

COS30045 Data Visualisation

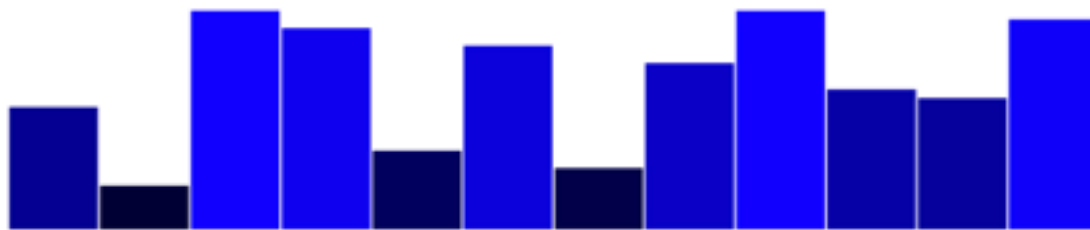
Task 2.4 D3 Importing data from CSV file

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:	<i>Textbook:</i> Murray Ch 5
To be marked as Complete your submission must:	Submit working code that meets the requirements specified in document below. Demonstrate appropriate use of HTML, CSS and D3. Properly formatted code Well commented code with references to code sourced from web, stack overflow etc. where appropriate. Demonstrate and explain code to tutor in class.
Submission	Submit to Doubtfire <ul style="list-style-type: none">• screenshot of final webpage• code Bring code to class to demonstrate to tutor

Overview

In this tutorial we will start using D3 to draw a bar charts. At the end of this Task 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:

Chart drawn from CSV file



COS30045 Data Visualisation
Joe Bloggs

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 Task 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 Task 2.2. 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 Task 2.2). 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., Task_2.4_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. In this case we want our data to behave like numbers so we need to tell D3 to treat them as numbers (i.e, parse them as floats). We can do this with the `parseFloat` function. There are other D3 functions available to parse your data into a number of different formats.

```
var dataset;  
  
var rowConverter = function(d) {  
  return {  
    Test: parseFloat(d.Test)  
  };  
}
```

To access the data from the file we need to use the column heading (e.g., `Test`). Hint: This is case sensitive, in this case, `test` will not work.

Step 4: Reading in the data

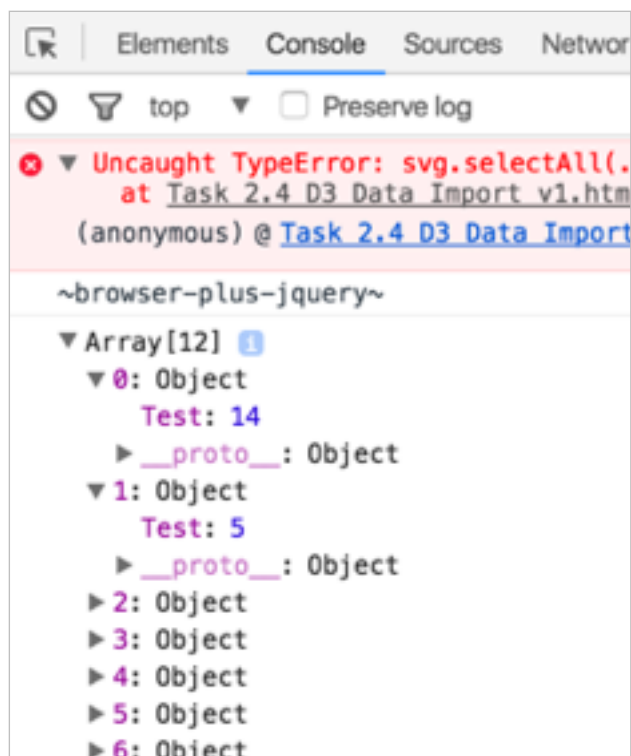
To read in the data we use the `d3.csv` function along with the parsing function we wrote earlier (i.e., `rowConverter`)

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

    dataset = data;
    console.log(dataset);

});
```

Add in a 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.



One way to give our chart code access to the data would be to cut and paste it into 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", rowConverter, function(data) {

    dataset = data;
    console.log(dataset);

    barChart(dataset);

});
```

```
function barChart() {

    //code goes here...

};
```

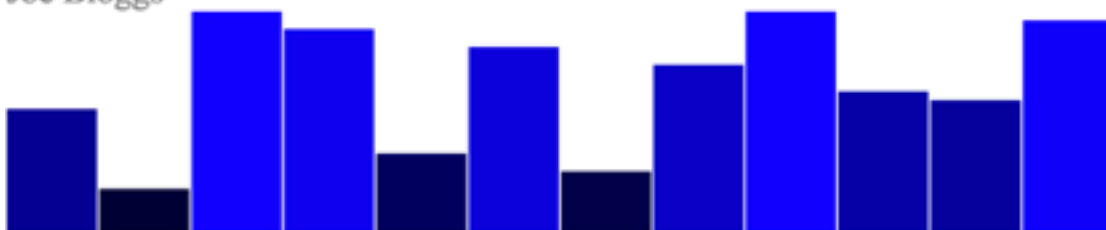
Cut and paste your chart code into the new function. Finally, you need to alter the way you reference the data points to tell D3 which column to get the data from.

```
.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

Chart drawn from CSV file

COS30045 Data Visualisation
Joe Bloggs



imported data.

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.

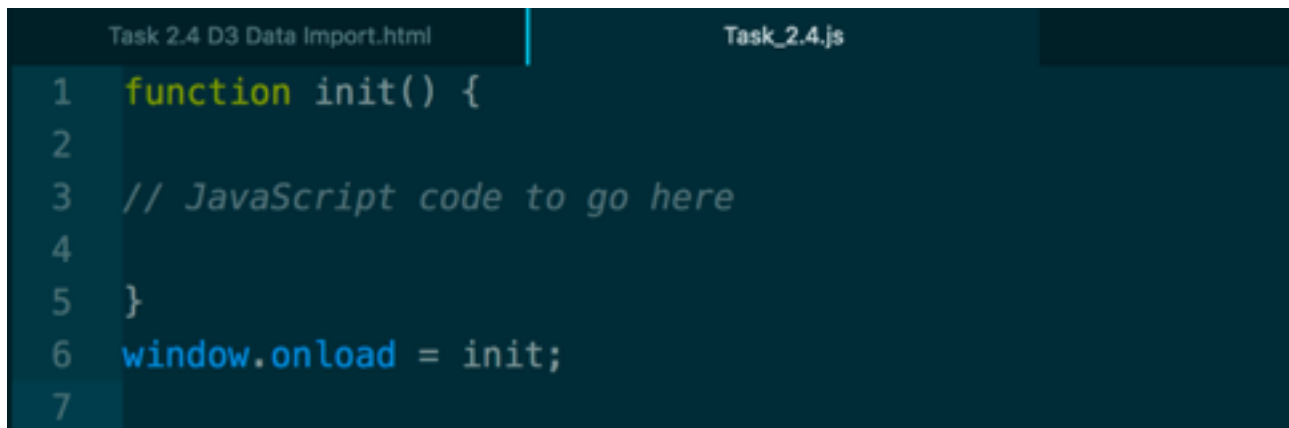
Finally, if you have done Web Application Design you will know that it is good practice to move your js script to a separate file. Submit your final code with the D3 script in a separate .js file.

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.



```
Task 2.4 D3 Data Import.html Task_2.4.js
1  function init() {
2
3  // JavaScript code to go here
4
5  }
6  window.onload = init;
7
```

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

```
<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.v4.min.js"></script>
  <script src="Task_2.4.js"></script>

</head>
```

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.