F21DV - Data Visualization and Analytics

Coursework Lab 1

Due on Friday 25 Feb 2022

Submitted By: Shashank Ayanikkatt Vengalapurath

Demonstrated to:

Demonstration on:

School of Mathematical and Computer Sciences

Heriot-Watt University

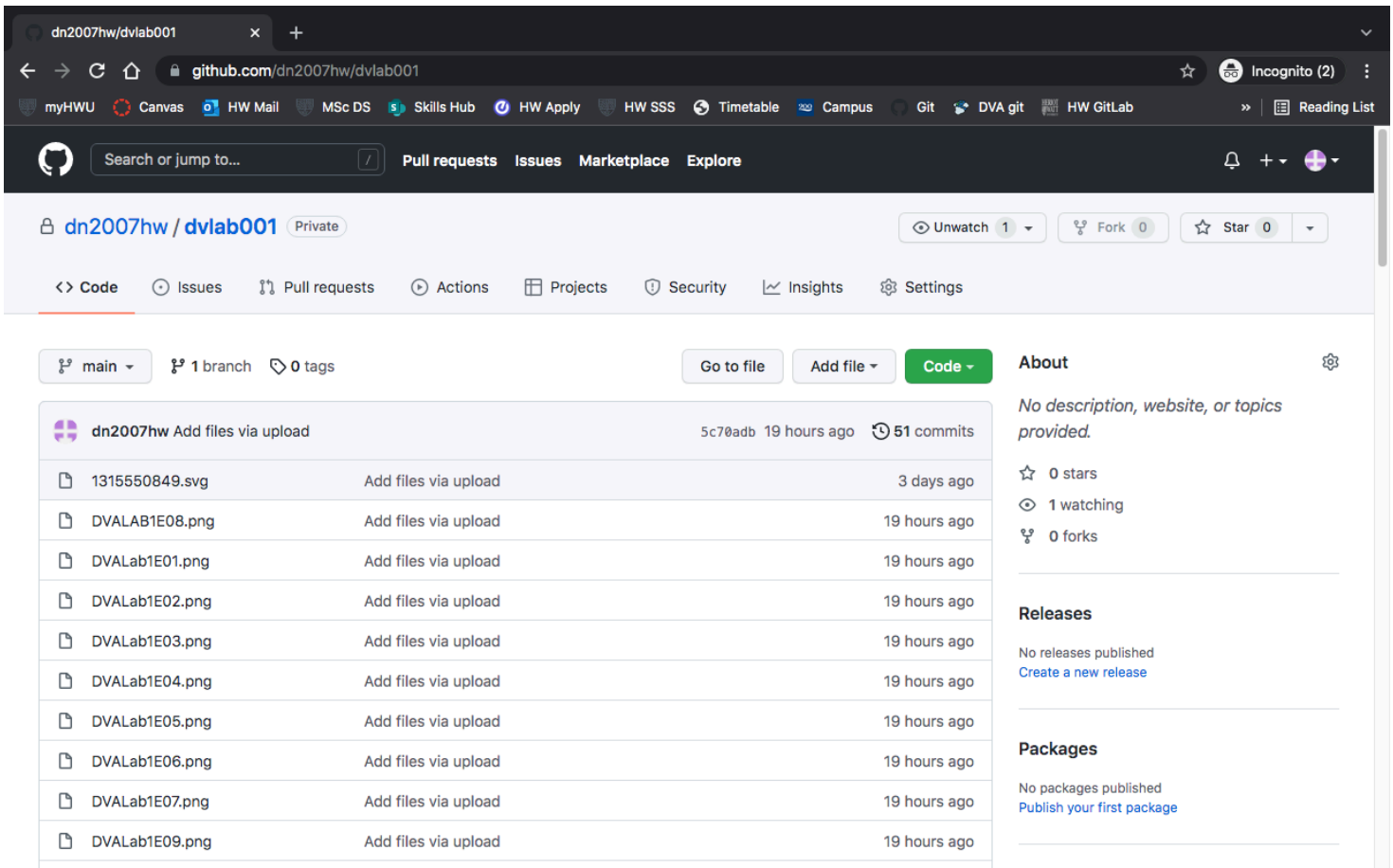
Table of Contents

Overview

The purpose of this course work is to have a basic understanding of D3 Visualization concepts and how the data is bind to the graphic to produce the visualization. The lab exercises have demonstrated how the data binding is done in different methods and how the different types of charts and graphs are prepared and rendered on the screen using HTML, CSS, Javascript and D3 V 7.3.0.

Part 1. Getting Started – GitHub Pages

A repository is created in GitHub to maintain all the coursework files. All the source codes and output screenshots are maintained in a private repository.



Part 2. D3 Setup

Exercise 1:

What version number is displayed in the console output window?

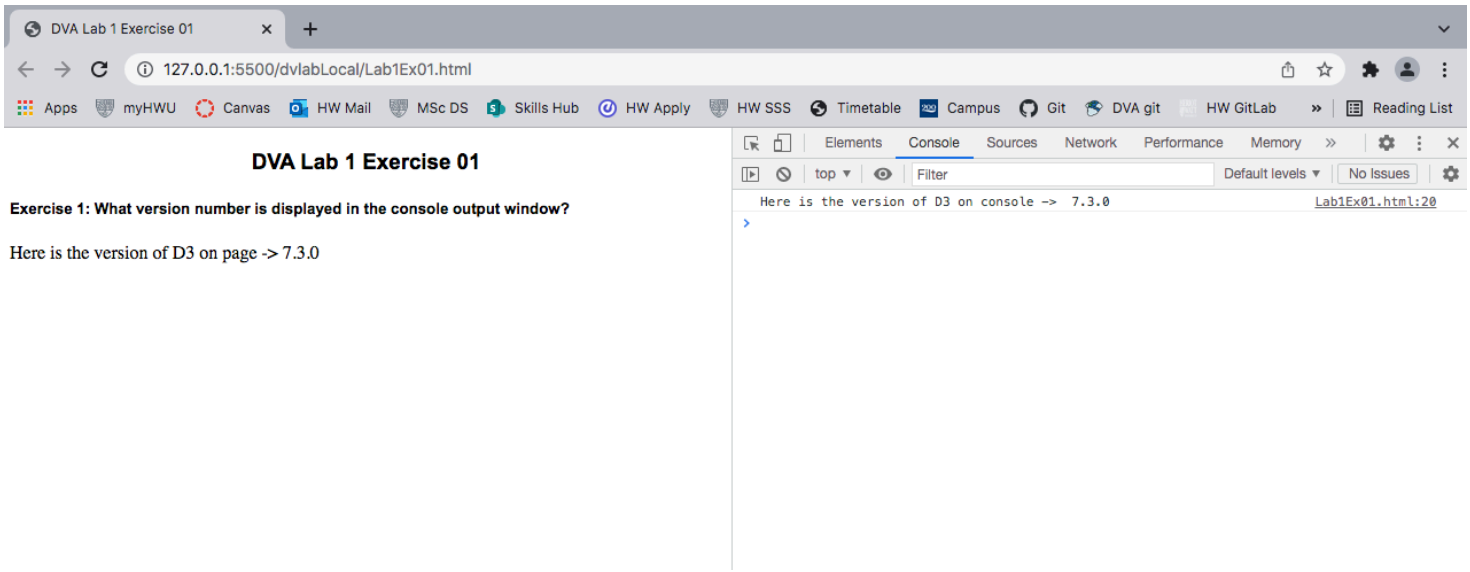
Code: Lab1Ex01.html (available in the attached zip file)

Snippet:

The current version of the D3 is obtained from pre-defined element D3.Version and displayed in both browser and console.

```
// Following line will print the version on the browser page.
d3.select("body")
  .append("p")
  .text("Here is the version of D3 on page -> " + d3.version);
// Following link will print the version on the console.
console.log("Here is the version of D3 on console -> ", d3.version);
```

Output:



Exercise 2:

Change other style properties of the paragraph tag (e.g., font-size, line-height, font-family, contents, …)

Code: Lab1Ex02.html (available in the attached zip file)

Snippet:
Sample text is added to the paragraph section with Id's and class names defined. Using the d3.select function, style and attributes of the paragraph text are updated using the class names and Id's.

```
// Changing the font style thru testclass with SelectAll command
d3.selectAll(".testclass1").style("font-style", "italic");

// Changing the font color, size, height and family thru testclass with Select command
d3.select(".testclass1")
  .style("color", "blue")
  .style("font-size", "xx-large")
  .style("line-height", 3)
  .style("font-family", "courier");

// Changing the font color thru id with Select command
d3.select("#p2").style("color", "magenta").style("text-align", "center");

// Changing the font alignment, size and family thru id with Select command
d3.select("#p3")
  .style("text-align", "right")
  .style("font-size", "xx-large")
  .style("font-family", "Brush Script MT");
```

Output:

Exercise 3:

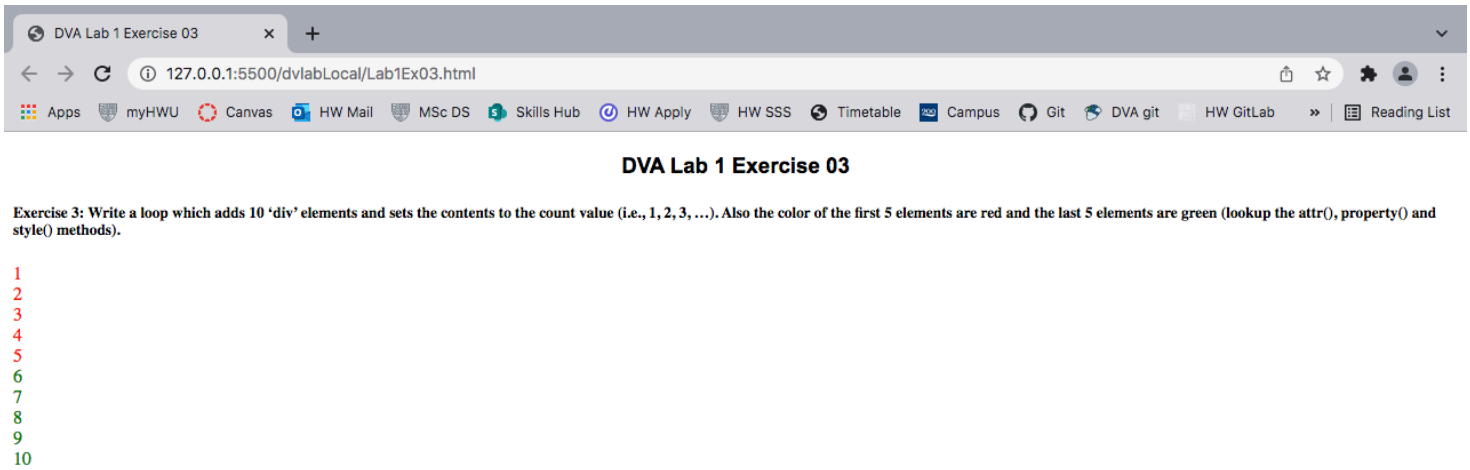
Write a loop which adds 10 'div' elements and sets the contents to the count value (i.e., 1, 2, 3, …). Also the color of the first 5 elements are red and the last 5 elements are green (lookup the attr(), property() and style() methods).

Code: Lab1Ex03.html (available in the attached zip file)

Snippet:

```
<script>
  //div elements are added in loop and color set as per the need.
  for (let i = 1; i <= 10; i++) {
    if (i <= 5) {
      d3.select("body").append("div").text(i).style("color", "red");
    } else {
      d3.select("body").append("div").text(i).style("color", "green");
    }
  }
</script>
```

Output:



Exercise 4:

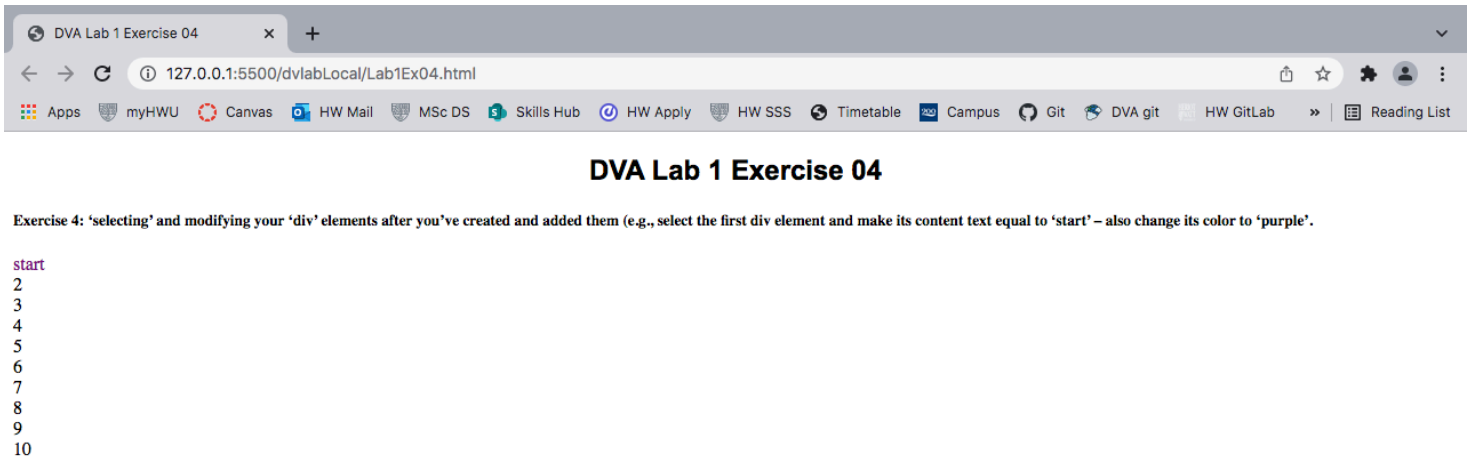
'selecting' and modifying your 'div' elements after you've created and added them (e.g., select the first div element and make its content text equal to 'start' – also change its color to 'purple'.

