infoVIS TA Session

Hwiyeon Kim
Email: gnldus28@unist.ac.kr

Overview

- ***Term Project*** is out!
- Small hints for implementation
  - Import json file to javascript
- Helpful sources to utilize

Term Project

Data to be used:

- A building layout for the GAStech offices, including the maps of the prox zones and the HVAC zones (Building Layout)
- A current list of employees, roles, and office assignments (Employee List.xlsx)
- A description of the data formats and fields provided (Data Formats.pdf)
- Proximity sensor data for each of the prox zone regions (BuildingProxSensorData)
- Proximity sensor data from Rosie the mobile robot (BuildingProxSensorData)
- HVAC sensor readings and status information from each of the building's HVAC zones (BuildingProxSensorData)
- Hazium readings from four sensors. (BuildingProxSensorData)

Tasks:

You will be asked to answer the following types of questions (Total 100pt):

1. (25pt) What are the typical patterns in the prox card data? What does a typical day look like for GAStech employees? Describe up to **five of the most interesting patterns** that appear in the building data.
2. (20pt) Describe up to **five of the most interesting patterns** that appear in the building data. Describe what is notable about the pattern and explain its possible significance.
3. (25pt) Describe up to **five notable anomalies or unusual events** you see in the data. Prioritize those issues that are most likely to represent a danger or a serious issue for building operations.
4. (30pt) Describe up to **three observed relationships between the proximity card data and building data elements**. If you find a causal relationship (for example, a building event or condition leading to personnel behavior changes or personnel activity leading to building operations changes), describe your discovered cause and effect, the evidence you found to support it, and your level of confidence in your assessment of the relationship.

Term Project – points possible

Case 1) Term project group with **2 people** (same as before):

1 (25pt) 2 (20pt) 3 (25pt) 4 (30pt), total 100pt

Case 2) Term project group with **1 person** (newly updated):

3 (40pt) 4 (60pt), total 100pt

Case 1

1. (25pt) What are the typical patterns in the prox card data? What does a typical day look like for GAStech employees? Describe up to **five of the most interesting patterns** that appear in the building data.
2. (20pt) Describe up to **five of the most interesting patterns** that appear in the building data. Describe what is notable about the pattern and explain its possible significance.

Case 2

3. (25pt) Describe up to **five notable anomalies or unusual events** you see in the data. Prioritize those issues that are most likely to represent a danger or a serious issue for building operations.
4. (30pt) Describe up to **three observed relationships between the proximity card data and building data elements**. If you find a causal relationship (for example, a building event or condition leading to personnel behavior changes or personnel activity leading to building operations changes), describe your discovered cause and effect, the evidence you found to support it, and your level of confidence in your assessment of the relationship.

Term Project

- **Prox data**

GASTech has installed proximity detectors throughout the building to enhance the safety and security of its employees. There are two types of proximity sensors, (1) fixed sensors that register when a badge crosses a boundary and a (2) mobile sensor reader. The mobile reader moves throughout the building and will register badges within a 4 foot radius. (Note: the mobile sensor scans for prox badges every second, but reports out in 1 minute aggregations).

There are 2 formats available for download, a CSV file or a file with JSON objects.

The CSV formats are as follows:

- Fixed-prox

```
timestamp, type, prox-id, floor, zone
2016-05-31 07:15:00, fixed-prox, proxName002, 1, 1
2016-05-31 07:18:00, fixed-prox, proxName1001, 1, 1
```

- Mobile prox

```
timestamp, type, prox-id, floor, x, y
2016-05-31 09:00:00, mobile-prox, proxName1001, 1, 174, 15
2016-05-31 09:09:00, mobile-prox, proxName2001, 2, 15, 78
```

The (x,y) coordinates are based per floor with the lower left of the provided map being (0,0) and the upper right being (189,111).

The JSON formats are as follows:

The message format for fixed prox sensors will be

```
"message": {
          "zone": "4",
          "floor": "1",
          "datetime": "2016-5-14 23:59:46",
          "type": "fixed-prox",
          "proxCard": "proxName002"

}
```

- **Building Data**

As part of this year's data, we are simulating a 3 story building with multiple air handling zones per floor. The data is sampled every 5 minutes.
Two main types of data are provided:

Building

This is provided as a CSV and a JSON structure for this project.

