Problem SET 9

Interaction and Animation

This is a modification of tutorials made by Mike Foster, Eric Huntley, and Carlos Olascoaga

Happy Thanksgiving Here is an Easy Assignment!! Due 11:59 Wednesday the Nov. 24th

Please go through the tutorials on your own and we would like you to post the web pages you make as a result to your github page. You will be submitting those pages as the answers to this week's problem set.

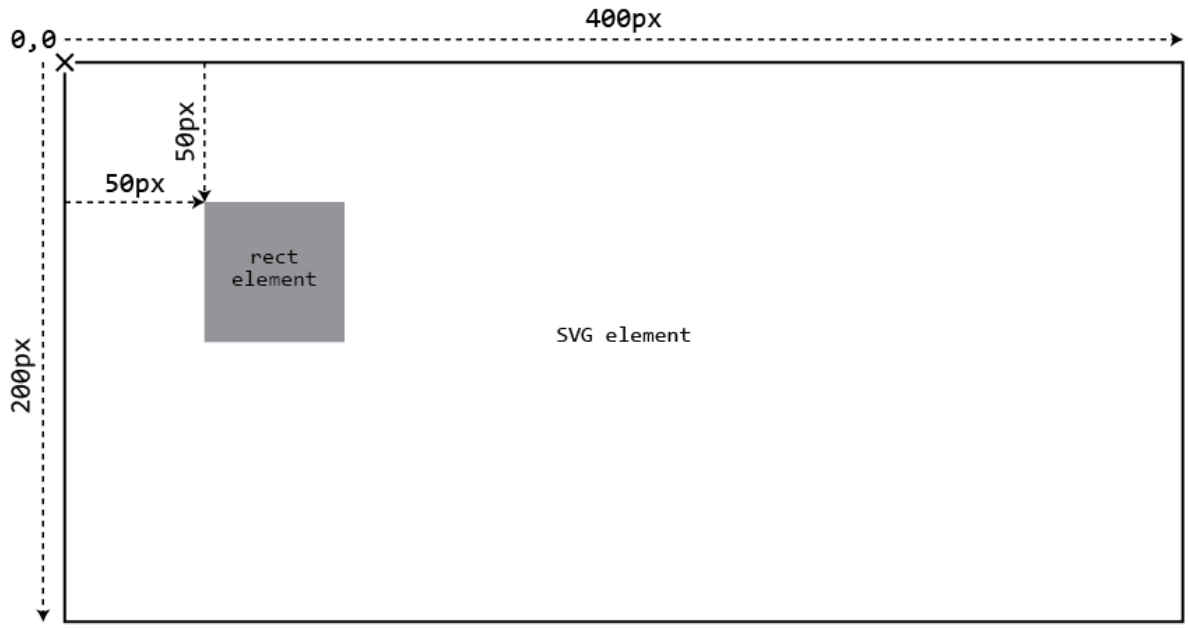
d3transitionexample.html
Interaction_scatterplot.hmtl.

Part 1: Transitions: Animation the D3 Way

The robust and real way to animate and modify elements using D3 is using the D3 Transition methods. You will be excited to learn that D3 has methods for transitions that you can activate through user input, such as clicks. These transitions can change size, color, position, and really anything that can be set in the attributes of a page element (view examples here). Let's explore this a little further.

To start, let's create a simple D3 visualization with a rectangle. To quote Jerome Cukier from his fantastic tutorial on Animations and Transitions, If you know how to draw in D3, you almost know how to animate! We use transitions to change the properties of the rectangle.

Our page layout, to give us a better idea, will look like the following.



Using Transitions

To implement a transition, you set up a trigger, be it a button or a click or some type of user event, and then within the trigger, change the respective attributes of the SVG you are seeking to modify. For example, set up a rectangle element and a button. When that button gets a click, have it change some of the attributes of the element.

Open the d3transitions.html file in your web browser and in a text editor. Add the following code to the file between the script tags.