Code: Lab1Ex04.html (available in the attached zip file)

Snippet:

```
<script>
  //Div element created with default black color
  for (let i = 1; i <= 10; i++) {
    d3.select("body").append("div").text(i);
  }
  //div elements are assigned a new color.
  d3.select("div").text("start").style("color", "purple");
</script>
```

Output:



Exercise 5:

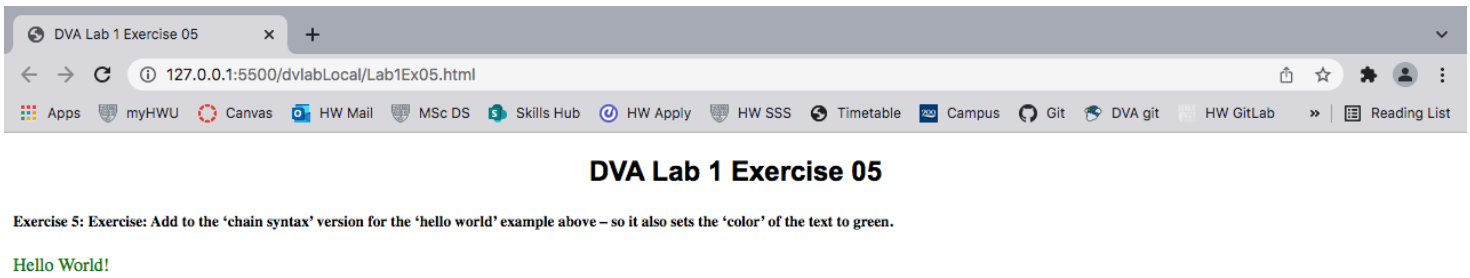
Add to the 'chain syntax' version for the 'hello world' example above – so it also sets the 'color' of the text to green.

Code: Lab1Ex05.html (available in the attached zip file)

Snippet:

```
<script>
  //d3 statement constructed as chain syntax
  d3.select("body")
    .append("div")
    .text("Hello World!")
    .style("color", "green");
</script>
```

Output:



Part 3. Data

Exercise 6

Exercise 6: Modify the example above so the 'otherdata' contains an additional variable called color (print this color value out in the 'text' method.

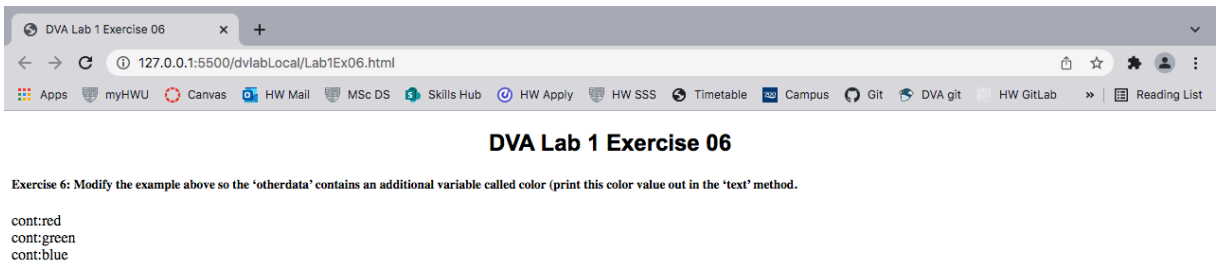
Code: Lab1Ex06.html (available in the attached zip file)

Snippet:

```
<script>
  let otherdata = [
    { name: "test", val: 1, color: "red" },
    { name: "other", val: 2, color: "green" },
    { name: "b", val: 3, color: "blue" },
  ];

  let paragraph = d3
    .select("body")
    .selectAll("div")
    .data(otherdata)
    .text(function (d, i) {
      console.log("d.name: " + d.name);
      console.log("d.val: " + d.val);
      console.log("d.val: " + d.color);
      console.log("i: " + i);
      console.log("this: " + this);
      return "cont:" + d.color; // return value is used to set the 'text'
    });
</script>
```

Output:



Exercise 7:

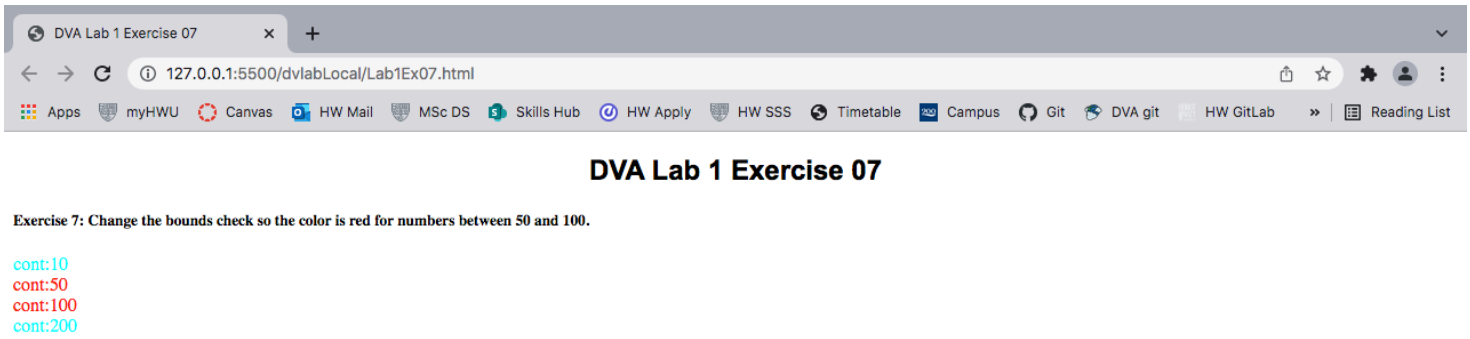
Change the bounds check so the color is red for numbers between 50 and 100.

Code: Lab1Ex07.html (available in the attached zip file)

Snippet:

```
<script>
  let num = [10, 50, 100, 200];
  let paragraph = d3
    .select("body")
    .selectAll("div")
    .data(num)
    .text(function (d, i) {
      return "cont:" + d; // return value is used to set the 'text'
    })
    .style("color", function (d, i) {
      if (d >= 50 && d <= 100) {
        return "red";
      } else {
        return "aqua";
      }
      return "blue";
    });
</script>
```

Output:



Part 4. Data Binding

Exercise 8:

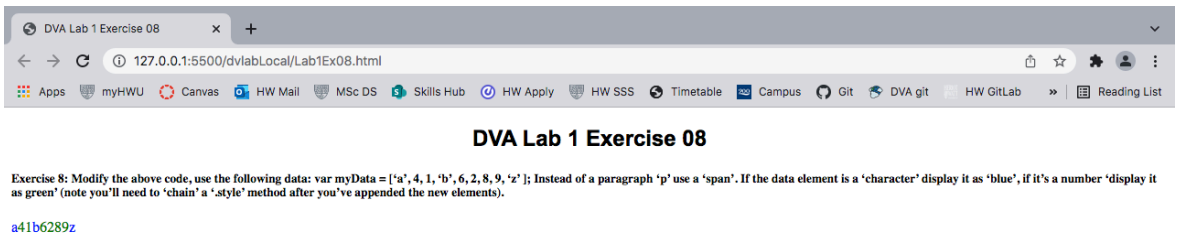
Modify the above code, use the following data:
var myData = [‘a’, 4, 1, ‘b’, 6, 2, 8, 9, ‘z’];
Instead of a paragraph ‘p’ use a ‘span’. If the data element is a ‘character’ display it as ‘blue’, if it’s a number ‘display it as green’ (note you’ll need to ‘chain’ a ‘.style’ method after you’ve appended the new elements).

Code: Lab1Ex08.html (available in the attached zip file)

Snippet:

```
<script>
  var myData = ["a", 4, 1, "b", 6, 2, 8, 9, "z"];
  var span = d3
    .select("body")
    .selectAll("span")
    .data(myData)
    .enter()
    .append("span")
    .text(function (d) {
      return d;
    })
    .style("color", function (d) {
      if (typeof d == "string") return "blue";
      else return "green";
    });
</script>
```

Output:



Part 5. Loading Data

Exercise 9:

For the example above, to count how many of the names include 'Mr.' and 'Mrs' (or other). Also print out other details using other column header information, such as, how many passengers are 'male' and how many 'female'.

Code: Lab1Ex09.html (available in the attached zip file)

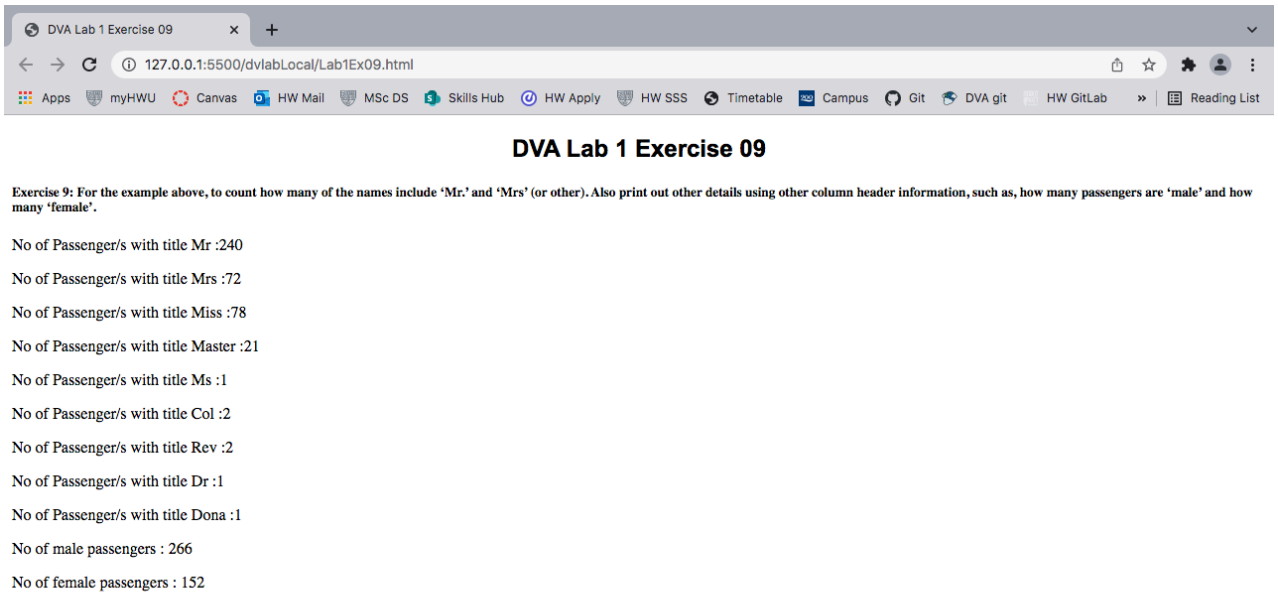
CSV file used: https://raw.githubusercontent.com/dsindy/kaggle-titanic/master/data/test.csv

Snippet:

Based on the data imported from the CSV file, two arrays are created for title and sex and each array will have an element for its different instances, each data item is scanned for title and sex, and appropriate array item is incremented for its count.

```
        for (var i = 0; i < data.length; i++) {
          titlesearch = data[i].Name.substring(
            data[i].Name.indexOf(", ") + 2,
            data[i].Name.indexOf(".")
          );
          /* Each unique item is stored as a key identifier and its count is stored as the value of
the item */
          titlecount[titlesearch] = 1 + (titlecount[titlesearch] || 0);
          sexcount[data[i].Sex] = 1 + (sexcount[data[i].Sex] || 0);
        }
```

Output:



Exercise 10:

Write an update to the example above, so extra elements are added to the window to display information. For instance, display paragraphs for the total patients with heart failure between 1-30, 31-40, 41-60, 61-100. Process the data, store it in an array then pass that array to 'selectAll()', 'data()' as discussed in previous sections.

Code: Lab1Ex10.html (available in the attached zip file)

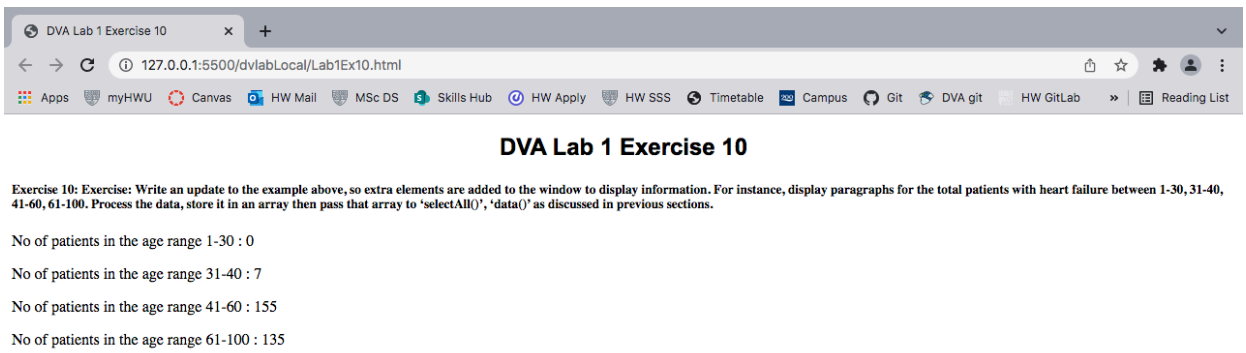
CSV file used: https://raw.githubusercontent.com/akmand/datasets/master/heart_failure.csv

Snippet:

Based on the data imported from the CSV file, an array is created for age range and with different ranges as element name. Each data item is scanned for its age, and appropriate age range element value is incremented for its count.

```
        //new agerange is determined and the count is added to the corresponding items in the list.
        for (var i = 0; i < data.length; i++) {
          if (Number(data[i]) >= 1 && Number(data[i]) <= 30) {
            agerange["1-30"] += 1;
          } else if (Number(data[i]) >= 31 && Number(data[i]) <= 40) {
            agerange["31-40"] += 1;
          } else if (Number(data[i]) >= 41 && Number(data[i]) <= 60) {
            agerange["41-60"] += 1;
          } else if (Number(data[i]) >= 61 && Number(data[i]) <= 100) {
            agerange["61-100"] += 1;
          }
        }
```

Output:



Part 6. SVG

Exercise 11:

Exercise: Modify the code so the example draws a 'square shape' (4 lines) – each side of the square a different color.

