D3.js presentation at Code Camp 10/11/2014 – janmilosh.com

Demo 1, Getting started

- Include d3.js file before your graph script.
- Create graph svg – here added directly with inline attributes and style.

Demo 2, Select, Append, and Chained Operators

- Operators can be used to get or set items such as attributes, properties, styles, text, and html.
- Operators can be chained (like jQuery). Each returns a selection, which the next operator in the chain acts on.

`select()` – selects the first element that matches the selector string and returns a single element selection.
`append()` – appends a new element with the current specified name as the last child of the current selection, returns a new selection containing the appended elements.

Demo 3, Binding Data to the SVG

- D3.js can load data from a variety of external resources (CSV, TSV, JSON, XML).
- Ultimately the data needs to be an array (can be array of JSON objects).
- We're supplying an array of objects for our dataset.

`selectAll()` – selects what we want, even if it's not there. Selects all elements that match the specified selector. If no elements match the selector, returns an empty selection.
`selection.data(values[,key])` – joins specified array of data with the current selection. The specified values is an array of data values (e.g. numbers or objects), or a function that returns an array of values. If no key is specified, the first datum in values is assigned to the first element in the current selection, the second datum to the second element, etc. The data is sticky and available on re-selection. The result is the update selection which represents the selected DOM elements that were successfully bound to the specified data elements. The update selection also contains a reference to the enter and exit selections…
`selection.enter()` – returns enter selection, placeholder nodes for which no corresponding existing DOM element was found. When there are fewer DOM nodes than data points…keeps the correspondence between data and DOM nodes. There is also an **`exit()`** function to collect DOM elements for removal if there are too many for the data.

Demo 4, Scaling our Data

- We're using a linear scale for both axes.
- Need to specify a domain and range.
- Domain refers to the data.
- Range refers to the SVG space.
- Origin of SVG is upper left.
- Origin of our graph will be lower left.
- Data attributes for the circles can be a function of some property of the data. The syntax for this is an anonymous function that operates like a forEach on the data array. The data is passed into the scale. The function can have two arguments, data and index.

`d3.scale.linear()` - Constructs a new linear scale with the default domain [0,1] and the default range [0,1]. Thus, the default linear scale is equivalent to the identity function for numbers; for example linear(0.5) returns 0.5.
`linear.domain([numbers])` – sets the scale's input domain to the specified array of two or more numbers

`linear.range([values])` – sets the scale's output range to the specified array of values. Must match the cardinality of the input domain.

Demo 5, Make data scaling more dynamic

- Calculate the max and min so that we can use unknown data and have it still fit.
- Use the max and min values for the domain.
- Use the anonymous function syntax for looping through the values to find the max or min.

`d3.max()` – returns the maximum value in a given array using natural order. It ignores undefined values.

`d3.min()` – like d3.max, but returns the minimum.

Demo 6, Add Labels to the Data Points

- Uses the SVG text element. It's added after the circles so it's on top.
- We've also added an offset.
- The text is appended to the SVG in the same manner as the circles.
- Notice that we set variables to break up the chains.

Demo 7, Add Some Margin

- We have defined our margins with an object.
- SVG height and width are defined, then height and width are defined as the SVG size minus the margins so that the height and width variables correspond to the actual graph area (rather than the outer SVG that contains it).
- Group elements must be positioned with transform.
- The g element is appended to the SVG, then the circles are appended to the g element.

`append('g')` – appends a group element to the SVG.

Demo 8, Add Axes

The axis component is designed to work with D3's scales (quantitative, time, and ordinal).
Notice that we append a group for the axis and we transform the group to move the axis.

`d3.svg.axis()` – Creates a new default axis.

`scale()` – sets the scale and returns the axis

`orient()` – determins whether horizontal or vertical and which way the ticks go ('top', 'bottom', 'left', 'right').

`ticks()` – for a linear scale, can specify the number of ticks as the argument.

`call()` – Invokes the function once, passing the function with any optional arguments.

Demo9, Add Title and Axis Labels

These are simple text elements