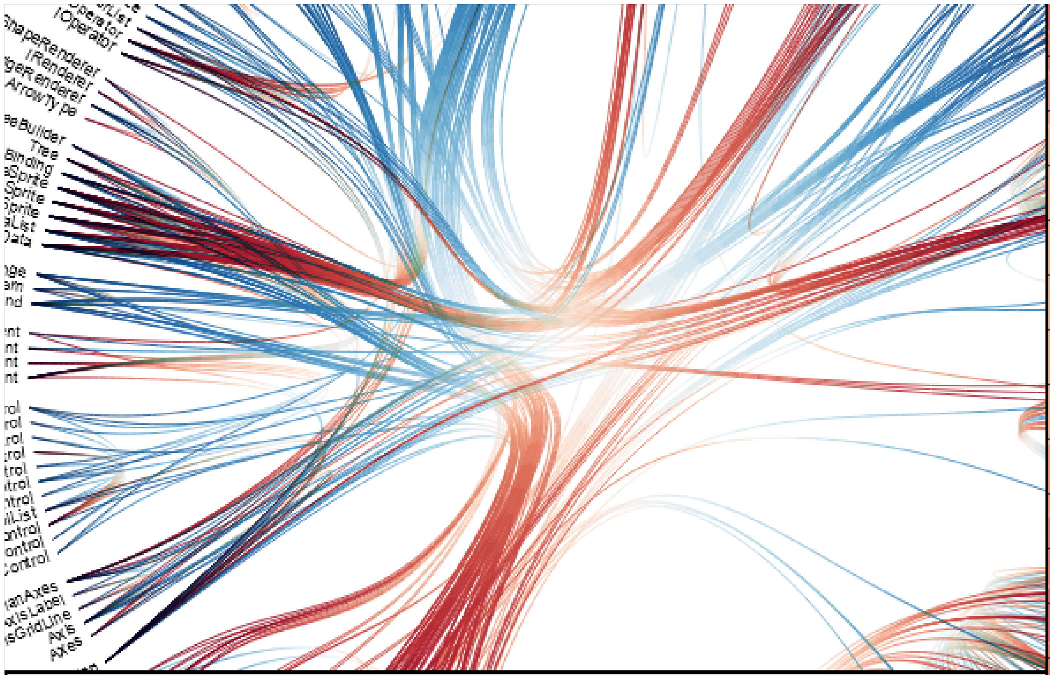
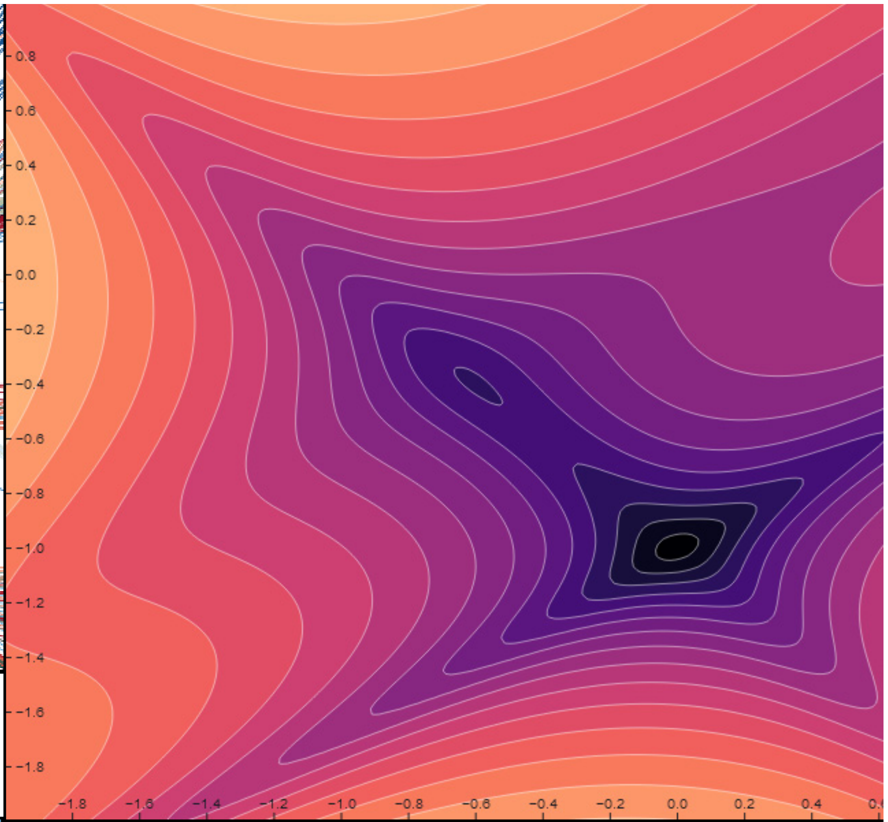
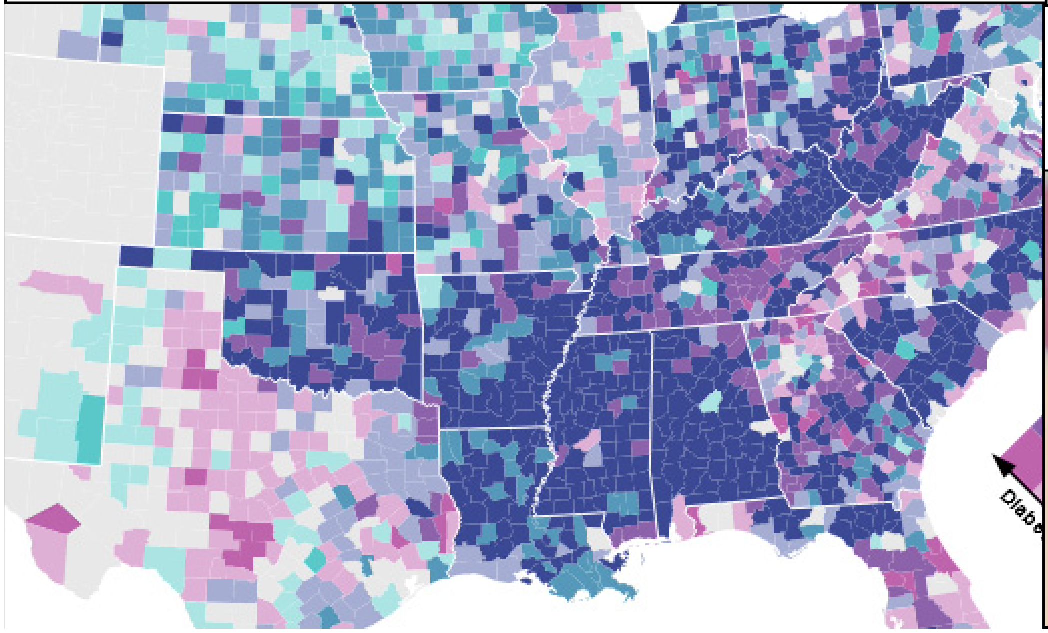


