School of Software and Electrical Engineering

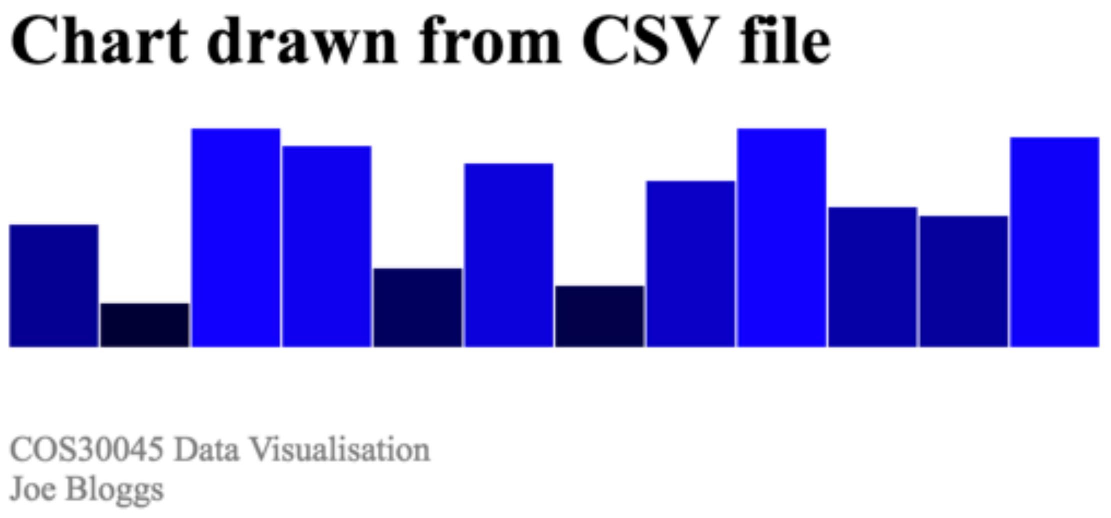
COS30045 Data Visualisation

Exercise 7.1 D3 Importing data from CSV file

ILO	Create web-based interactive visualisations using real-world data sets.
Aim:	Use D3 to populate the data set for a bar chart from a CSV file
Resources:	*Textbook:* Chapter 5 Data CSV - 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

Overview

In this tutorial 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, but populated from an imported CSV file:



In addition:

- your JavaScript/D3 code must be in a separate file (see Step 6)
- an error message is generated if the data fails to load (see Step 7)

Feel free to choose your own data and styling at the end.

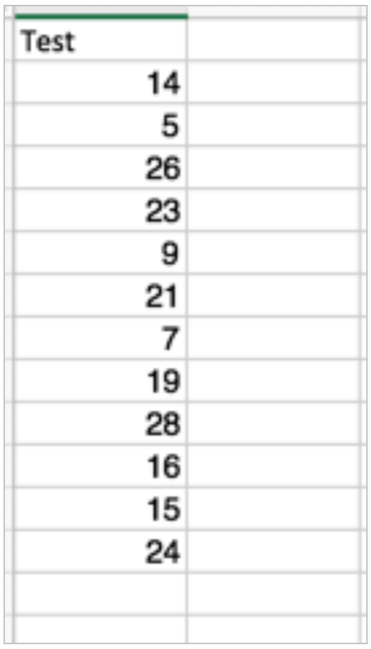
Note: This Exercise Guide is not meant to be fully explanatory. You may also need to work through the examples in the text book *Interactive Data Visualisation* for the Web by Murray.

Step 1: Start a basic HTML template with D3

Start with your code from Drawing with Data Bar Chart. Rename the file and update the meta data and title to reflect this new task.

