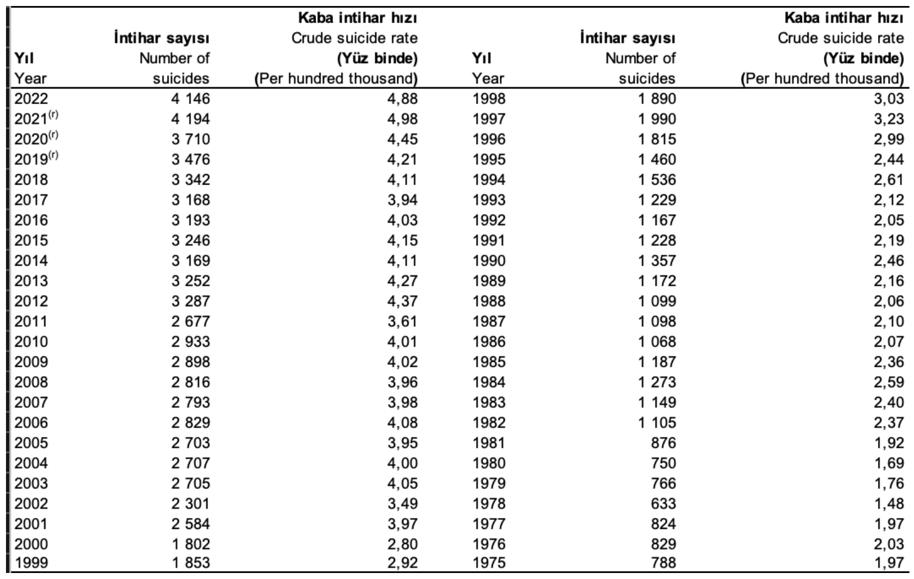
CS 405 HW1 Report

Gorkem Yar

27970

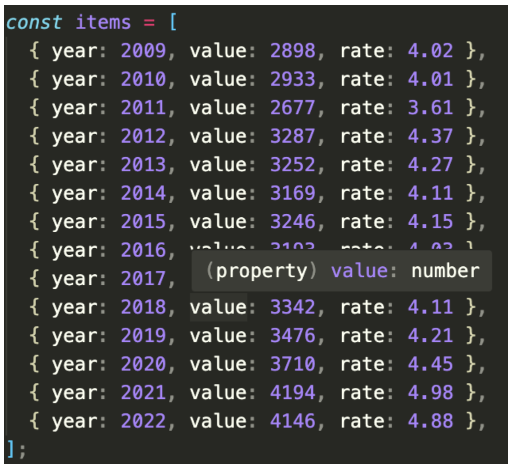
For this homework, I used the dataset "Kaba intihar hızı". I used both the number of suicides and the rate of suicide per 100,000 to visualize.



Yıl Year	İntihar sayısı Number of suicides	Kaba intihar hızı Crude suicide rate (Yüz binde) (Per hundred thousand)
2022	4 146	4,88
2021(r)	4 194	4,98
2020(r)	3 710	4,45
2019(r)	3 476	4,21
2018	3 342	4,11
2017	3 168	3,94
2016	3 193	4,03
2015	3 246	4,15
2014	3 169	4,11
2013	3 252	4,27
2012	3 287	4,37
2011	2 677	3,61
2010	2 933	4,01
2009	2 898	4,02
2008	2 816	3,96
2007	2 793	3,98
2006	2 829	4,08
2005	2 703	3,95
2004	2 707	4,00
2003	2 705	4,05
2002	2 301	3,49
2001	2 584	3,97
2000	1 802	2,80
1999	1 853	2,92
1998	1 890	3,03
1997	1 990	3,23
1996	1 815	2,99
1995	1 460	2,44
1994	1 536	2,61
1993	1 229	2,12
1992	1 167	2,05
1991	1 228	2,19
1990	1 357	2,46
1989	1 172	2,16
1988	1 099	2,06
1987	1 098	2,10
1986	1 068	2,07
1985	1 187	2,36
1984	1 273	2,59
1983	1 149	2,40
1982	1 105	2,37
1981	876	1,92
1980	750	1,69
1979	766	1,76
1978	633	1,48
1977	824	1,97
1976	829	2,03
1975	788	1,97

For my SVG plot, I used the data between the years 2009 to 2022. I created a static array to use the data in the visualization.