Visualization



Animated D3



Credit: d3js.org/

Animated Transitions

- Animated transitions can be applied to selections using the **transition** operator
- Transitions change the **style** and **attr** operators of selections
- Values are interpolate from the current to specified value gradually over time
- The delay and duration of transitions can be specified as functional operators
- Easing can also be set as **“elastic”**, **“cubic-in-out”** and **“linear”**

Transitions

- Transitions are a form of key frame animation
- Starting frame is the current state of the DOM
- Ending frame is a set of attributes, styles and/or properties you specify
- Use the **transition()** function to make the change

Transitions Between Datasets

- To transition all the data values at once:
  - Modify the values in your dataset.
  - Rebind the new values to the existing elements
  - Set new attribute values as needed to update the visual display.

Reacting to an Event

- To initiate the change, we can react to an event
- Add a paragraph to the HTML's body:

```
<p>Click on this text to update the chart with new data values
(once).</p>
```

- Then, add the following:

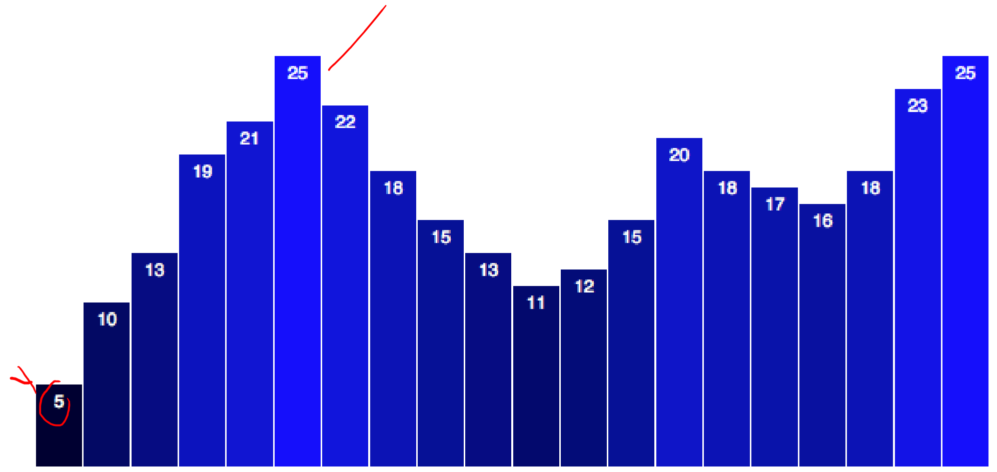
```
d3.select("p")
    .on("click", function() {
        //Do something on click
    });
```

Recall our Bar Chart

- Our 1D data:

```
var dataset = [ 5, 10, 13, 19, 21, 25, 22, 18, 15, 13,
                11, 12, 15, 20, 18, 17, 16, 18, 23, 25 ];
```

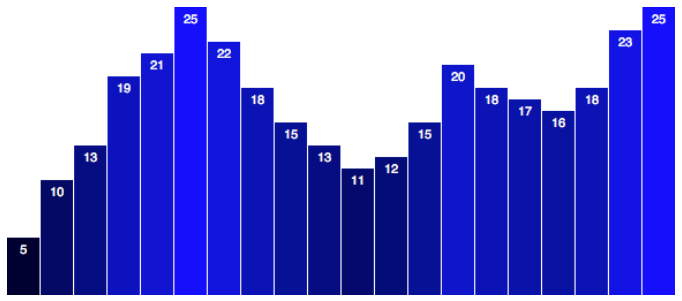
- Color coded, with padding and labels:

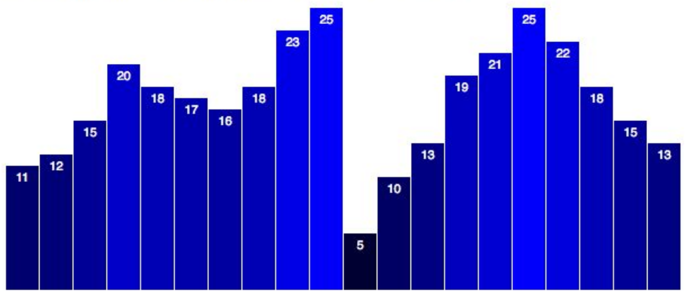


Changing the Data

- We need to bind and use new data values

```
//New values for dataset
dataset = [ 11, 12, 15, 20, 18, 17, 16, 18, 23, 25,
            5, 10, 13, 19, 21, 25, 22, 18, 15, 13 ];
```





```
//On click, update with new data
d3.select("p")
    .on("click", function() {
        //New values for dataset
        dataset = [ 11, 12, 15, 20, 18, 17, 16, 18, 23, 25,
                    5, 10, 13, 19, 21, 25, 22, 18, 15, 13 ];

        //Update all rects
        svg.selectAll("rect")
            .data(dataset)
            .attr("y", function(d) {
                return h - yScale(d);
            })
            .attr("height", function(d) {
                return yScale(d);
            });
           .attr("fill", function(d) {
                return "rgb(0, 0, " + Math.round(d * 10) + ")";
    svg.selectAll("text")
        .data(dataset)
        .text(function(d) {
            return d;
        })
        .attr("y", function(d) {
            return h - yScale(d) + 14;
        });

});
```

Transitions Control

```
//Update all rects
svg.selectAll("rect")
    .data(dataset)
    .transition()      // <-- This is new!

    .attr("y", function(d) {
        return h - yScale(d);
    })
    .attr("height", function(d) {
        return yScale(d);
    })
    .attr("fill", function(d) {
        return "rgb(0, 0, " + Math.round(d * 10) + ")";
});
```

Transitions – Duration

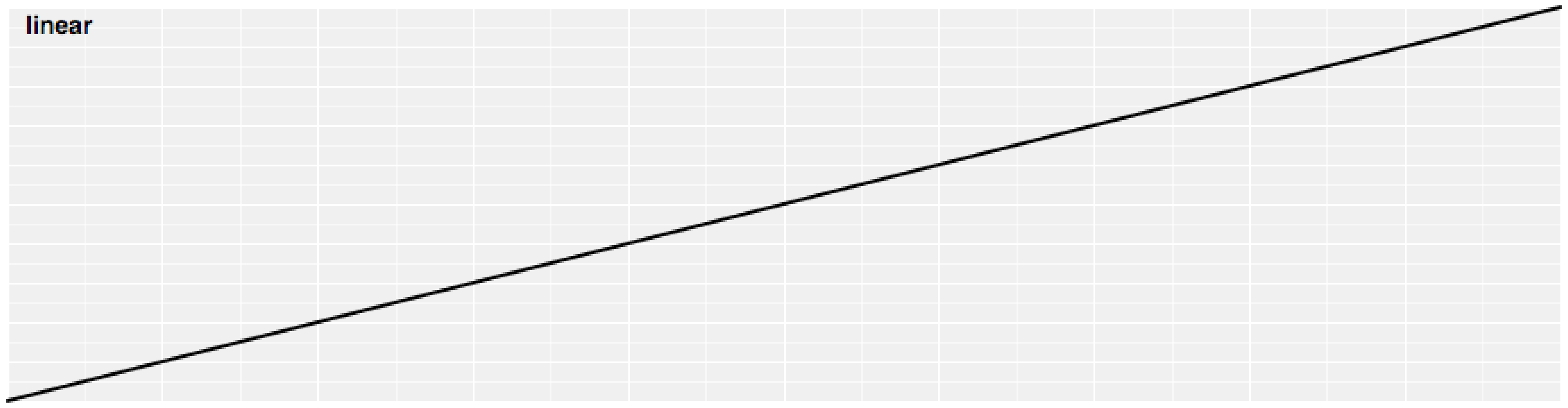
- Can override duration with gxudwlrq+ploolvhfv,>

```
svg.selectAll("rect")
   .data(dataset)
   .transition()
   .duration(1000)   // <-- Now this is new!
   .attr("y", function(d) {
       return h - yScale(d);
   })
   .attr("height", function(d) {
       return yScale(d);
   })
   .attr("fill", function(d) {
       return "rgb(0, 0, " + Math.round(d * 10) + ")";
   });
```

