# 

# **Using D3 to Create a Bar Graph: A Guide for Web Designers and Students**

By Kevin Zheng

# 

# 

# 

# 

# 

# 

# 

# 

# 

**Table of Contents**

[**Introduction**](#_2i410vnql9nu)**……………………………………………………………………..2**

**Installation……………………………………………………………………….2**

**Setup Work Environment……………………………………………………..3**

**Data Binding…………………………………………………………………….4**

**SVG……………………………………………………………………………….7**

**The SVG Element………………………………………………………………7**

**Rectangle………………………………………………………………………..8**

**Drawing…………………………………………………………………………..9**

**The FInal Product……………………………………………………………..15**

**Conclusion……………………………………………………………………..16**

# 

# 

# 

# **Introduction**

Welcome to this manual on using D3 to create a bar graph. This manual is intended to help expert web designers learn new skills and techniques for producing interactive data in web browsers. It is also intended to help engineering, statistics, and business students to efficiently customized and communicate their data to improve the understanding of their argument. This manual assumes that the reader has previous knowledge of basic HTML, CSS, and JavaScript

D3 is a JavaScript library that formats large amount of raw data from inefficient communication to efficient communication. Excess amounts of information are overwhelming, and they are often hard to understand. It is much easier to create a graph and use it to represent your data. For that reason, data visualization is a powerful method. Data visualization is the process of mapping information to visuals graphs. Visualizing data is the fastest way to communicate it to others, by using charts, colors, and animation to represent large amount of raw data in a more easier to understand visual form.

In this manual, we will take a large sample of raw data and use JavaScript, HTML, CSS, and D3 to format it into a bar graph on a web browser. A bar graph is useful for comparing things between different groups or to track changes over time.

## **Installation**

There are three files that will need to be download in order for this project to work: the D3 library, CSS bootstrap, and the data file for this project.

The CSS bootstrap will be use to compile the CSS and JavaScript source code to make it work in the web browser and to import all the CSS functions from the bootstrap file.

To download and find the CSS bootstrap:

1. Go to the website: <https://getbootstrap.com/docs/4.1/getting-started/download/> and press the first download button. It will download a zip folder
2. Unzip the folder.
3. The file that that we want is located inside the css folder. Go there and find a file with the name bootstrap.min.css.
4. Remember the location of that file and leave it there for now, it will be shown where to be place later.

To download the D3 JavaScript library:

1. Go to the official website:<https://d3js.org>.
2. Scroll down until you see the download section with the link d3.zip.
3. Click on the link and it will download a zip folder.
4. Unzip the d3.zip folder.
5. There will be multiple files, focus on d3.min.js.
6. Remember the location of that file and leave it there for now, it will be show where to be place later.

The data file will be provided on my GitHub page: @42kevinzheng. It will be explained later on in the manual.

**Setup Work Environment**

This section will help you set up a work environment so that you can follow along with the editing and example of this project. The text editor that you will use for this project is Notepad++ and the web browser is Microsoft Edge.

1. To begin our project, start by creating a new folder for your project. Call it whatever you like. (I will call my folder Project-Example).
2. Inside Project-Example, create three more folder and call them css, js, and data.
3. Then place the latest version of d3.min.js into the js folder.
4. Inside the js folder, create a file call main.js. This is where you’ll place all the d3 code and JavaScript code.
5. Next, for the css folder, place the bootstrap.min.css file in the folder.
6. Inside the css folder, create another file called style.css. This is where you’ll place all the css code.
7. For the data folder place the data file that was downloaded from my GitHub.
8. Finally, create a index.html file. This is where the main program will be run and where the bar graph will be draw.

A template of the HTML is shown below. Go and copy all the code into your index.html. Once you copy all the code into your index.html, you’re done. You’ve finished your index.html and the HTML section. You will never touch the HTML or the index.html in this project ever, again.



(Figure 4-1)

Here are a few things to note about this template:

* The meta tag is set as utf-8, to ensure that the browser can use all the D3 functions and the data properly.
* The first link tag points to the CSS bootstrap, to import all the CSS functions.
* The second link tag points to the style.css file, where you’ll customize the css code.
* All the div tags holds specific area on where the bar graph will be placed. Typically, the div tag is used to layout the web page by grouping similar elements together. In this case, our div tags will make sure that the bar graph is be located in the middle of the browser, because it is easier to view it that way.
* The first script tag, points to the d3.min.js file, to import all the d3 functions.
* The second script tag points to our customize JavaScript code. Right now, main.js is empty.