- CSV

The row header and sample row can be found in the .csv attachment. All data for all floors is in a single record.

- JSON

The data is divided into 4 JSON objects, "general", "floor 1", "floor 2" and "floor 3". Each floor may have a different number of field/value pairs because each floor has a differing number of HVAC zones.

Hazium

The GasTech headquarters has installed a limited number of Hazium sensors throughout the building. The sensors are part of the building air handling system. They are placed in an HVAC zone. Not all HVAC zones have sensors.
So F_1_Z_1 is floor 1, zone 1.
For this project, this data is provided in CSV and JSON formats.

- CSV

```
Date/Time , F_#_Z_#: Hazium Concentration
2016-05-31 00:00:00, F_1_Z_1, 0.0
2016-05-31 00:05:00, F_1_Z_1, 0.0
2016-05-31 00:10:00, F_2_Z_1, 0.0
```

- JSON

```
"message": {
          "F_1_Z_1: Hazium Concentration": "0.0",
          "type": "sensor",
          "Date/Time": "2016-05-31 00:00:00"

}
```

Due Date: 2019-12-13 11:59pm (extended)

Documents:

- Your analysis report (pdf, doc,..)
- Your presentation file (pptx, pdf)
- Code

*Note

1. You should answer these questions with figures (e.g., screenshot of web page you implemented) in your report. Please describe the patterns you found with fully finished sentences. Note that you must do **demonstration of your implemented system in the Final presentation**.
2. You can earn additional points (up to 30pt) by submitting a runnable code (e.g., github) or website to your report.
3. Late submissions can be received up to three days after the due date, with 10% penalty per day (1 day late – 10% / 2 days late – 20% / 3 days late – 30%).

Import JSON file to javascript

JSON
- text-based data format following JavaScript object syntax

AJAX (**A**synchronous **Ja**vaScript and **X**ML)
- technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

For this time, we will use AJAX for reading JSON file to Javascript ☺

Import JSON file to javascript

AJAX (**A**synchronous **Ja**vaScript and **X**ML)
- technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

For this time, we will use AJAX for reading JSON file to Javascript ☺

- How to use

```
$.ajax({
    dataType: json     // json, xml, html,
    url:'example.php' // Address to request
    async:true,        // true: Asynchronous, false: Synchronous
    type:'POST'        // GET, PUT
    data: {            // Data to send
        Name:'ajax',
        Age:'10'
    }).done (function(data){

        // do something if connected successfully
    };
```

Import JSON file to javascript - example 1

load_data.js

```
var filename = path + ylist[i] + "_" + this.name + ".json";
$.ajax({
        dataType: 'json',
        cache: false,
        async: false,
        url: filename,
        data: json
})
.done(function(json){
        var us = json[0].Times;
        var ue = json[json.length -1].Times;
        var ustart = convUnix(us);
        var uend = convUnix(ue);
        u1 = new Date(ustart.y, ustart.m, ustart.d);
        u2 = new Date(uend.y, uend.m , uend.d);
    })
    .fail (function(){
        console.log(filename + " is not found");
        return false;
    });
```

10_말똥가리.json


```
[
  {
    "Times": 1267887600,
    "Coords": [
      129.0266667,
      35.1180556
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.0266667,
    "long": 129.0266667,
    "lat": 35.1180556
  },
  {
    "Times": 1267974000,
    "Coords": [
      129.1208333,
      35.1319444
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.1208333,
    "long": 129.1208333,
    "lat": 35.1319444
  },
  {
    "Times": 1267974000,
    "Coords": [
      129.4075,
      35.135
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.4075,
    "long": 129.4075,
    "lat": 35.135
  },
```


Learn more about json:
https://developer.mozilla.org/ko/docs/Learn/JavaScript/Objects/JSON

Import JSON file to javascript - example 1

load_data.js

Path to the JSON file

```
var filename = path + ylist[i] + "_" + this.name + ".json";
$.ajax({
        dataType: 'json',
        cache: false,
        async: false,
        url: filename,
        data: json
})
.done(function(json){
        var us = json[0].Times;
        var ue = json[json.length -1].Times;
        var ustart = convUnix(us);
        var uend = convUnix(ue);
        u1 = new Date(ustart.y, ustart.m, ustart.d);
        u2 = new Date(uend.y, uend.m , uend.d);
    })
    .fail (function(){
        console.log(filename + " is not found");
        return false;
    });
```