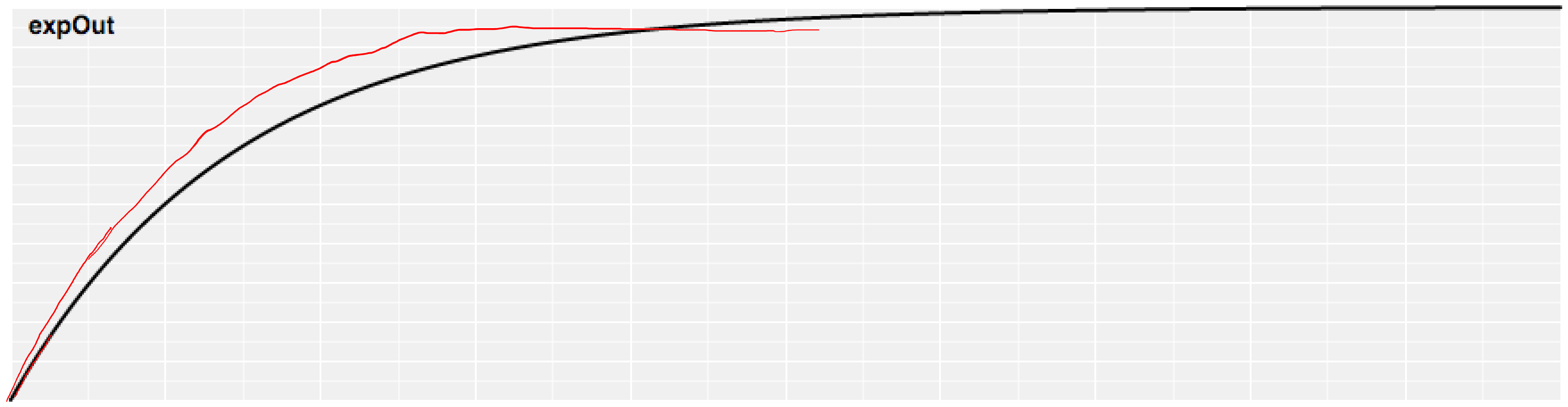
Transitions - Ease

- Ease controls the rate of change of the transition
- In addition to the default linear, lots of other easing types are supported:
  - d3.**easeLinear**(*t*)
  - d3.**easePolyIn**(*t*)
  - d3.**easePolyOut**(*t*)
  - d3.**easePoly**(*t*)
  - d3.**easePolyInOut**(*t*)
  - d3.**easeQuadIn**(*t*)
  - Etc, etc, etc

