School of Software and Electrical Engineering

COS30045 Data Visualisation

Exercise 1.5 D3 Drawing with Data - Bar Chart

ILO	Create web-based interactive visualisations using real-world data sets.
Aim:	Use D3 to generate elements on a webpage to create a bar chart from a data set.
Resources:	*Textbook:* Chapter 6 Drawing with Data - Murray (2017) Interactive Data Visualisation (2nd Ed) on ProQuest Murray on Safari
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: This Exercise Guide is not meant to be fully explanatory. You may also need to work through the examples in the text book *Interactive Data Visualisation* for the Web by Murray.

Overview

In this exercise we will start using D3 to draw a bar charts. At the end of this exercise you should end up with a bar chart drawn using D3 generated SVGs that looks something like this:
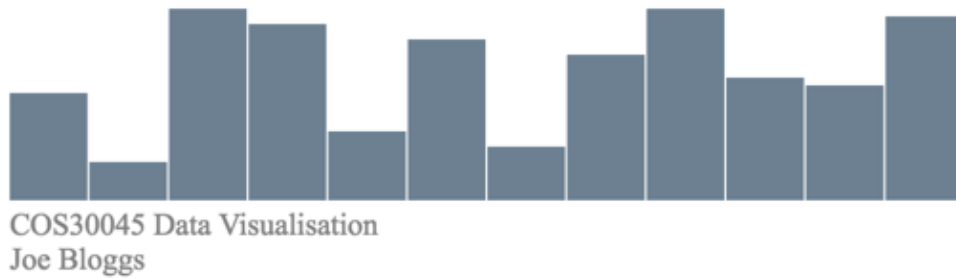
Feel free to choose your own data and styling at the end.

Step 1: Start a basic HTML template with D3

As in Task 2.1, you need to set up a basic HTML template and add in reference to the D3 library in the header.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8"/>
    <meta name="description"    content="Data Visualisation"/>
    <meta name="keywords"       content="HTML, CSS, D3"/>
    <meta name="author"         content="Your name here"/>

    <title>Task 2.1 D3 Data Binding</title>

    <script src="https://d3js.org/d3.v5.min.js"></script>

</head>
<body>

    <h1>The D3 Journey starts here...</h1>

    <script>

        //D3 Code goes here

    </script>
```

reference to v5 of D3 library

D3 is now up to v6!

Step 2 Create a SVG element on which to display the chart

Next you need to create an SVG element on which we can build our chart. This will be done by creating a new variable ('svg') so we can easily reference it later. We can use the `.attr ()` method to assign the SVG a width and a height.

```html
<script>
//Example from Murray

var w = 500;
var h = 100;

var dataset = [14, 5, 26, 23, 9];

var svg = d3.select("body")
            .append("svg")
            .attr("width", w)
            .attr("height", h);

</script>
```
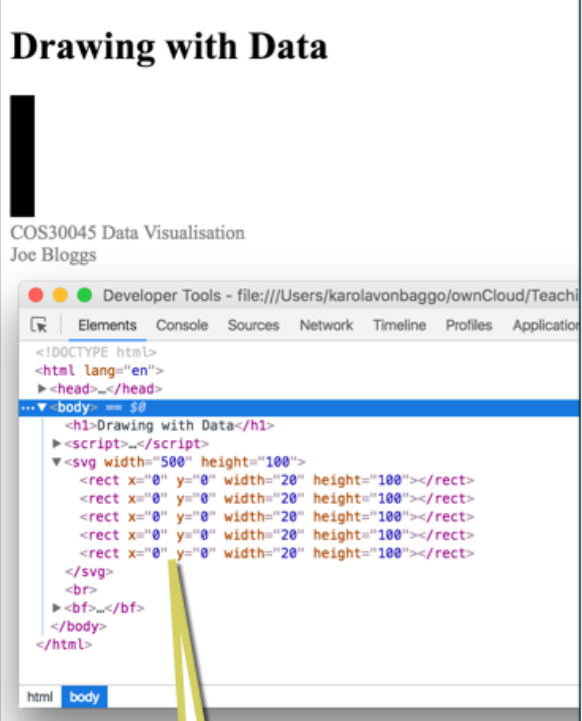
Using variables to specify height and width of the SVG will help make our code more flexible.

Step 3 Add rectangles to the chart

We will be using rectangles to visualise the data. So the next step is to attach our data to a set of rectangle shapes. We can use the the same sequence we used to attach data to paragraphs in Task 1.3, only this time we will be attaching the data to rectangles. To start with we will use the x and y values as 0 and somewhat arbitrarily use 20 and 100 for width and height.

If you run this code you will only see one black bar, but if you check the DOM you will see the five rectangles are ready there and waiting to be used to express our data. But before we do that lets make sure we will be able to see your bars properly.

Step 4 Space out the bars

At the end of this process we want a bar chart with our bars spaced out nicely. We need to move our bars along the x axis. To do this we can use the index of our data points. As we saw in Task 1.3, an index is stored with our data points so we can use the index with a handy anonymous function to get better x position's for our bars.