Set path to the JSON file

1) Use variable

If you have many files that have some text patterns like:
years_birdname1.json
years_birdname2.json
years_birdname3.json
..
You may utilize variables in your code:

```
var filename
= years + '_' + birdname + '.json'

url: filename
```

2) Type directly

```
url: '../data/10_말똥가리.json'
```

10_말똥가리.json


```
[
  {
    "Times": 1267887600,
    "Coords": [
      129.0266667,
      35.1180556
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.0266667,
    "long": 129.0266667,
    "lat": 35.1180556
  },
  {
    "Times": 1267974000,
    "Coords": [
      129.1208333,
      35.1319444
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.1208333,
    "long": 129.1208333,
    "lat": 35.1319444
  },
  {
    "Times": 1267974000,
    "Coords": [
      129.4075,
      35.135
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.4075,
    "long": 129.4075,
    "lat": 35.135
  },
```


Learn more about json:
https://developer.mozilla.org/ko/docs/Learn/JavaScript/Objects/JSON

Import JSON file to javascript - example 1

Can be any name
(e.g., json, data, bird..)

```
var filename = path + ylist[i] + "_" + this.name + ".json";
$.ajax({
        dataType: 'json',
        cache: false,
        async: false,
        url: filename,
        data: json
})
.done(function(json){
        var us = json[0].Times;
        var ue = json[json.length -1].Times;
        var ustart = convUnix(us);
        var uend = convUnix(ue);
        u1 = new Date(ustart.y, ustart.m, ustart.d);
        u2 = new Date(uend.y, uend.m , uend.d);
    })
    .fail (function(){
        console.log(filename + " is not found");
        return false;
    });
```

10_말똥가리.json


```
[
  {
    "Times": 1267887600,
    "Coords": [
      129.0266667,
      35.1180556
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.0266667,
    "long": 129.0266667,
    "lat": 35.1180556
  },
  {
    "Times": 1267974000,
    "Coords": [
      129.1208333,
      35.1319444
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.1208333,
    "long": 129.1208333,
    "lat": 35.1319444
  },
  {
    "Times": 1267974000,
    "Coords": [
      129.4075,
      35.135
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.4075,
    "long": 129.4075,
    "lat": 35.135
  },
```

Import JSON file to javascript - example 1

```
var filename = path + ylist[i] + "_" + this.name + ".json";
$.ajax({
        dataType: 'json',
        cache: false,
        async: false,
        url: filename,
        data: json
})
.done(function(json){
        var us = json[0].Times;
        var ue = json[json.length -1].Times;
        var ustart = convUnix(us);
        var uend = convUnix(ue);
        u1 = new Date(ustart.y, ustart.m, ustart.d);
        u2 = new Date(uend.y, uend.m , uend.d);
    })
    .fail (function(){
        console.log(filename + " is not found");
        return false;
    });
```

- You can also check the data variable whether it meets with your condition:

```
// for i = 0 ~ json[json.length-1]

var time = json[i].Times;
if (time > 1267974000) { // do something }
```

10_말똥가리.json


```
[
  {
    "Times": 1267887600,
    "Coords": [
      129.0266667,
      35.1180556
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.0266667,
    "long": 129.0266667,
    "lat": 35.1180556
  },
  {
    "Times": 1267974000,
    "Coords": [
      129.1208333,
      35.1319444
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.1208333,
    "long": 129.1208333,
    "lat": 35.1319444
  },
  {
    "Times": 1267974000,
    "Coords": [
      129.4075,
      35.135
    ],
    "Bird_name": "말똥가리4",
    "altitude": 129.4075,
    "long": 129.4075,
    "lat": 35.135
```


json[0]

json[0].Times

json[0].Coords

json[0].Bird_name

json[0].long

json[1]

json[2]

proxMobileOut-MC2.json

proxMobileOut-MC2.json 〉 No Selection

data[0]


```json
[
    {
        "message": {
            "Y": "15",
            "datetime": "2016-05-31 09:00:00",
            "X": "174",
            "proxCard": "vlagos001",
            "type": "mobile-prox",
            "floor": "1"
        },
        "offset": 32400.0
    },
    {
        "message": {
            "Y": "15",
            "datetime": "2016-05-31 09:00:00",
            "X": "174",
            "proxCard": "vlagos001",
            "type": "mobile-prox",
            "floor": "1"
        },
        "offset": 32400.0
    },
    {
        "message": {
            "Y": "15",
            "datetime": "2016-05-31 09:00:00",
            "X": "174",
            "proxCard": "vlagos001",
            "type": "mobile-prox",
            "floor": "1"
        },
        "offset": 32400.0
    },
    {
        "message": {
            "Y": "15",
            "datetime": "2016-05-31 09:01:00",
            "X": "165",
            "proxCard": "sfusil001",
            "type": "mobile-prox",
            "floor": "1"
```