**Data Binding**

The next concept you’ll need to learn is data binding: this is the process that will populate or manipulate the DOM elements. With D3, we bind our data input values to elements in the DOM. Binding is like “attaching” or associating data to specific elements, so that later you can reference those values to apply mapping rules. Without the binding step, we have a bunch of data-less, unmappable DOM elements.

In this project the data file we will be using is formatted as a JavaScript Object Notation (JSON) file. JSON is a specific syntax for organizing data as JavaScript objects. The syntax is optimized for use with JavaScript. D3 works well with it, because the data format is predictable and it is easy to figure out what the next value will be. For example, a picture of our JSON file is shown below:



(Figure 6-1)

In (Figure 6-1) it shows the months, revenue, and profit. The square bracket [] is an array that holds objects. Each curly bracket {} is a different object with key/value pairs. For example, the third object is the third element in the array. It has three keys and three values. The first key is month and the first value is March. The second key is revenue and the second value is 16443. The third key is profit and the third value is 15423. In this project, we will be mapping the month and revenue on the bar graph. The month will be the x-axis and the revenue will be the y-axis.

**SVG**

Scalable Vector Graphics (SVG) is a way to render graphical elements and images in the DOM. Think of SVG as a canvas, that an artist will paint his or her work on. D3 uses SVG to create all of its visuals; it is the area where the bar graph will be drawn.

**The SVG Element**

The SVG has its own element tag. D3 has lots of specific function that are design specifically for SVG. In this case, SVG is the area where you will draw the bar graph. Now, It is time to put the first line of code in the main.js file, open it up and we will create a SVG that has a width of 500px and a height of 500px. The picture below is the code for the SVG.

