Computing and Web Development with Foundation

**Programme:** Computing & Web Development

**Module:** CPU5001

**Module Title:** Web Programming

**Assignment Title:** Web Based Data Driven Graphics

**Student Identification Number:** 1401501

**Tutor Name:** Martin Stanhope

**Date of Submission: Friday 6th January 2017**

Contents Page

[Abstract 4](#_Toc471204353)

[1. Introduction 4](#_Toc471204354)

[2. Data Capture 4](#_Toc471204355)

[3. Data Storage 5](#_Toc471204356)

[4. Data Conversion 6](#_Toc471204357)

[CSV to MySQL conversion 7](#_Toc471204358)

[Stage One 7](#_Toc471204359)

[Stage Two 7](#_Toc471204360)

[Stage Three 8](#_Toc471204361)

[MySQL to JSON Conversion 9](#_Toc471204362)

[Stage One 9](#_Toc471204363)

[Stage Two 9](#_Toc471204364)

[Stage Three 10](#_Toc471204365)

[Stage Four 10](#_Toc471204366)

[5. Scalable Vector Graphics (SVG) 11](#_Toc471204367)

[6. D3 Data Driven Documents 11](#_Toc471204368)

[D3 Basic Example One 12](#_Toc471204369)

[Stage One 12](#_Toc471204370)

[Stage Two 12](#_Toc471204371)

[D3 Basic Example Two 13](#_Toc471204372)

[Stage One 13](#_Toc471204373)

[Stage Two 14](#_Toc471204374)

[D3 Basic Bar Chart Example 15](#_Toc471204375)

[Stage One 15](#_Toc471204376)

[Stage Two 15](#_Toc471204377)

[Stage Three 16](#_Toc471204378)

[Final Result 17](#_Toc471204379)

[D3 Advanced Bar Chart Example 17](#_Toc471204380)

[Stage One 17](#_Toc471204381)

[Setting the scales 18](#_Toc471204382)

[Setting the Axis 18](#_Toc471204383)

[Creating the SVG element 19](#_Toc471204384)

[Data Acquisition 19](#_Toc471204385)

[Data Visualisation 21](#_Toc471204386)

[Adding the bars 23](#_Toc471204387)

[7. Evaluation 27](#_Toc471204388)

[8. Conclusion 27](#_Toc471204389)

[9. Appendices 27](#_Toc471204390)

# Abstract

This technical report details the skills and knowledge required to insert data into a MySql database followed by extracting the data into a JSON format while using a PHP script. D3 data driven documents library is explained with demonstrations provided, which explain how to begin developing with the D3 data driven document library.

# Introduction

A Manchester based company is currently researching into a system, which tracks the number of passengers entering and leaving a public transport vehicle. The vehicle in question is owned by a Californian company who provide guided tours around several tourist attractions.

An area of concern is the presentation of their findings; the system currently uses a data logging device, together with a camera, counting the number of passengers going both on and off the vehicle. This information is then sent back to the headquarters where it is prepared to be analysed and formatted into a useable source.

# Data Capture

Using a modern approach to data capture, a system of cameras is used to count the number of customers getting on and off the vehicle. This data logging equipment is also linked to a global positioning system (GPS) which gives the latitude and longitude coordinates showing which attraction the vehicle is currently stationary at.

The data which is captured from the combined system is automatically stored in a ‘comma separated value’ (CSV) file.

*“CSV is a simple file format used to store tabular data, such as a spreadsheet or database. Files in the CSV format can be imported to and exported from programs that store data in tables, such as Microsoft Excel or OpenOffice Calc. CSV stands for "comma-separated values".* (Computer Hope, n.d.)

An example of a simple csv file is shown in fig 1, this shows the source data which is automatically generated.

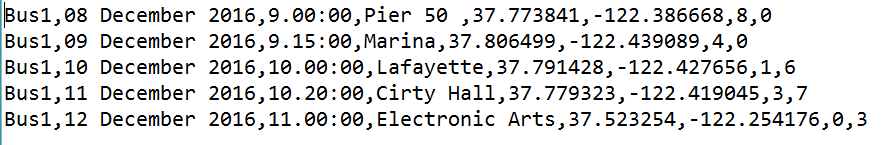


Figure - Example of a CSV file

The recorded data can be explained further by breaking down each value which has been recorded. As shown below in fig 2.



* Value 1 shows the bus identification number i.e. Bus1
* Value 2 shows the departure date i.e. 08 December 2016
* Value 3 shows the departure time i.e. 9.00:00
* Value 4 shows the departure location i.e. Pier 50
* Value 5 shows the Latitude coordinate of the departure location i.e. 37.773841
* Value 6 shows the longitude coordinate of the departure location i.e. -122.386668
* Value 7 shows the number of passengers entering the bus i.e. 8
* Value 8 shows the number of passengers leaving the bus i.e. 0

Figure - CSV file explanation

Upon data capture and conversion into a CSV format data can be processed by a number of applications such as d3.js. At the request of the organisation, it is essential this captured data is stored within a database to enable enabling future development and usage.

# Data Storage

A database is defined as being a *“collection of*[*information*](http://searchsqlserver.techtarget.com/definition/information)*that is organized so that it can easily be accessed, managed, and updated. In one view, databases can be classified according to types of content: bibliographic, full-text, numeric, and images.”* (TechTarget, n.d.)

There are many different types of database systems available to store data. A popular method is to use a relational database system which allows the user to query or interrogate the stored data. To access the data stored within the database a relational database management system (RDBMS) is required, this example uses the popular MySQL RDBMS as it is open source and free to use.

To administer the MySQL RDBMS, an application named phpMyAdmin is used as shown in figure 3.

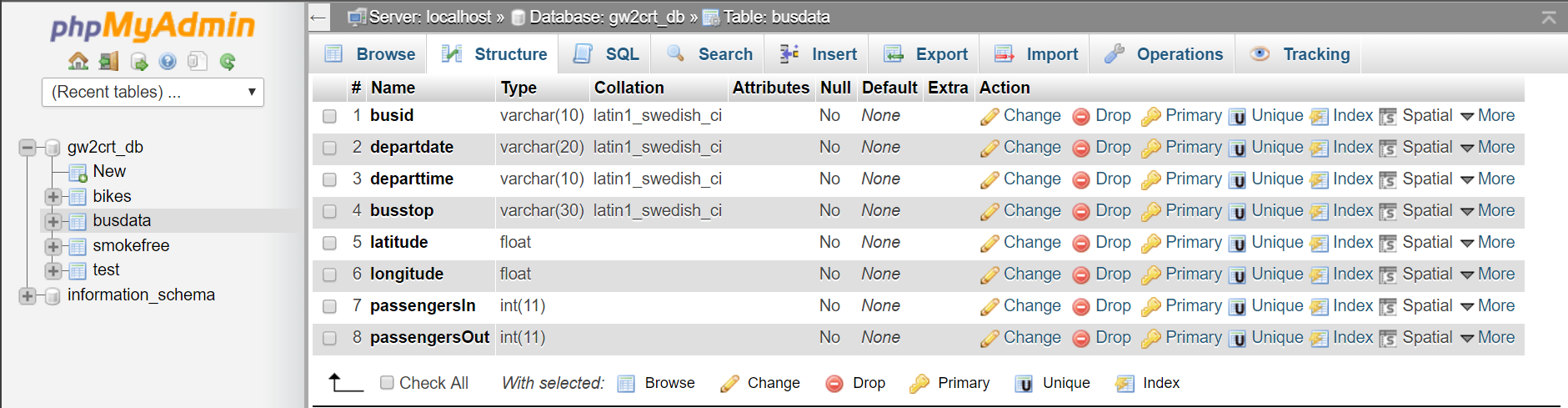


Figure – phpMyAdmin

The use of phpMyAdmin simplifies the four basic functions of persistent data storage which are Create, Read, Update and Delete (C.R.U.D). Creating the database structure is a simple procedure using the phpMyAdmin application.

