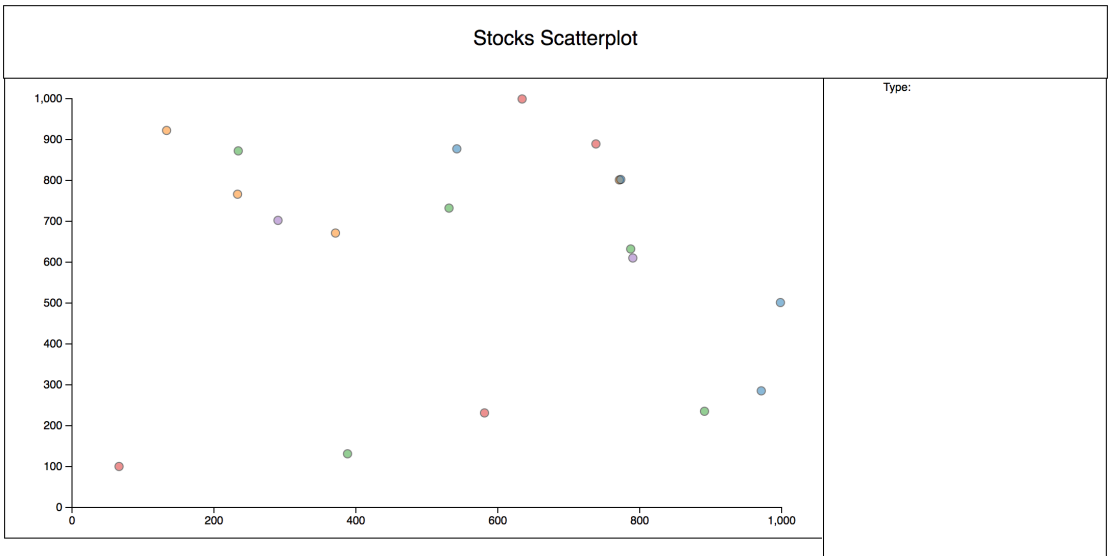
D3 Lab 1

Please name your file with your first name (e.g., yeaseul.html and yeaseul.js) and submit them to the discussion board.

In this lab we'll work with a basic scatterplot of stock data (written in d3 v4 for this lab. The first step will review skills you may have already mastered: implementing details on demand via tooltips that appear on hover interactions. **If you feel comfortable with this step, you may wish to jump ahead to steps 2, 3, and 4. Your files don't need to include step 1 if you skipped it.**

When you first load step1.html, you should see the following:



Step 1)

It would be nice to see the stock name and all variable values for each data point in the chart on demand. **Implement labels on mouse-hover which disappear on mouse-out**.

Do not change the variable names. If you need help remembering the different ways to add labels, look at the in-class slides on adding labels.

Step 2)

One of the variables associated with each stock is called "type" and describes the sector that the stock falls in (Tech, etc.). Implement filtering by type using a dropdown. Note that I created a div called "layout", a div called "title", and a div called "filters" where you can put the dropdown and any other interactive inputs. Also note that there is already a label and text input element for displaying whatever type has been selected with the dropdown in step1.html. In your dropdown handler function, you'll want to set the value of this text input whenever the dropdown selection changes.

In step1.js, you will find a function stub called "filterType()." You can add the filtering code here. Make sure your dropdown makes it possible for the user to again view all the data even after they have filtered to only one type.

Step 3)

One of the quantitative data variables in the csv (vol, which gives the volume sold) is not visualized. We can implement a slider to allow users to filter on values of this variable.

Jquery offers a two-handled slider range (here assuming a baseball visualization in which the variable 'assists' can be filtered using the slider):

```
<p>Assists</p>
        <div id="assists" class="slider-range"></div>
        <label for="assistamount">Assists range:</label>
        <input type="text" id="assistamount" readonly style="border:0; color:#f6931f; font-
weight:bold;">
```

We can define a handler function to handle the presentation of the selected values and the manipulation of the visualization:

```
$(function() {
   $( "#assists" ).slider({
     range: true,
     min:  0,
     max: maxAssists,
     values: [ 0, maxAssists ],
     slide: function( event, ui ) {
       $( "#assistamount" ).val( ui.values[ 0 ] + " - " + ui.values[ 1 ] );
       filterAssists(ui.values);
      }  //end slide function
     }); //end slider
   $( "#assistamount" ).val( $( "#assists" ).slider( "values", 0 ) +
     " - " + $( "#assists" ).slider( "values", 1 ) );
}); //end function
```

Note that I have already loaded jquery (and associated css) for you at the top of the body tag:

```
 <link rel="stylesheet" href="http://code.jquery.com/ui/1.11.1/themes/smoothness/jquery-
ui.css">
  <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
```

<script src="http://code.jquery.com/ui/1.11.1/jquery-ui.js"></script>

To implement your slider for vol, first copy and paste the slider range and handler code into your code. Put the definition of the slider range in your html, and the handler in the js. Then, create a filter function that filters the data based on the selected range and then updates the visualization. Note that you will also need to have created a variable for the max value of the vol variable *before* you reference this variable in the handler function.

Hint: To avoid buggy code where the slider overwrites the dropdown filter and vice versa, make the filter functions associated with each interaction compatible. One way to do this is to keep a record of the current selected filter criteria from the dropdown and of the current selected filter range from the slider. Then, when a user interacts with either the dropdown or slider, you can maintain the other interaction's filter criteria.

Step 4)

There is another quantitative data variable in the csv that is also not visualized (deltaPrior, which gives the change in price over the previous day). Add an additional slider for this variable.

Hint: To avoid buggy code where the vol slider overwrites the deltaPrior filter and vice versa, make the filter functions associated with each interaction compatible (also remembering to keep track of the current value of the dropdown filter on type). For this step, you'll need to think about what information you need to keep track of for each slider in order to make sure that an interaction with one slider doesn't overwrite the current selection of the prior slider. Given that a two-handled slider controls the minimum and maximum value of the data that are plotted, you will probably want to keep track of this information for each slider, and consult each time either slider is interacted with.

Step 4)

A user may want to filter the data by more than one quantitative variable at once. Add an additional slider that filters either price or tValue. Hint: There are a couple ways you can do this. The less elegant approach is to create a separate data filtering function for each function, and just make sure that they keep track of the other slider's (and the dropdown's) selections. A more elegant approach would be to use a single filter data function to handle filtering events triggered by all sliders.

Step 5)

A user may want to see more than one type of stock at once, for instance to compare Tech stocks to Retail stocks. Revise your dropdown implementation by substituting a different type of HTML input that will allow selection of multiple types at once, like a set of checkboxes.