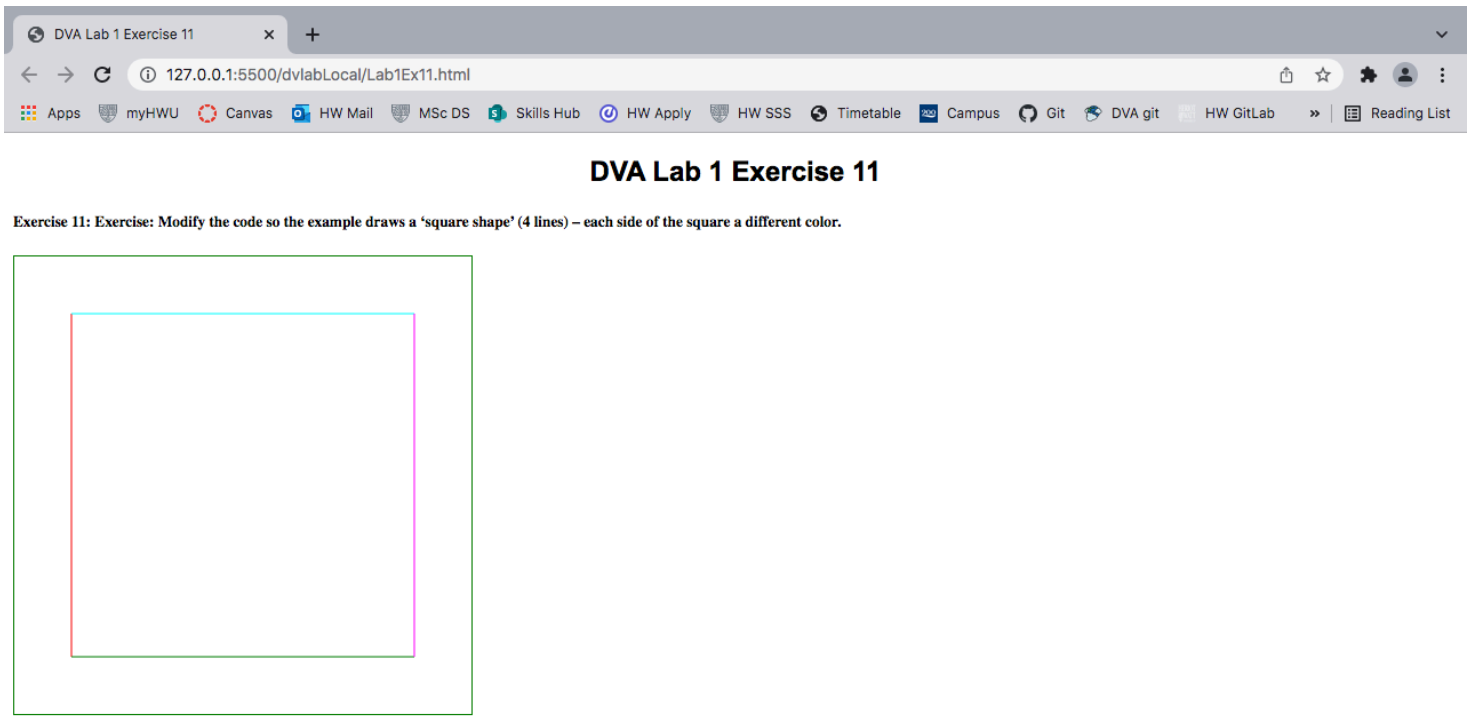
Code: Lab1Ex11.html (available in the attached zip file)

Snippet:

```
    svg
       .append("line")
       .attr("x1", 50)
       .attr("y1", 50)
       .attr("x2", 50)
       .attr("y2", 350)
       .attr("stroke", "red");

     svg
       .append("line")
       .attr("x1", 50)
       .attr("y1", 50)
       .attr("x2", 350)
       .attr("y2", 50)
       .attr("stroke", "cyan");
     svg
       .append("line")
       .attr("x1", 50)
       .attr("y1", 350)
       .attr("x2", 350)
       .attr("y2", 350)
       .attr("stroke", "green");
     svg
       .append("line")
       .attr("x1", 350)
       .attr("y1", 50)
       .attr("x2", 350)
       .attr("y2", 350)
       .attr("stroke", "magenta");
```

Output:


DVA Lab 1 Exercise 11
127.0.0.1:5500/dvlabLocal/Lab1Ex11.html
Apps
myHWU
Canvas
HW Mail
MSc DS
Skills Hub
HW Apply
HW SSS
Timetable
Campus
Git
DVA git
HW GitLab
Reading List
DVA Lab 1 Exercise 11
Exercise 11: Exercise: Modify the code so the example draws a 'square shape' (4 lines) -- each side of the square a different color.

Exercise 12:

Build an SVG scene which is created from an external file. You need to create a csv with the information about the shapes. You should include columns in your csv file for the type of shape (circle, rectangle, ellipse, line), its dimensions and position, and color. Your program reads the data and creates and displays the shapes to the screen.

Code: Lab1Ex12.html (available in the attached zip file)

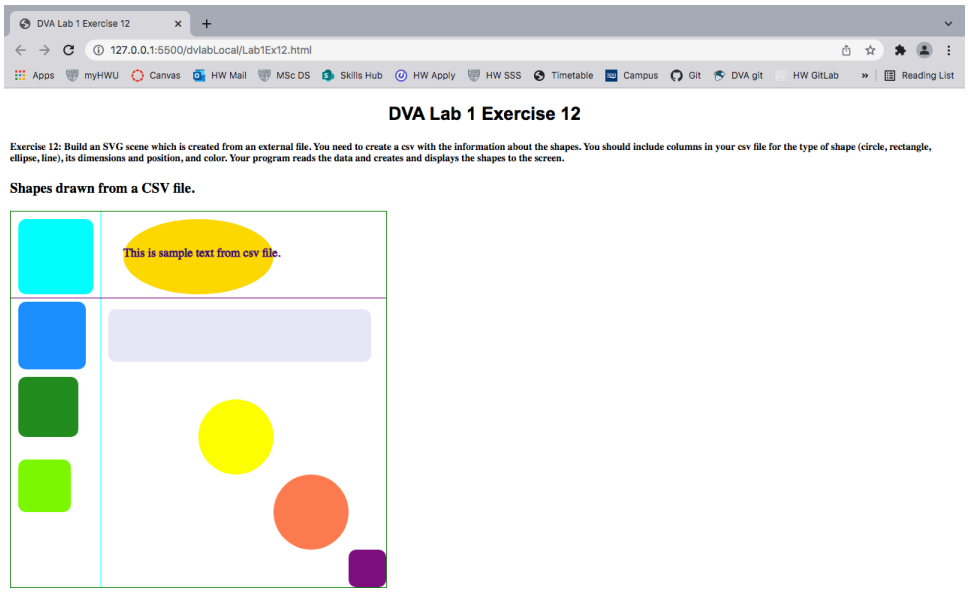
CSV file used: shapes.csv (available in the attached zip file)

Snippet:

Based on the shape details imported from the CSV file, an append statement is constructed for each shape and drawn.

```
d3.csv(shapes, function (data) {
        return data;
      }).then(function (data) {
        console.log(data);
        for (let x in data) {
          let str = "";
          for (let y in data[x]) {
            if (data[x][y] != "") {
              if (y == "Shape") {
                str = 'svg.append("' + data[x]["Shape"] + '")';
              } else if (y == "text") {
                str += '.text("' + data[x][y] + '")';
              } else {
                str += '.attr("' + y + '","' + data[x][y] + '")';
              }
            }
          }
          console.log(str);
          Function(str)();
        }
```

Output:



Exercise 13:

Exercise 13: Extend the example to include the 'enter' and 'exit' concepts. So that the svg elements are updated, created or removed based on the csv data from your csv file.

Code: Lab1Ex13.html (available in the attached zip file)

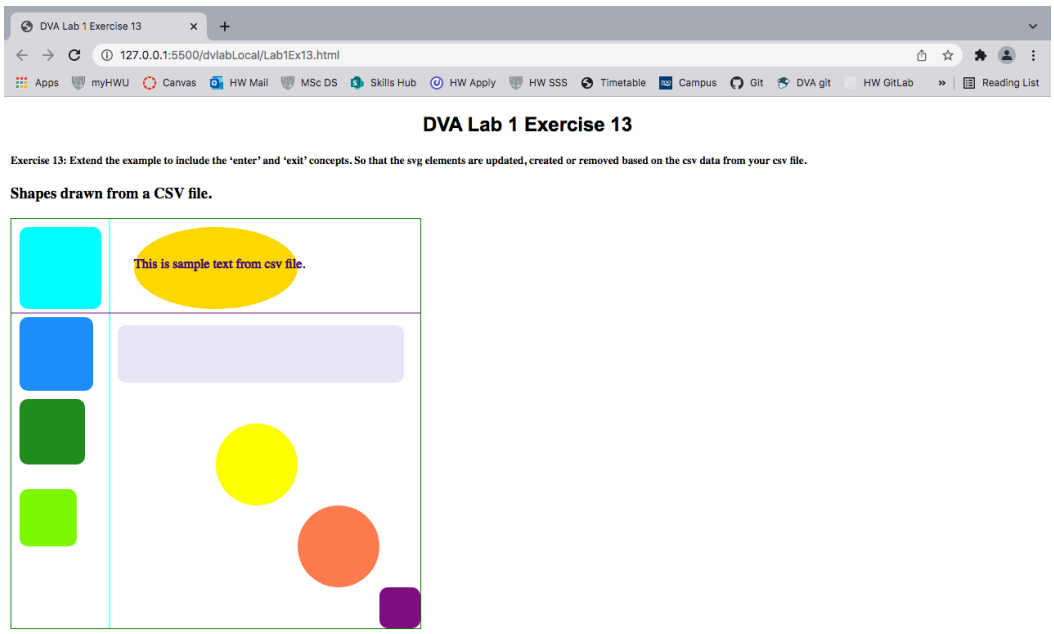
CSV file used: shapes.csv (available in the attached zip file)

Snippet:

Based on the shape details imported from the CSV file, an append statement is constructed for each shape and drawn.

```
d3.csv(shapes, function (data) {
        return data;
      }).then(function (data) {
        console.log(data);
        for (let x in data) {
          let str = "";
          for (let y in data[x]) {
            if (data[x][y] != "") {
              if (y == "Shape") {
                str = 'svg.append("' + data[x]["Shape"] + '")';
              } else if (y == "text") {
                str += '.text("' + data[x][y] + '")';
              } else {
                str += '.attr("' + y + '","' + data[x][y] + '")';
              }
            }
          }
          console.log(str);
          Function(str)();
        }
```

Output:



Part 7. Bar Chart

Exercise 14:

Extend the simple bar chart example to display the heart failure data you processed in Part 5 (Part 5 - Loading Data) from the csv file. (i.e., age ranges for people with heart failure).

Code: Lab1Ex14.html (available in the attached zip file)

CSV file used: https://raw.githubusercontent.com/akmand/datasets/master/heart_failure.csv

Snippet:

Based on the details imported from the CSV file, various age range info and its count are calculated and store in an array. A bar chart is created and drawn from the data available in the age range array.

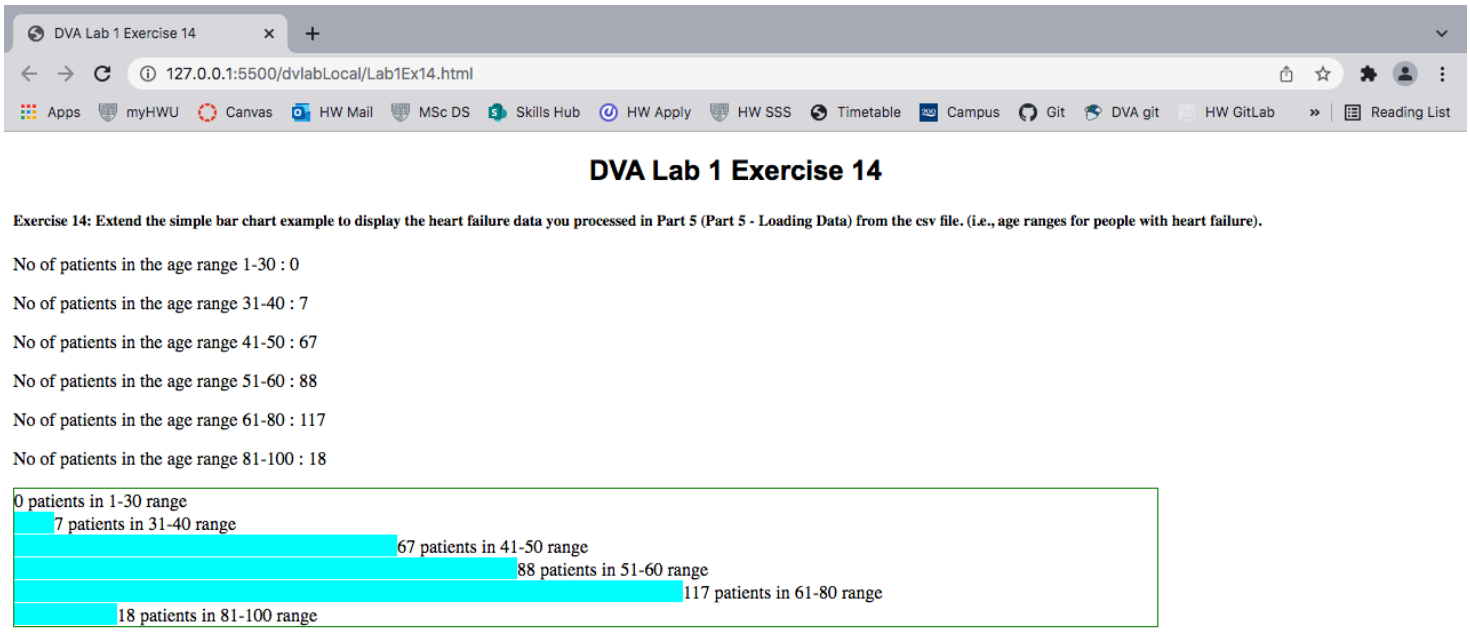
```
d3.csv(heartfaillocal, function (data) {
        //d3.csv(heartfailurecsv, function (data) {
        return (data = +data.age);
      }).then(function (data) {
        for (var i = 0; i < data.length; i++) {
          if (Number(data[i]) >= 1 && Number(data[i]) <= 30) {
            agerange["1-30"] += 1;
          } else if (Number(data[i]) >= 31 && Number(data[i]) <= 40) {
            agerange["31-40"] += 1;
          } else if (Number(data[i]) >= 41 && Number(data[i]) <= 50) {
            agerange["41-50"] += 1;
          } else if (Number(data[i]) >= 51 && Number(data[i]) <= 60) {
            agerange["51-60"] += 1;
          } else if (Number(data[i]) >= 61 && Number(data[i]) <= 80) {
            agerange["61-80"] += 1;
          } else if (Number(data[i]) >= 81 && Number(data[i]) <= 100) {
            agerange["81-100"] += 1;
          }
        }

        let paragraph = d3.select("body").selectAll("p").data(data);
        for (let x in agerange) {
          console.log(
            "No of patients in the age range " + x + " : " + agerange[x]
          );
          keyArray[barindex] = x;
          dataArray[barindex] = agerange[x];
          ++barindex;

          paragraph
            .append("p")
            .text("No of patients in the age range " + x + " : " + agerange[x]);
        }

        drawagerange();
      });
```

