CDS 6324
Lab 11: D3 Animation and Interaction

Learning Objectives

- Understand how D3 animation and transitions work
- Learn to add transitions between data fields
- Learn to add transitions between two separate datasets
- Learn to animate a line chart
- Learn to animate axes and attributes of SVG elements
- Learn to create simple interaction via filtering

D3 Transition

Transition is the process of changing from one state to another of an item. D3.js provides a **transition()** method to perform transition in the HTML page. Let us learn about transition in this chapter.

The transition() method

The transition() method is available for all selectors and it starts the transition process. This method supports most of the selection methods such as – attr(), style(), etc. But, It does not support the append() and the data() methods, which need to be called before the transition() method. Also, it provides methods specific to transition like duration(), ease(), etc. A simple transition can be defined as follows −

```
d3.select("body")
   .transition()
   .style("background-color", "lightblue");
```

A transition can be directly created using the d3.transition() method and then used along with selectors as follows.

```
var t = d3.transition()
   .duration(2000);

d3.select("body")
   .transition(t)
   .style("background-color", "lightblue");
```

A Minimal Example

Let us now create a basic example to understand how transition works.

Create a new HTML file, **transition_simple.html** with the following code.

```
<!DOCTYPE html>
<html>
   <head>
      <script type = "text/javascript" src =
                              "https://d3js.org/d3.v7.min.js"></script>
```

```
   </head>
   <body>
      <h3>Simple transition</h3>
      <script>
         d3.select("body")
               // make the background-color lightblue
               .transition().style("background-color", "lightblue")
      </script>
   </body>
</html>
```

Here, we have selected the **body** element and then started transition by calling the transition() method. Then, we have instructed to transit the background color from the current color, **white** to **light blue**.

Lifecycle of Transition

Transition has a four-phased lifecycle −

- The transition is scheduled.
- The transition starts.
- The transition runs.
- The transition ends.

Let us go through each of these one by one in detail.

The Transition is scheduled

A transition is scheduled when it is created. When we call **selection.transition**, we are scheduling a transition. This is also when we call **attr(), style()** and other transition methods to define the ending key frame.

The Transition Starts

A transition starts based on its delay, which was specified when the transition was scheduled. If no delay was specified, then the transition starts as soon as possible, which is typically after a few milliseconds.

The Transition Runs

When the transition runs, it is repeatedly invoked with values of transition ranging from 0 to 1. In addition to delay and duration, transitions have easing to control timing. Easing distorts time, such as for slow-in and slow-out. Some easing functions may temporarily give values of t greater than 1 or less than 0.

The Transition Ends

The transition ending time is always exactly 1, so that the ending value is set exactly when the transition ends. A transition ends based on the sum of its delay and duration. When a transition ends, the end event is dispatched.

D3 Animation

D3.js supports animation through transition. We can do animation with proper use of transition. Transitions are a limited form of **Key Frame Animation** with only two key frames – start and end. The starting key frame is typically the current state of the DOM, and the ending key frame is a set of attributes, styles and other properties you specify. Transitions are well suited for transitioning to a new view without a complicated code that depends on the starting view.

Example − Modify the code in "transition_simple.html" page to add in an additional transition as follows:

```
// make the background-color gray
  .transition().style("background-color", "gray");
```

Here, the Background color of the document changed from white to light gray and then to gray.

The duration() Method

The duration() method allows property changes to occur smoothly over a specified duration rather than instantaneously. Create a new HTML file, **transition-duration.html** to create a transition which takes 5 seconds using the following code.

```
<!DOCTYPE html>
<html>
   <head>
      <script type = "text/javascript" src =
                                 "https://d3js.org/d3.v7.min.js"></script>
   </head>

   <body>
      <h1>If there is a will there's a way!</h1>
      <script>
       d3.selectAll("h1").transition().style("color","blue").duration(3000);
      </script>
   </body>
</html>
```

Here, the transitions occurred smoothly and evenly. We can also assign RGB color code value directly using the following method.

```
d3.selectAll("h3").transition().style("color","rgb(0,150,120)").duration(3000);
```

Now, each color number slowly, smoothly and evenly goes from 0 to 150. To get the accurate blending of in-between frames from the start frame value to the end frame value, D3.js uses an internal interpolate method, **`d3.interpolate(a, b).`**

D3 also supports the following interpolation types −

- **interpolateNumber** − support numerical values.
- **interpolateRgb** − support colors.
- **interpolateString** − support string.

D3.js takes care of using the proper interpolate method and in advanced cases, we can use the interpolate methods directly to get our desired result. We can even create a new interpolate method, if needed. More details on d3 interpolation can be obtained from https://github.com/d3/d3-interpolate

The transition in the previous example, **transition_simple.html**, occurs too fast for the user to notice the intermediate transition to lightblue. Edit the program to slow down the transition using the duration method.

The delay() Method

The delay() method allows a transition to take place after a certain period of time. Consider the following code in **transition_delay.html**.

```
<!DOCTYPE html>
<html>
   <head>
      <script type = "text/javascript" src =
                              "https://d3js.org/d3.v7.min.js"></script>
   </head>

   <body>
      <h2  style="color:steelblue;">Life is all about the choices we make.</h2>
      <script>
         d3.selectAll("h2").transition()
            .style("font-size","48px").delay(1000).duration(2000);
      </script>
   </body>
</html>
```

A. Simple Interaction and Animation with D3 Charts

For this exercise, we will create simple and useful interaction and transition for a bar chart. You can use the **barchart_base.html**, which consist a static bar chart as as a starting point.