The example shown in figure 3 shows the database structure, the design is based on the information provided in figure 2. The object busdata has been created with the corresponding attributes busid, departdate, departtime, busstop, latitude, longitude, passengerIn and passengerOut. Data from the CSV file can now be input into the correct attribute and stored correctly, this preparation enables the data to be converted from csv into the database automatically with the use of a PHP script.

# Data Conversion

Entering the data created from the data logging process could take many hours if done manually. Due to the amount of data which is logged daily, it is essential the data is converted automatically. This not only improves data processing speeds but also increases efficiency and data integrity.

The use of a Hypertext Pre-Processor (PHP) script is well suited to this role where communication between web bases applications are needed.

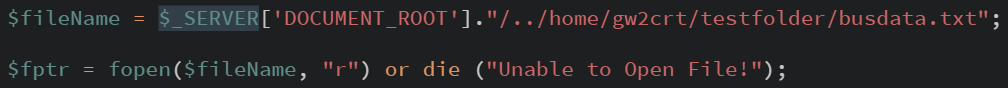
*“PHP is a general-purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites or elsewhere.” (PHP, n.d.)*

An overview of the PHP scripts used within this technical report is shown in appendix one and appendix in two, this will give a visual representation and help the end user to understand what has been achieved.

## CSV to MySQL conversion

The following breakdown explains the programming statements used to create the application, which converts the CSV into the MySQL database.

### Stage One

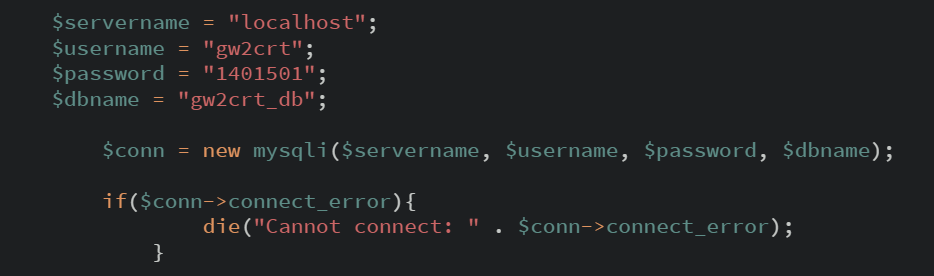


The variable “fileName” is assigned the text file that is found using the specified path name from the document root. This allows the variable to be called instead of using the full path name. Variable name “fptr” is assigned the built-in function fopen which opens the file situated at the address stored in the variable “fileName” followed by “r” which states the file is to be opened in a read only state.

*“$\_SERVER is an array containing information such as headers, paths, and script locations. The entries in this array are created by the web server.”* (php.net, n.d.)

*“***fopen()** binds a named resource, specified by **filename**, to a stream.” (php.net, n.d.)

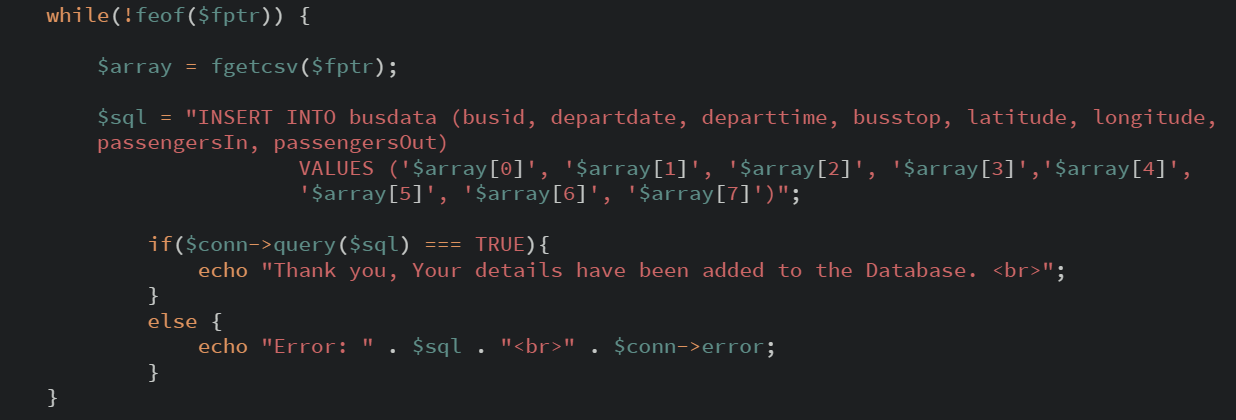
### Stage Two



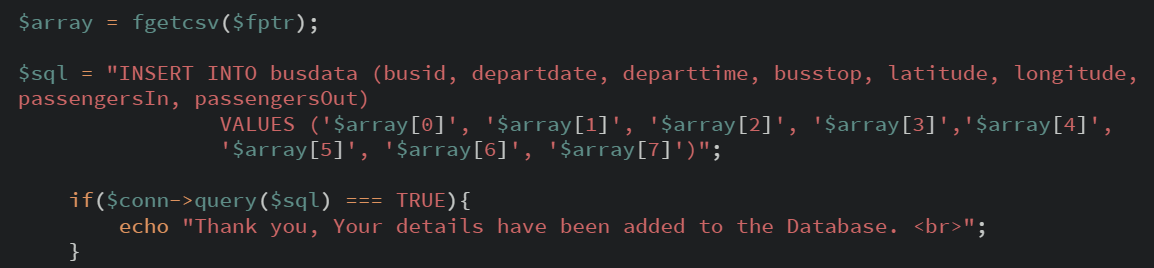
Stage two describes the connection method to the database, using the object orientated programming method. Each variable is assigned a value, which is called within the “conn” object. A conditional statement is used to check if the connection was successful or not returning a “cannot connect” error message if the connection fails.

“Error is the base class for all internal PHP errors” (php.net, n.d.)

### Stage Three



Stage three is an iteration condition which continues to perform the actions until the condition is met. The “While” statement is checking against the condition of not “feof” which is a built-in function looking for the end of file pointer, “$fptr” is called which opens the requested file for the iteration statement to check.



Once inside the iteration statement the variable named “$array” creates an associative array which is used to store the data from the CSV file. The built-in function “fgetcsv” pulls the comma separated values from the “$fptr” file and adds them line by line to the associative array.

“fgetcsv — Gets line from file pointer and parse for CSV fields” (php.net, n.d.)

The following variable $sql pushes the data from the array into the busdata database. Using an SQL INSERT command, the data from the associative array is to be input into the corresponding attribute within the database. A final check is run at the end of the ‘while’ iteration to check if the data within the associative array has been entered successfully into the database. The “Thank You, your details have been added to the Database” message is displayed when each record is entered. If the record fails to …. then an error message is produced.

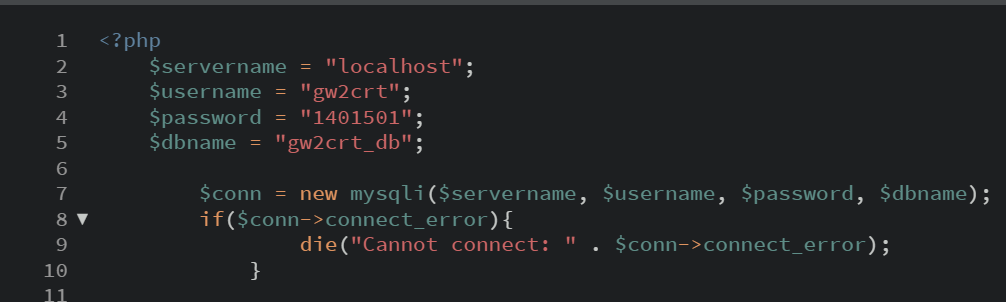
## MySQL to JSON Conversion

The second step to the process is to convert the data stored within the MySQL database into JavaScript Object Notation (JSON) format.