Output:


DVA Lab 1 Exercise 14
127.0.0.1:5500/dvlabLocal/Lab1Ex14.html
Apps
myHWU
Canvas
HW Mail
MSc DS
Skills Hub
HW Apply
HW SSS
Timetable
Campus
Git
DVA git
HW GitLab
Reading List
DVA Lab 1 Exercise 14
Exercise 14: Extend the simple bar chart example to display the heart failure data you processed in Part 5 (Part 5 - Loading Data) from the csv file. (i.e., age ranges for people with heart failure).
No of patients in the age range 1-30 : 0
No of patients in the age range 31-40 : 7
No of patients in the age range 41-50 : 67
No of patients in the age range 51-60 : 88
No of patients in the age range 61-80 : 117
No of patients in the age range 81-100 : 18
0 patients in 1-30 range
7 patients in 31-40 range
67 patients in 41-50 range
88 patients in 51-60 range
117 patients in 61-80 range
18 patients in 81-100 range

Exercise 15:

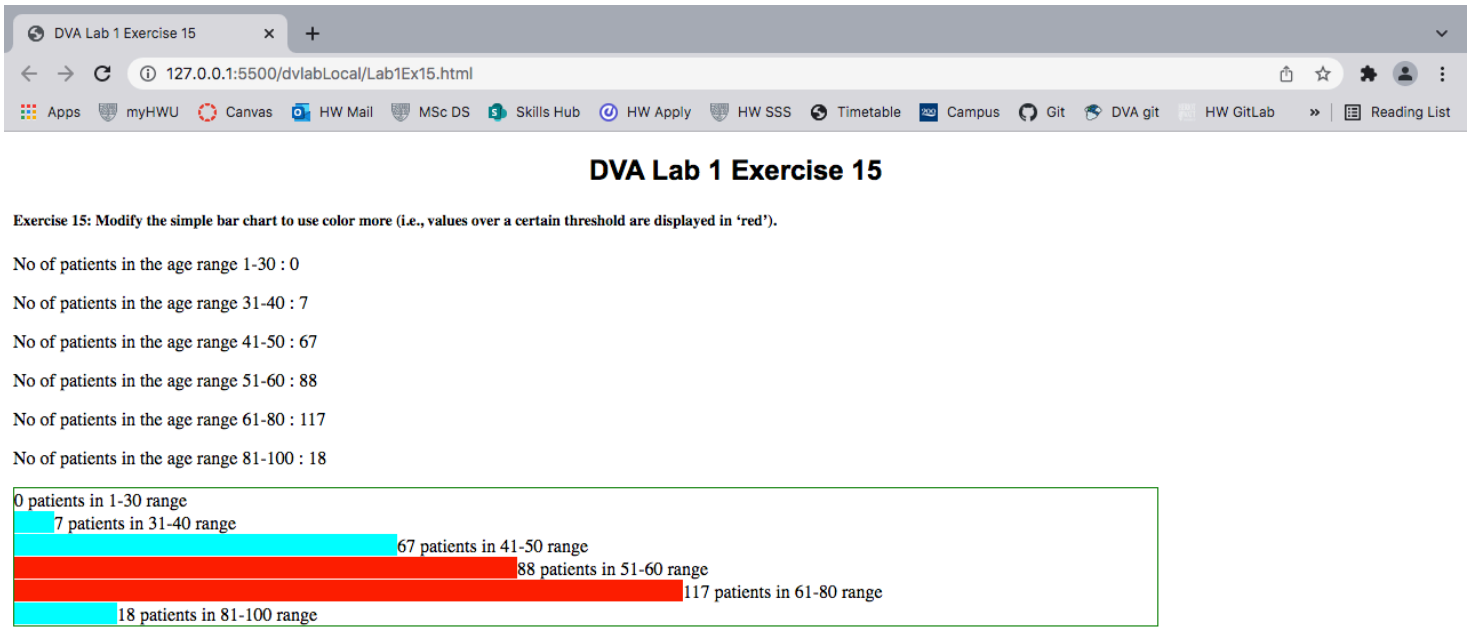
Modify the simple bar chart to use color more (i.e., values over a certain threshold are displayed in 'red').

Code: Lab1Ex15.html (available in the attached zip file)

CSV file used: https://raw.githubusercontent.com/akmand/datasets/master/heart_failure.csv

Based on the details imported from the CSV file, various age range info and its count are calculated and store in an array. A bar chart is created and drawn from the data available in the age range array. Individual bars crossing the threshold of 80(count) are colored in red.

Output:



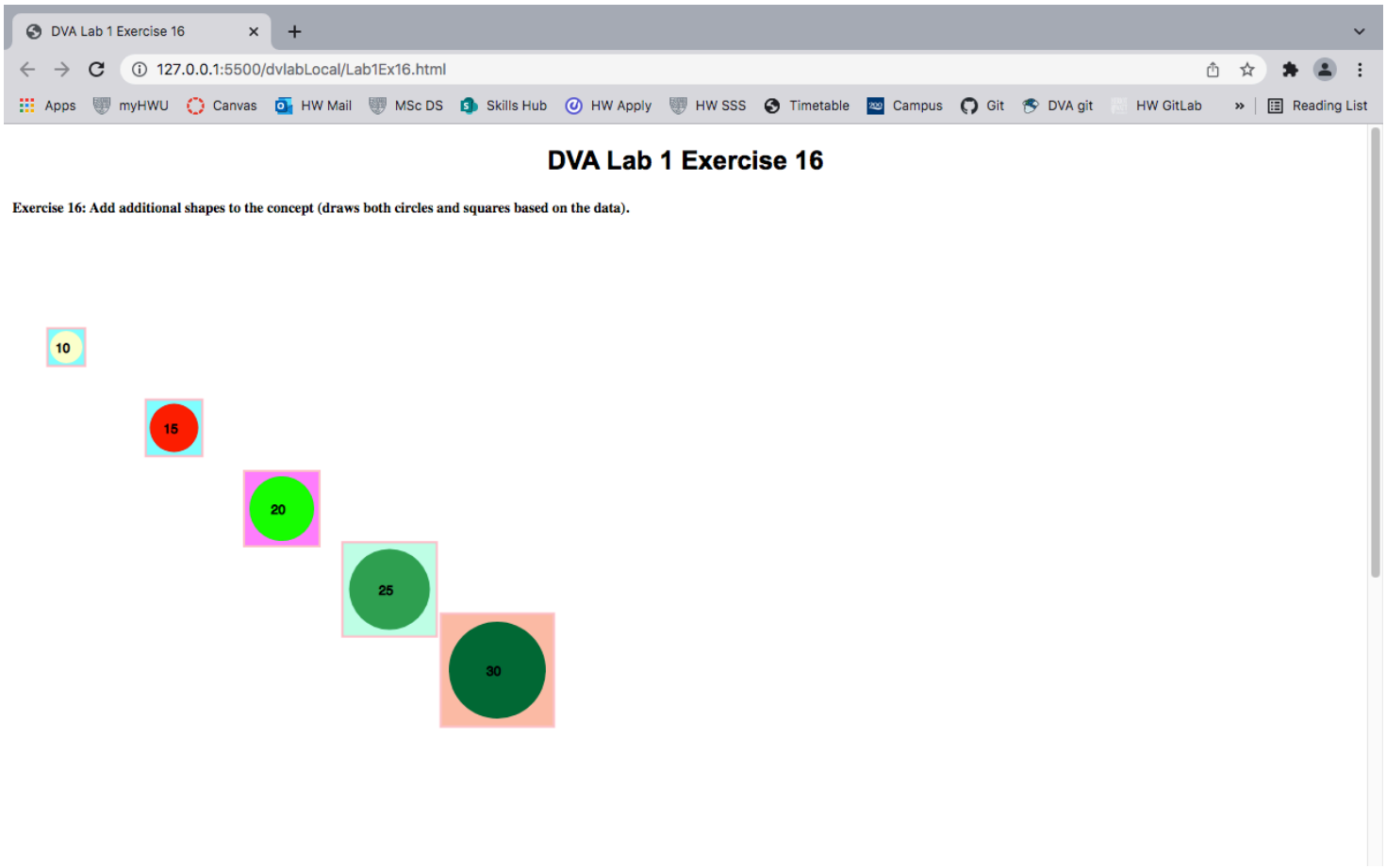
Part 8. Circle Chart

Exercise 16:

Add additional shapes to the concept (draws both circles and squares based on the data).

Code: Lab1Ex16.html (available in the attached zip file)

Output:

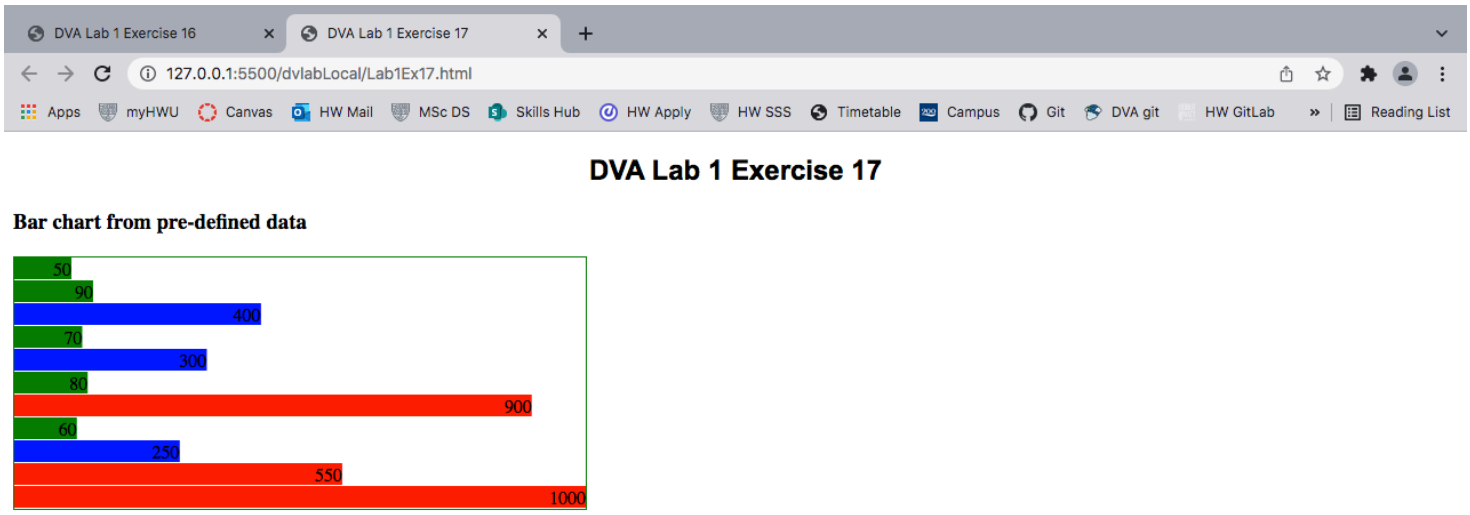


Exercise 17:

Modify the example above so the bars are green if below 100 and red if above 500.

Code: Lab1Ex17.html (available in the attached zip file)

Output:



Exercise 18:

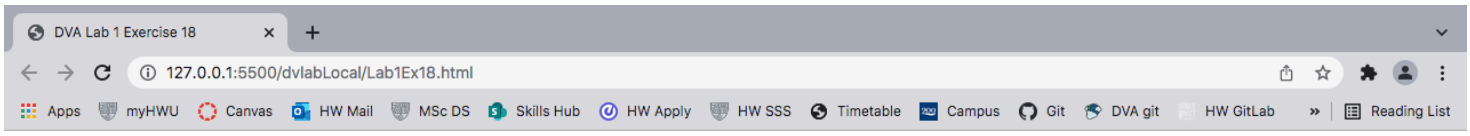
Extend the example, so the 'bar chart' data is displayed from an external file (e.g., csv).

Code: Lab1Ex18.html (available in the attached zip file)

CSV file used: ex18bardata.csv (available in the attached zip file)

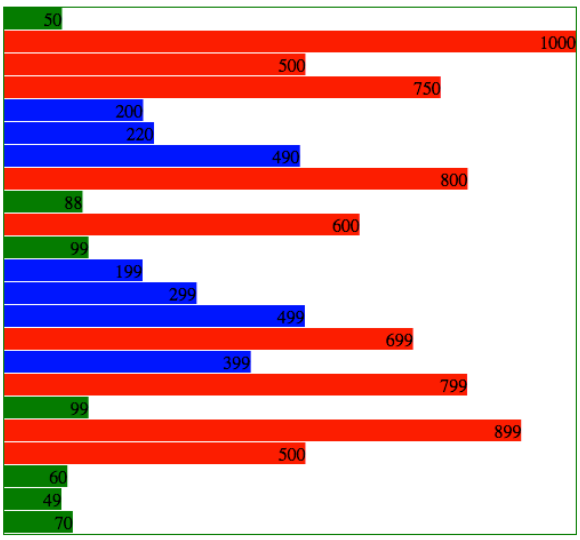
Bar chart is prepared and drawn based on the data imported from the CSV file.

