*

*firstName: "John",*

*lastName: "Doe"*

*},*

*{*

*firstName: "Jane",*

*lastName: "Doe"*

*}];*

*d3.select("ul").selectAll("li")*

*.data(customerData)*

*.enter()*

*.append("li")*

*.text(function(d) {*

*return d.firstName;*

*});*

* Activity: 02-Ins\_Complex\_Data, 03-Stu\_D3\_Table

Concept: Styles can be applied to DOM elements using the ***style*** method. For example:

*d3.selectAll("p")*

*.data(["red", "green", "blue"])*

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

*return d;*

*});*

You can control the class of an element using D3's ***classed*** method. For example:

*d3.selectAll("p")*

*.data(["red", "green", "blue"])*

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

Using ***merge*** after appending new elements changes the selection object to include both new and old elements. For example:

*var austinTemps = [76, 59, 59, 73, 71];*

*var selection = d3.select("#content").selectAll(".temps")*

*.data(austinTemps);*

*selection.enter()*

*.append("div")*

*.classed("temps", true)*

*.merge(selection)*

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

*return d + "px";*

*});*

* Activity: 04-Evr\_Enter\_Exit\_Update

Concept: There are a number of different options for displaying graphics in a webpage. Scalable Vector Graphics (**SVG**) operates by using text to describe the shapes used to create your graphical objects. Creating graphics in this manner has the advantage of being able to be scaled to fit different size screens and can require less data than pixel based images if the shapes are not too complex. This approach is not ideal for photographic images but is well suited for logos and vector art. To add a simple rectangle to your webpage you can create an SVG element using inline HTML. For example:

<svg width="100" height="150">

<rect x="0" y="0" width="50" height="50" fill="green" />

</svg>

* Activity: 05-Ins\_SVG, 06-Stu\_SVG\_Stickman

Concept: SVG elements can be created and edited using D3 just like any other HTML elements. To add a rectangle to your code using D3 you could use the following code:

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

*svg.attr("width", "100px").attr("height", "100px");*

*var rectangles = svg.selectAll("rectangle");*

*var dimensions = [{width: 50, height: 100}];*

*rectangles.data(dimensions)*

*.enter()*

*.append("rectangle")*

*.attr("x", 0)*

*.attr("y", 0)*

*.attr("width", function(d) {*

*return d.width;*

*})*

*.attr("height", function(d) {*

*return d.height;*

*})*

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

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

*.attr("fill", "red");*

* Activity: 07-Evr\_D3\_Bullseye, 08-Stu\_Data\_Rectangles

Concept: You can combine D3's ability to update HTML elements according to a data set with SVG's ability to create shape-based images to generate charts on your webpage.

* Activity: 09-Stu\_UpsideDownBarChart, 10-Evr\_Bar\_Chart\_Refactored