“JSON(JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999.” (json.org, n.d.)

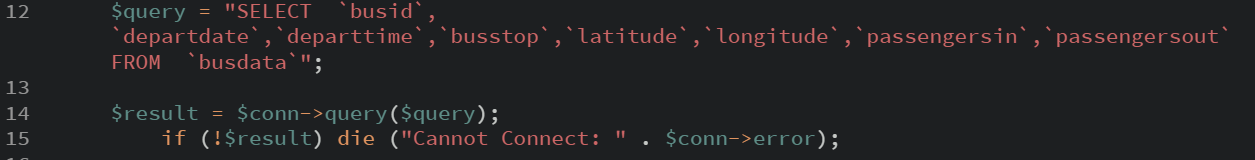
This process is completed to enable the data to be processed and consumed by a third-party application giving visually appealing results.

### Stage One



A connection to the database is created, this is identical to the method used within the CSV to MySQL conversion.

### Stage Two



Using a variable named “$query” an SQL statement is formed to retrieve the data from the database. The example shows the SELECT statement has been used to select all the fields from the busdata database. The “$result” variable stores the query results, a conditional check is also performed to check if the query was successful, if it was not an error message is displayed.

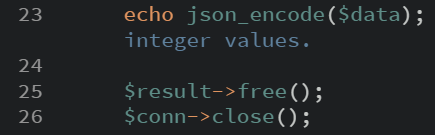
### Stage Three



The resulting query is passed to an associative array which is created using the same method shown previously, again, an iteration statement is used to loop through each record testing if there is an array of values from the SQL query which was previously performed. Each row is fetched in turn and stored as a record in the “$data” associative array using the PHP function MYSQLI\_ASSOC.

*“mysqli\_result::fetch\_array -- mysqli\_fetch\_array — Fetch a result row as an associative, a numeric array, or both“* (php.net, n.d.)

### Stage Four



Finally, the associative array is passed to the built-in function called json\_encode. This function converts the array into a valid JSON format that enabling use with a third-party application.

*“json\_encode — Returns the JSON representation of a value”* (php.net, n.d.)

The final lines of code free the memory and clear the “$result” variable with the free() function together with the close() function which closes the connection.

This final step produces a string containing the JSON representation of the associative array value as which can be seen in appendix three. The JSON output can now be used with a scalable vector graphics application to create a visual representation of the logged and processed data.

# Scalable Vector Graphics (SVG)

Scalable Vector Graphics (SVG) is an application of the eXtensible Mark-up Language (XML) which makes it achievable to represent graphic data in a compact and portable manner. (David Eisenberg & Bellamy-Royds, 2014)

There is colossal number of image formats which are used within the Print and Digital Media Industries. One such image format used within the digital media industry is the ‘vector’ format.

“Vector graphics is the creation of digital images through a sequence of commands or mathematical statements that place lines and shapes in a given two-dimensional or three-dimensional space. In physics, a vector is a representation of both a quantity and a direction at the same time.” (TechTarget, n.d.)

A vector graphic is useful as it allows the image to be manipulated without distortion as shown in figure 4, the graphic is represented through geometric shapes at specified coordinates and when compared to a ‘Bitmap’ graphic it is clear to see that there is very little degradation of the image when it is magnified.



Figure - Vector example (Schwartz & Solis, 2012)

This data is stored as a plain text which is a key advantage of the XML language. This stored data is both human and machine-readable which is simple to create and useable by several various platforms giving great versatility and adaptability. An example of an SVG image written in XML is shown in appendix four.

Making use of the SVG format is the D3.js or D3 Data Driven Documents JavaScript library. Using JavaScript programming, JSON and SVG stunning data drive visuals can be created.

# D3 Data Driven Documents

D3 delivers information in a visual way, this method of delivery is a powerful tool in the communication of results to any audience removing the need to interpret data from rows and columns. Data values are mapped to visuals, which is a process of information mapping. This process gives an insight to trends, patterns that would otherwise be difficult to perceive.

To create a visual, D3 takes the users data which is converted from source data into a JSON format, as described in the data conversion section of this technical report. This data is processed using the D3 framework which creates the visual output by binding the data and SVG elements to the web browsers ‘Document Object Model’ (DOM). The D3 library does not follow a template-based model, such as the Microsoft Office package. Instead, the library works by manipulating web documents with data, transforming various elements in response to the user input. The following example will describe the creation of a simple web page using the D3 library.

## D3 Basic Example One

This demonstration shows how the D3 library uses the DOM to add elements to a webpage. This simple explanation shows the structure on which more advanced features are developed.

Using codepen, an online development tool, to develop a test website using the D3 JavaScript library. To start the example a blank html document is created with a link to the JavaScript library. A DIV element allows D3 to target it as shown in figure 5.

### Stage One



Figure - HTML code Basic Example

### Stage Two

D3 has an element to target and manipulate, JavaScript adds two elements to the web page. The first is a heading one (h1) element. The second is a paragraph (p) element. Both elements contain text processed by the D3 library and added to each element.

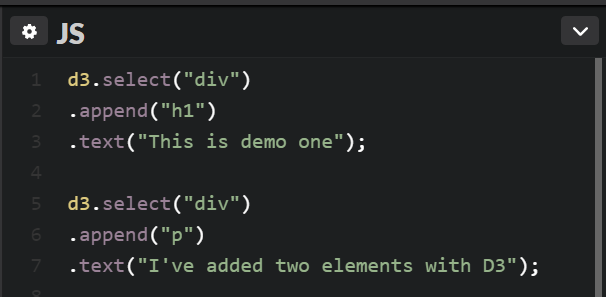


Figure - Javascript Code Example

The example shows several lines of code, the first section is a collection of commands using a period to string them together. This method of joining commands is called chaining or chain syntax, which executes several lines of code in one action.

Using the method d3.select(), we can see the div is targeted, .append is chain linked to the d3.select() method. This creates an h1 element and appends this element inside the selected div.

The addition of the .text method to the chain adds a string of text in between the opening and closing h1 element. Due to the chain syntax, which has been used for this example, the code follows its normal flow process and executes this as a block as shown in figure 6.