I. D3 Enter, Update and Exit

D3's enter, update and exit methods can be used to achieve **fine grained control over the behaviour of entering, updating and exiting elements**. This is particularly useful when dealing with dynamic behaviors, such as transitioning elements and providing interaction.

The following is the Enter-Update-Exit code in **barchart_base.html**.

```
// DATA JOIN use the key argument for ensuring that the same
// DOM element is bound to the same data-item

const rect = g
          .selectAll("rect")
          .data(new_data, (d) => d.location.city)
          .join(
            // ENTER:
            // new elements
            (enter) => {
              const rect_enter = enter.append("rect").attr("x", 0);
              rect_enter.append("title");
              return rect_enter;
            },
            // UPDATE:
            // update existing elements
            (update) => update,
            // EXIT:
            // elements that aren't associated with data
            (exit) => exit.remove()
);
```

To understand more about how Enter-Update-Exit works, go through the following tutorial:

D3 Enter, Exit and Update: https://www.d3indepth.com/enterexit/

II. Basic Interaction with CSS

CSS can also be used to create some simple interaction. Add the following code to the CSS section of the HTML file to change mourseover attribute:

```
rect:hover {
        fill-opacity: 1;
}
```

Observe its effect when you mouseover a bar in the bar chart.

III. D3 Transition

Transition can be easily applied to a DOM object such as SVG elements

a) To apply transition to a SVG Element such as a rectangle, we can add the transition() method to the element vis the chaining method as follows:

```
        rect
          .transition()
          .attr("height", yscale.bandwidth())
          .attr("width", (d) => xscale(d.temperature))
          .attr("y", (d) => yscale(d.location.city));
```

b) The above code only apply transition to the rectangle, but not the axes. Next, modify the code that draws the axes to apply transition to the s- and y-axes as follows:

```
        //render the axis
        g_xaxis.transition().call(xaxis);
        g_yaxis.transition().call(yaxis);
```

III. Simple D3 Interaction

Filtering and tooltip are two useful approaches to create interactive visualizations that allow users to explore the visualization to obtain useful insights.

a) Add the following code to launch an event that changes the chart when the value of a checkbox changes:

```
// This code be triggered when the user selects or unselects the checkbox
d3.select("#filter-us-only").on("change", function () {

        const checked = d3.select(this).property("checked");

        if (checked === true) {
          // Checkbox was just checked
          // Keep only data element whose country is US
          const filtered_data = data.filter((d) =>
                                          d.location.country === "US");
          update(filtered_data);
        } else {
          // Checkbox was just unchecked
          update(data);
        }
});
```

b) Add the following code to add a tooltip to show the temperature value as a tooltip when the mouse is hovered over a bar element of the bar chart:

```
rect.select("title").text((d) => d.location.city);
```

In addition to the temperature, add the city name to the toolip.

Exercise 1: Create a dropdown menu to allow viewers to sort the bar chart based on city, and temperature (in both ascending and descending order), with transition. You can refer to the following example:

https://observablehq.com/@d3/bar-chart-transitions#chart

Hint: Ensure that when the visualization reflect the selected data for the sorted chart.

B. More on D3 Interaction and Animation

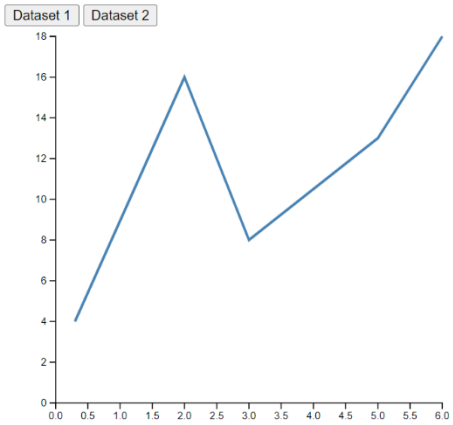
Next, we will go through and explore the following Observable tutorials from the "25 Days of D3.js" tutorial by *Tyler Wolf,* to learn more about interaction and animation.

Day 11 - Introduction to Animations
Day 12 - Interactions 1 - Click and Hover
Day 15 - Interactions 4 - Brush

Transfer the above code example to HTML/CSS/Javascript files and run it from the web server.

C. Transition between two datasets

The given **linechart_animate_datasets.html** program is an example of performing transition between two simple datasets as follow:



Study this example and understand how the the transition between the two datasets is programmed.

Exercise 2: Using the dataset from **plane_salestrend.tsv,** create a animated visualization that allow user to compare the sales trend (%) between Airbus and Boeing. You can either choose one of the following form of animation:

1) Show one lines representing Airbus or Boeing at one time and animate between two line charts as in the example given above.
2) Show two lines together on the chart with different colors for Airbus and Boeing respectively, with the two lines animating from left to right.

Relevant references:

- Time-series line chart (Lab 7): https://observablehq.com/@thetylerwolf/day-7-a-line-chart
- Animating a line chart: https://observablehq.com/@blosky/animated-line-chart

More Examples from the Web

a) Learn D3: Animation
   https://observablehq.com/@d3/learn-d3-animation
b) D3 Transition
   https://github.com/d3/d3-transition
c) Animation Loops
   https://observablehq.com/@mbostock/animation-loops
d) Brush and Zoom
   https://medium.com/@xoor/brush-and-zoom-with-d3-js-and-canvas-71859cd28832