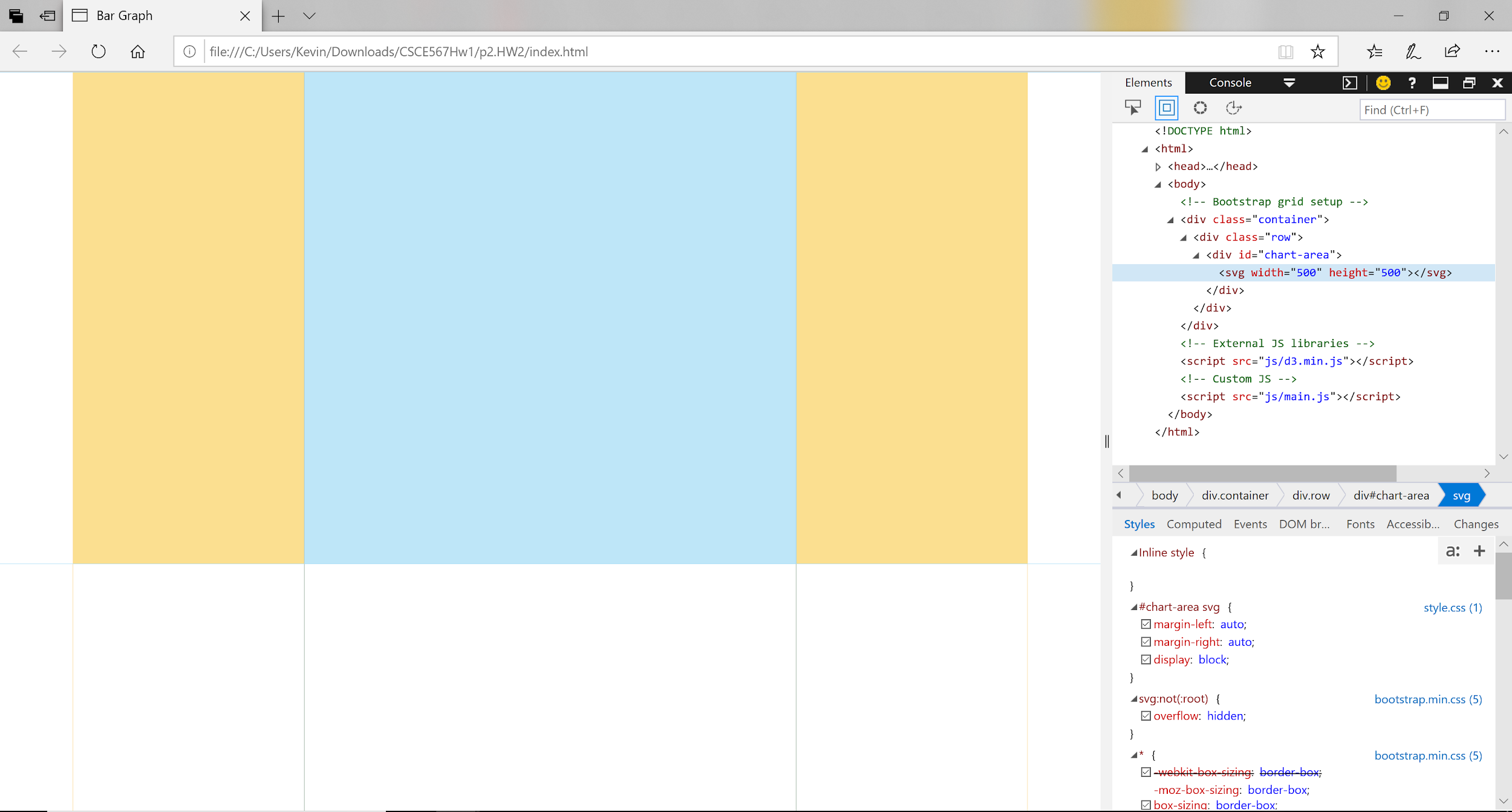


(Figure 7-1)

Here are a few things to note in this example:

* I have created a variable called svg and use “d3.select()” to select the third div tag that has the id “chart-area.”
* The third div tag is where the bar graph will be draw. D3 is smart enough to look through the HTML structure and find the specific div tag, because of the “#chart-area” name.
* The “#” is used to specify that the name “chart-area” is an id. Once the id is found, the svg tag is placed there, along with its attributes.
* The “.append(“svg”)“ add the svg tag to the chart-area and the “.attr()” gives the SVG its width and height attribute.The first script tag, points to the d3.min.js file, to import all the D3 functions.
* The “.attr()” takes in the name and value as the input. The name has to be spell correctly with double or single quotation, and the value has to be an integer.

Right now, the SVG is a blank white canvas, it blends in with the white background, when you run Microsoft Edge you are not able to see it. A useful technique to see the HTML structure in the web browser is to right click the screen and click on the Inspect Element option. In this example, the blue area is the SVG. We can see that the SVG is successfully created.



(Figure 8-1)

**Rectangle**

SVG is able to support a number of visual shapes between those svg tags. In this section, we will focus only on the rectangle shape. Once the SVG is created, the starting point for the x position and y position is 0,0 in the top-left corner of the drawing space. To Increase x values, move to the right, to increase y values, move down.

The keyword ‘rect’ draws a rectangle. Use x and y to specify the coordinates of the upper-left corner. Use width and height to specify the dimensions. A example is shown below:



(Figure 8-2)

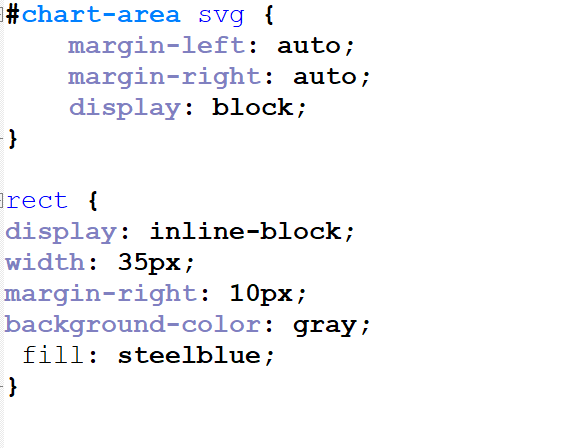
In this example, we want to specify that the rectangle must be drawn inside the SVG by using “svg.append(“rect”).” If we draw the rectangle outside the SVG it would not show up anywhere in the web browser. There are five attributes to the rectangle: x, y, width, height and fill. The x and y attribute gives the position of the rectangle by starting from position 0,0 in the top-left corner of the SVG. The width and height are used to specify the dimensions. The fill gives the color for the rectangle. These five attribute are required for the rectangle to be drawn.

We’ll use rectangle to generate a bar graph. The bar graph is essentially just multiple vertical rectangles.

**Drawing**

**Step 1:**

Setup the style.css file.