Step 2: Create a CSV file to read your data from

Open excel and put in one column of data (you can use your data from the Drawing with Data Bar Chart exercise). Make sure you give your column of data a heading. D3 expects a heading. Save your data as an CSV file with a meaningful name (e.g., Exercise_1.7_data.csv)



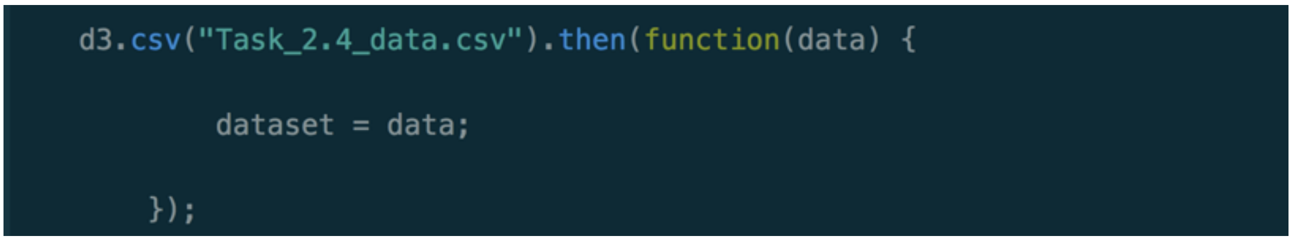
Test	
14	
5	
26	
23	
9	
21	
7	
19	
28	
16	
15	
24	

Step 3: Parsing the data

Remove the hard coded data from your code as we will now be reading in from the CSV file you just created. You should now just be left with an empty dataset variable. D3 automatically assumes that any data it reads in will 1) have a heading and 2) is a string.

Step 4: Reading in the data

To read in the data we use the d3 function `csv`

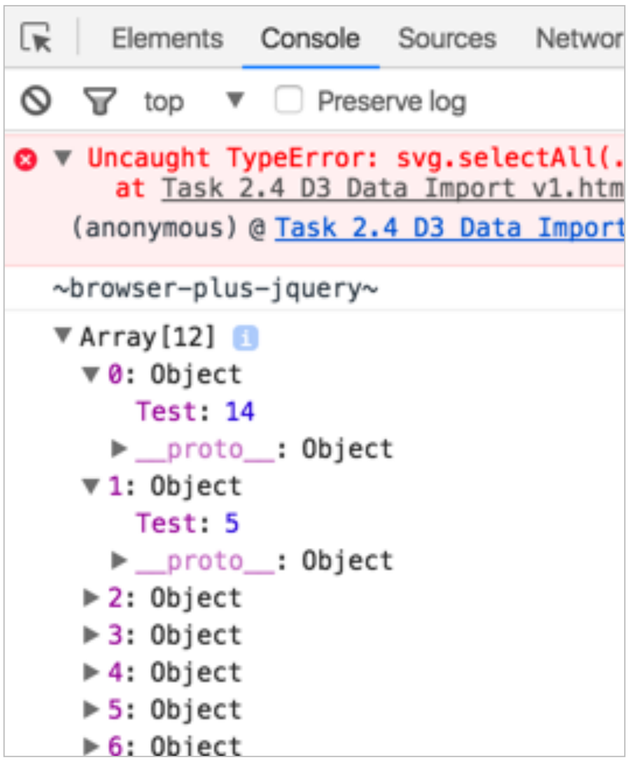


```
d3.csv("Task_2.4_data.csv").then(function(data) {

        dataset = data;

    });
```

Review the console log to demonstrate the the data is loaded in. At the moment the bar chart won't work because it doesn't have access to the data.



```
Uncaught TypeError: svg.selectAll(.
    at Task 2.4 D3 Data Import v1.htm
  (anonymous) @ Task 2.4 D3 Data Import
  ~browser-plus-jquery~
  ▼Array[12]
    ▼0: Object
       Test: 14
      ▶__proto__: Object
    ▼1: Object
       Test: 5
      ▶__proto__: Object
    ▶2: Object
    ▶3: Object
    ▶4: Object
    ▶5: Object
    ▶6: Object
```