To do

1. Create a javascript file
2. Write the path to the json file
3. Decide what variables to export (time, coordinates,…)
4. Do something in the function(data)

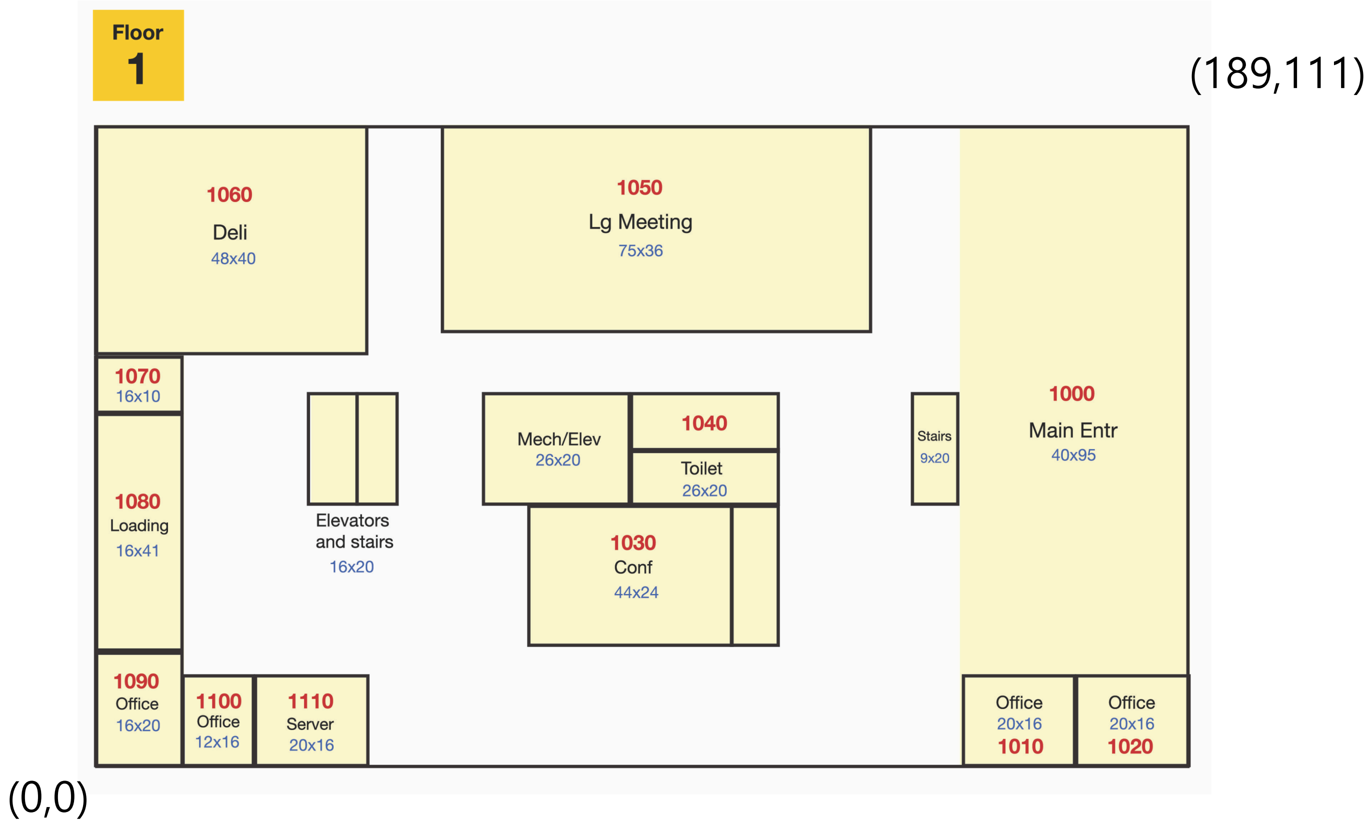
Example

```
$.ajax({
        dataType: json
        url:'/path-to-the-json-file/proxMobileOut-
MC2.json'
        async:true,
        data: data
        .done (function(data){

        var time = data[0].datetime;
        var coord_X = data[0].X;
        var coord_Y = data[0].Y;

        }
```

