Lab 5: Graphing sensor data

Sensors for reading environmental data

Background

This is the fifth lab exercise in the mobile dev course for web developers. It's about reading data captured using sensors from the world around us.

After completing this lab you will be able to pick up a generic sensor feed and present its contents to a easy-to-use service. The exercise is tailored to help you learn how to traverse a JSON object using javascript, and to use your favorite framework (such as jQuery Mobile, Bootstrap, Angular et cetera) to create an agreeable interface for the user to consume the data via a mobile browser.

Pro tip: Use "hyper.log" instead of "console.log" to get debugging results back in Evothings Workbench from a mobile device

Links, starting points

Sensor source database (based on Phant, by Sparkfun)

http://80.69.174.27:8080/streams/

Example of a single stream:

http://80.69.174.27:8080/streams/dPjyGN0bx0IDgO000D4II6leJGK

Its cover image:

https://evothings.com/demos/dome_pics/IMG_1758.JPG

And the corresponding data (here is json format):

http://80.69.174.27:8080/output/dPjyGN0bx0IDgO000D4II6leJGK.json?limit=5

(limit=5 produces the 5 last values only)

There is a also a minimum viable webapp (below, under Objectives), for you to exploit to get going quickly. You can get the data using other methods, if you're more comfortable with another library.

Legend

p = barometric pressure (Pa)

t = temperature (°C)

l = lumination (lux)

h = humidity (%)

c = carbon dioxide (ppm)

PIR (photoelectric sensors, currently not operational):

- np = no movement
- pp = registered movement

(where np + pp = 1500 samples/10minutes)

Objective

Your objective is to make an app to monitor the data variations in the Rindö office. There are two streams, one in the office, and one in recreational space.

http://80.69.174.27:8080/streams/dPjyGN0bx0IDgO000D4Il6leJGK

http://80.69.174.27:8080/streams/GqXO9z3ae9tODKPPPO2YHxw39By

(the latter misses some data from during the week-end)

The objective is compare data from the two spaces, .e.g their temperatures, lighting conditions et cetera to find similarities or other patterns. You will need to compare the readings from the two sensors, and be able to step forwards and back in time.

Then you can filter using timestamps, gt = "greater than" & lt = "less than";

http://80.69.174.27:8080/output/dPjyGN0bx0IDgO000D4Il6leJGK.json?lt[timestamp]=2018-04-30&gt[timestamp]=2018-04-29

or

'.json?gt[timestamp]=now-3day&lt[timestamp]=now-2day

This lab is more open-ended than the previous ones, and it's up to you how to visualise the data. Here are some links to open source graphing tools, that you might want to use. Look them through before you begin if you like.

Google Charts https://developers.google.com/chart/

Chart JS https://www.chartjs.org

Flotr2 http://www.humblesoftware.com/flotr2/

D3.js https://d3js.org/

Smoothie Charts http://smoothiecharts.org/

Converting ISO time format to internal JavaScript datetime object

Some graphing libraries can't read ISO format:

Conversion can be done in two steps:

2018-05-07T09:26:21.638Z --> milliseconds using Date.parse()

milliseconds --> JS Datetime object using Date()

e.g:

```
var ts = new Date(Date.parse(sensor.fullData[Number(row)].timestamp));
```

Picking up data

Picking up data using Apache Cordova (embedded in Evothings Viewer)

```
// Function to retrieve data, placing it in a "response" object
function getJSON()
{
  if (window.cordova)
```

```
{
  console.log('Using the Cordova HTTP GET function');
  cordovaHTTP.get(
    mSensorDataURL + sensor.key +
    '.json?gt[timestamp]=now-1day&page=1',
    function (response)
     {
       if (response)
          {
             sensor.data = JSON.parse(response.data)[0];
             sensor.fullData = JSON.parse(response.data);
             printData();
           }
       },
       function (error)
        {
          console.log(JSON.stringify(error));
        });
    }
```

Picking up the data, $ajax style using jsonp

// In this case, there is no Apache cordova to save you, make a traditional ajax call instead. Here **sensor.data** holds the last collected values, whereas **sensor.fullData** is the entire array for a given sensor over the last 24 hrs.

….

else

```
 {
   console.log('Not using Cordova, fallback to AJAX via jquery');
   $.ajax({
     url: mSensorDataURL + sensor.key
    + ".json?gt[timestamp]=now-1day",
     jsonp: "callback",
     cache: true,
     dataType: "jsonp",
     data:
      {
          page: 1
       },
      success: function(response)
        {
          if (response && response[0])
              {
                sensor.data = response[0];
                sensor.fullData = response;
                printData();
              }
         }
   });
}
```

Code for a simplistic template app

Here’s a starting point, doing the bare minimum

https://evothings.com/demos/dome_pics/sensorer.zip (right-click to download, you know)

What, don’t developers do everything from scratch? Is not uncommon in the industry, that there are building blocks on how to do things made by others, often simplified and lacking of functionality which you need to add to complete the app.

Good luck!