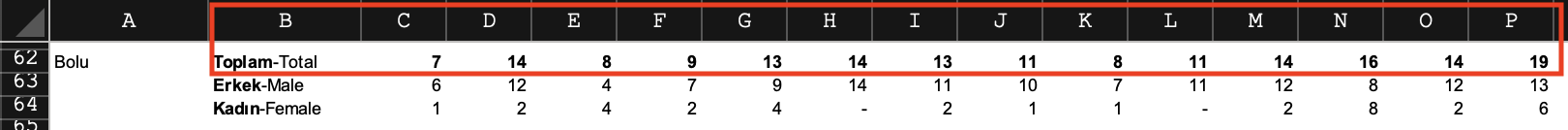
**CS405 Assignment 1**

**Barış Ulaş Çukur**

**29461**

**The dataset**

I’ve chosen to work with suicide rates by cities and genders table. Here’s a quick summary of the dataset and the subset I’ve chosen to work with.

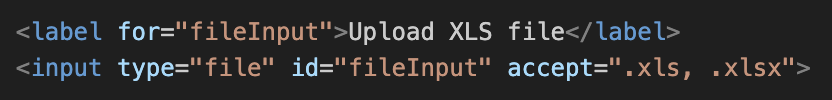


The subset of the suicide rates by cities and genders table that I’ve chosen to work with.

The subset I’ve chosen to work with was about the overall suicide rates by all genders in Bolu between 2009 and 2022. The dataset included year information on the 5th row of the table and the overall suicide rates summary in Bolu on the 62nd row.

**Reading the data**

The code includes a *fileInput* label for selecting an *xls* file from the local storage. After the file has been selected, the website assigns it an *id* called “fileInput”. Later on, the other parts of the code are able to reference this file using this *id*.



Label prompts the user to upload an xls file from the local storage and input field assigns it “fileInput” id.

**Filtering the data**

While *JavaScript* works fine with *CSV* files, the dataset was in *xls (excel datasheet)* format. To work with the file, I needed a library to handle the reading process. After researching about it, I’ve decided to use *SheetJS* library. The code is able to access the specific rows and columns of the excel sheet using this library.

The code works by reading the xls file and storing the resulting object in the *sheet* variable. Later, code iterates over each row and column of the file. When it reaches the rows and columns of the specified parts in the *columnNumbers* and *rows* variables, it appends the data inside the *currentList* variable. After the loop reaches the end of the current row, it appends the *currentList* inside the *datamatrix* variable.

In the end, we end up with a matrix that consists of the data filtered by the *columnNumbers* and *rows* lists.

A computer code on a black background

Description automatically generated

JS code accessing the certain columns and rows of the uploaded excel sheet to get the data.

**Mistakes and Regrets**

I’ve recently come into a realization that I did not need to parse the excel sheet and I could’ve manually imported the values of the cells into a list or a matrix. I wish the homework document was clearer about this part. At least now I know how to parse excel sheets with *JavaScript*.

**Visualizing the data**

* **Creating the Axes**

Before visualizing the data, I needed to add an *SVG* with dimensions 1000 to 600 pixels to place the graph in. Data visualization is handled by the two functions of the code called *createAxis* and *createLabel*.