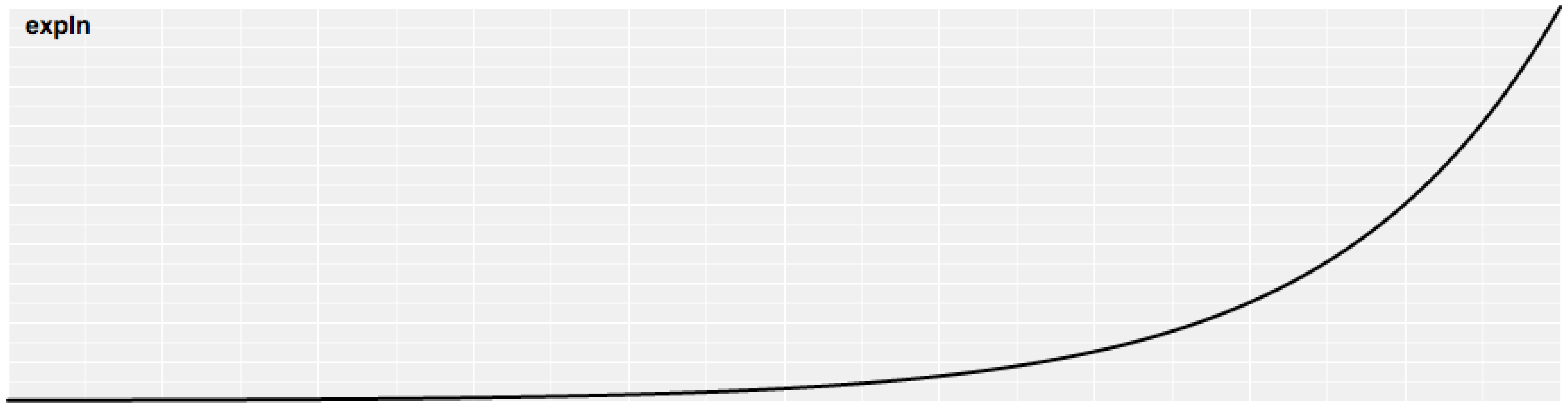
Transitions – Linear Ease



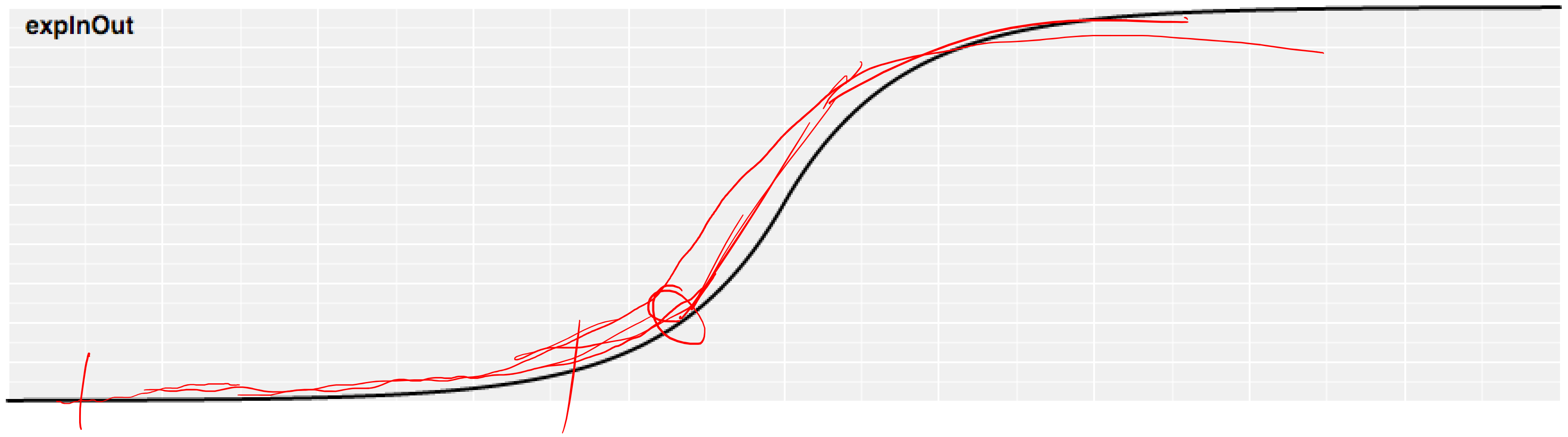
Transitions – ExpOut Ease



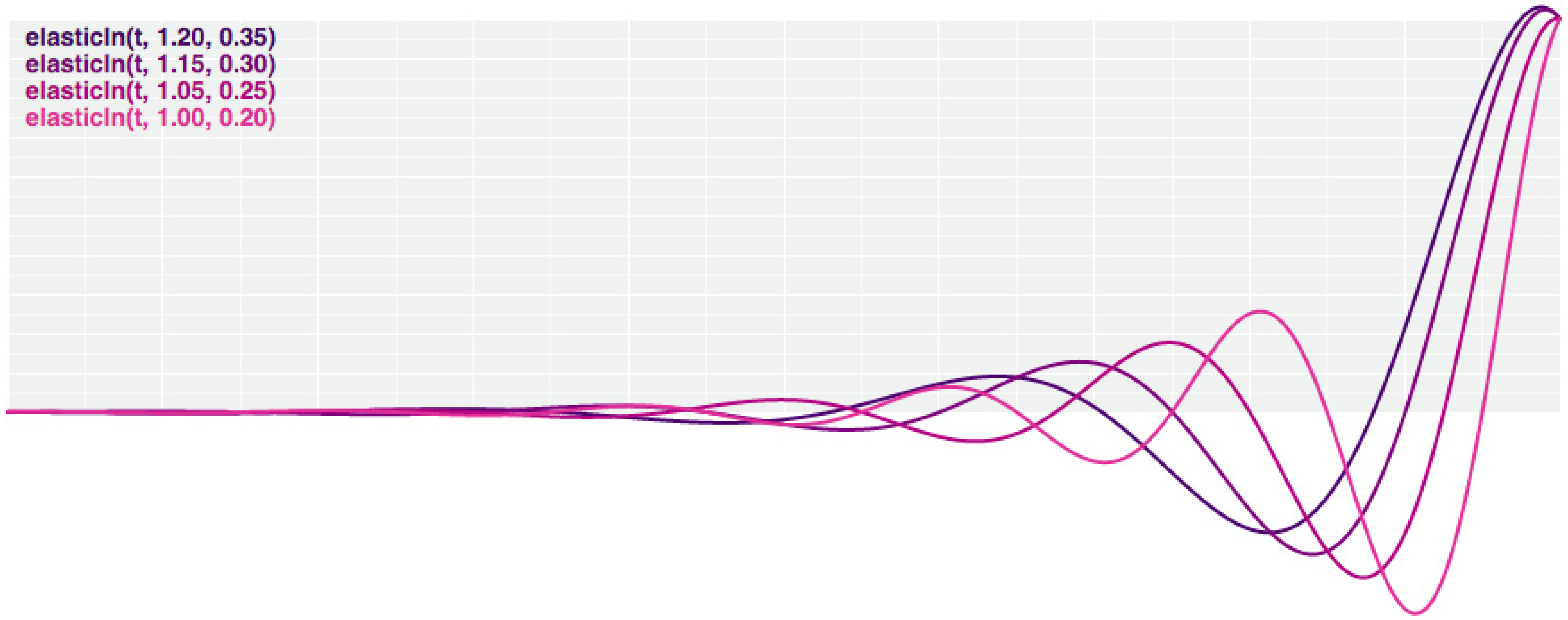
Transitions – ExpIn Ease



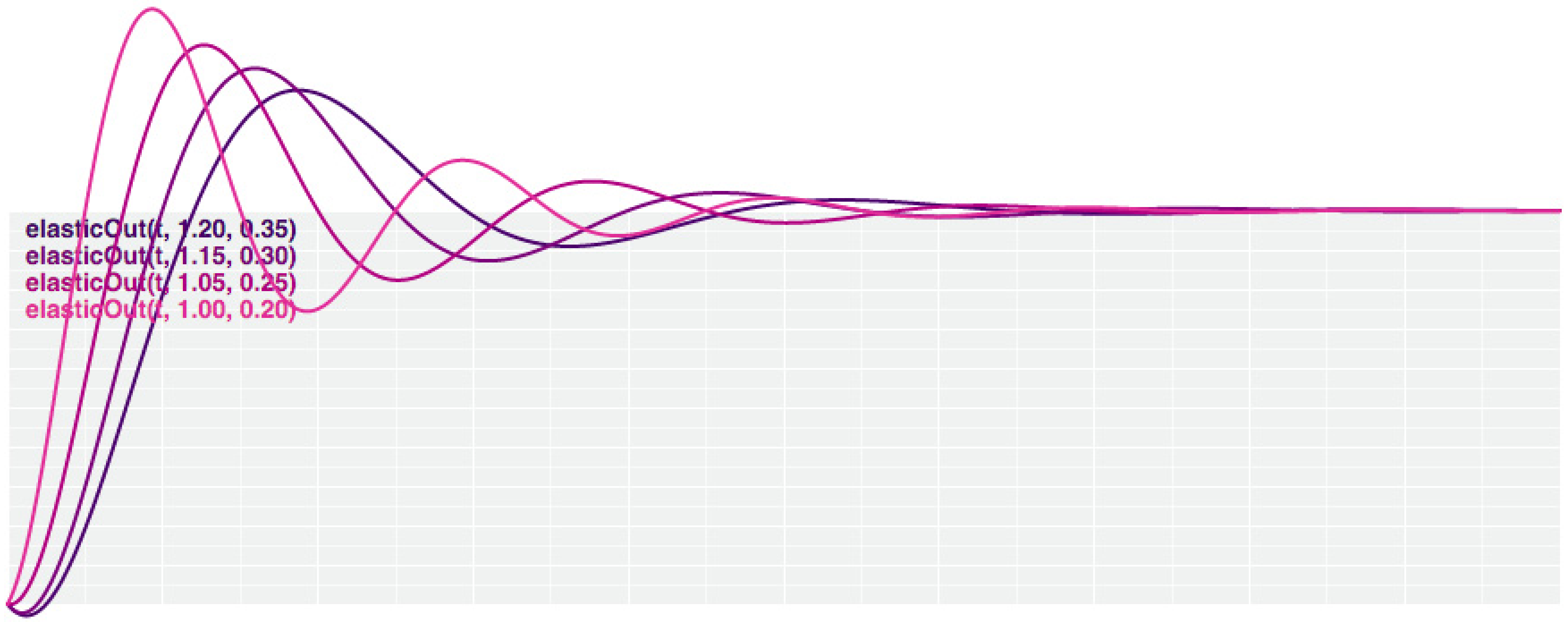
Transitions – ExpInOut Ease