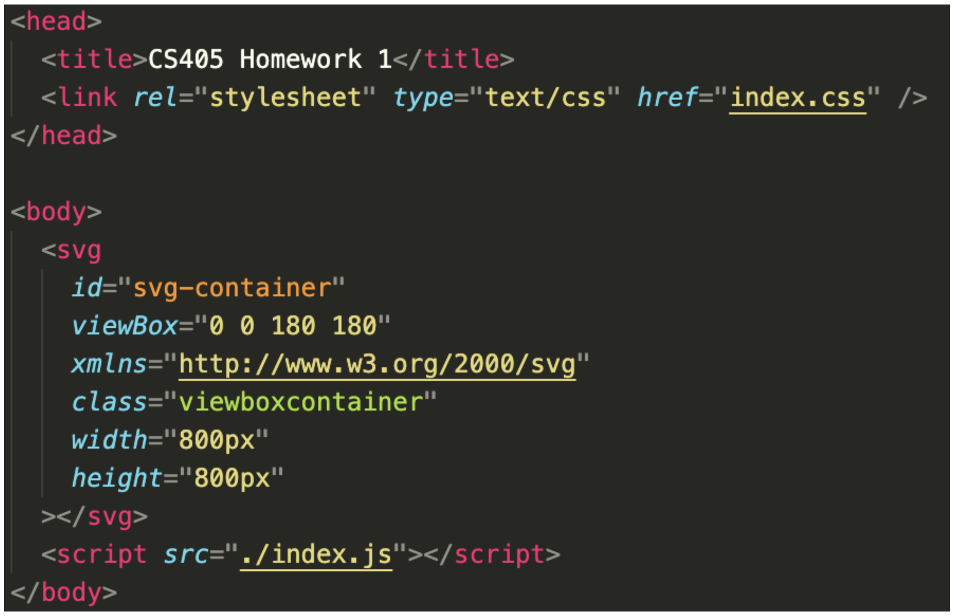
Here is my static array in the code:



```
const items = [
  { year: 2009, value: 2898, rate: 4.02 },
  { year: 2010, value: 2933, rate: 4.01 },
  { year: 2011, value: 2677, rate: 3.61 },
  { year: 2012, value: 3287, rate: 4.37 },
  { year: 2013, value: 3252, rate: 4.27 },
  { year: 2014, value: 3169, rate: 4.11 },
  { year: 2015, value: 3246, rate: 4.15 },
  { year: 2016, value: 3193, rate: 4.03 }
  { year: 2017,  (property) value: number
  { year: 2018, value: 3342, rate: 4.11 },
  { year: 2019, value: 3476, rate: 4.21 },
  { year: 2020, value: 3710, rate: 4.45 },
  { year: 2021, value: 4194, rate: 4.98 },
  { year: 2022, value: 4146, rate: 4.88 },
];
```

My code has three files: index.js, index.html, index.css. In index.css there is almost no code. I did not use CSS directly. I created my components in Javascript and added them to the SVG element.

index.html:

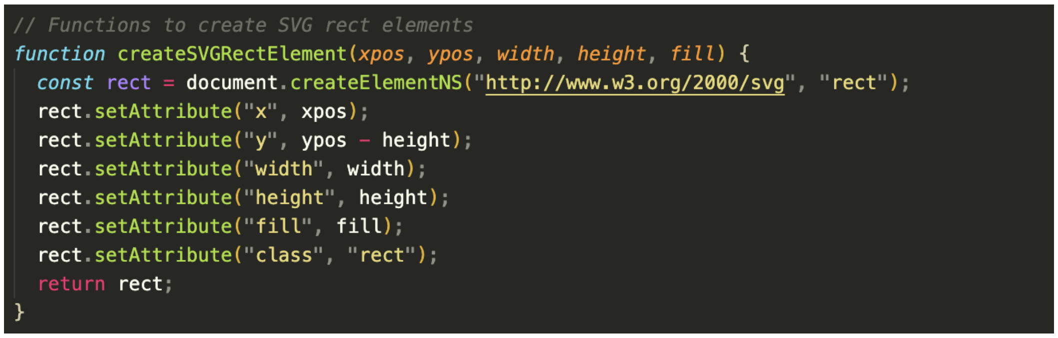


```html
<head>
  <title>CS405 Homework 1</title>
  <link rel="stylesheet" type="text/css" href="index.css" />
</head>

<body>
  <svg
    id="svg-container"
    viewBox="0 0 180 180"
    xmlns="http://www.w3.org/2000/svg"
    class="viewboxcontainer"
    width="800px"
    height="800px"
  ></svg>
  <script src="./index.js"></script>
</body>
```