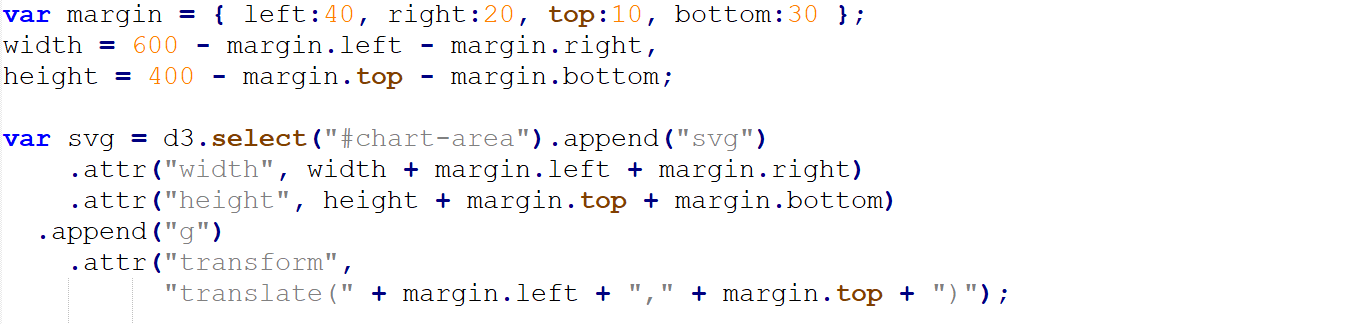
****

(Figure 9-1)

“#chart-area” make sure that the SVG is place in the center of the page. “Rect” styles all the rectangles to make sure that they have a width of 35px with 10px of spaces between each rectangle. It also color the rectangles as steelblue.