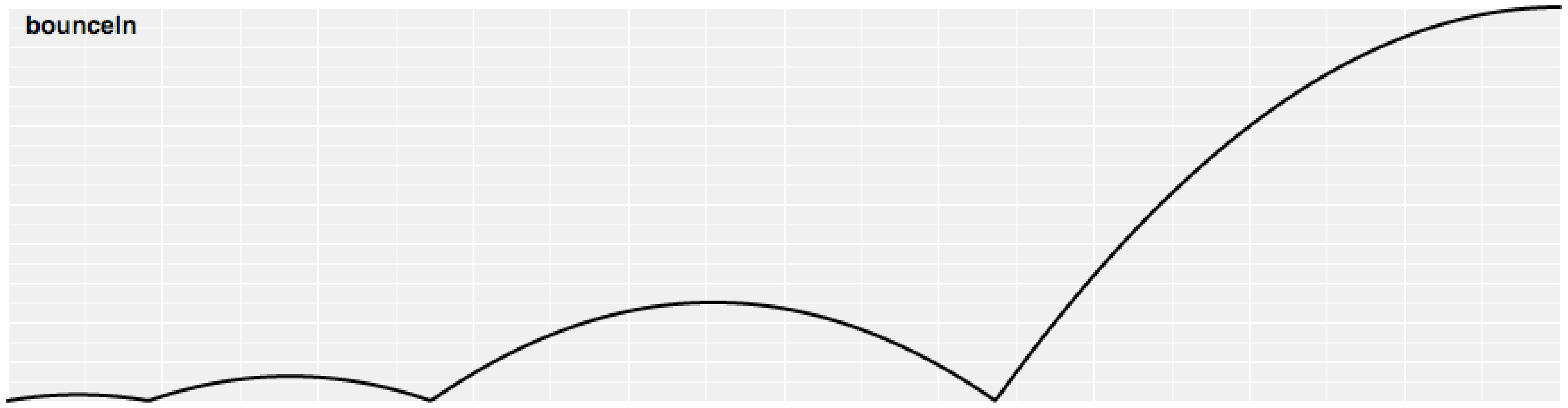
Transitions – ElasticIn Ease



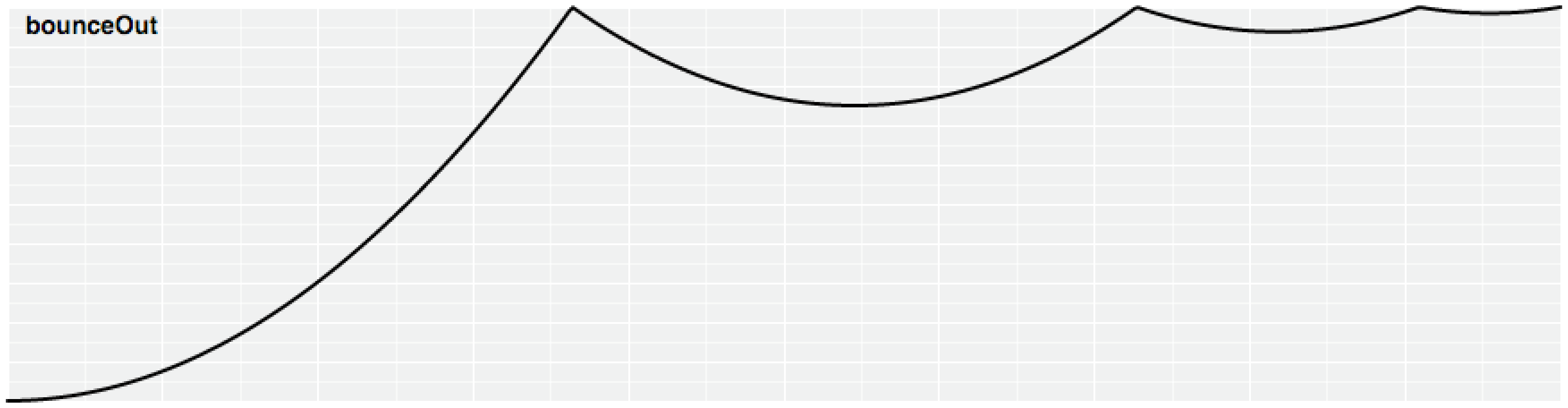
Transitions – ElasticOut Ease



Transitions – BounceIn Ease



Transitions – BounceOut Ease



Transitions – Ease

- Can set transition style with a call to Ease()

```
…    //Selection statement(s)