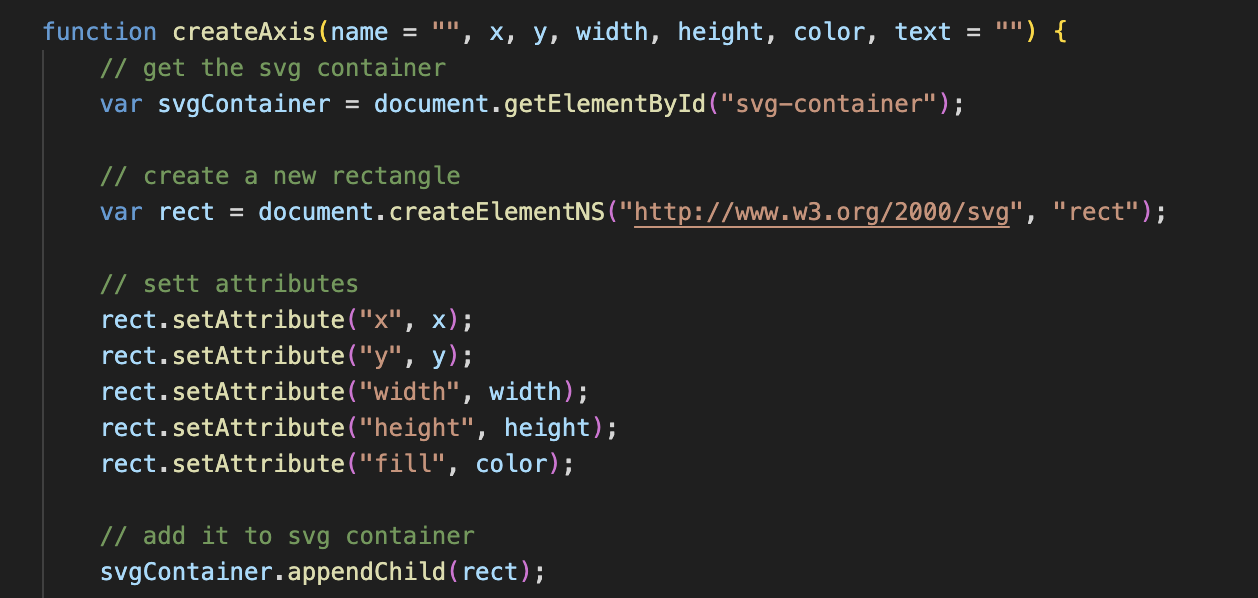
*createAxis* is responsible by creating the x and y axis of the graph. It takes *name, x, y, width, height, color* and *text* parameters and creates an axis accordingly.

A black background with white text

Description automatically generated

createAxis function calls to create x and y axes.

The *createAxis* function works by creating a rectangle of specified *width* and *height* integers from the function parameter. Similarly, it also places the rectangle to *x* and *y* coordinates given by the parameter of the function.



createAxis function creating a rectangle to be used as an axis.

* **Adding labels to axes**

*createLabel* function is responsible by the creation of the axis labels, such as *0, 10, 20* for *y* *axis* and *2009, 2010, 2011* for the *x axis*. The function takes *type, x, y* and *text* elements as parameters.

It basically works by adding a *text* element to the specified position by the function caller. It also adds a small vertical rectangle in the shape of a line “|” 20 pixels to the left of the text.

The logic of adding labels dynamically to the graph required finding a generalized formula for the *x axis* offset of the labels. Instead of explaining it in a long text, I’ll go over the visualization I’ve created.

A graph of a bar graph

Description automatically generated

Alignment of x-axis labels with respect to their indices.

Basically, the generalized formula for the x offset of the label n at index n is calculated using:

x\_offset = (2n + 1) \* a + n \* w + d

*n: Index of the label*

*a: offset of the bar graph from the left = 20px*

*w: width of the bar graph = 25px*

*d: display deadzone = 50px*

For example, the second label of the x-axis with the value “label 2” and index 1 would have the x offset of: (2\*2 + 1)\*20 + 2\*25 + 50 = 200px. This generalized formula ensured that each bar graph labels are placed in place correctly.

Adding labels to *y-axis* required the development of a similar generalized formula. Similarly, I’ve created a basic visualization to go over.

A graph of a bar graph

Description automatically generated

Alignment of y-axis labels with respect to their indices.

Similarly, I’ve devised a generalized formula for the alignment of y labels with respect to their indices.

y\_offset = y – n\*l

y: length of the y-axis = 350px

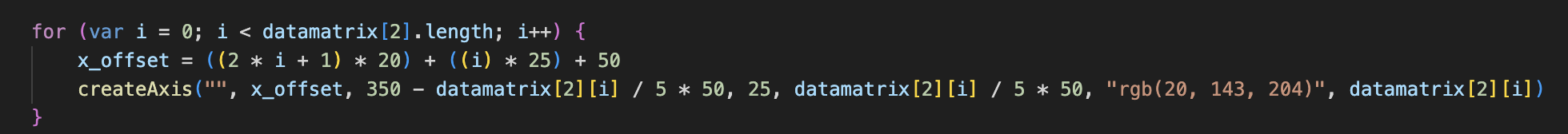
n: index

l: the space between two labels = 50px

For example, the second element of the graph with the value 10 and index 1 would have a y offset of: 350 – 1\*50 = 300px.