Output:



DVA Lab 1 Exercise 18

Bar chart from CSV File



Exercise 19:

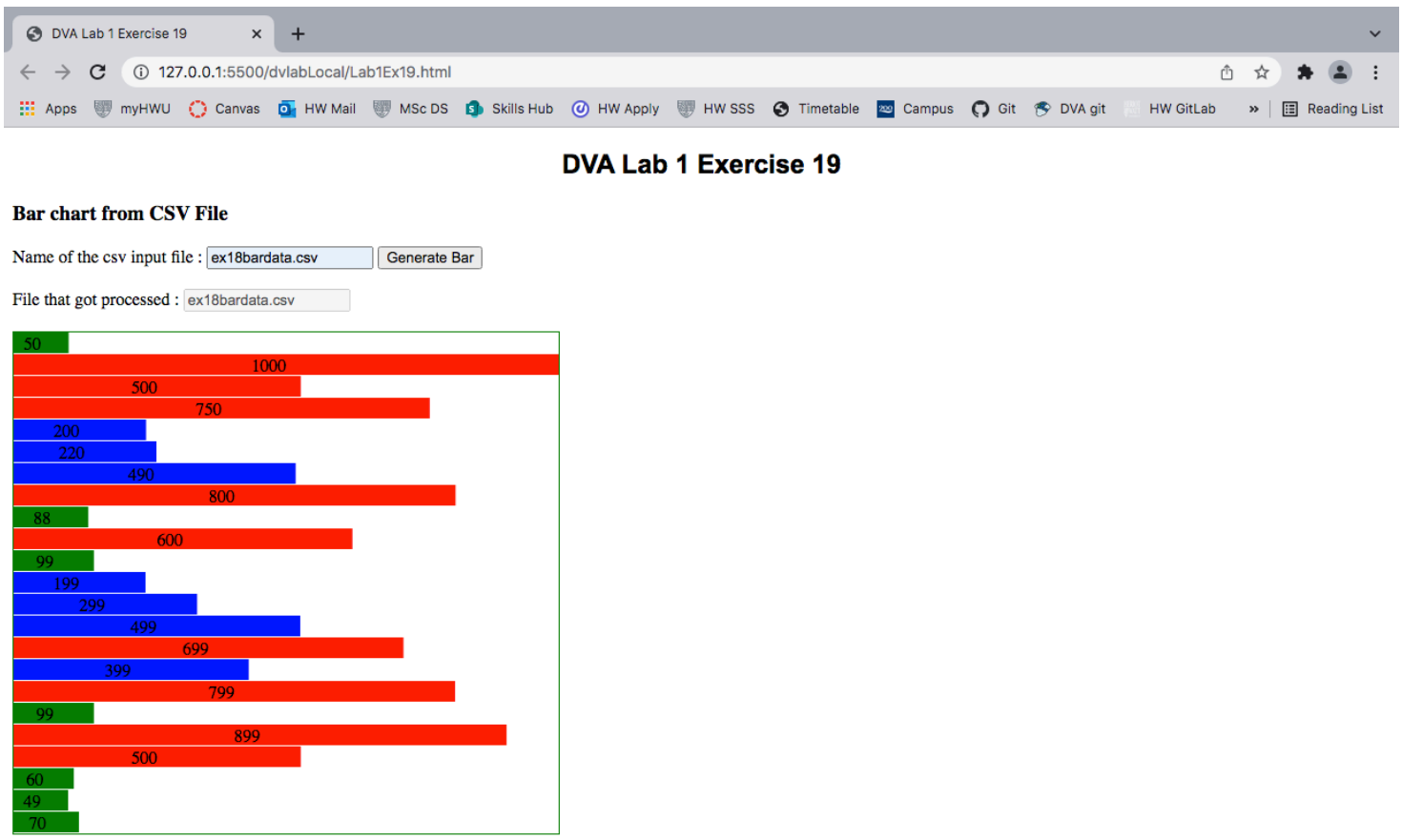
Put the code in a 'function' so the bar chart is only displayed when the function is called. Also if the function is called twice, then it will show the bar chart twice on screen. Extend this function so it takes a 'csv' file name as the input argument. Call it twice and it displays two different bar charts using different data on screen.

Code: Lab1Ex19.html (available in the attached zip file)

CSV file used: ex18bardata.csv (available in the attached zip file)

Bar chart is prepared and drawn in a function, based on the data imported from the CSV file given as input. The function is called upon by onlick of button 'Generate Bar'.

Output:



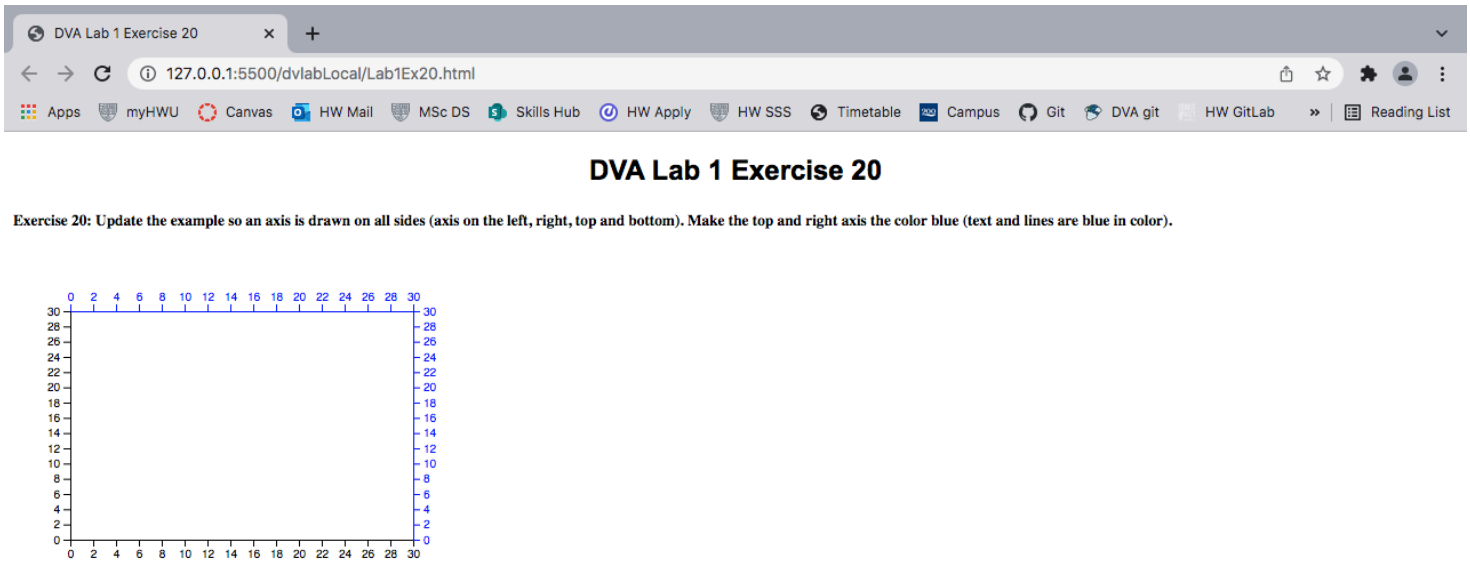
Part 10. Axis

Exercise 20:

Update the example so an axis is drawn on all sides (axis on the left, right, top and bottom). Make the top and right axis the color blue (text and lines are blue in color).

Code: Lab1Ex20.html (available in the attached zip file)

Output:



Exercise 21:

Add an ‘axis’ to the bar chart example (bottom and left axis for the bar chart).

Code: Lab1Ex21.html (available in the attached zip file)

Output:

Part 12. Line Chart

Exercise 22:

Modify the code so it's contained within a function (pass the data to the function, so you're able to draw sine wave, cosine, or other type).

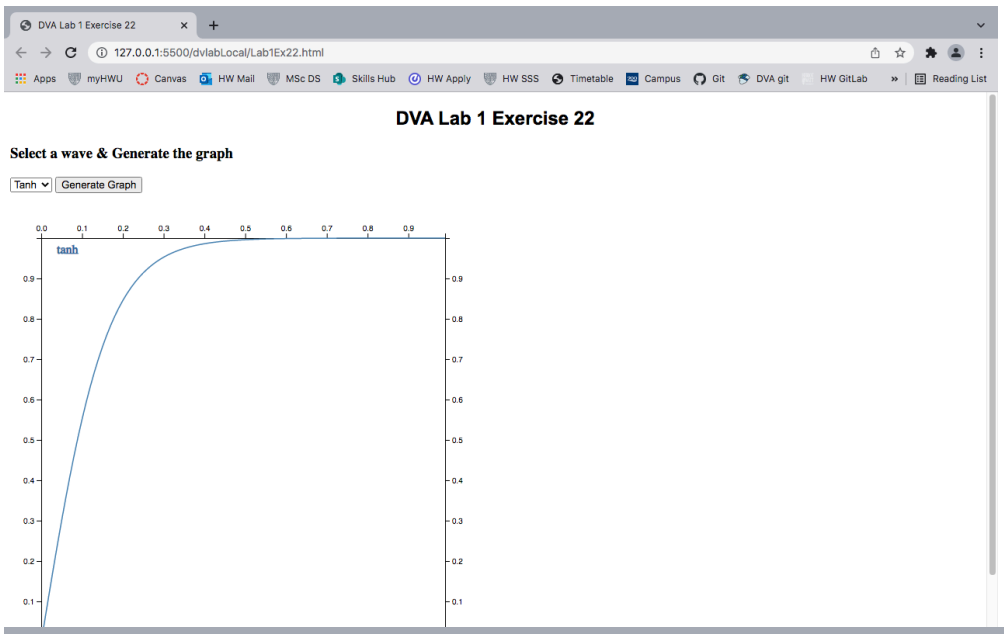
Code: Lab1Ex22.html & LabEx22 Multi.html (available in the attached zip file)

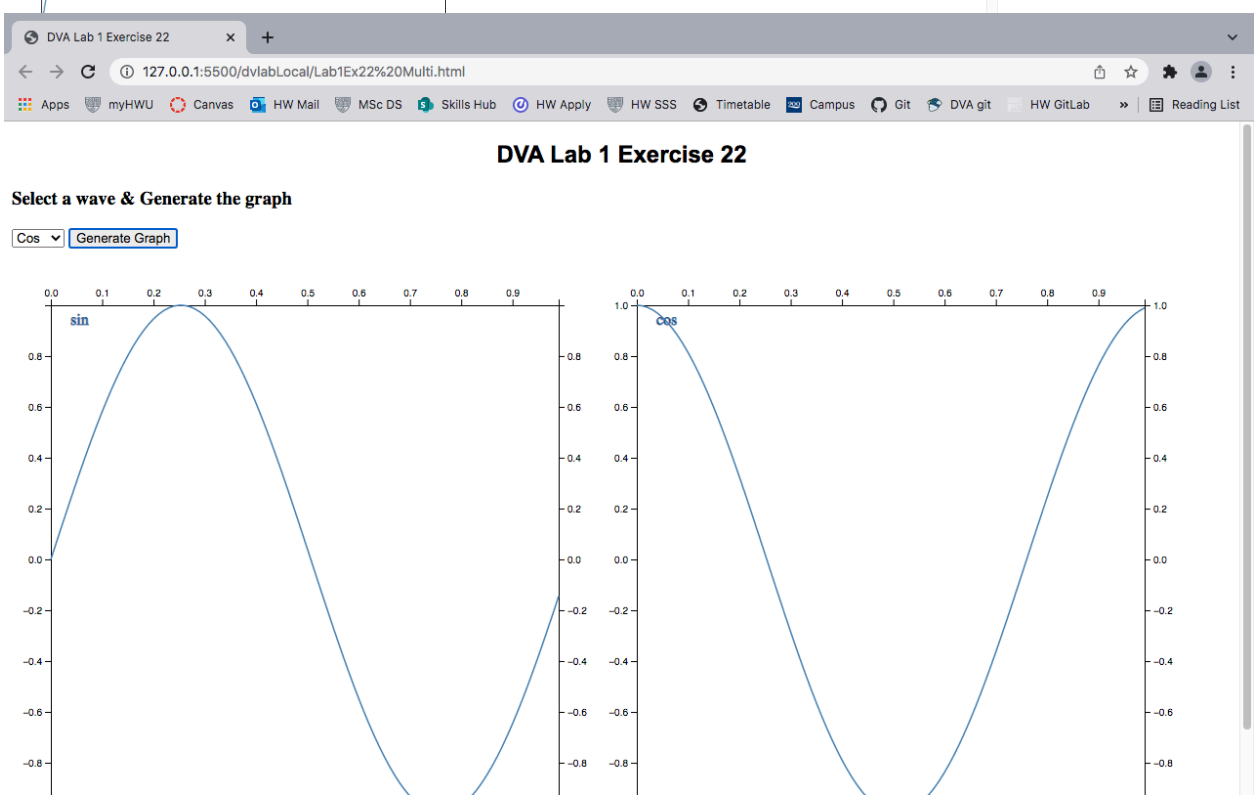
The Graph is drawn in a function which is called upon by onclick of button 'Generate Graph'.

LabEx22.html – Generates a graph with the selected wave and overwrites it when a new wave is selected.

LabEx22 Multi.html - Generates a selected wave graph and adds a new graph when a new wave is selected.