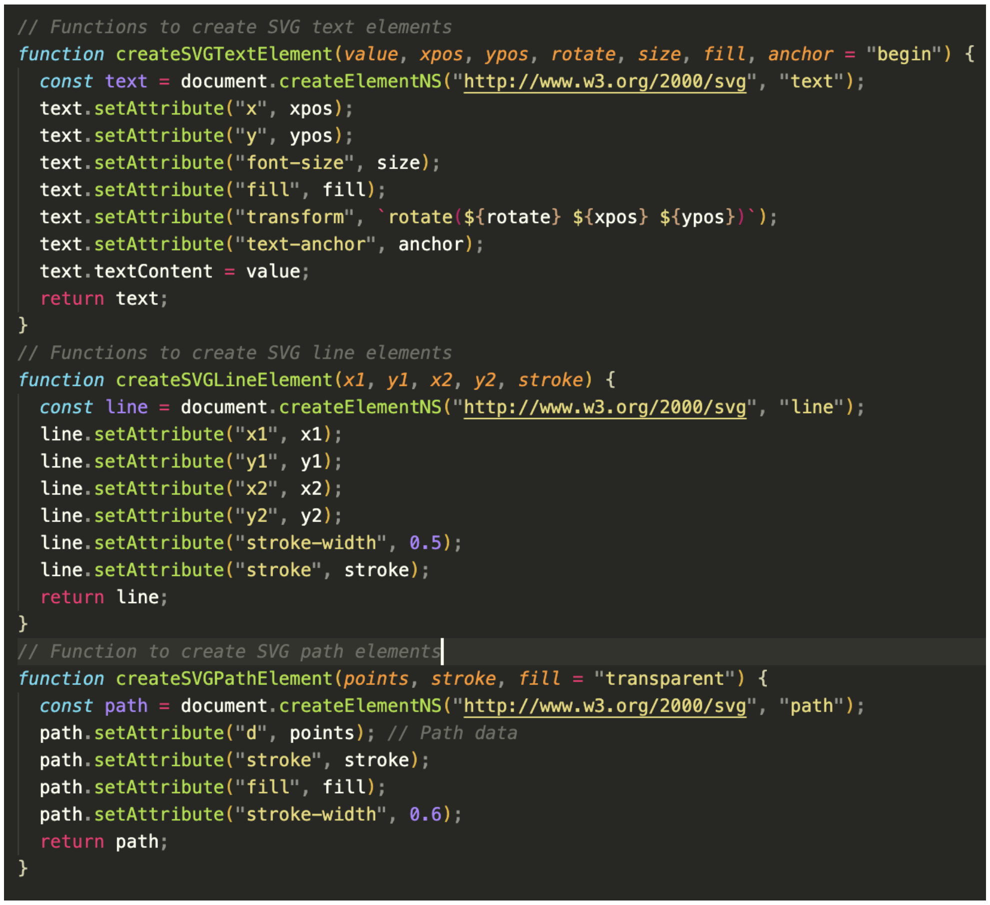
I only created an SVG element in index.html and assigned it to a unique ID. The SVG element is a parent element in order to add other SVG elements like rect, line, and path. My main functionality and components reside in the index.js file.

In my index.js file, I created four functions to create rect, text, line, and path SVG elements.



```javascript
// Functions to create SVG rect elements
function createSVGRectElement(xpos, ypos, width, height, fill) {
  const rect = document.createElementNS("http://www.w3.org/2000/svg", "rect");
  rect.setAttribute("x", xpos);
  rect.setAttribute("y", ypos - height);
  rect.setAttribute("width", width);
  rect.setAttribute("height", height);
  rect.setAttribute("fill", fill);
  rect.setAttribute("class", "rect");
  return rect;
}
```