```html
<script>
//Example from Murray

var w = 500;
var h = 100;

var dataset = [14, 5, 26, 23, 9];

var svg = d3.select("body")
            .append("svg")
            .attr("width", w)
            .attr("height", h);

svg.selectAll("rect")
   .data(dataset)
   .enter()
   .append("rect")
   .attr("x", function(d, i) {
       return i * 21;
   })
   .attr("y", 0)
   .attr("width", 20)
   .attr("height", 100);

</script>
```
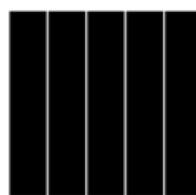
Drawing with Data

COS30045 Data Visualisation
Joe Bloggs

Our rectangles are 20 px wide, so multiplying each index by 20 will give us x's at 0, 20, 40, 60 etc. However, unless we put a gap between them they will be one big block (try and see), so we are using 21 so we have a little space between them.

Now our bars are spaced out nicely, but what if we added more data points? The current code is not very flexible. We can make it more flexible by replacing the 21 with a formula that takes into account the length of the data set and the width of the SVG.

```javascript
.attr("x", function(d, i) {
    return i * (w / dataset.length);
})
```

Luckily we have a method we can use to get the length of the data set. Note there is nothing special about 'dataset'. If your data set was called 'wombats' and could access the length with `wombats.length`.

Now our bars are equally spaced, but they look a bit skinny and too spaced out. In addition the size of the gap will be dependent on the number of bars. It would be better if we could make these relative as well. First, instead of having an arbitrary rectangle width of 20, make the **width** relative to the width of the SVG and the data set length (i.e., `w / dataset.length`), then add a variable to specify the gap between the bars (i.e, `barPadding`) and subtract this from the calculated width to give a gap (i.e, `w / dataset.length - barPadding`). Now you should have something like this:

Add some more data to the data set (between 0 and 30) and see what happens.

Step 5 Using rectangles to visualise our data

At the moment our rectangles are all the same height, but traditionally in a bar chart you want the height of the bars to reflect the data value. It is the `height` value that determines the height of the rectangle, so we need to use the data in our data set to populate the `height` value attribute. Use another anonymous function to populate the `height` attribute with your data values. Depending on the extra values you added it will probably look something like this:

Firstly the height of the data is not scaled well, it's not making full use of the full 100 px SVG height. Multiplying by 4 will make it look a bit better. (Note: This is not a very elegant way of

doing it, we will look at how to properly deal with the scaling issue in the next D3 lab). Another problem you may have noticed is that traditionally we like our bar charts to grow from the bottom, not the top. This one is not quite right because in SVGs the origin (0, 0) is in the top left hand corner and the y values increase as they go down the page.

Adjust the `height` and `y` attributes to get the bars to go the right way up. If you are having trouble check out Ch 6 of Murray. Finally add some colour to your bars using the `fill` attribute.

If you have time, follow the guidance in Murray Ch 6 to add labels to your bar graph.