Import JSON file to javascript - example 2



You may draw building layout and
draw coordinates (X, Y) with datetime / proxCard
on the layout.

proxMobileOut-MC2.json

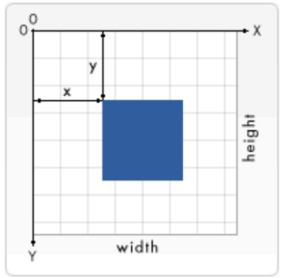
proxMobileOut-MC2.json 〉 No Selection


```
[
    {
        "message": {
            "Y": "15",
            "datetime": "2016-05-31 09:00:00",
            "X": "174",
            "proxCard": "vlagos001",
            "type": "mobile-prox",
            "floor": "1"
        },
        "offset": 32400.0
    },
    {
        "message": {
            "Y": "15",
            "datetime": "2016-05-31 09:00:00",
            "X": "174",
            "proxCard": "vlagos001",
            "type": "mobile-prox",
            "floor": "1"
        },
        "offset": 32400.0
    },
    {
        "message": {
            "Y": "15",
            "datetime": "2016-05-31 09:00:00",
            "X": "174",
            "proxCard": "vlagos001",
            "type": "mobile-prox",
            "floor": "1"
        },
        "offset": 32400.0
    },
    {
        "message": {
            "Y": "15",
            "datetime": "2016-05-31 09:01:00",
            "X": "165",
            "proxCard": "sfusil001",
            "type": "mobile-prox",
            "floor": "1"
```

Drawing shapes

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_shapes

The grid



Before we can start drawing, we need to talk about the canvas grid or **coordinate space**. Our HTML skeleton from the previous page had a canvas element 150 pixels wide and 150 pixels high. To the right, you see this canvas with the default grid overlayed. Normally 1 unit in the grid corresponds to 1 pixel on the canvas. The origin of this grid is positioned in the *top left* corner at coordinate (0,0). All elements are placed relative to this origin. So the position of the top left corner of the blue square becomes x pixels from the left and y pixels from the top, at coordinate (x,y). Later in this tutorial we'll see how we can translate the origin to a different position, rotate the grid and even scale it, but for now we'll stick to the default.

Drawing rectangles

Unlike SVG, `<canvas>` only supports two primitive shapes: rectangles and paths (lists of points connected by lines). All other shapes must be created by combining one or more paths. Luckily, we have an assortment of path drawing functions which make it possible to compose very complex shapes.

First let's look at the rectangle. There are three functions that draw rectangles on the canvas:

`fillRect(x, y, width, height)`
: Draws a filled rectangle.

`strokeRect(x, y, width, height)`
: Draws a rectangular outline.

`clearRect(x, y, width, height)`
: Clears the specified rectangular area, making it fully transparent.

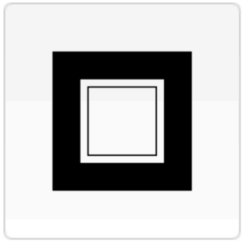
Each of these three functions takes the same parameters. `x` and `y` specify the position on the

Rectangular shape example

```
function draw() {
  var canvas = document.getElementById('canvas');
  if (canvas.getContext) {
    var ctx = canvas.getContext('2d');

    ctx.fillRect(25, 25, 100, 100);
    ctx.clearRect(45, 45, 60, 60);
    ctx.strokeRect(50, 50, 50, 50);
  }
}
```

This example's output is shown below.