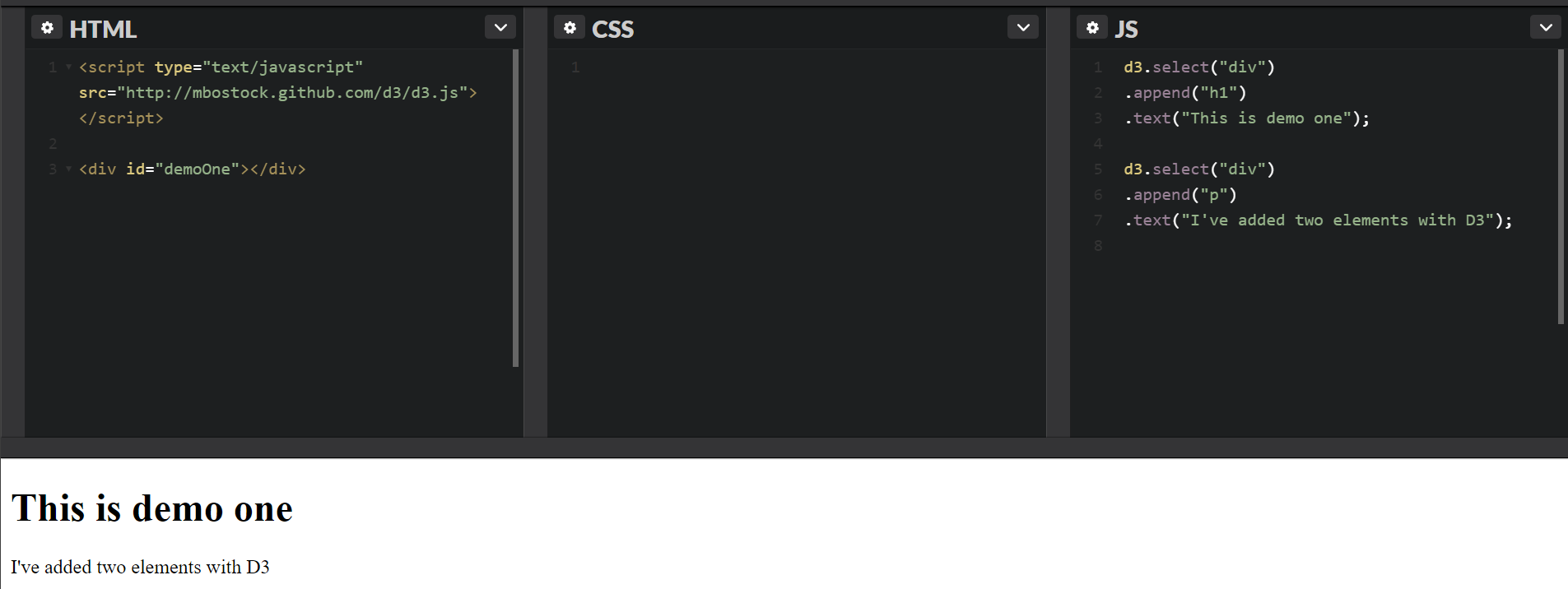


Figure - Final Code Example

## D3 Basic Example Two

This second demonstration shows how d3 is used to create a single circle. Using codepen to display the results.

### Stage One

Create a blank HTML document and create a link to the D3.js library using the script tags. Open a blank div with an identification to enable JavaScript to append.



Figure - Example 2 HTML Code

### Stage Two

Using JavaScript and the D3.js library, we can now manipulate the Document Object Model to create a circle within the created div.

Create a variable that selects the div using the .select() function. Using the .append() function, we are telling the JavaScript to add an SVG element. Next, we set the size of the objects container using the .attr() function.



Figure - Example 9 JavaScript Code

Finally, a call is made to the variable demoSVG which creates and appends the following code to the div. This method contains the styling of the circle built into the JavaScript code as opposed to using a separate CSS file for the styling. Using the .attr() function the circle radius, x and y axis are set.



Figure - JavaScript Code Example

Below is the circle which is created using HTML, JavaScript with the D3 library.

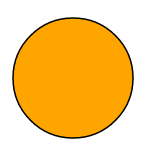


Figure - Circle created using D3.js

This demonstration creates a single orange circle with a black stroke using the functions within the D3.js library. D3 works by simply manipulating the document object model using JavaScript to create an SVG image with its own attributes, which are defined within the JavaScript code.

## D3 Basic Bar Chart Example

To demonstrate the simplicity of D3.js the following example creates a bar chart using codepen.

### Stage One

* Create the HTML document
* Link the D3.js library
* Create an empty div



Figure - Barchart HTML code

### Stage Two

Add the styling using CSS

* Add a font style
* Add a background colour
* Set the text alignment
* Add some padding
* Add the margins
* Add some colour

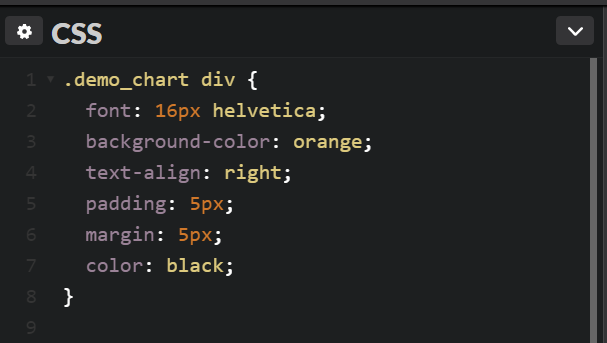


Figure - Bar Chart CSS Code

### Stage Three

Creating the JavaScript code is the most important part of the d3.js framework this binds the data to a visual.

* Create an array which contains the data to be used
* Set the data transformation quantitative scale using a built in d3 function
* Select the div using its class name as the target output
* The Selectall() function targets the empty div element based upon the parent for each data point
* Invoke the data to create the bind between the data and the visual
* .enter().append(“div”) appends the data to a child div
* Set the width of each bar which is a multiple of the associated data
* Add the associated data text labels

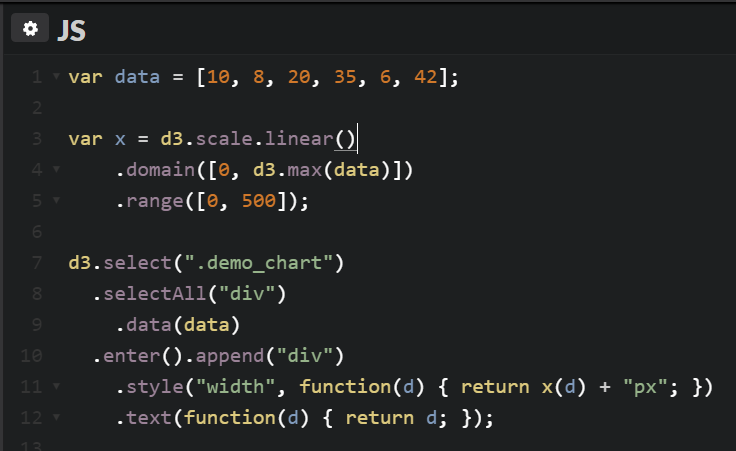


Figure - Bar Chart JavaScript Code

Final Result

From the collection of stages, the results are shown below. This example shows a simple bar chart showcasing a small selection of tools D3.js has to offer.

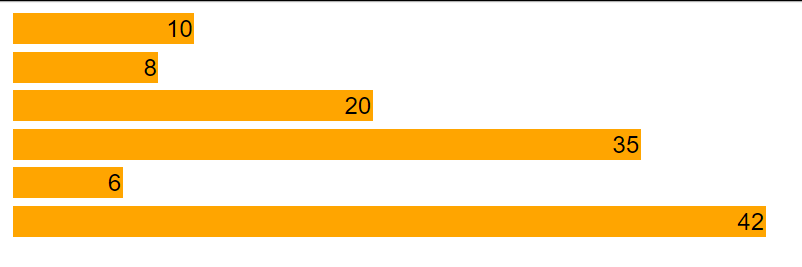


Figure - Bar Chart Result

## D3 Advanced Bar Chart Example

This demonstration based on the data produced in direct relation to the research data provided by the Manchester based data logging company within this technical report.

This data has been processed and converted to a JSON format as explained in Section 4 entitled ‘MYSQL to JSON Conversion’. The HTML document is broken down and explained in several stages.

## Stage One

Declaration of the Variable used to create the bar chart.