.transition()
.duration(2000)
.ease(d3.easeLinear)
…
```

Transitions – Delays

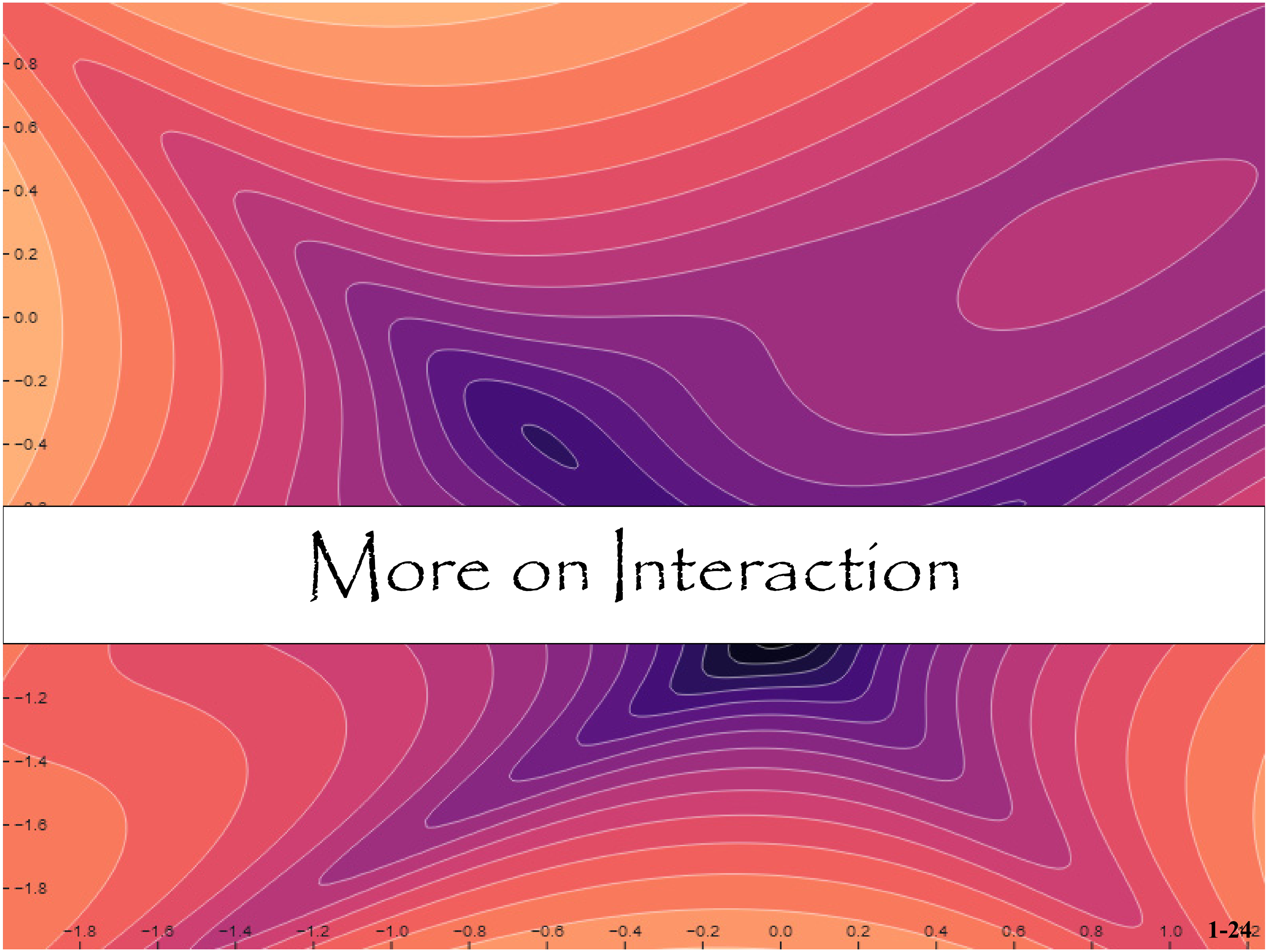
- Can also set a delay() timer, before the transition

```
...
.transition()
.delay(1000)      //1,000 ms or 1 second
.duration(2000)   //2,000 ms or 2 seconds
...
```