One way to give our chart code access to the data would be to cut and paste it into the the `d3.csv` function. However, that would reduce our ability to reuse the code, so instead we will turn it into an independent bar chart function and call it from within the `d3.csv` function.

```
    d3.csv("Task_2.4_data.csv").then(function(data) {

            dataset = data;
            barChart(dataset);

        });
```

Cut and paste your chart code into the new function. Finally, you need to alter the way you reference the data points to tell D3 which column to get the data from.

```
function barChart() {

        //code goes here...

};
```

```
    .attr("y", function(d) {
        return  h - (d.Test * 4);
    })
```

Update all references to d to d.Test and now your code should produce a chart that uses the imported data. Hint: This is case sensitive, in this case, `d.test` will not work.

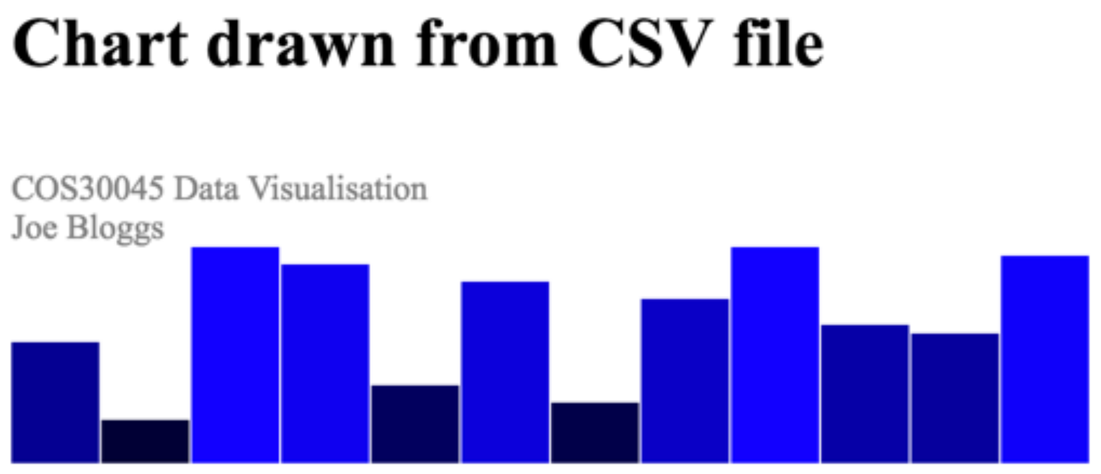


Chart drawn from CSV file

COS30045 Data Visualisation
Joe Bloggs

Step 5: Refining the presentation

Unfortunately, the chart is not in the right place (i.e., it is below the footer). We can place the chart where we want by using

```
<p id="chart"></p>
```

and getting D3 to select "#chart" instead of "body". Fix so that the chart displays above the footer.

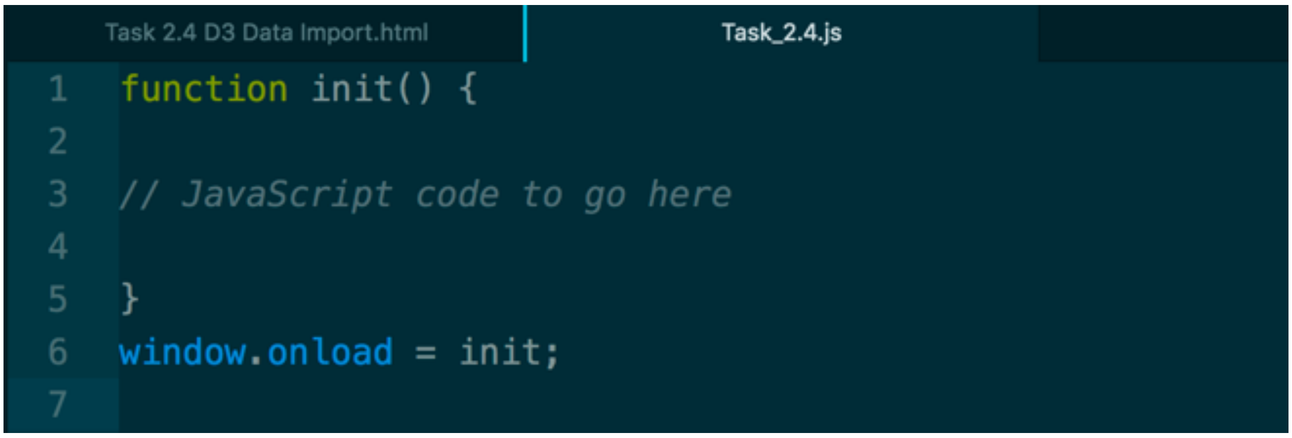
Finally, if you have done Web Application Design you will know that it is good practice to move your js script to a separate file. Submit your final code with the D3 script in a separate .js file.