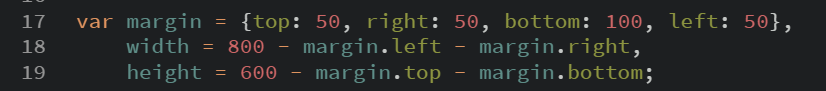


Figure - Advanced Example Margin

The variable name margin sets the margins around the SVG image

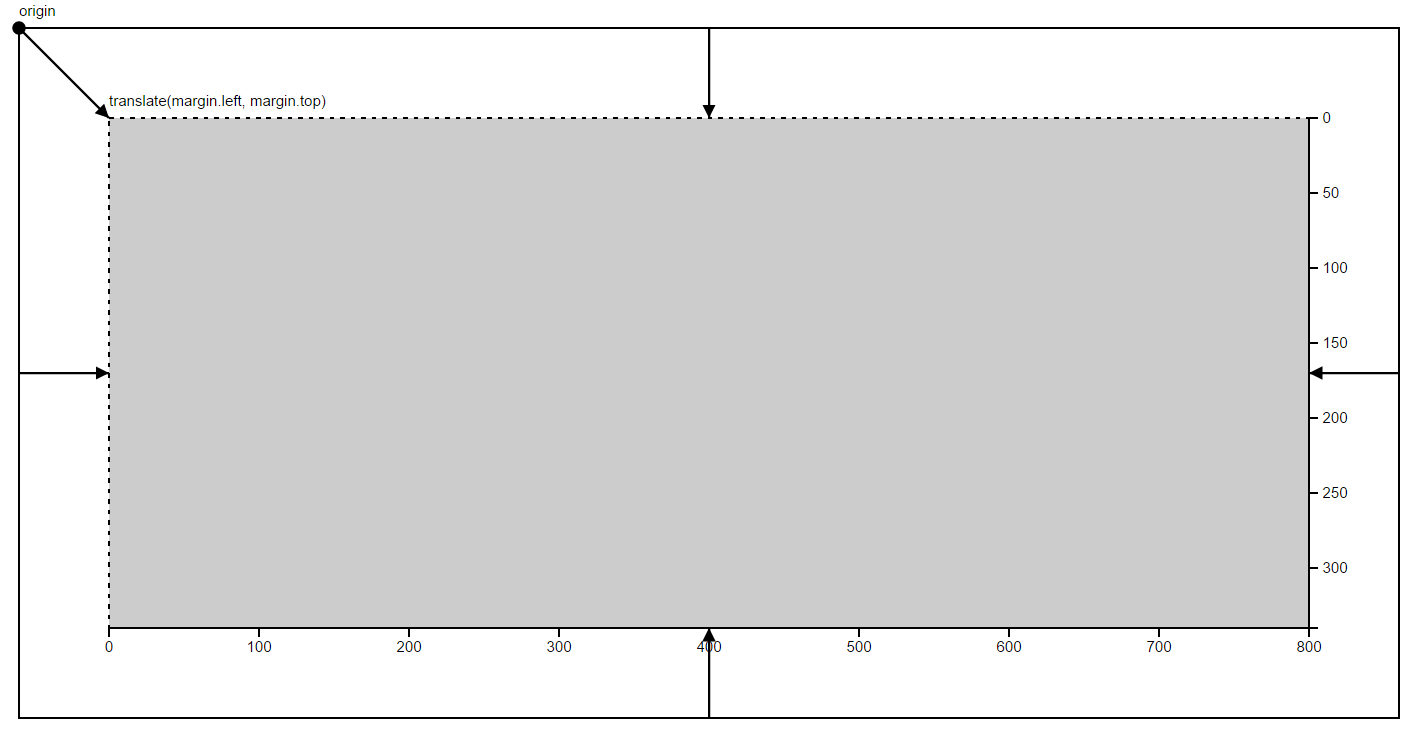


Figure - Advanced Example Margin Explanation

### Setting the scales

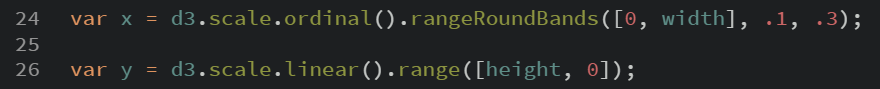


Figure - Advanced Example Scales

Variable X is used to create an ordinal scale with the specified range intervals and padding between visual bands for the X axis using a built-in function named scale. This scale function maps the data from an input domain to an output range.

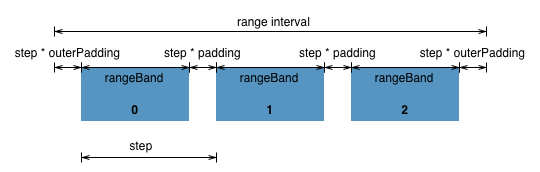


Figure - Advanced Example Scale Ranges (d3noob, 2016)

Variable Y constructs a continuous scale map using the domain and range for the Y axis using a built in function named scale().

These scales calculate the corresponding pixels within the SVG image to be processed, this is dependent on the values that have been input upon creation (Murray, 2015).

### Setting the Axis

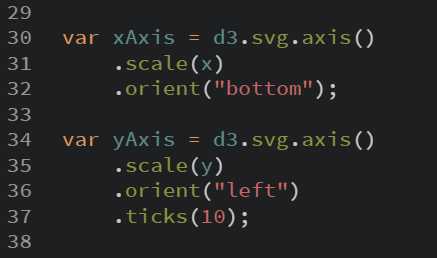


Figure - Advanced Example Axis

The x and y axis is defined to give a textual representation to the graph within the data visualisation.

Variable xAxis is used to store a function call to d3.svg.axis. This sets the X axis using the scale function with the argument stored within variable x (see “setting the scales”) together with its label position which is set to “bottom” showing the labels below the axis line. The yAxis variable is works in the same way with the orientation or label position set to left and ticks set to the value of 10. The .ticks() function is called to display information on the data values which are displayed. Using the ‘ticks’ method is subjective, the methods within the d3 library will automatically overwrite this value automatically to a value which makes more sense to data graphic produced.

### Creating the SVG element

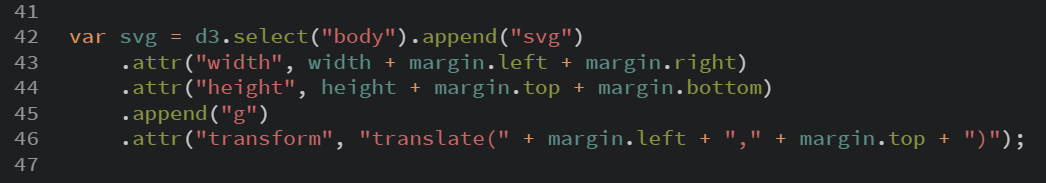


Figure - Advanced Example SVG element

The previously declared variables are called within this block of code to create a placeholder for the SVG element. Because there isn’t an SVG element within the HTML body the first line of code is selecting the body element to append an SVG element using the d3.select() and .append() functions. This targets the body of the HTML document and appends an SVG element using the “body” and “svg” argument.

Both width and height is set by passing the .attr() function the relevant variable adding to it the values stored with its margin properties. The .attr() function is used to set an HTML attribute and its value on an element. (Murray, 2013)

Appending a “g” element is used to group SVG shapes together, once grouped this allows for the collection of shapes to be transformed as if it was a single shape. Linked to this is the .attr() function, this applies control of the created group of SVG shapes. (Jenkov, 2015)

### Data Acquisition

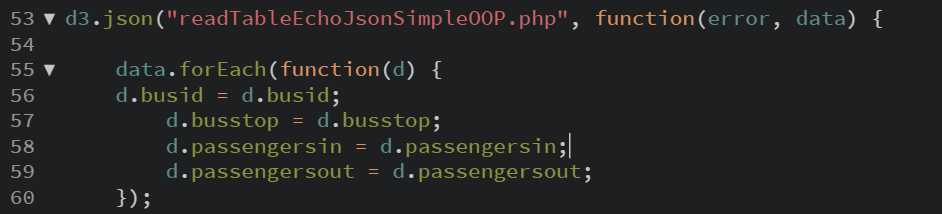


Figure - Advanced Example Data Acquisition

In previous examples the use of data has been attained from a manually declared array, this example shows how the data can be retrieved using a PHP script. Section 4, Data Conversion aids the description of this block of code.