Transitions – Staggered Delays

- Can also set a delay() timer, before the transition

```
...
.transition()
.delay(function(d, i) {
    return i * 100;
})
.duration(500)
...
```



More on Interaction

User Input

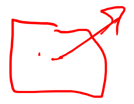
- User interaction supported with **event listeners**
- Need to set even listeners with the '.on' method
- Other topics:
  - Mouseover, mouseout, mousedown, mouseup

User Interface Event Listeners

- To bind a function to a mouse event:

  .on(*type*, function() { });

- There are several kinds of events:
  - mouseover - mouse is hovering over the object
  - mouseout - mouse leaves the object
  - mousedown - mouse button is held down
  - mouseup - let go of your mouse button
  - click - click mouse button



User Input

- You can bind event listeners when you first create elements

```
//Create bars
svg.selectAll("rect")
   .data(dataset)
   .enter()
   .append("rect")
   …   //Set attributes (omitted here)
   .on("click", function(d) {
       //This will run whenever *any* bar is clicked
   });
```

User Input – selecting bars

- You can bind event listeners when you first create elements

```
//Create bars
svg.selectAll("rect")
    .data(dataset)
    .enter()
    .append("rect")
    …   //Set attributes (omitted here)
    .on("click", function(d) {
        console.log(d);
    });
```

User Input - hovering

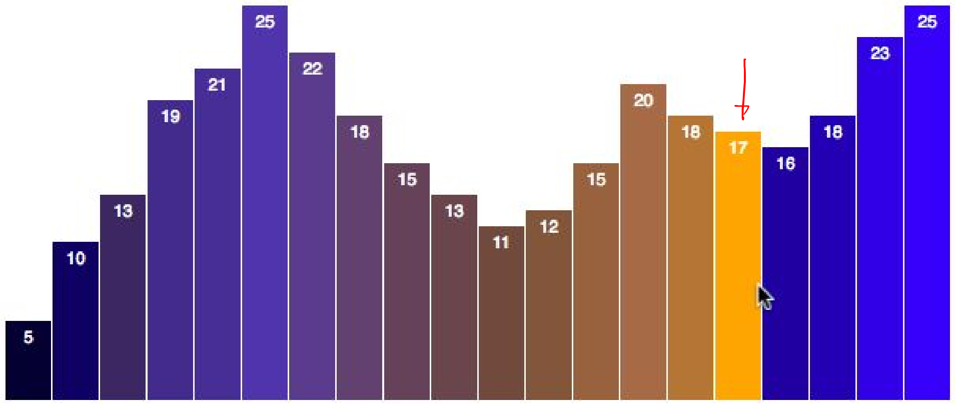
```
//Create bars
svg.selectAll("rect")
    .data(dataset)
    .enter()
    .append("rect")
    …   //Set attributes (omitted here)
    .on("mouseover", function() {
        d3.select(this)
          .attr("fill", "orange");
    });
```

User Input – hovering (fixed)

```
//Create bars
svg.selectAll("rect")
    .data(dataset)
    .enter()
    .append("rect")
…      //Set attributes (omitted here)
    .on("mouseover", function() {
        d3.select(this)
          .attr("fill", "orange");
    });
    .on("mouseout", function(d) {
        d3.select(this)
          .attr("fill", "rgb(0, 0, " + (d * 10) + ")");
    });
```

User Input – hovering (fancy)

```
.on("mouseout", function(d) {
    d3.select(this)
      .transition()
      .duration(250)
      .attr("fill", "rgb(0, 0, " + (d * 10) + ")");
});
```



Even More to Learn about

Layouts & Behaviours

- Layouts supply reusable, flexible visualization techniques, including:
  - partition layout
  - chord layout
  - force layout
  - stack layout
  - squarified treemap layout
  - Etc, etc, etc…
- Common interaction techniques, including:
  - zoom behavior

Help Hours for D3 assignments and projects

- Lab times (for both 4166 & 6406)
  - Wednesday, 13:35-14:25
  - Friday, 10:35-11:25
- TA Office hours (for both 4166 & 6406)
  - Tuesday 11:05 – 11:55
- Dipankar Mazumdar is the TA

More Reference Material

- *Interactive Data Visualization for the Web* by Scott Murray, O'Reilly.
- *Getting Started with D3,* by Mike Dewar, O'Reilly.
- *SVG Essentials*, by J. Eisenberg, O'Reilly.
- The D3 website:
  - http://d3js.org/
  - https://github.com/mbostock/d3/wiki/API-Reference

DFA Copyright Information