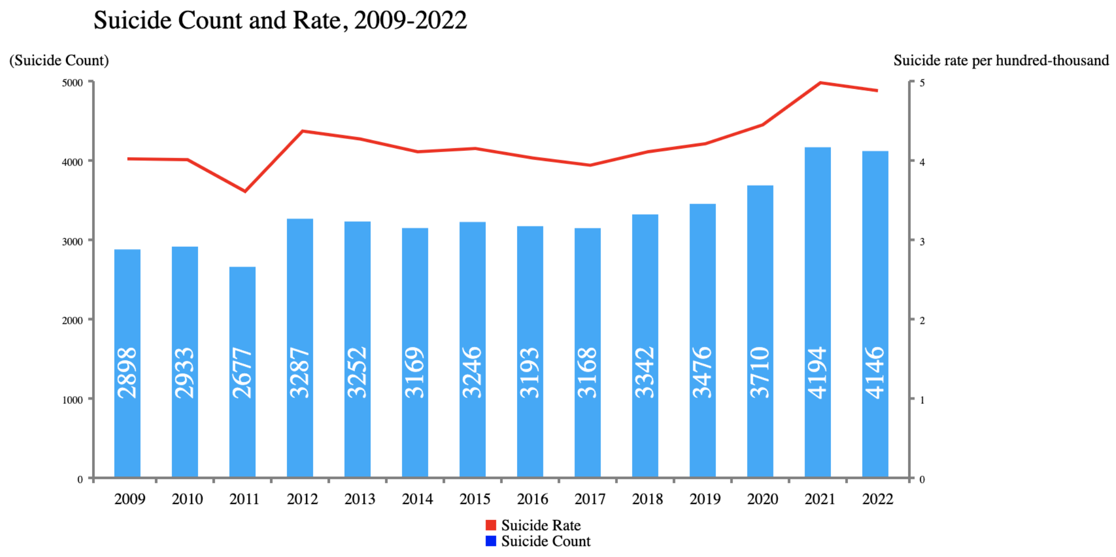
This function takes the x and y coordinates, width, height, and color then returns an SVG rectangle object.



```
// Functions to create SVG text elements
function createSVGTextElement(value, xpos, ypos, rotate, size, fill, anchor = "begin") {
  const text = document.createElementNS("http://www.w3.org/2000/svg", "text");
  text.setAttribute("x", xpos);
  text.setAttribute("y", ypos);
  text.setAttribute("font-size", size);
  text.setAttribute("fill", fill);
  text.setAttribute("transform", `rotate(${rotate} ${xpos} ${ypos})`);
  text.setAttribute("text-anchor", anchor);
  text.textContent = value;
  return text;
}
// Functions to create SVG line elements
function createSVGLineElement(x1, y1, x2, y2, stroke) {
  const line = document.createElementNS("http://www.w3.org/2000/svg", "line");
  line.setAttribute("x1", x1);
  line.setAttribute("y1", y1);
  line.setAttribute("x2", x2);
  line.setAttribute("y2", y2);
  line.setAttribute("stroke-width", 0.5);
  line.setAttribute("stroke", stroke);
  return line;
}
// Function to create SVG path elements
function createSVGPathElement(points, stroke, fill = "transparent") {
  const path = document.createElementNS("http://www.w3.org/2000/svg", "path");
  path.setAttribute("d", points); // Path data
  path.setAttribute("stroke", stroke);
  path.setAttribute("fill", fill);
  path.setAttribute("stroke-width", 0.6);
  return path;
}
```

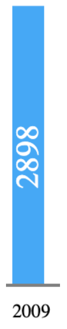
Similar to the createSVGRectElement function, these functions are responsible for creating the necessary SVG components.

Before going further into the details of my code. I would like to share the plot that I created. By doing this my aim is to discuss the details in a more concrete way.