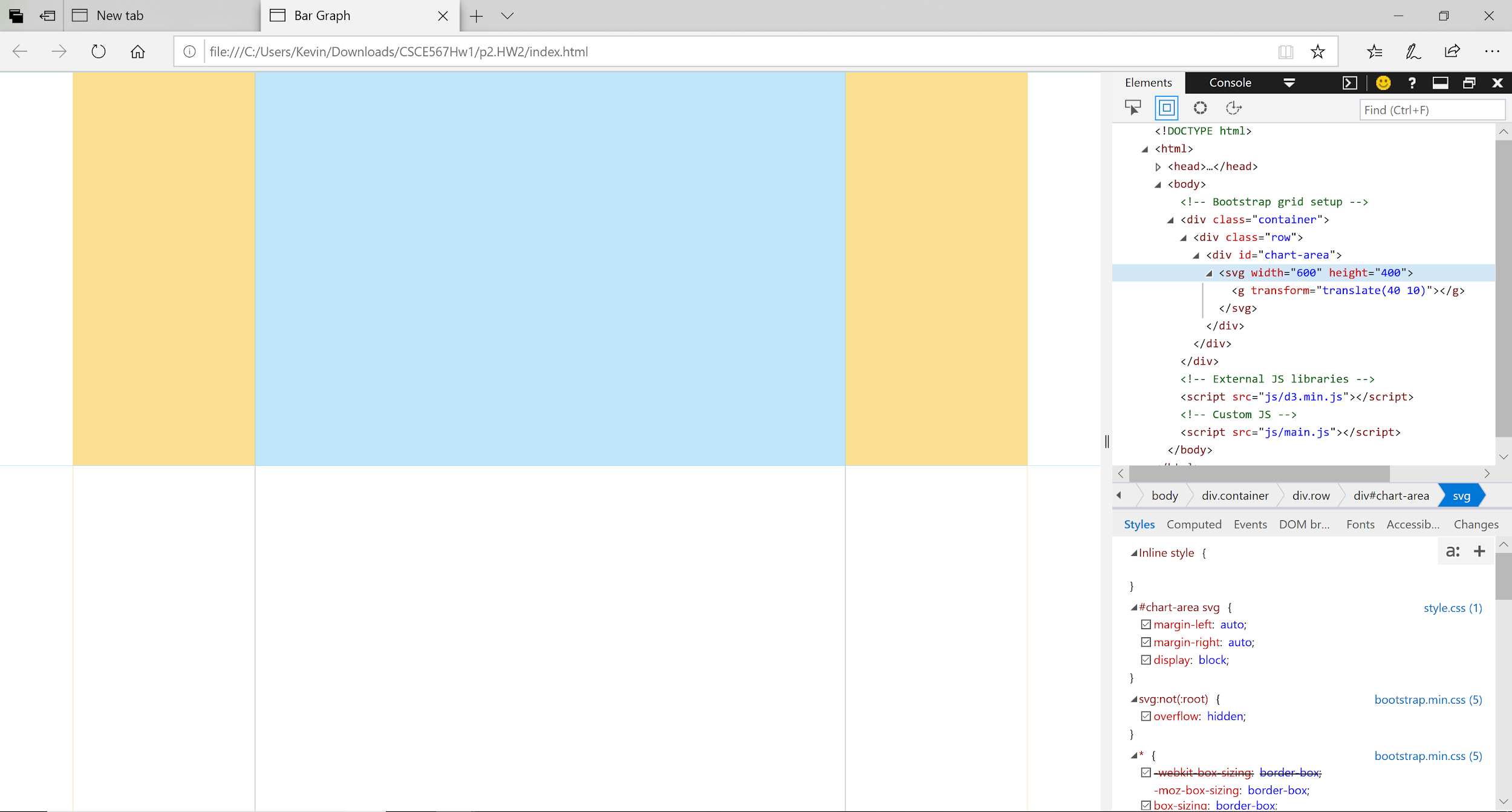
**Step 2:**

The second step is to setup the SVG chart and the margins for the SVG chart. Remember that SVG is the area where the bar graph will be created. The margins are to make sure that the graph won’t go out of bound on the chart. It ensure that the graph will always fit inside the SVG by giving the graph some wiggle room.



(Figure 10-1)

In (Figure 10-1) the total size of the SVG is 400px height and 600px width. The second “.append()” method adds the g tag. The g has no meaning in D3, we can change it to any other name. It is being used as a placeholder. Right now, it is used to group the SVG shapes together. The picture below shows that the SVG is created.

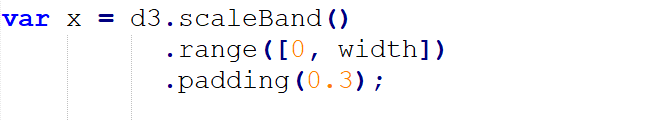


(Figure 10-2)

**Step 3:**

The third step is find the x range and the y range to create the x-axis and y-axis. We will begin with finding the x domain first.

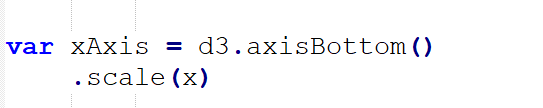
To do this, a new D3 function will be shown below. The “scaleBand()” function, is used to divide a range into a uniform range set. This is useful, we would want the width of the rectangle to be equal size for different category dimension. In this case, the range attribute takes in a range of 0 to 540.



(Figure 11-1)

In (Figure 11-1) a variable named x is created and is given a value of the scaleBand that has a range of 0 to 540. The padding attribute is used to give thickness to the rectangles.

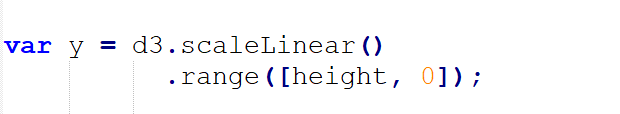
We have created the x domain, now we will create the x-axis. Another new D3 function will be shown, the “axisBottom()” function. The axisBottom() function is self explanatory. It creates an axis on the bottom of the SVG by drawing a line on the bottom. If you want to create an axis on the top of the SVG, use “axisTop().”



(Figure 11-2)

In (Figure 11-2) a variable called xAxis is created and is assigned the bottom axis. The scale attribute is asking for the range. We have x as the range, so we’ll input it into “scale().”

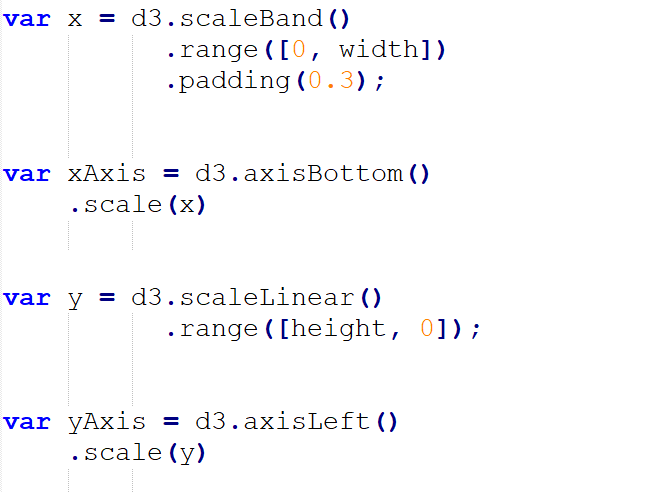
Now, for the y range. It will use a D3 function called “scaleLinear().” The function scaleLinear() is used to create a continuous range. This is useful, because we don’t want the line to be broken apart.