D3 accepts several various ways to bind data to the visualisations. One such way is to use a JSON formatted dataset, which is an example of an associative array. Using the d3.json() function with the argument readTableEchoJsonSimpleOOP.php a function call executes the PHP script which converts the data stored in a MySQL database into JSON format. The JSON format encapsulates data into a name-value pair which is known as a JavaScript Object Literal as shown in figure 23 below (Dynamic Web Coding, n.d.).



Figure - Advanced Example Object Literal

Upon retrieval name-value object literals are stored within the console ready to be called by the data() function which counts and parses the values.

The first line used within this data() function is simply an anonymous function which creates a placeholder (d) for the data to be held. Using a forEach() iteration loop function, each object literal is retrieved as an associated name-value pair dataset.

Using the placeholder (d) stringed together with a variable name creates an object literal variable, storing the dataset as a name-value pair array. This can later be called during the creation of the data visualisation using the object literal name-value pair ie busstop or passengersin. This method of data binding is later used to create the d3 visualisation by mapping the data to various elements within the document object model.

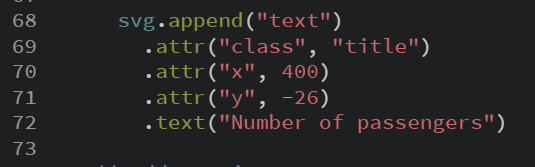
### Data Visualisation

Following the data binding process, a visualisation is created.

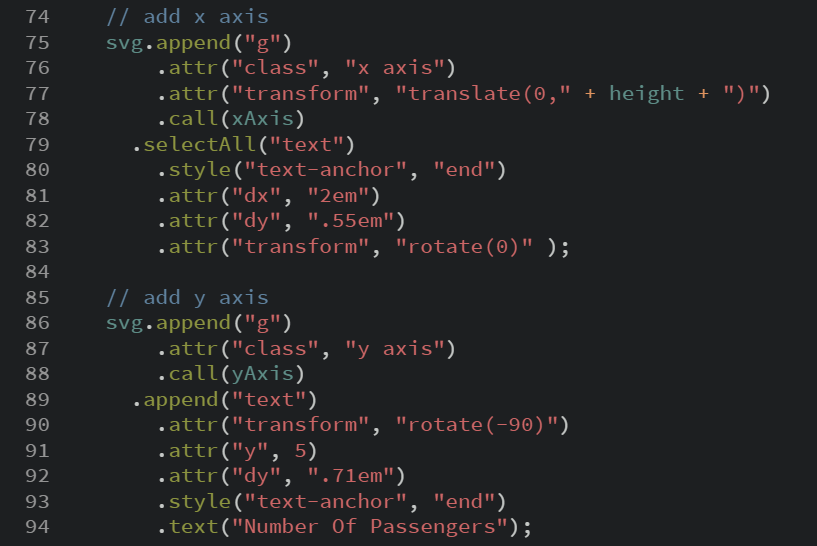




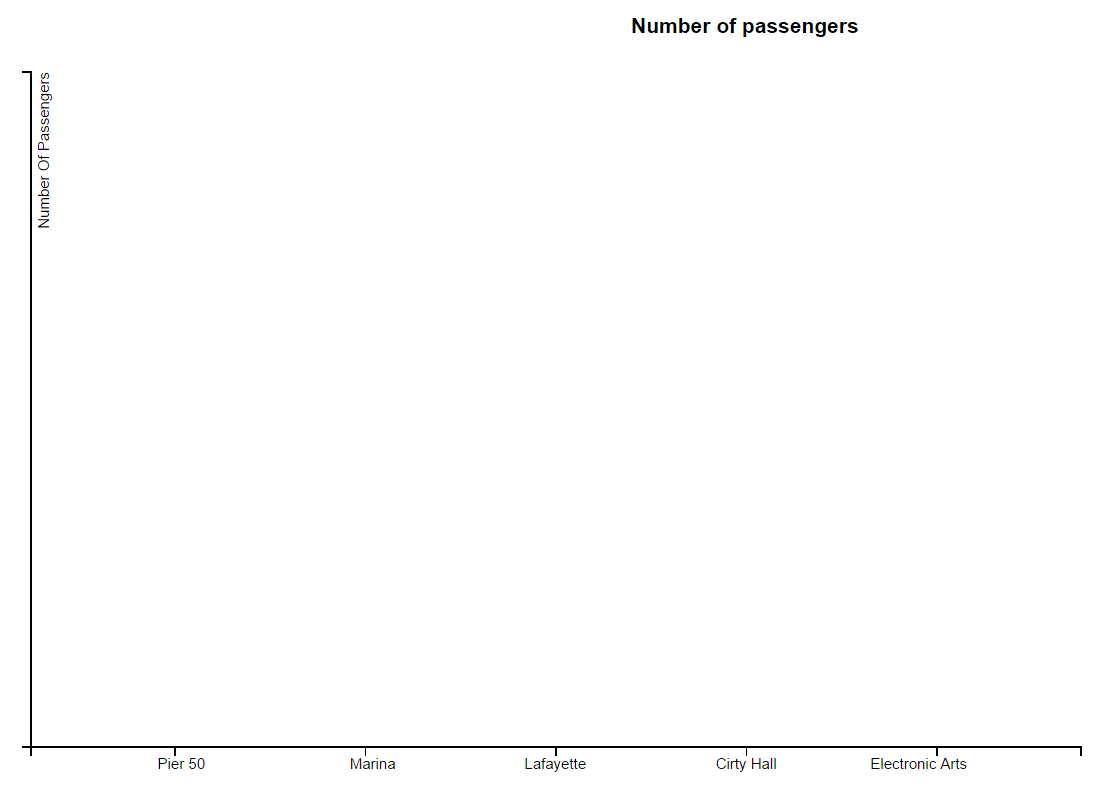
Initially setting the X and Y axis scale domain, this maps the d.busstop and d.passengerIn data to the graph. The domain is the range of possible input data values. The X axis uses an anonymous function to return the d.bustop data. The Y axis uses a .max() function against the data to calculate its values. An anonymous function is again called to label the values with the data held in the d.passengersIn variable.



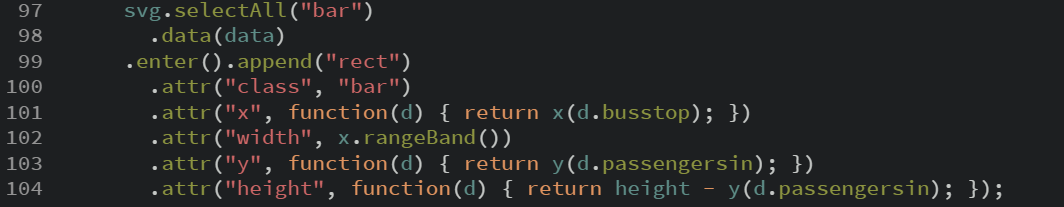
This block of code simply appends a text block to the corresponding x and y coordinates of the SVG image. Using a .text() function a string of text is appended to be used as a chart title.



Following on from the append() function, this block of code appends to the SVG image using the “g” element grouping each element together. Styling and position Is set for the X and Y axis together with the padding, margins, axis title positioning, tick and value label information relating to the data visualisation.