All of our transitions will go in the "start on click" function!

```
// Width and height of the SVG object
 let w = 350;
 let h = 175;

  //Make an SVG Container
  var svgContainer = d3.select("div#main")
        .append("svg")
        .attr("width", 400)
        .attr("height", 200)
        .style("border-color", "black")
        .style("border-style", "solid")
```

```
            .style("border-width", "1px");

  // Draw the Rectangle
  var rectangle = svgContainer.append("rect")
            .attr("x", 50)
            .attr("y", 50)
            .attr("width", 50)
            .attr("height", 50);

 // Set up the reset button to move it back to 50,50
 d3.select("#reset").on("click", function() {
            rectangle
            .transition()
            .attr("x", 50)
            .attr("y", 50);
 });

 d3.select("#start").on("click", function() {
            rectangle
            .transition(); // ALL OF OUR TRANSITIONS WILL GO HERE!
 });
```

Your code should look like the following:

```
<script type="text/javascript">

    // Width and height of the SVG object
    let w = 350;
    let h = 175;

    //Make an SVG Container
    var svgContainer = d3.select("div#main")
        .append("svg")
        .attr("width", 400)
        .attr("height", 200)
        .style("border-color", "black")
        .style("border-style", "solid")
        .style("border-width", "1px");

    // Draw the Rectangle
    var rectangle = svgContainer.append("rect")
        .attr("x", 50)
        .attr("y", 50)
        .attr("width", 50)
        .attr("height", 50);

    d3.select("#reset").on("click", function() {
    rectangle
      .transition()
      .attr("x", 50)
      .attr("y", 50);
    });

     d3.select("#start").on("click", function() {
      rectangle
      .transition();
        });
```

The following examples show some basic transitions. Two buttons have been added to help showcase the example, and demonstrate how to start the transition and then reset the transition.

Position

Add the following code to your HTML to add buttons (add this above your div element):

```
<button id="start">Transition</button>
<button id="reset">Reset</button>
```

Your code should look like the following:

```
<body>
    <button id="start">Transition</button>
    <button id="reset">Reset</button>
    <div id="main">

    </div>
<script type="text/javascript">
```

Modify your start and reset selections as follows:

```
d3.select("#start").on("click", function() {
        rectangle
                .transition()
                .attr("x", 250); // New Position
  });

  d3.select("#reset").on("click", function() {
        rectangle
                .transition()
                .attr("x", 50); // Old Position
  });
```

Your code should now look like the following:

```
<script type="text/javascript">

    // Width and height of the SVG object
    let w = 350;
    let h = 175;

    //Make an SVG Container
    var svgContainer = d3.select("div#main")
        .append("svg")
        .attr("width", 400)
        .attr("height", 200)
        .style("border-color", "black")
        .style("border-style", "solid")
        .style("border-width", "1px");

    // Draw the Rectangle
    var rectangle = svgContainer.append("rect")
        .attr("x", 50)
        .attr("y", 50)
        .attr("width", 50)
        .attr("height", 50);

     d3.select("#start").on("click", function() {
        rectangle
            .transition()
            .attr("x", 250); // New Position
    });

    d3.select("#reset").on("click", function() {
        rectangle
            .transition()
            .attr("x", 50); // Old Position
    });
```

In Class Exercise: Opacity

Modify the code below to modify the opacity when you click on the button (make sure to also update the IDs on your buttons).

```
d3.select("#start2").on("click", function() {
          rectangle
          .transition()
      .attr("x", 250)
      .attr("opacity", 0.5)
    });

d3.select("#reset2").on("click", function() {
          rectangle
          .transition()
          .attr("x", 50) //old position
          .attr("y", 50)
         .attr("opacity", 1)
        });
```

NOW CHANGE THE COLOR:

Modify the code below to modify the color when you click on the button.

```
d3.select("#start3").on("click", function() {
          rectangle
          .transition()
      .attr("x", 250)
      .attr("opacity", 0.5)
      .attr("fill", 'blue'); // ALL OF OUR TRANSITIONS WILL GO HERE!
    });

d3.select("#reset3").on("click", function() {
          rectangle
          .transition()
          .attr("x", 50) //old position
          .attr("y", 50)
         .attr("opacity", 1)
         .attr("fill", 'green');
        });
```

Starting Transitions: Duration, Delay, and Easing

Some additional transition properties you can easily set are duration (or the amount of time taken to complete the transition), delay (the delay observed before the transition starts after initializing the transition), and easing type (the method in which the transition is completed). Read more in the D3 Documentation on Starting Transitions.

