School of Software and Electrical Engineering

COS30045 Data Visualisation

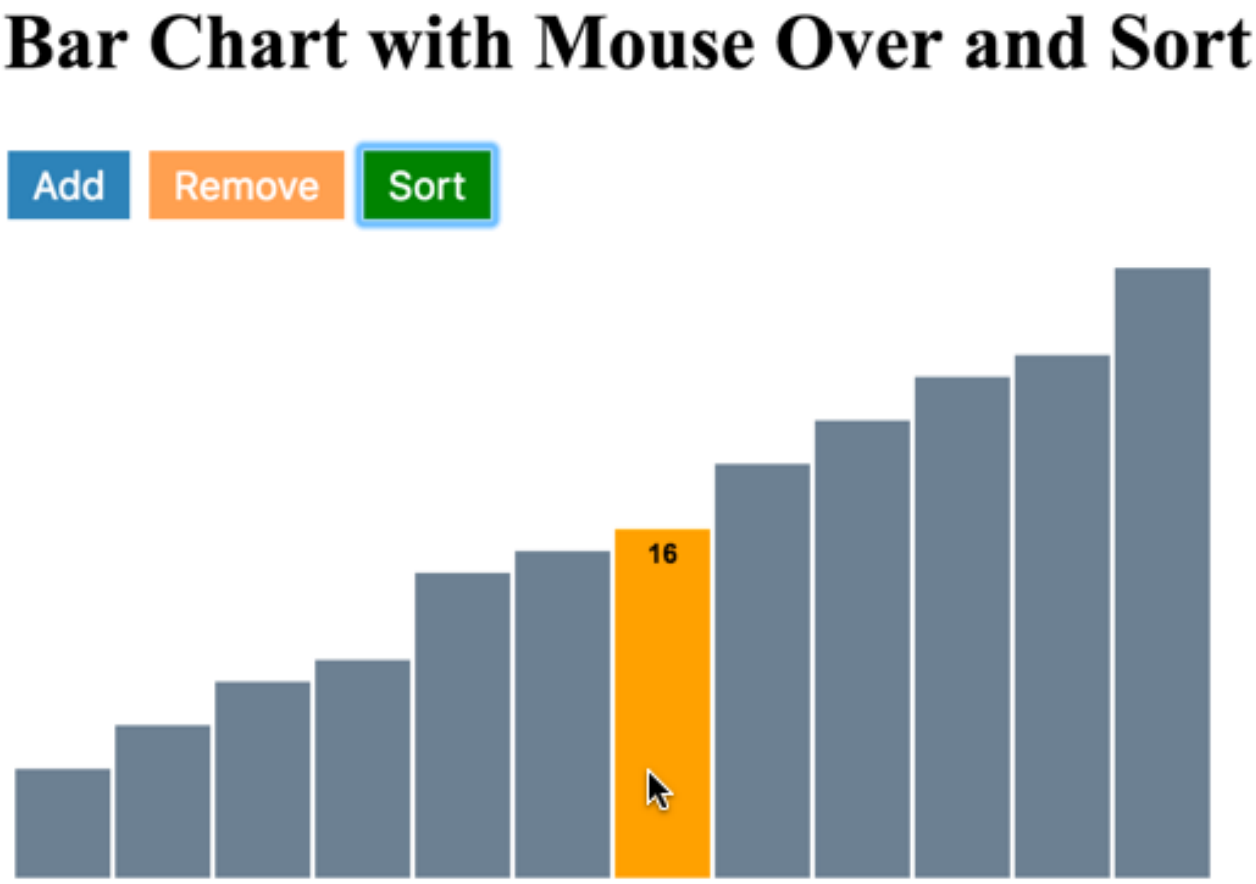
Exercise 3.5 D3 Interactivity - Sort

ILO	Create web-based interactive visualisations using real-world data sets.
Aim:	Allow users to sort data in a bar chart
Resources:	*Textbook:* Chapter 10 Interactivity - Murray (2017) Interactive Data Visualisation (2nd Ed) on ProQuest
Demonstration	If you are required to demonstrate this exercise we will be looking for: - code that is appropriate for exercise, well formatted and commented - code that runs correctly and meets the requirements specified in this exercise - an explain programming features and concepts in the code - the ability to successfully edit code to change a specified feature of the program

Note: The functions handling scale have changed between D3 v3 and D3 v4. This is something to be aware of if you are doing your own research into this topic. Make sure you use Murray Ed 2. Code examples from Ed 1 will not work.

Overview

At the end of the Interactivity - Mouse Over Effects exercise we had a bar chart that we could add and remove data from and demonstrated some mouse over effects. In this exercise we will add a sort feature.



Step 1: Set up event listener for sort button

Start with the code from Interactions Mouse Over Effects. Add a button for your new Sort function. Give it a unique id (e.g., `id="sort"`) so we can set up an event listener for it. Then set up the event listener to launch our the sort function we will build.

```
d3.select("#sort")
  .on("click", function() {
    sortBars();
});
```

Step 2: Write Sort Function

Our sort function will select all the rectangle elements then use D3 functions `sort()` and `d3.ascending()` to sort the data. Finally the rectangles need to be redrawn with new x values.

```
var sortBars = function() {

        svg1.selectAll("rect")
           .sort(function(a, b) {
                return d3.ascending(a, b);
            })
            .attr("x", function(d, i) {
                return xScale(i);
            });
        };
```

Save and run. You will notice that the transition happens straight away. Add a transition so the user can see the sort occur.

Step 3: Add a descending sort

Currently the sort only works one way. Allow the user to resort in the other direction by clicking the sort button again.

```
var sortOrder = false;

var sortBars = function() {

            sortOrder = !sortOrder;

              svg.selectAll("rect")
                 .sort(function(a, b) {
                     if (sortOrder) {
                         return d3.ascending(a, b);
                     } else {
                         return d3.descending(a, b);
                     }
                  })
                  // ....
```