We will be using this convention for the rest of our tasks.

Step 6: Creating separate JavaScript files

If you haven't done this before, here are some tips….

Create a new file on your text editor and save it as a .js file. Write a function to run when the window loads and add your chart code into it.

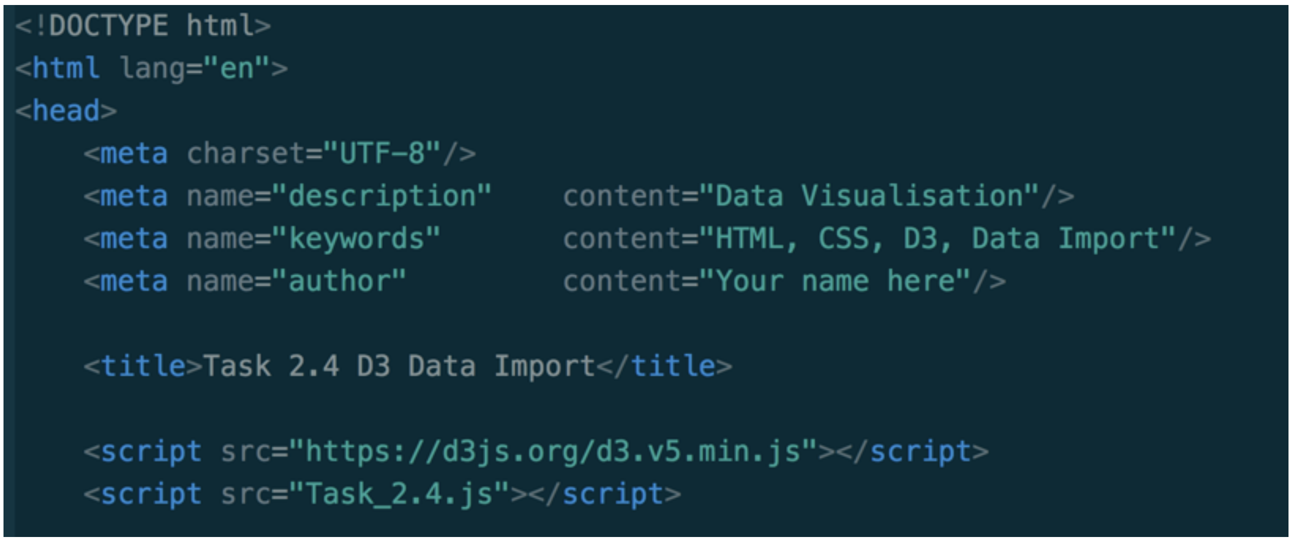


```
function init() {

// JavaScript code to go here

}
window.onload = init;
```

Don’t forget to call your new .js file in the header of your HTML file.



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

    <title>Task 2.4 D3 Data Import</title>

    <script src="https://d3js.org/d3.v5.min.js"></script>
    <script src="Task_2.4.js"></script>
```

Step 7: Add Error message to CSV import

Sometimes data does not get imported properly. Rather than letting the user sit and wonder what has happened, write an error message that will let the user know if the data has failed to load.