Screenshot	Live sample
	

Sources

Timelines

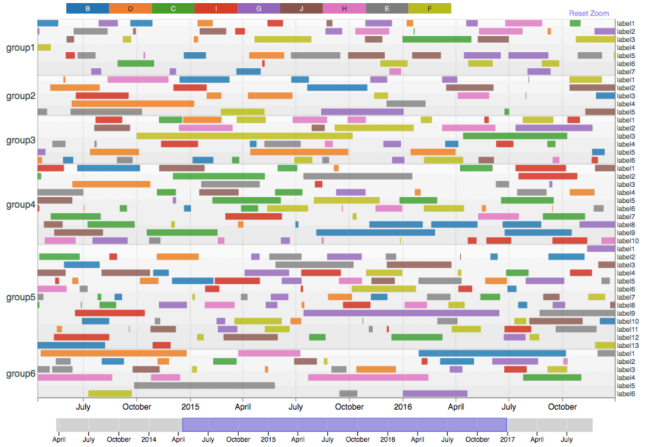
https://www.npmjs.com/package/timelines-chart
https://www.npmjs.com/package/timeline-plus

Heatmaps

Plugin	Description	Maintainer
MaskCanvas	Canvas layer that can be used to visualize coverage.	Dominik Moritz
HeatCanvas	Simple heatmap api based on HTML5 canvas.	Sun Ning
heatmap.js	JavaScript Library for HTML5 canvas based heatmaps. Its Leaflet layer implementation supports large datasets because it is tile based and uses a quadtree index to store the data.	Patrick Wied
Leaflet divHeatmap	Lightweight and versatile heatmap layer based on CSS3 and divIcons	Daniele Piccone
WebGL Heatmap	High performance Javascript heatmap plugin using WebGL.	Benjamin J DeLong
Leaflet.heat	A tiny, simple and fast Leaflet heatmap plugin. Uses simpleheat under the hood, additionally clustering points into a grid for performance. (Demo)	Vladimir Agafonkin
Leaflet-Solr-Heatmap	A Leaflet plugin for rendering heatmaps and clusters from Solr's Heatmap Faceting. High performance for millions of points or polygons.	Jack Reed / Steve McDonald

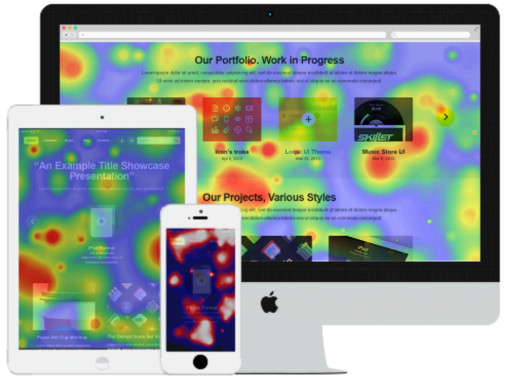
Timelines Chart

npm v2.8.4 | minzipped size 47.6 kB | dependencies up to date



heatmap.js

Dynamic Heatmaps for the Web



heatmap.js is a lightweight, easy to use JavaScript library to help you visualize your three dimensional data!

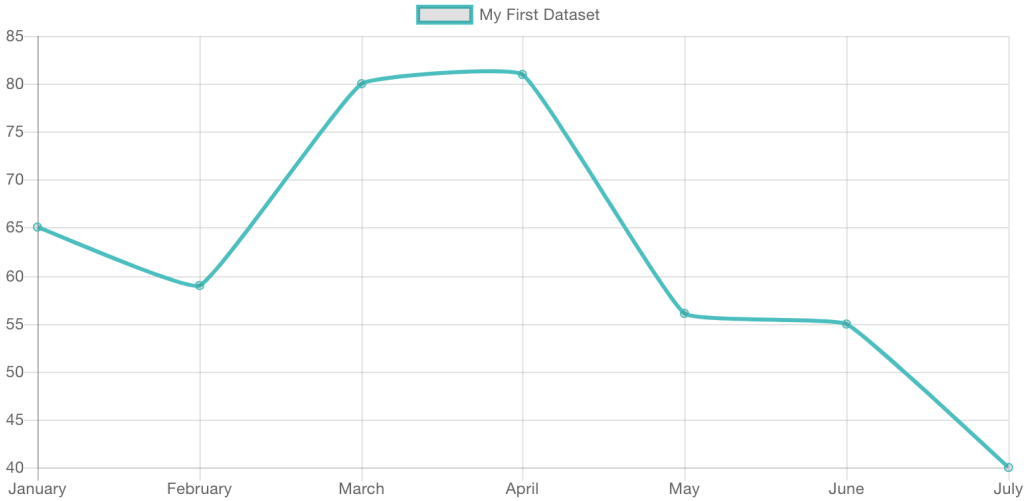
Sources

Chart.js

https://www.chartjs.org/docs/latest/charts/line.html

Line

A line chart is a way of plotting data points on a line. Often, it is used to show trend data, or the comparison of two data sets.