Output:



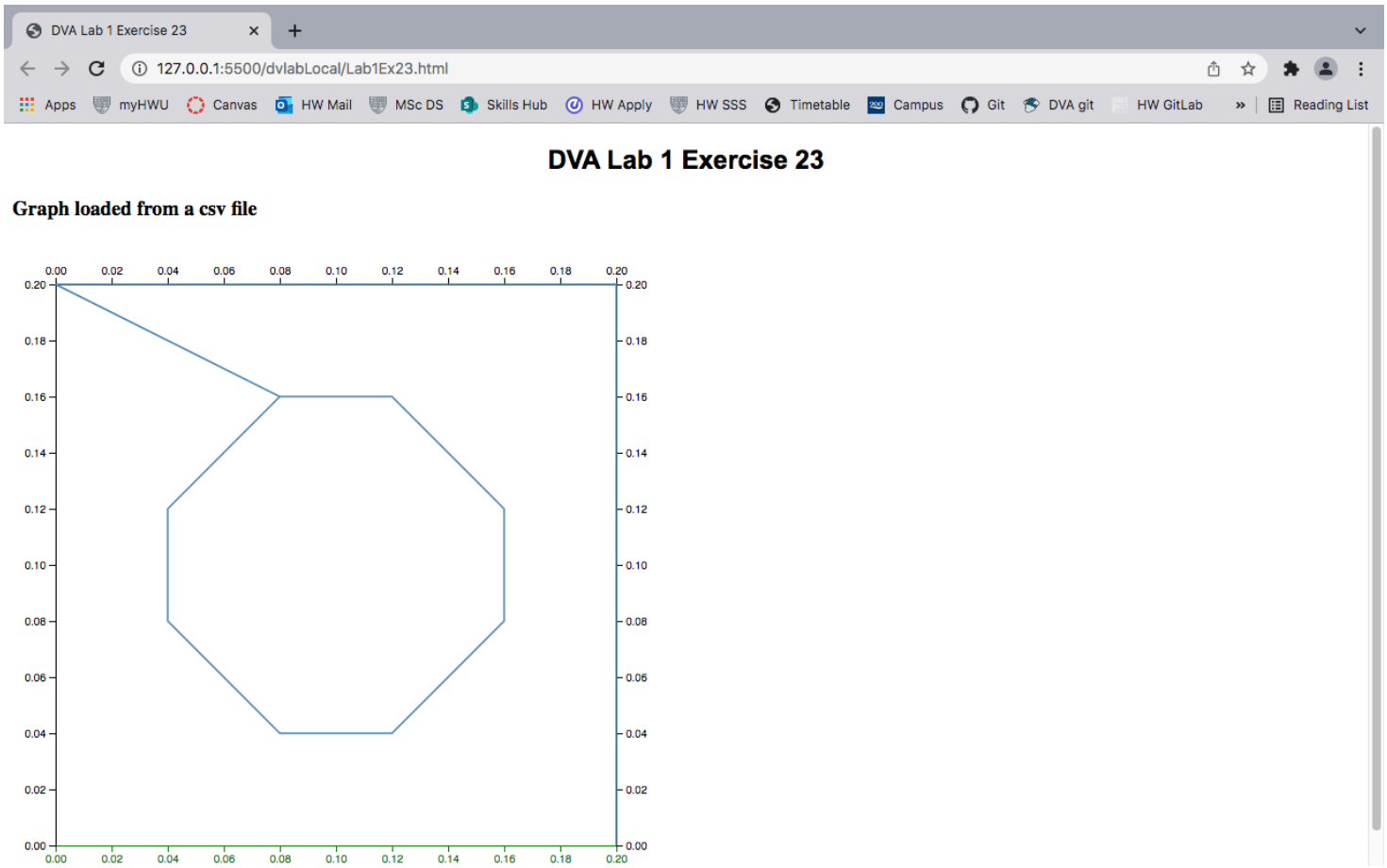


Exercise 23:

Load in some test data from a csv and plot the line (instead of 'generating' the data you load it from an external file).

Code: Lab1Ex23.html (available in the attached zip file)

Output:



Exercise 24:

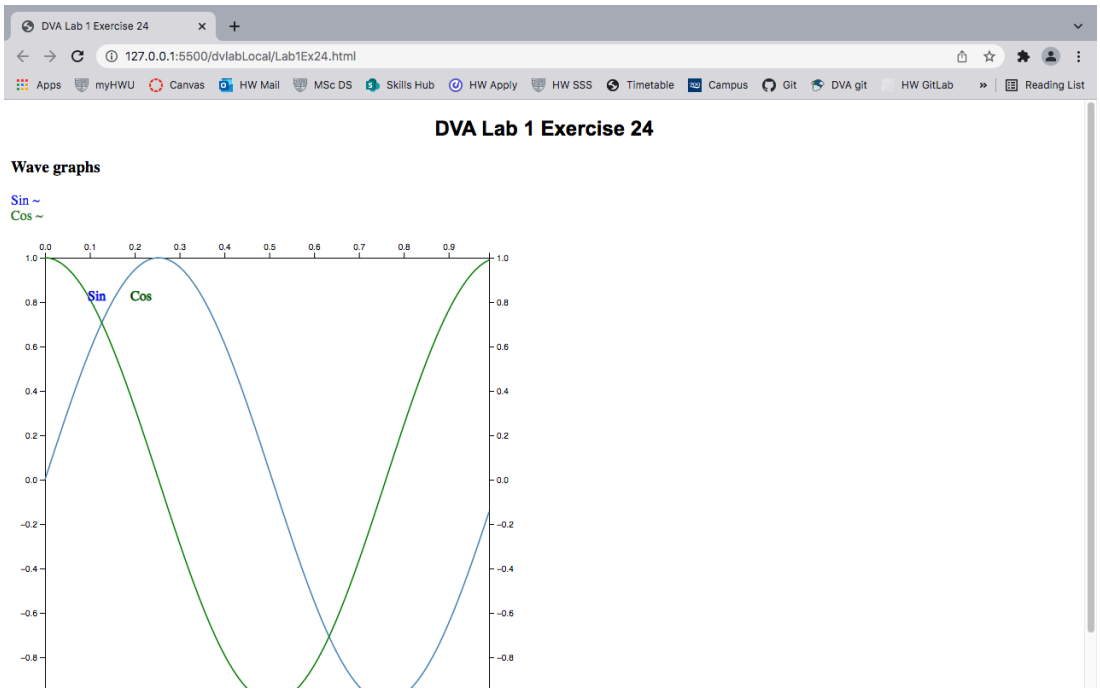
Draw multiple lines on the same chart (e.g., sinewave and a cosine wave). Make one blue and the other green.

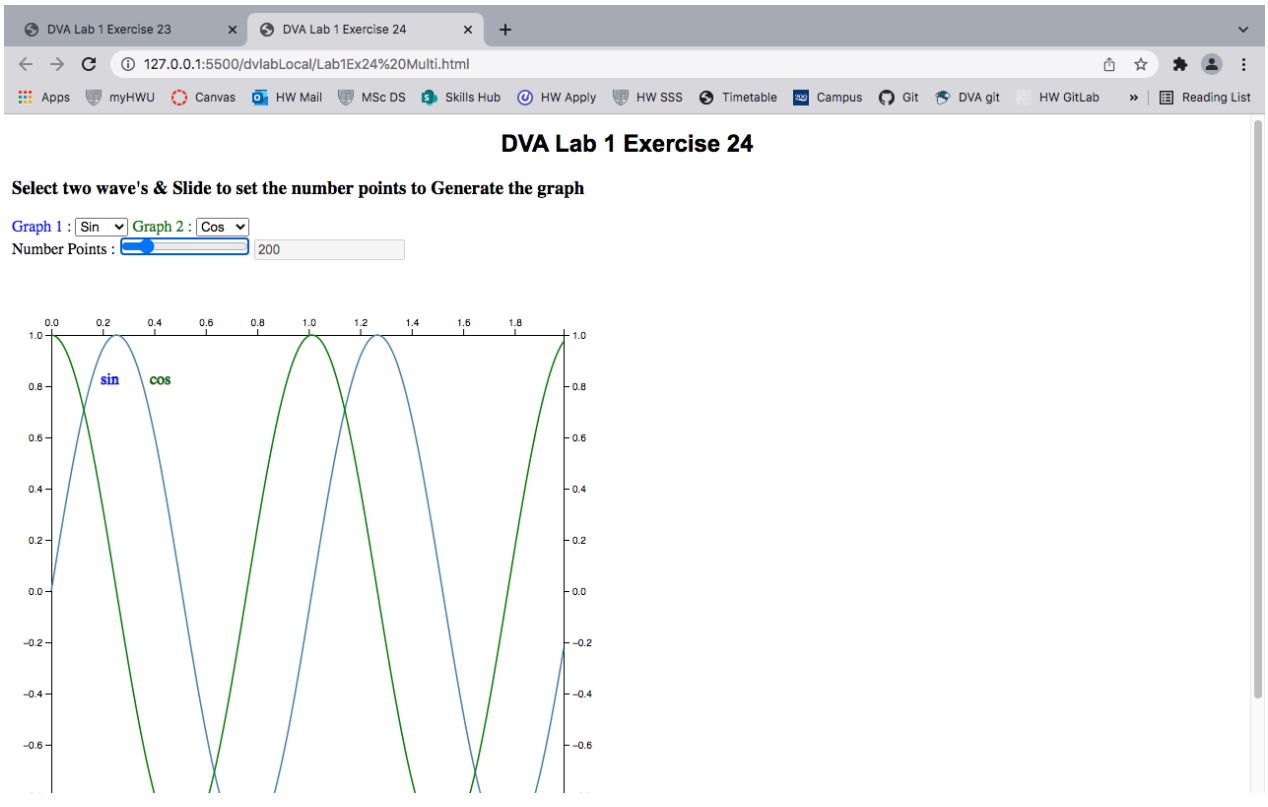
Code: Lab1Ex24.html and LabEx24 Multi.html (available in the attached zip file)

Lab1Ex24.html – Two waves, sin and cos are drawn on the same graph.

Lab1Ex24.html – Two selected waves are drawn on the same graph and are overwritten when the number point is changed over the slider. The Graph is drawn in a function which is called upon by slider change.

Output:





Part 13. Markers

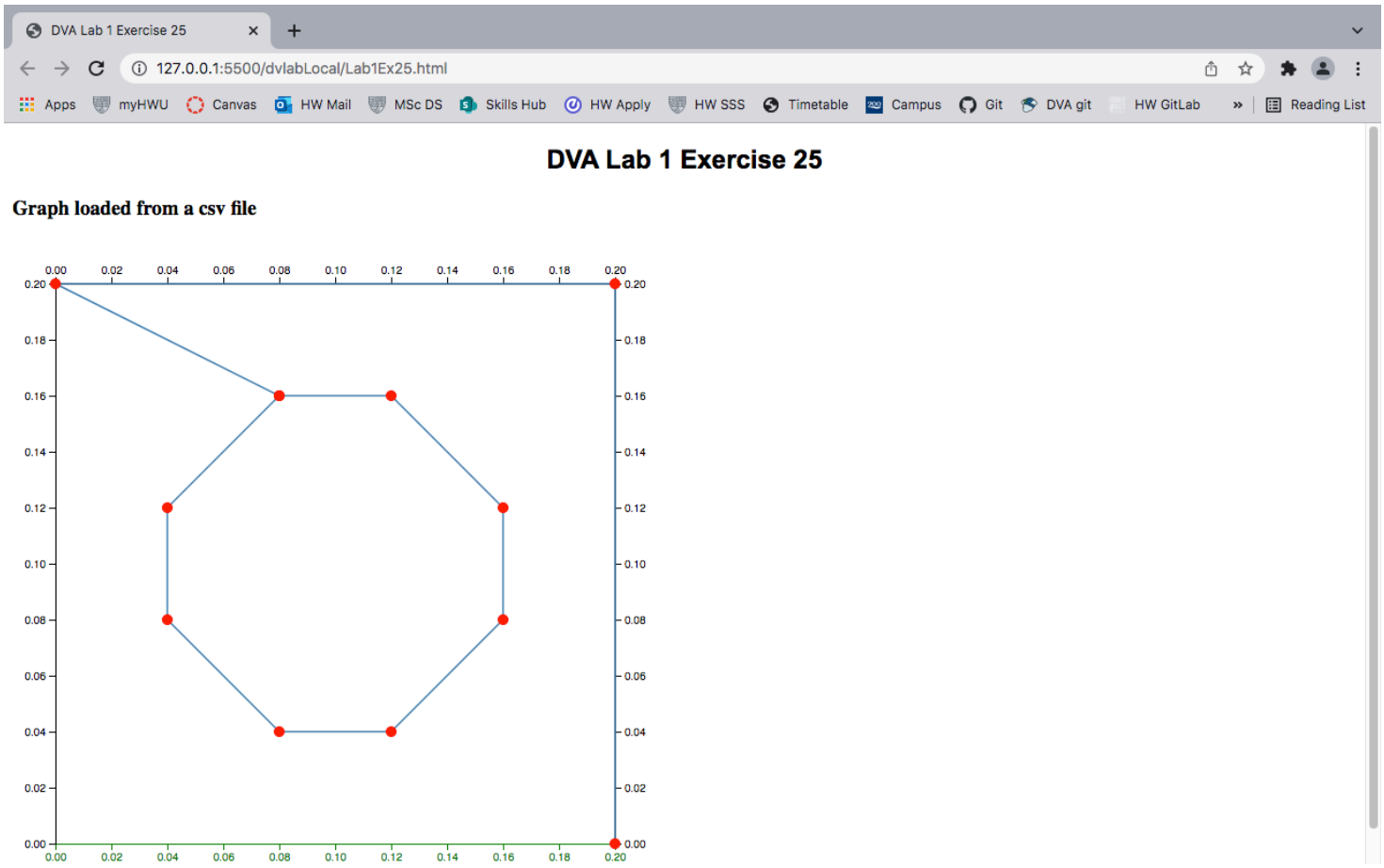
Exercise 25:

Add a 'circle point' to the line graph, so that each data point is displayed on the graph as circle.