* **Creating the bar charts**

For the creation of the bar charts, *createAxis* function is repurposed and used again but this time is called a bit differently.



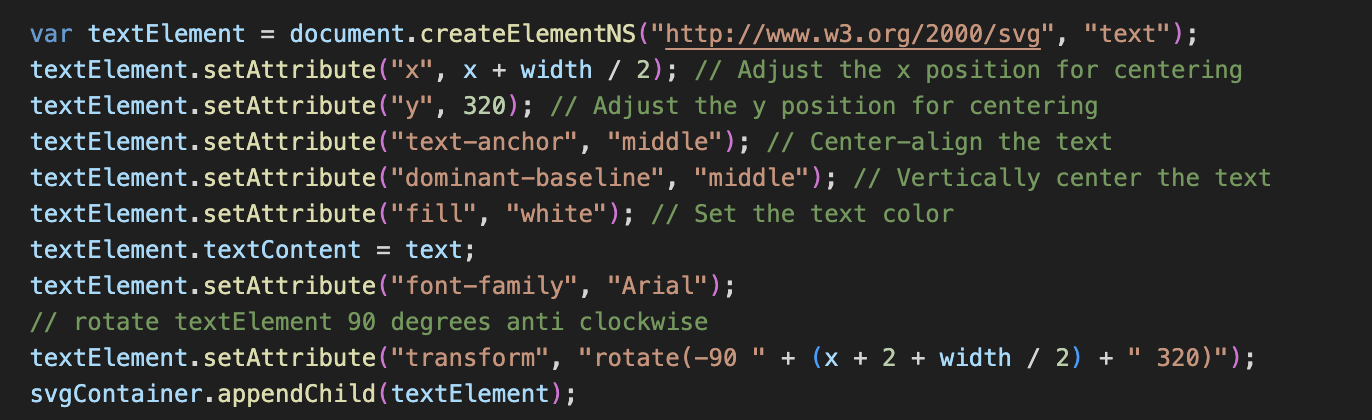
Similar to the previous offset calculations, I calculated the required offsets for the x-axis of the bar elements. The offset formula for calculating the offset here is the same that was used for calculation of the x-axis labels, so I won’t go into details of the calculation again.

What’s new here is the calculation of the bar graph (a rectangle basically) height with respect to y-axis offset. As JavaScript starts drawing the rectangle from the top, I needed to move the draw point to the place where the remaining part of the rectangle would be equal to the value it holds.

A basic formula for this could be devised as follows:

Starting point of the rectangle (y) = y-axis length – value of the rectangle

This function is also responsible from adding the value of the current element inside the rectangle rotated 90 degrees anti-clockwise.



createAxis function adding the value of the current element inside the rectangle.

If needed, *createAxis* function also takes a *name* parameter. This is to be used as the label of the *y* graph, “Ölüm Sayısı” for this instance. If a name is provided, the function adds it to the top of the *y* graph by creating a new text element with the x-value of the SVG container and y-value of the y-axis length - 20.

* **Adding legend**

Lastly, I added a legend to the graph to visualize the kind of the value shown on the graph. To create the label, I just added a simple text and rectangle elements side by side at the center of the graph and a little bit below it.

A screen shot of a computer program

Description automatically generated

addLegend function to add legend to the graph.



The function call of the addLegend function to create a legend with specified text color.

Finally, we end up with a graph that has correctly aligned x and y labels combined with accurate bar lengths including their values inside it rotated 90 degrees anti-clockwise.

A graph of blue bars

Description automatically generated with medium confidence

The final graph we end up with.