(Figure 12-1)

In (Figure 12-1) a variable called y is created and is assigned the “scaleLinear()” function with a range of 0 to 360.

Now for the y-axis.



(Figure 12-2)

The code for the y-axis is similar to the code for the x-axis, used the x-axis example. The x-axis and y-axis are now created, but not drawn yet.

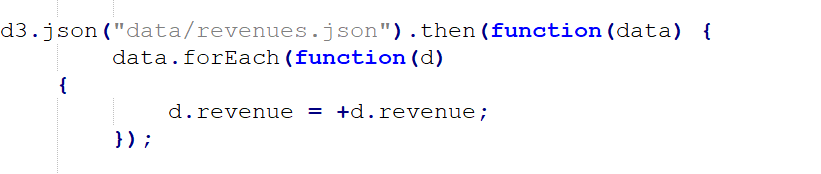
**Step 4:**

The fourth step is to map the data. This will be the most complex part. It involves reading the data and creating a rectangle for each object in the data file. This is where everything will be draw. The picture below, represent all the code for step four.

****

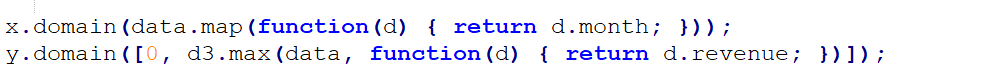
(Figure 13-1)

I will break down each important sections in the code.



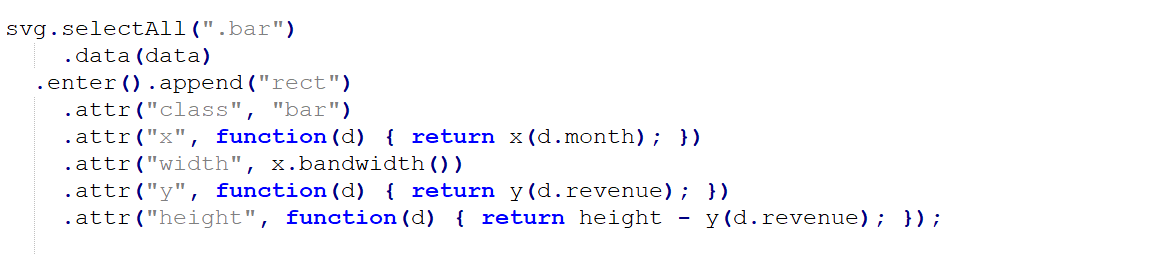
(Figure 13-2)

In (Figure 13-2) the first section is to select the data file using “d3.json()” and input the path of the file location inside parenthesis (). Next, use the “.then()” function to make sure that the file is found and then we can continue on with the code. Next, use the D3 anonymous function, which is a function that does not need to defined. In this case the parameter is named “data.” The next function, “data.forEach()” is the D3 version of a for-loop. Inside the for-loop use another anonymous function. Inside that second anonymous function it will find all the revenue value in the data file and will change each value from string to int. Everything from here to the end of the code, will be inside the second anonymous function.



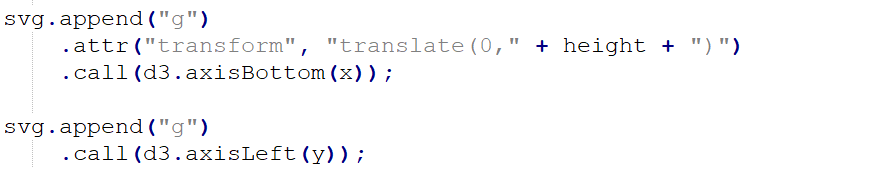
(Figure 14-1)

The second section shows that the “x.domain()” finds the name of the month and holds it for further use later. The “y.domain()” finds the max value of the data and use the “d3.max()” function to average each revenue. If you look at the revenue values in the data file, you’ll see that it has extreme ranges. The graph would have a hard time fitting all the values in a single small graph. The “y.domain()” ensure that the values can fit into a graph by giving the average size of all the revenue.