Delay

A delay to the start of your transition can be implemented by including the following in your script. Delay is in milliseconds (ms), and the default is no delay.

Modify your selection code according to the following:

```
d3.select("#start4").on("click", function() {
        rectangle
                .transition()
                .attr("fill", "red") // New Color (Hex, RGB, or Web Safe)
                .attr("opacity", 0.5) // New Opacity
                .attr("width", 100) // New Width
                .attr("height", 100) // New Height
                .attr("x", 250)
                .delay(100); // New Position
  });

  d3.select("#reset4").on("click", function() {
        rectangle
                .transition()
                .attr("fill", "black") // New Color (Hex, RGB, or Web Safe)
                .attr("opacity", 1) // New Opacity
                .attr("width", 50) // New Width
                .attr("height", 50) // New Height
                .attr("x", 50); // New Position
  });
```

Duration

Duration is the time it takes to complete the transition. Duration is in milliseconds (ms), and the default is 250ms.

Modify your selection code according to the following:

```
d3.select("#start5").on("click", function() {
        rectangle
                .transition()
                .attr("x", 250) // New Position
                .duration(2500); // Set Duration of 2500ms (2.5 seconds)
  });

  d3.select("#reset5").on("click", function() {
        rectangle
                .transition()
                .attr("x", 50) // New Position
                .duration(2500); // Set Duration of 2500ms (2.5 seconds)
  });
```

Easing

Easing is the method in which the transition is completed.

The default is 'cubic-in-out', which resembles a slow start, fast middle, and slow finish. There are many other types. To see an illustration of the easing types, view the D3js v4 Easing Example on bl.ocks. Add some of these to the above examples to see what they do! Start with 'bounce'.

Modify your selection code according to the following:

```
d3.select("#start6").on("click", function() {
        rectangle
                .transition()
                .attr("x", 250) // New Position
                .ease(d3.easeBounce); // Use the Bounce Transition Ease
  });

  d3.select("#reset6").on("click", function() {
        rectangle
                .transition()
                .attr("x", 50) // New Position
                .ease(d3.easeBounce); // Use the Bounce Transition Ease
```

```
});
```

DO MORE :

Change the type of ease for the example above. You can use some of these.

Double Transitions

You might also find that one transition is not enough. There are a couple different methods available to address this. Say you want the transition to first go to one spot, then to a second spot. You can run a function at the end of a transition using **.each("end", function(){...})**. Your function will run when the transition ends, meaning in here, you can actually put another transition. Let's take a look.

Add the code below. This will run two transitions!

```
d3.select("#start7").on("click", function() {
        rectangle
                .transition()
                        .attr("x", 250)
                        .attr("width", 100)
                        .attr("height", 100)
                        .attr("opacity", 0.5)
                        .attr("fill", "red")
                        .delay(500)
                        .duration(2500)
                        .ease(d3.easeBounce)
                        .on("end",function() { // on end of transition...
                                d3.select(this)
                                        .transition() // second transition
                                                .attr("x", 150) // second x
                                                .attr("width", 75) // second width
                                                .attr("height", 75) // second height
                                                .attr("opacity", 0.75) // second opacity
                                                .attr("fill", "blue") // second color
                                                .delay(500) // second delay
                                                .duration(2500) // second time
                                                .ease(d3.easeBounce); // second ease
```

```
                        });
});

d3.select("#reset7").on("click", function() {
        rectangle
                .transition()
                .attr("x", 50)
                .attr("y", 50); // New Position
});
```

This may seem pretty basic, but you can build pretty sophisticated graphics from these principles along. Let's dive a little deeper, and animate something real. In the following sections, we will make an interactive scatterplot, and then an interactive bubble chart.