Code: Lab1Ex25.html (available in the attached zip file)

CSV file used: wavegraph.csv (available in the attached zip file)

Output:



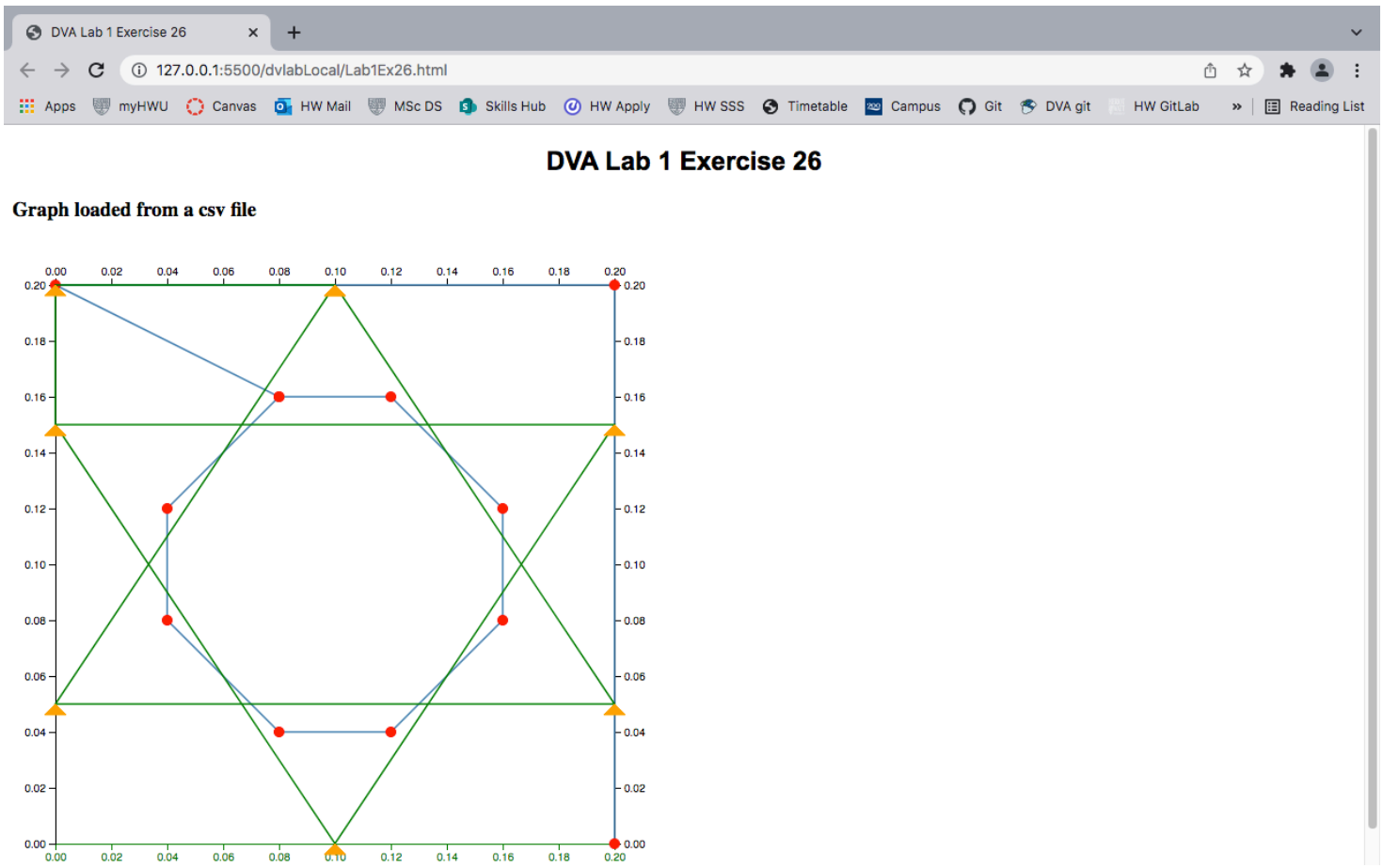
Exercise 26:

Plot two lines on the same graph, and mark the data points as circles for the first line and triangles for the second line.

Code: Lab1Ex26.html (available in the attached zip file)

CSV file used: wavegraph.csv (available in the attached zip file)

Output:



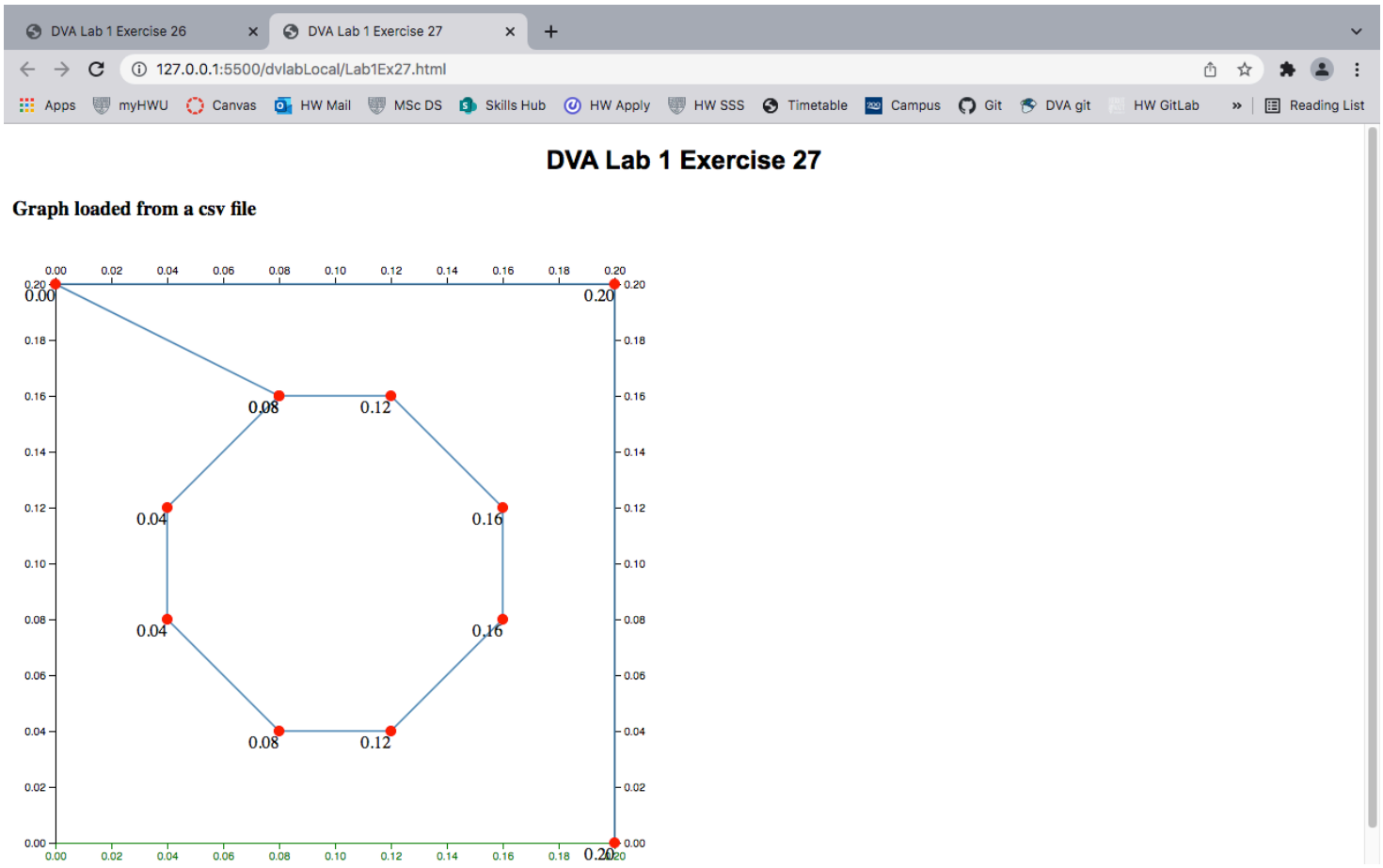
Exercise 27:

Exercise 27: Add 'text' to certain points on the line plot (e.g., next to the 'circle' dot also write the number of the data value). Limit the text information to only a few points so it doesn't get over crowded.

Code: Lab1Ex27.html (available in the attached zip file)

CSV file used: wavegraph.csv (available in the attached zip file)

Output:



Part 14. Colors

Exercise 28:

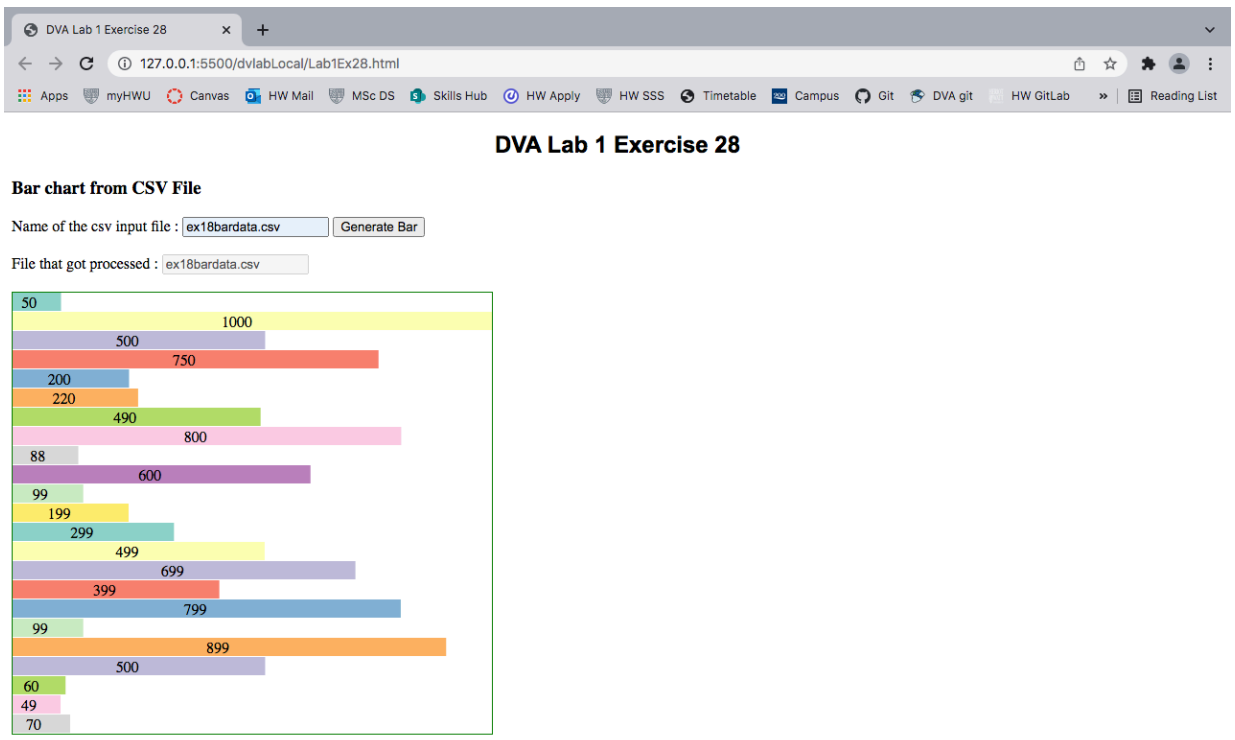
Take one of the d3 color methods and apply it to the bar chart example (from earlier). So the color is related to the value (not just fixed).

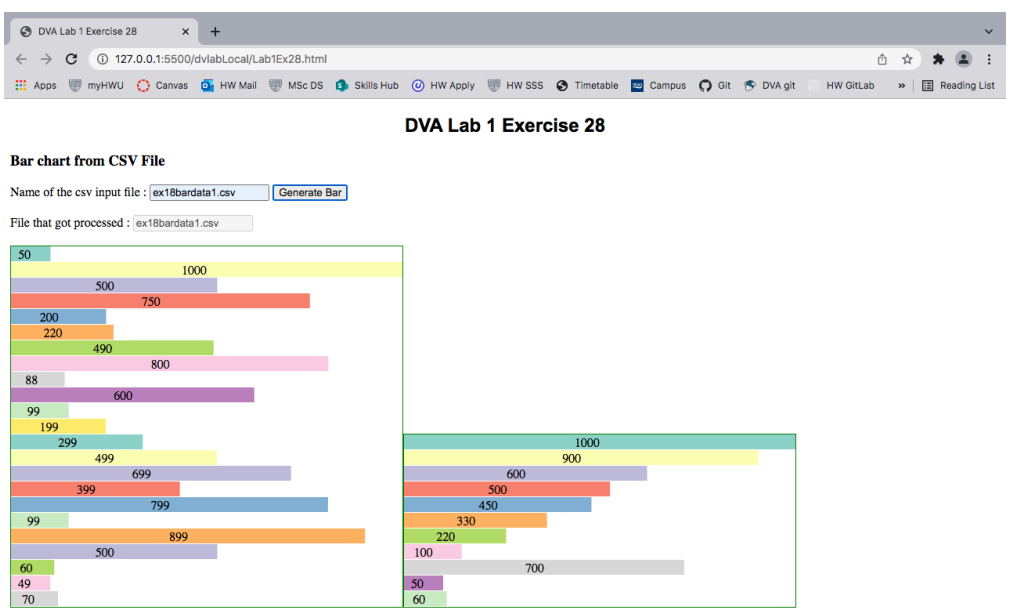
Code: Lab1Ex28.html (available in the attached zip file)

CSV file used: ex18bardata.csv & ex18bardata1.csv (available in the attached zip file)

The bar chart is drawn based on the data from the CSV file given as input and in a function, which is called upon by onclick of button 'Generate bar'. A new chart is added when a new CSV file is given.