(Figure 14-2)

For the third section, after finding the x domain and y domain. We’ll begin to draw all the rectangles for the bar graph. Select the SVG and use “.selectAll(“rect”)” to add the placeholder tag named ‘.bar’ to the SVG. Next, use “.enter()” function for the SVG to accept inputs to be mapped to the all the ‘rect’ tags. Then, use “.append(“rect”)” to add the rect tags. Now, the “x.domain()” will be used. For the x attribute, use “x(d.month)” to find the name of the month. The next attribute ‘width’ is given the value of “x.bandwidth()”, that function finds the width of each bar. The third attribute is y with a value of revenue. The fourth attribute height returns the height minus the revenue to fit inside the graph. Remember that “y(d.revenue)” is the average size of all revenue values.

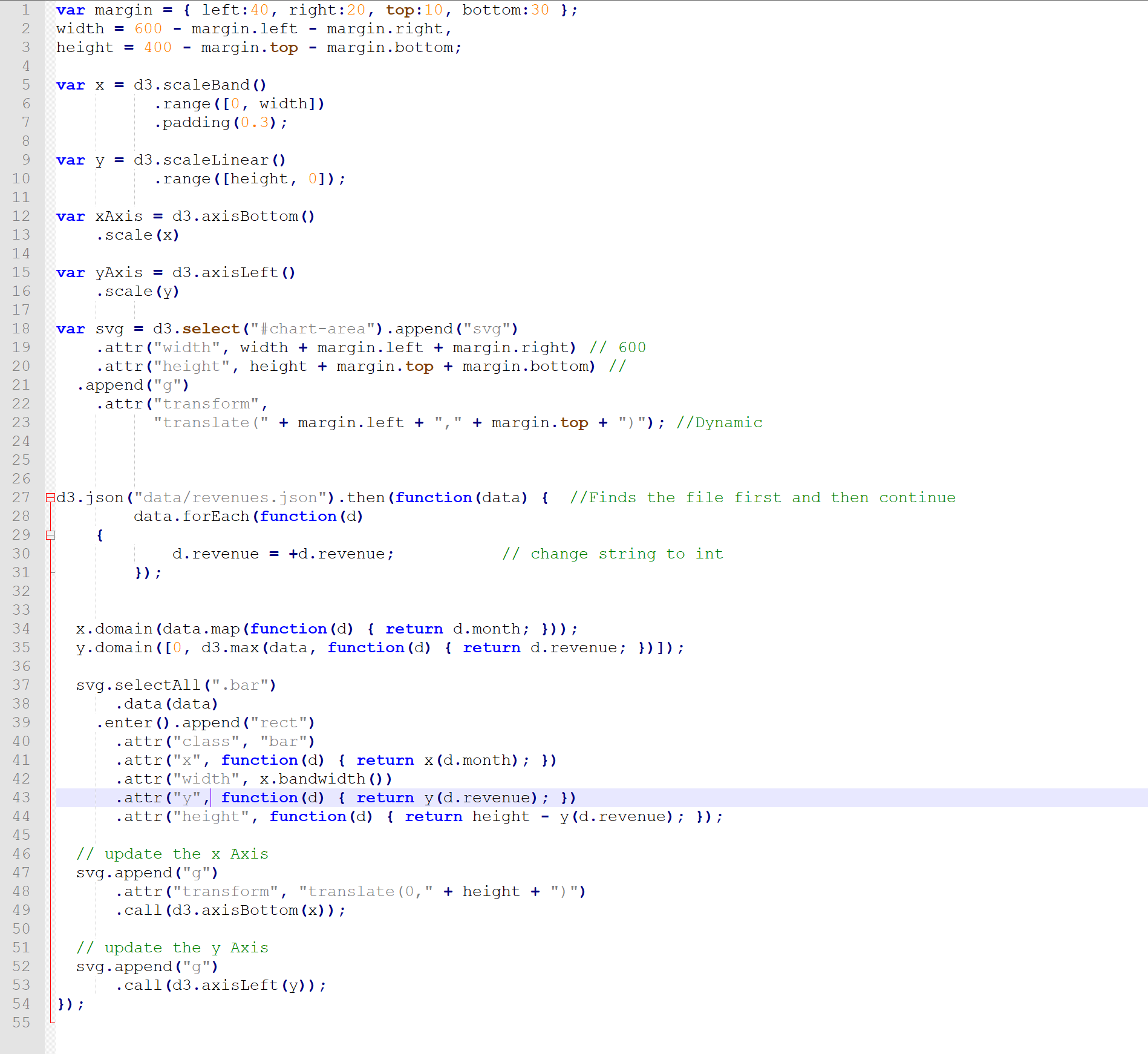


(Figure 14-3)