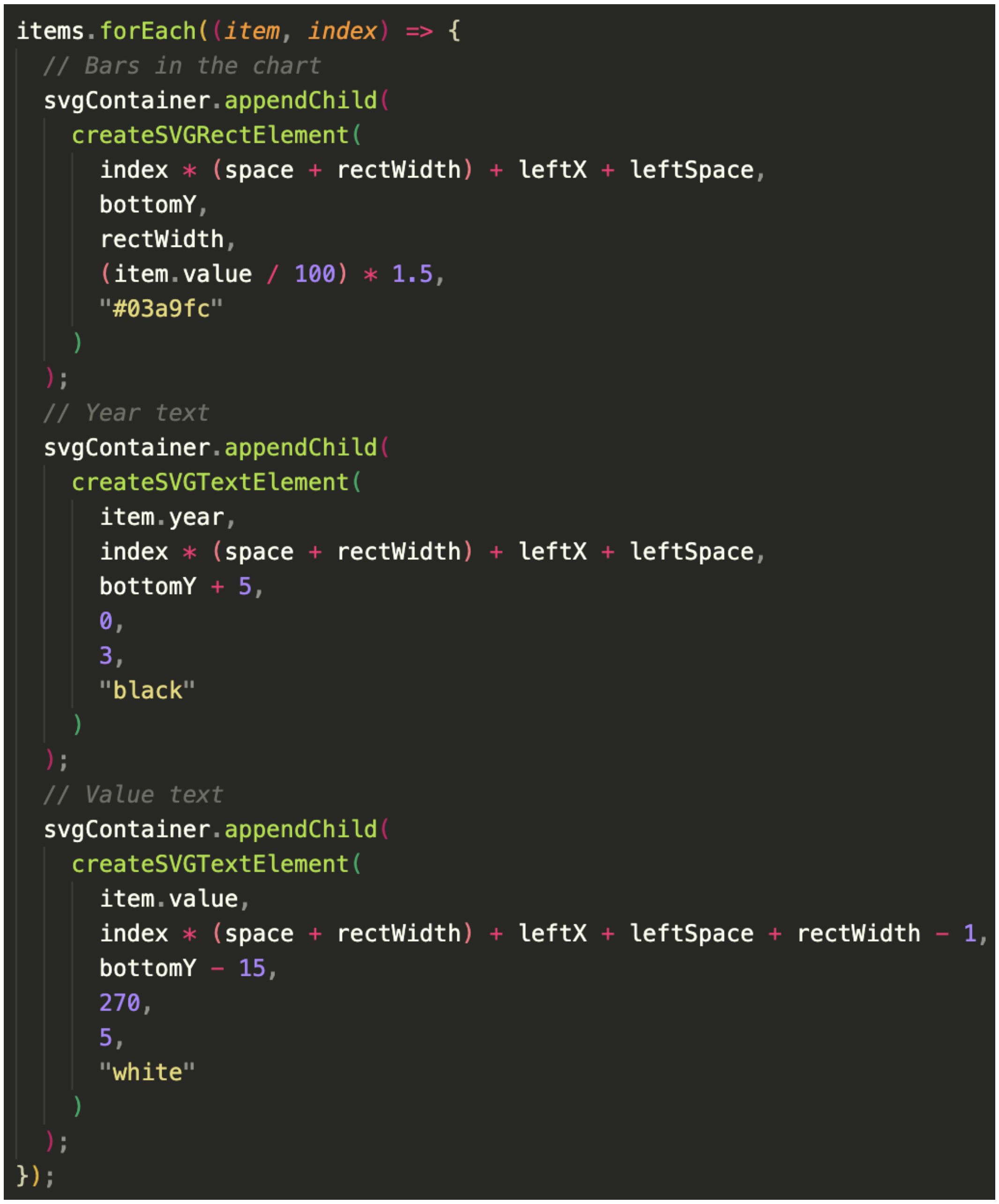


One important part of my code is constants for aligning the SVG elements. These constants help me to define the position of each element respective to their index.
Here is the list of constant that I use:

valueStepSize: The left Y axis on the graph shows the suicide count. ValueStepSize constant is for deciding what should be the step size between the suicide count values. For this graph it is 1000.
valueStepCount: This constant also help me to show the total number of steps. For this example it is 5 since the maximum suicide count is 4194 and step size is 1000.
rateStepSize: The right Y axis on the graph shows the suicide rate per 100000. This is the respective constant for the right Y axis. For this dataset is 1.
rateStepCount: It is the respective version of the valueStepCount in the right Y axis.
space: The distance between rect elements.
rectWidth: The width of the rect elements.
bottomY: The position of X axis.
topY: The position of the highest point in the graph. It can be calculated through the following formula: bottomY - ((valueStepCount * valueStepSize) / 100) * 1.5;
100 and 1.5 are just constants for better visualization.
leftX: The position of the left Y axis.
rightX: The position of the right Y axis. It can be calculated through the following formula:
leftX + leftSpace + dataCount * rectWidth + (dataCount - 1) * space + rightSpace
dataCount is the number of items in the static data array.
leftSpace: The distance between the left Y axis and the first bar.
rightSpace: The distance between the right Y axis and the last bar.