POST THIS WEBSITE TO YOUR GITHUB ACCOUNT WE WILL BE GRADING BASED ON COMPLETION.

d3transitionexample.html

Part 2: Scatterplot and bubble chart - Adding Transitions between Data Fields

A common operation will be to change the dataset, then apply a transition to move the points in the dataset to the new locations. Say we now have data that looks like this, with two values we want to transition between.

City	# of Rats (2015)	# of Coffee Shops (2015)	# of Rats (2016)	# of Coffee Shops (2016)
Brookline	40	50	60	30
Boston	90	120	10	30
Cambridge	30	90	80	100
Chelsea	10	10	100	120
Somervillle	60	40	40	60

And we want the scatterplot to transition the data between the two years. You can apply the transitions learned above to complete this task.

Start with our basic scatterplot (open interacton_scatterplot.html file in a text editor and in your web browser). It should look like the following:

1. Add some buttons

From here, we can make some initial adjustments. First, let's add two buttons that can be used to trigger the transition. We will set them up to listen for click events. Put these in the body of the page above the div element.

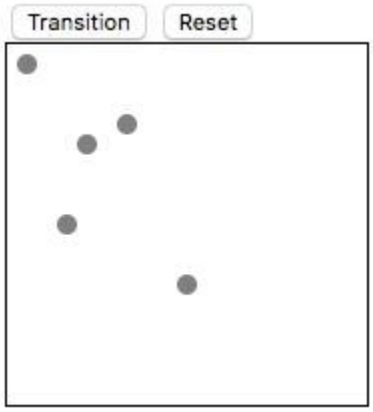
<button id="start">Transition</button>

<button id="reset">Reset</button>

```
<body>
    <button id="start">Transition</button>
    <button id="reset">Reset</button>
    <div id="main">

    </div>
<script type="text/javascript">
```

Your site should now look like the following:



2. Write the transition functions

Our next step is to write two functions that will transition the data between our two datasets when they hear a click on one of the buttons. We can use the D3 on click functionality to listen for this. The functions will look as follows. Add this code to your file below the rest of your D3 code.

```
  d3.select("#start").on("click", function() {
  svg.selectAll("circle")
      .transition()
      .attr("cx", function(d) {
      return d[2];
      })
      .attr("cy", function(d) {
      return d[3];
      })
});

d3.select("#reset").on("click", function() {
svg.selectAll("circle")
      .transition()
      .attr("cx", function(d) {
      return d[0];
      })
      .attr("cy", function(d) {
      return d[1];
      })
});
```

These functions transition the circle objects on the page to new locations based different fields in the dataset when a click is heard on either **#start or #reset**. These functions take each circle and apply a new X and Y.

In this example, we only transitioned location, but we could also transition color, size, and other attributes as described above.

Transitions in a bubble chart

Using the data above, on rats and coffee shops, we will make a bubble chart instead of a scatter plot. Your chart should have one row of bubbles (for coffee shops). Your chart should have two buttons, one button for 2015 and one button for 2016. Each bubble should be sized according to the quantity of coffee shops in that neighborhood. You will use a function to set the radius to reflect the values in the dataset.