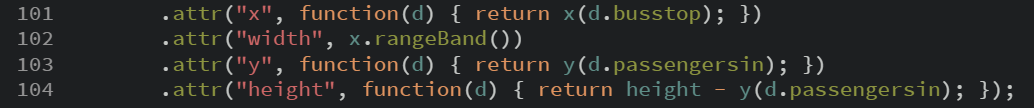
### Adding the bars



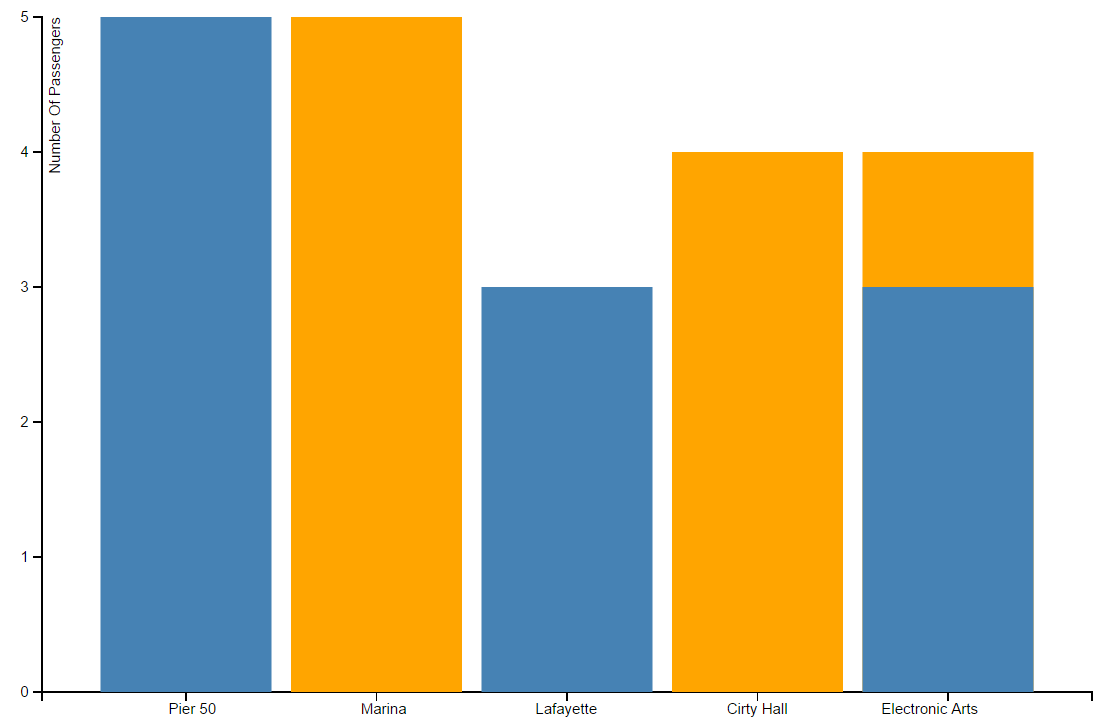
Creating the bars is the next block of code to be executed, this block uses many functions to display the data which was consumed by the d3.json() function.

Using a selectall() function with the argument SVG the function selects the SVG elements which at this point are non-existent. The .data() function loops through the passed argument which binds the data to the SVG elements creating the visual.

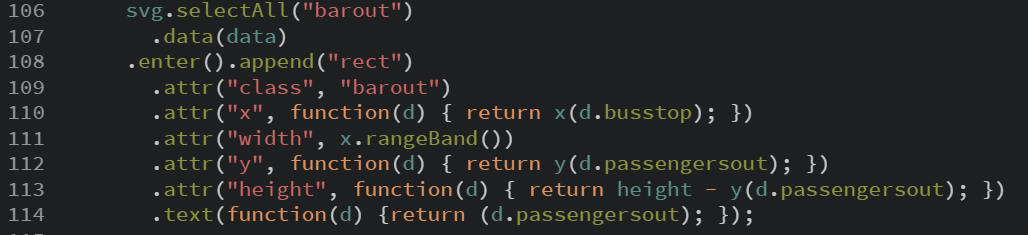
The .enter().append(“rect”) function works directly with the document object model, this method looks at the DOM selection, in this case bar, to see if there is an element in existence. If there is not a placeholder is created and passes a reference to the next line of code which calls the CSS styling properties.



This part of the function describes dynamic creation of the visualisation. Using the .attr() function an argument followed by an anonymous function(d) which returns the values of d.bustop and passengerin together with the relevant variable containing the scaling data. The data is then passed off to the following function that contains the values of width and height and x.rangeband. Height of the visualisation is set using the height minus the y value. Shown below is the results of the above code.



The same method is used to create the passengerOut data; however, a CSS styling class of barout is used to change the colour to orange giving a more distinguishing presence.



# Conclusion

The Use of the D3.js library with JavaScript provides an incredible way to display data in visual way. Providing a vast array of tools, a developer can use to produce stunning interactive and scalable visuals from a range of datasets.

This report has covered the technical aspects in manipulating the DOM to create a simple header to bar charts; however, the D3 library can create interactive maps using Geo Mapping through the GeoJSON encoding method. This provides a visual way to track the use of data within a specific area. Virtually any dataset has the potential to be manipulated and bound to a visual representation. This is not only useful for large presentations and conferences, but within company reports and websites. Combing D3 with data allows the predication of trends, patterns and associations, which in turn promotes growth within businesses having a positive impact on the economy and further project funding.

# Appendices

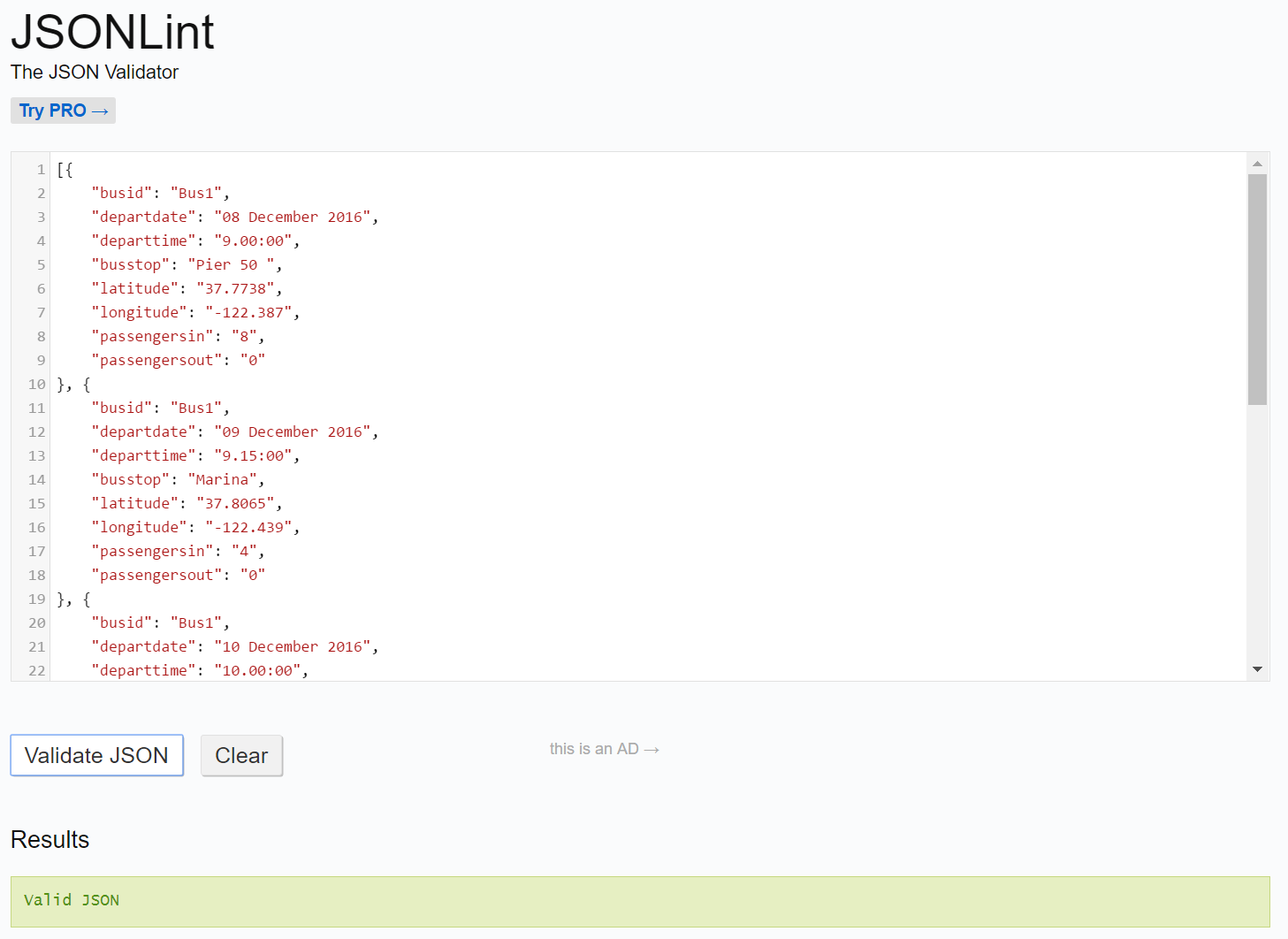
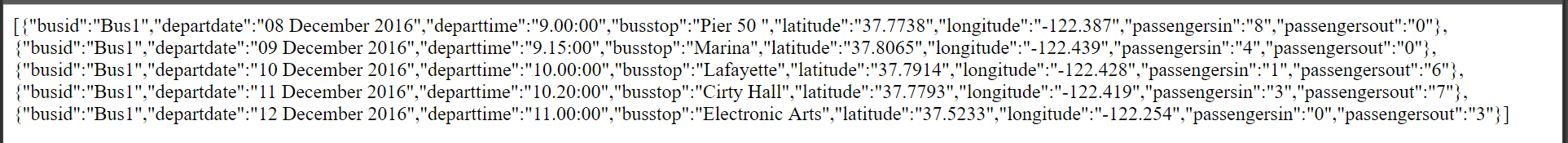
Appendix One - PHP Script Overview