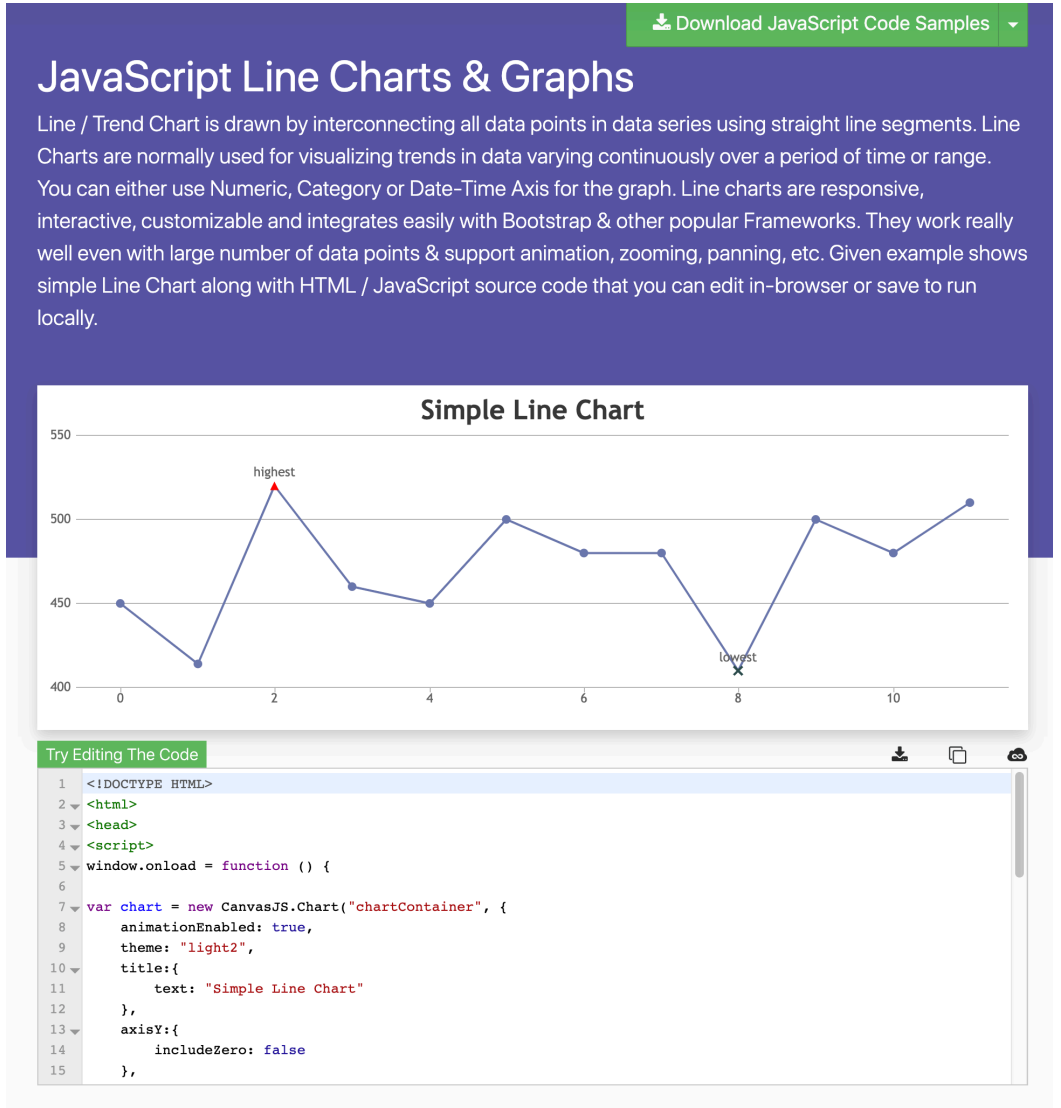


Example Usage

```
var myLineChart = new Chart(ctx, {
    type: 'line',
    data: data,
    options: options
});
```

canvasJS

https://canvasjs.com/html5-javascript-line-chart/



Any questions? ☺