2. Change the buttons

<button id="firstyear">2015 coffee shops</button>
<button id="secondyear">2016 coffee shops</button>

```
<body>
    <!-- Add buttons with IDs -->
    <button id="firstyear">2015 coffee shops</button>
    <button id="secondyear">2016 coffee shops</button>

    <div id="main">

    </div>
```

2. Change the width and height of the SVG

Change the width variable to 800 and the height variable to 600.

```
<script type="text/javascript">


    data2015 = [[40,50,60,30],[90,120,

    //Width and height
    var w = 800;
    var h = 600;
```

3. Change the attributes of the circles

Change the cx, cy, and r attributes of the circles. Make the r attribute change according to the values in the dataset. You can use the following code:

```
svg.selectAll("circle")
        .data(data)
        .enter()
        .append("circle")
        .attr("cx", function(d, i) {
                return 100 + i*100;
                })
        .attr("cy", 200)
        .attr("r", function(d) {
                return d[1]/2.5
        })
        .attr("fill", "orange");
```

4. Add functions for changing the data when user clicks buttons

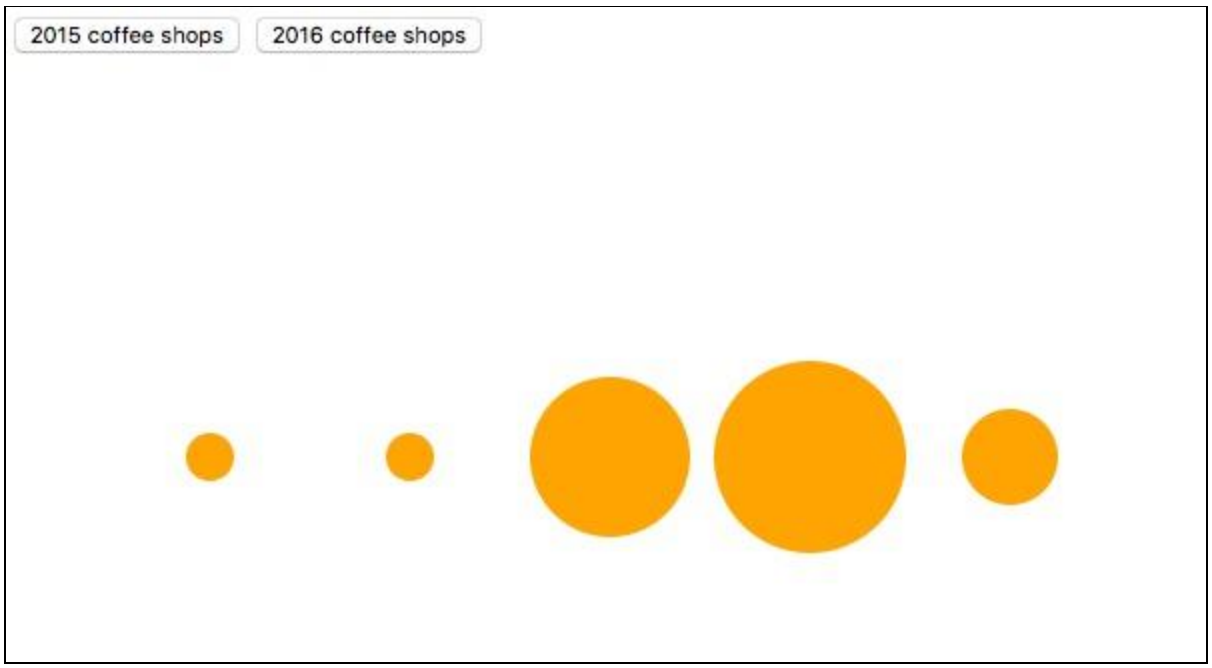
Add functions when a user clicks on either button. You can use on click to change the radius of the circles when the user clicks on a button. Use the following code to change the circle size between 2015 and 2016:

```
        d3.select("#firstyear").on("click", function() {
        svg.selectAll("circle")
        .transition()
          .attr("r", function(d) {
                return d[1]/2.5
        })
});

d3.select("#secondyear").on("click", function() {
svg.selectAll("circle")
        .transition()
```

```
        .attr("r", function(d) {
                return d[3]/2.5
        })
});
```

Your final visual should look like the following:



POST THIS WEBSITE TO YOUR GITHUB ACCOUNT WE WILL BE GRADING BASED ON COMPLETION.

interacton_scatterplot.html