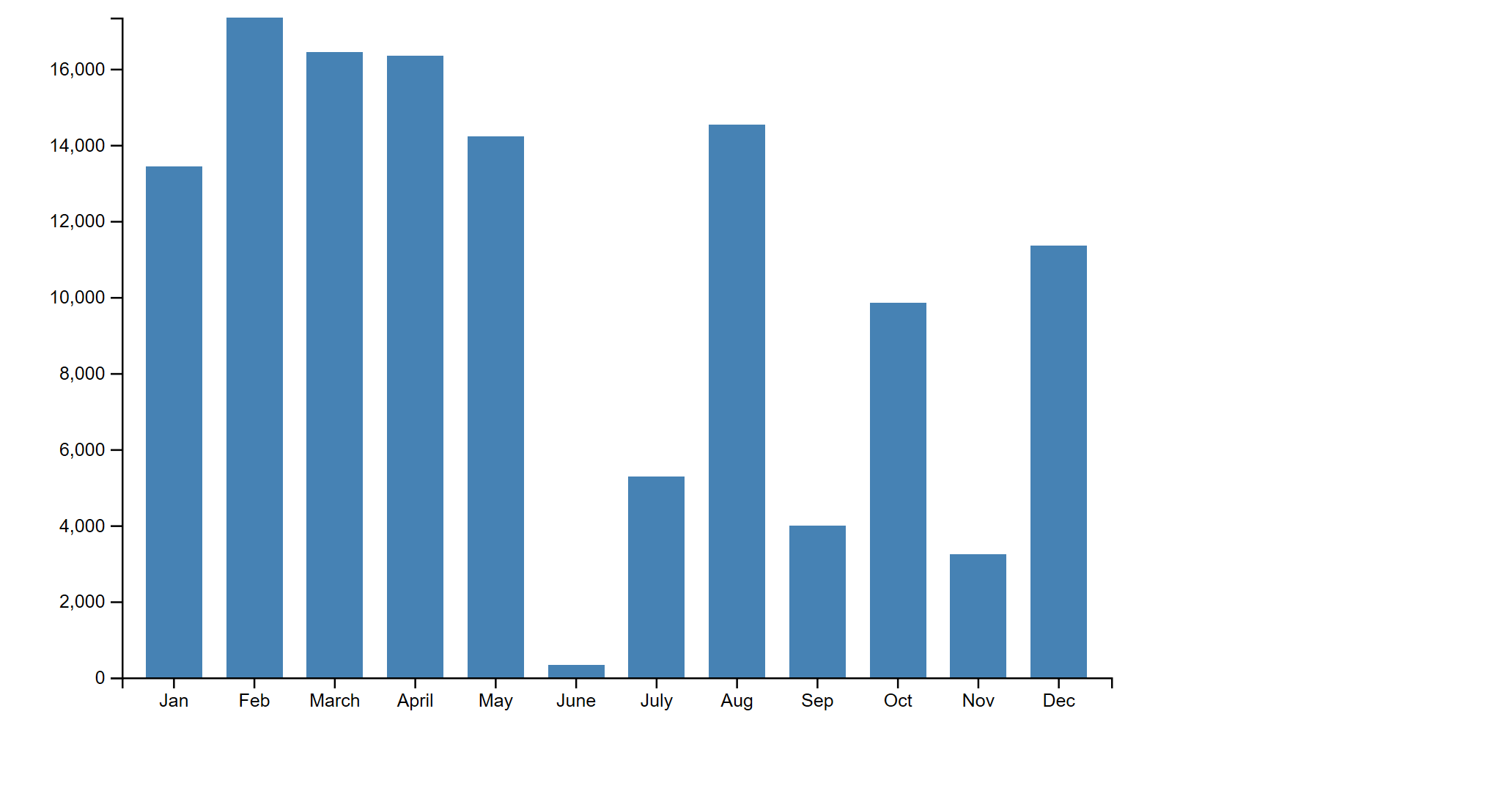
Now for the final section, to add the x-axis. Append a placeholder tag called g to the SVG. Use the “.call(d3.axisBottom(x))” to add the bottom axis with the value of x. The y-axis is done the same way, but with the “.axisLeft(y).” We have now created the bar graph.

**The Final Product**

This picture below, is all the code that was used in this project.

****

(Figure 15-1)

****

(Figure 16-1)

**Conclusion**

The main point of this project to show how D3 is able to transform raw data into a bar graph to communicate the data more efficiently. It is much easier to view a graph rather than reading directly from the JSON file. I hope you’ve learned something from this manual. Thank you for reading.