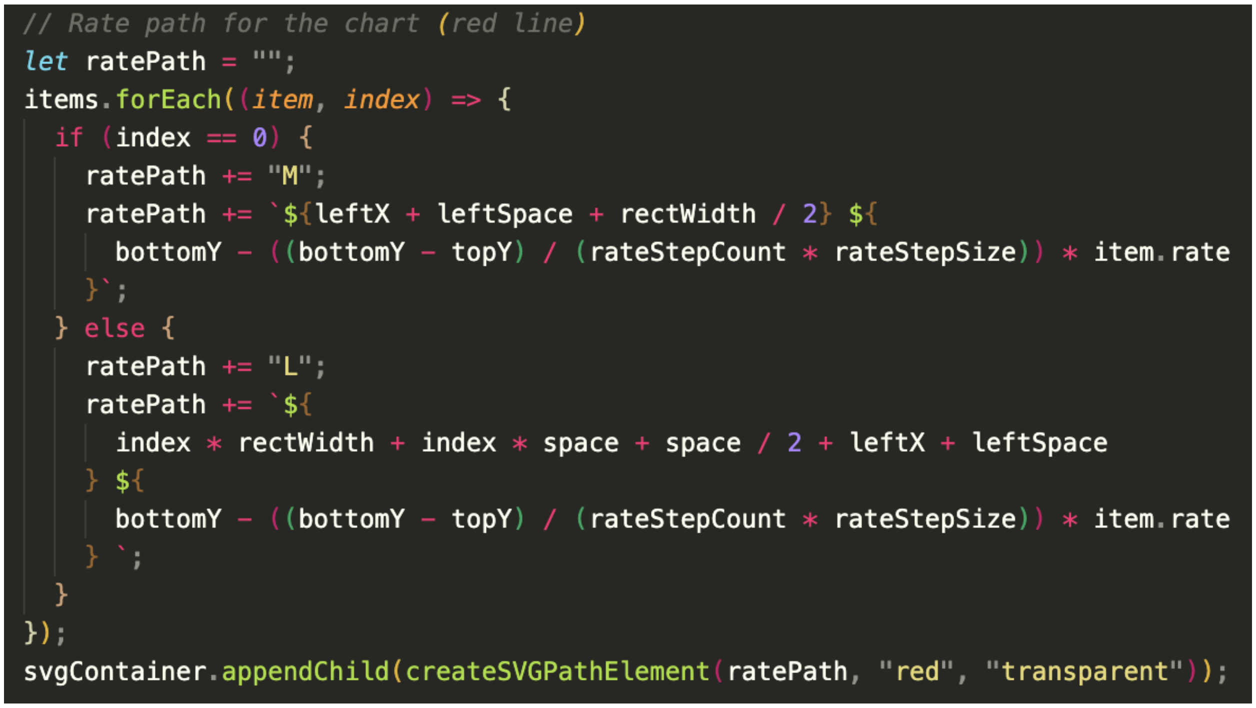


Now, it is time to mention the creation of SVG elements. The code that is in the following page creates the bar elements for the graph with the count inside them. It also creates the year of these data below them. One of the examples of the resulting code is the image in the left.



```
items.forEach((item, index) => {
  // Bars in the chart
  svgContainer.appendChild(
    createSVGRectElement(
      index * (space + rectWidth) + leftX + leftSpace,
      bottomY,
      rectWidth,
      (item.value / 100) * 1.5,
      "#03a9fc"
    )
  );
  // Year text
  svgContainer.appendChild(
    createSVGTextElement(
      item.year,
      index * (space + rectWidth) + leftX + leftSpace,
      bottomY + 5,
      0,
      3,
      "black"
    )
  );
  // Value text
  svgContainer.appendChild(
    createSVGTextElement(
      item.value,
      index * (space + rectWidth) + leftX + leftSpace + rectWidth - 1,
      bottomY - 15,
      270,
      5,
      "white"
    )
  );
});
```

Another important factor of my SVG image is the suicide rate path. For creating that path, I calculated the positions of each points. The position of each point depends on both x and y position. X position can be calculated using the index of the position. Y position can be calculated using the suicide percentage rate. The code doing that functionality resides in the next page.



```javascript
// Rate path for the chart (red line)
let ratePath = "";
items.forEach((item, index) => {
  if (index == 0) {
    ratePath += "M";
    ratePath += `${leftX + leftSpace + rectWidth / 2} ${
      bottomY - ((bottomY - topY) / (rateStepCount * rateStepSize)) * item.rate
    }`;
  } else {
    ratePath += "L";
    ratePath += `${
      index * rectWidth + index * space + space / 2 + leftX + leftSpace
    } ${
      bottomY - ((bottomY - topY) / (rateStepCount * rateStepSize)) * item.rate
    } `;
  }
});
svgContainer.appendChild(createSVGPathElement(ratePath, "red", "transparent"));
```

The last part of my SVG image is the X and Y axes. I added them by using the line function and the constants that were defined before. In each axis, there are separators as well. These separators were created by line function. I also added names for these axes with the text creation function. Finally, I added a legend at the bottom of the plot.