Output:





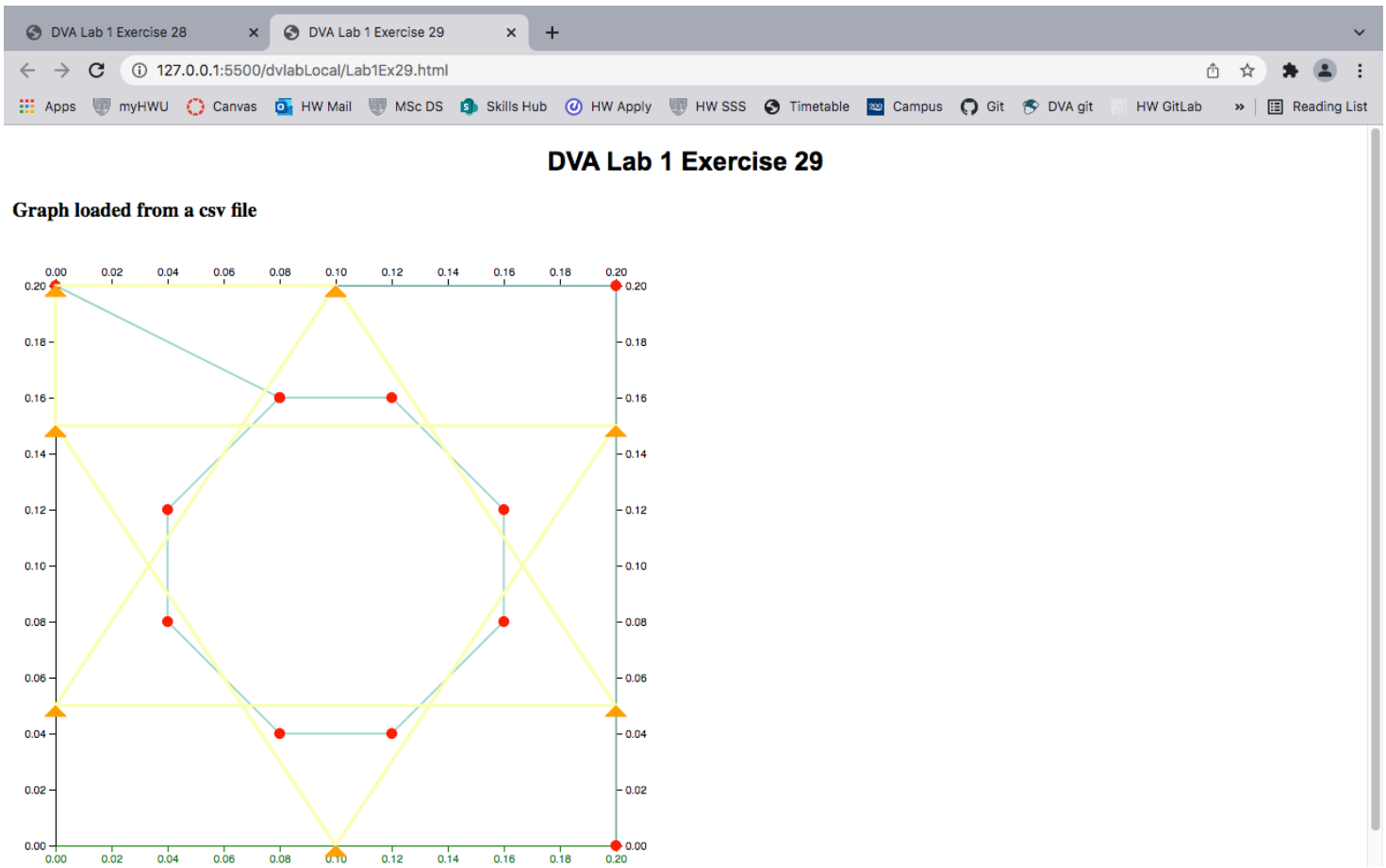
Exercise 29:

Take one of the d3 color methods and apply it to the line chart example (from earlier).

Code: Lab1Ex29.html (available in the attached zip file)

CSV file used: wavegraph.csv (available in the attached zip file)

Output:



Part 15. Pie Chart

Exercise 30:

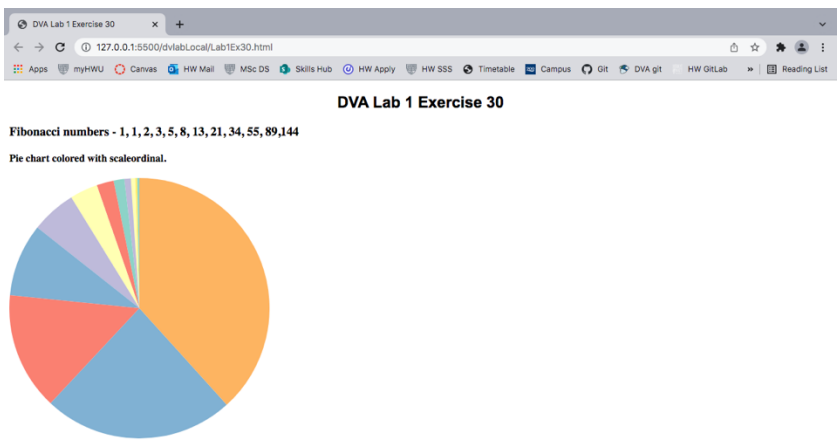
Add more data values (e.g., 12 different numbers)

Code: Lab1Ex30.html (available in the attached zip file)

Snippet:

```
    const svg = d3
      .select("body")
      .append("svg")
      .attr("width", xSize)
      .attr("height", ySize)
      .append("g")
      .attr("transform", "translate(" + xSize / 2 + "," + ySize / 2 + ")");
    const radius = Math.min(xSize, ySize) / 2;
    var color = d3.scaleOrdinal().domain(data).range(d3.schemeSet3);
    // Generate the pie
    var pie = d3.pie();
    // Generate the arcs
    var arc = d3.arc().innerRadius(0).outerRadius(radius);
    //Generate groups
    var arcs = svg
      .selectAll("arc")
      .data(pie(data))
      .enter()
      .append("g")
      .attr("class", "arc");
    //Draw arc paths
    arcs
      .append("path")
      .attr("fill", function (d, i) {
        return color(i);
      })
      .attr("d", arc);
```

Output:



Exercise 31:

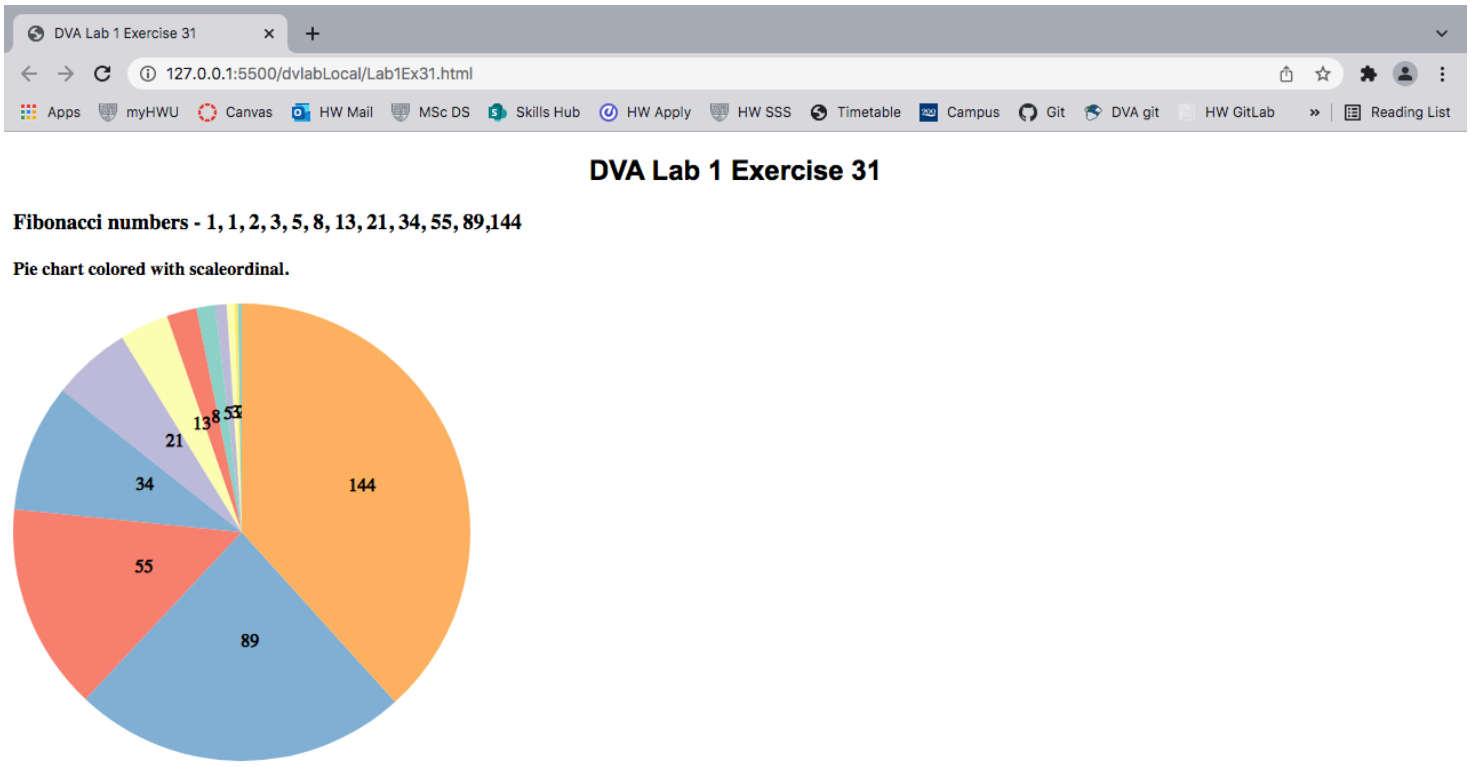
You've already learned about svg text items (previous sections). Add a text item to each 'arc' (e.g., draw the values for the data on the pie chart).

Code: Lab1Ex31.html (available in the attached zip file)

Snippet:

```
const svg = d3
        .select("body")
        .append("svg")
        .attr("width", xSize)
        .attr("height", ySize)
        .append("g")
        .attr("transform", "translate(" + xSize / 2 + "," + ySize / 2 + ")");
      const radius = Math.min(xSize, ySize) / 2;
      var color = d3.scaleOrdinal().domain(data).range(d3.schemeSet3);
      // Generate the pie
      var pie = d3.pie();
      // Generate the arcs
      var arc = d3.arc().innerRadius(0).outerRadius(radius);
      //Generate groups
      var arcs = svg
        .selectAll("arc")
        .data(pie(data))
        .enter()
        .append("g")
        .attr("class", "arc");
      //Draw arc paths
      arcs
        .append("path")
        .attr("fill", function (d, i) {
          return color(i);
        })
        .attr("d", arc);
      arcs
        .append("text")
        .text(function (d) {
          return d.value;
        })
        .attr("transform", function (d) {
          return "translate(" + arc.centroid(d) + ")";
        })
        .attr("stroke", "black");
```

Output:


DVA Lab 1 Exercise 31
127.0.0.1:5500/dvlabLocal/Lab1Ex31.html
Apps
myHWU
Canvas
HW Mail
MSc DS
Skills Hub
HW Apply
HW SSS
Timetable
Campus
Git
DVA git
HW GitLab
Reading List
DVA Lab 1 Exercise 31
Fibonacci numbers - 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,144
Pie chart colored with scaleordinal.
144
89
55
34
21
13
8
5
3

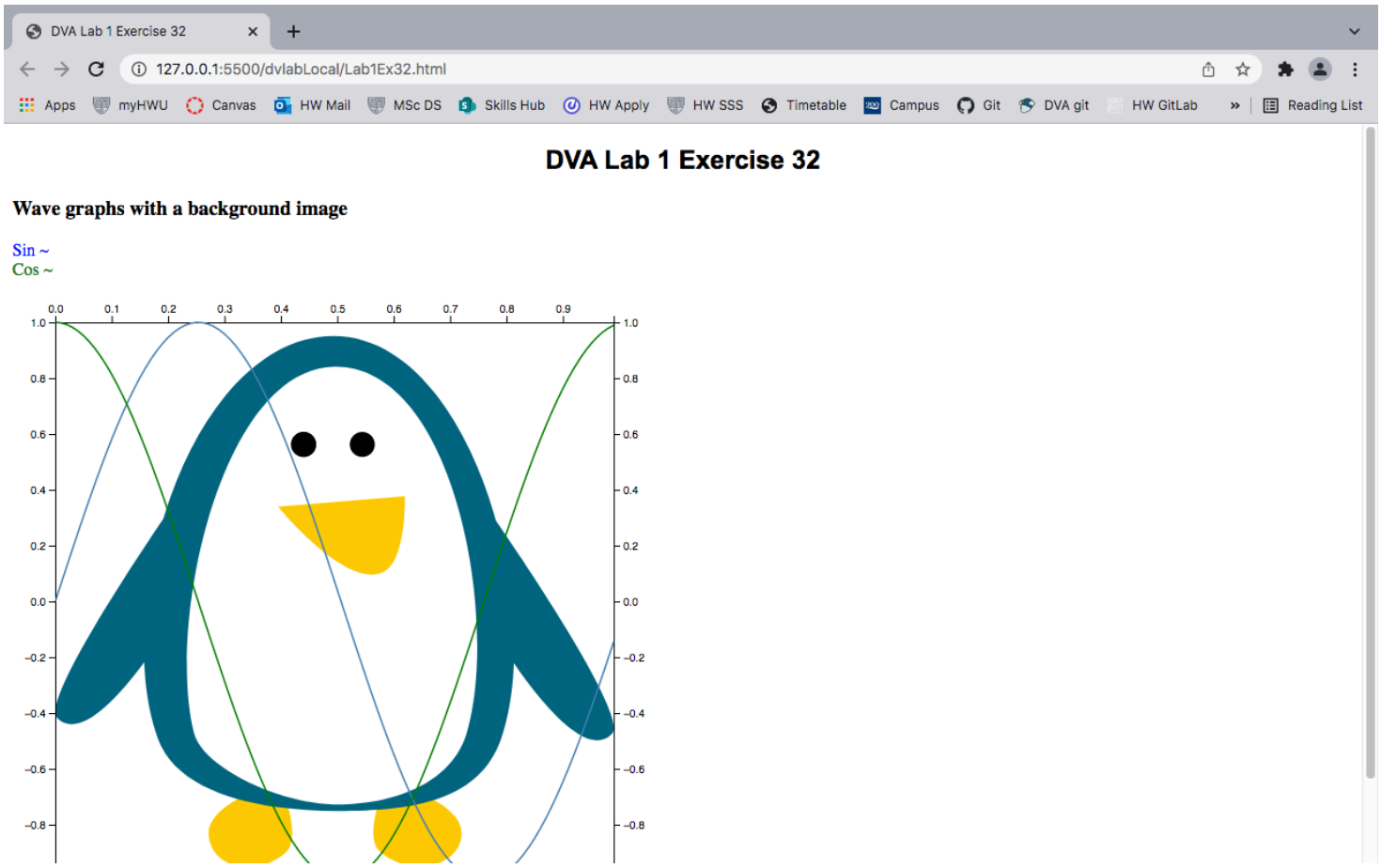
Part 16. SVG Graphics

Exercise 32:

Add a graphical image (e.g., png/jpg) to the background of one of the graphs (e.g., bar chart or line plot). Scale the image to fit the size of the svg bounds (covers background).

Code: Lab1Ex32.html (available in the attached zip file)

Output:



End of Report.

Thank You.