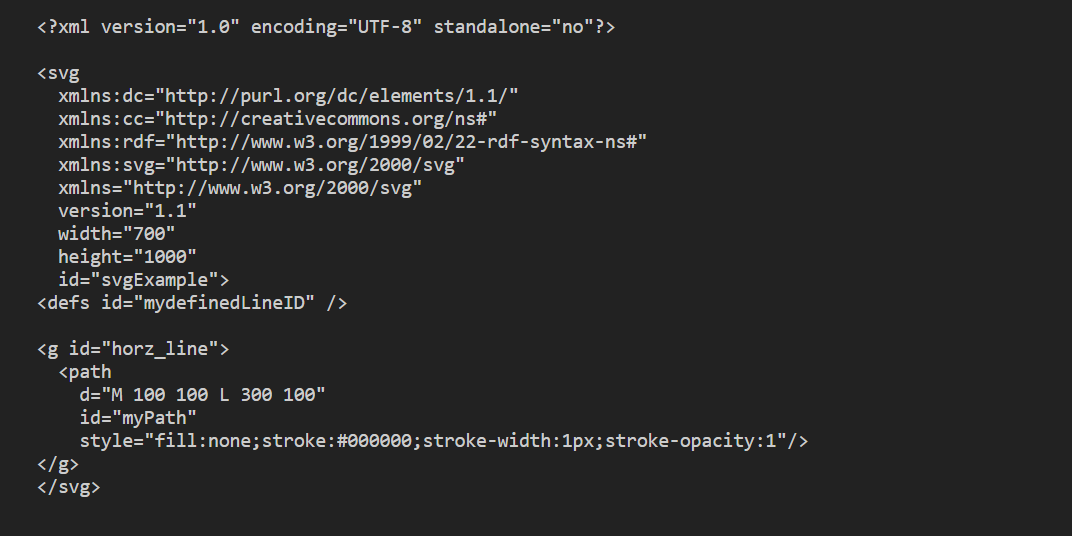
Appendix Two - PHP Script Overview



Appendix Three – Example of JSON format



Appendix Four – Example of an SVG element using XML





# Bibliography

BSUPPORT, n.d. *What is a CSV file and how do I save my spreadsheet as one?.* [Online]   
Available at: https://support.bigcommerce.com/articles/Public/What-is-a-CSV-file-and-how-do-I-save-my-spreadsheet-as-one  
[Accessed 19th December 2016].

Computer Hope, n.d. *How to Create a CSV File.* [Online]   
Available at: http://www.computerhope.com/issues/ch001356.htm  
[Accessed 19th December 2016].

d3noob, 2016. *Simple d3.js Graph.* [Online]   
Available at: http://bl.ocks.org/d3noob/b3ff6ae1c120eea654b5  
[Accessed 22nd December 2016].

David Eisenberg & Bellamy-Royds, A., 2014. *SVG Essentials.* 2nd Edition ed. Milton Keynes: O'Reilly Media.

Dynamic Web Coding, n.d. *JavaScript Object Literal.* [Online]   
Available at: http://www.dyn-web.com/tutorials/object-literal/  
[Accessed 3rd January 2017].

Jenkov, J., 2015. *SVG g element.* [Online]   
Available at: http://tutorials.jenkov.com/svg/g-element.html  
[Accessed 28th December 2016].

json.org, n.d. *Introducing JSON.* [Online]   
Available at: http://www.json.org/  
[Accessed 20th December 2016].

Murray, S., 2013. Setting Attributes. In: M. Blanchette, ed. *Interactive Data Vizulization.* Calafornia: O'Reilly Media, p. 86.

Murray, S., 2015. *Scales.* [Online]   
Available at: http://alignedleft.com/tutorials/d3/scales  
[Accessed 27th December 2016].

php.net, n.d. *$\_Server.* [Online]   
Available at: http://php.net/manual/en/reserved.variables.server.php  
[Accessed 20th December 2016].

php.net, n.d. *Error.* [Online]   
Available at: http://php.net/manual/en/class.error.php  
[Accessed 20th December 2016].

php.net, n.d. *fgetcsv.* [Online]   
Available at: http://php.net/manual/en/function.fgetcsv.php  
[Accessed 20th December 2016].

php.net, n.d. *fopen.* [Online]   
Available at: http://php.net/manual/en/function.fopen.php  
[Accessed 20th December 2016].

php.net, n.d. *json\_encode.* [Online]   
Available at: http://php.net/manual/en/function.json-encode.php  
[Accessed 20th December 2016].

php.net, n.d. *mysqli.* [Online]   
Available at: http://php.net/manual/en/function.mysql-fetch-array.php  
[Accessed 21th December 2016].

PHP, n.d. *What is PHP.* [Online]   
Available at: http://php.net/manual/en/intro-whatis.php  
[Accessed 19th December 2016].

Schwartz, A. & Solis, R., 2012. *D3 for Beginners.* [Online]   
Available at: http://website.education.wisc.edu/~swu28/d3t/baseline.html  
[Accessed 22nd December 2016].

Schwartz, A. & Solis, R., 2012. *D3 for Beginners.* [Online]   
Available at: http://website.education.wisc.edu/~swu28/d3t/images/VectorBitmapExample.png  
[Accessed 22nd December 2016].

TechTarget, n.d. *database.* [Online]   
Available at: http://searchsqlserver.techtarget.com/definition/database  
[Accessed 19th December 2016].

TechTarget, n.d. *vector graphics.* [Online]   
Available at: http://searchwindevelopment.techtarget.com/definition/vector-graphics  
[Accessed 21st December 2016].