**Cookbook for web**

**Description of technologies used for web application**

Jan Boril

Masaryk University/Yerevan State University

Sep 2021 – Jan 2022

borekj.jan@gmail.com

The whole web is using basically three technologies – geoserver, Leaflet and SentinelAPI. This Cookbook will lead through all of the used technologies. I will not explain single piece of code, rather how the web work in general.

Sections:

**WEB in general  
LEAFLET  
SENTINEL HUB  
LEGEND  
GEOSERVER  
SLIDERS  
LOCAL GEOJSON FEATURES  
CURRENT WEATHER   
GITHUB**

**WEB as general**

The whole web is controlled by these green buttons – an infinity of them can be added as other layers.

Obsah obrázku stůl

Popis byl vytvořen automaticky

For each button you click, a function is javascript is called, see the onclick value

Obsah obrázku text

Popis byl vytvořen automaticky

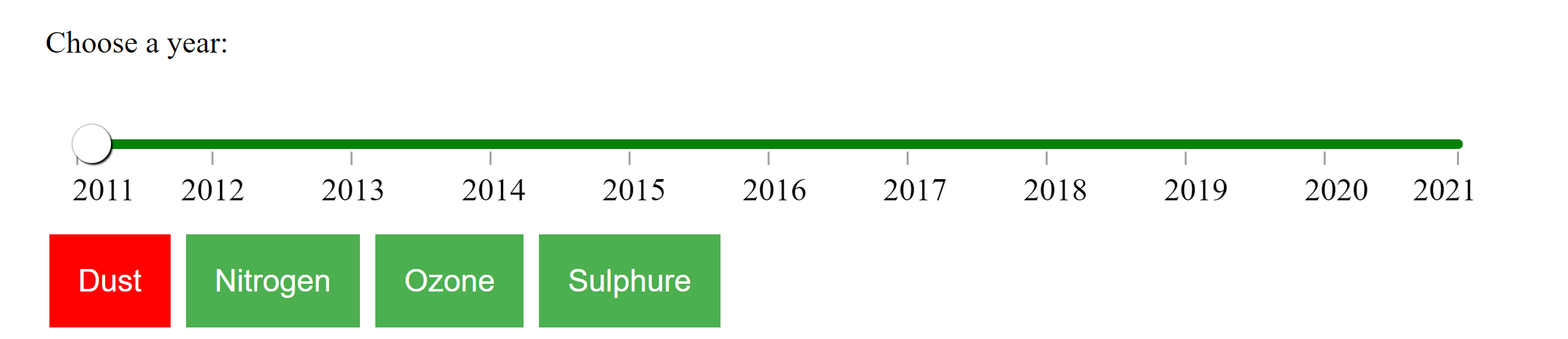
Each function has some elements that are the same for everything.

1. Obsah obrázku text

   Popis byl vytvořen automaticky A map view is set to desired coordinates, in case user moved the map somewhere else. Then, a legend is deleted. We will talk about legend in a minute
2. Obsah obrázku text

   Popis byl vytvořen automaticky

Then, all additional information are hidden or shown. For density, no additional information is shown. For some other layers, usually one of the containers has display = “block”, and there is additional information shown, e.g. this



1. Obsah obrázku text, podepsat

   Popis byl vytvořen automaticky Then, all layers are removed
2. Obsah obrázku text

   Popis byl vytvořen automatickyThen, the baselayer (OSM) is called with desired opacity
3. Obsah obrázku text

   Popis byl vytvořen automaticky Then, the desired layer is called (about that later)
4. Obsah obrázku text

   Popis byl vytvořen automaticky Finally, a Legend is called (not always though). A legend is just a picture called through wmsLegend – about that later
5. Some functions are more complicated, but about that later

The whole web is styled via simple css. There is not really anything to explain regarding that.

**LEAFLET**

[Leaflet](https://leafletjs.com/) is an opensource javascript library used to add map window (and layers on top of that) to the project. It basically takes only few added lines to the code in html

Obsah obrázku text

Popis byl vytvořen automaticky

Obsah obrázku text

Popis byl vytvořen automaticky

This is then a simple code how to add OSM layer to that window.

Obsah obrázku text

Popis byl vytvořen automaticky

Other can be then added – in the case of this web, L.tileLayer and L.geoJson are used. See documentation about how many more are possible

**LEGEND**

Legend is using a plugin called [WMSLegend](https://github.com/kartoza/leaflet-wms-legend). The plugin is free to download, the documentation can be seen in the link on github, so there is no need to explain it more. It basically takes only a picture and places it its on the upper corner of leaflet map. The picture can be download through GetLegendGraphic via WMS, but generally just creating a picture and adjusting its size is much simpler and more reliable.

Because the legend has to be changed every time a layer is changed, all Legends (sometimes there is more than one) are deleted when a function of a new layer is called.

Obsah obrázku text

Popis byl vytvořen automaticky

It basically collects all Elements with given tag, and through series of DOM removals removes the whole legendDOM (not only the picture, the whole DOM). A new DOM with a new legend is then always called

**Geoserver**

(on <http://vgse.geology.am/geoserver/web>) is used to store locally produced data – see the PROJECT document for what data are stored there and how were they created. The data are transferred there in either .tif or .shp format, through geonode (http://vgse.geology.am/) however only WMS is used to transfer stored data to the leaflet.

For that, a standard WMS request is used. Obsah obrázku text

Popis byl vytvořen automaticky

* L.tileLayer.wms = simple call to geoserver
* Layers: which layer from geoserver to choose
* Format: in which format to send the picture – usually png
* Maxcc, minZoom, maxZoom: limiting zoom levels – there is no need to do that now

There are currently these layers store on geoserver – all air quality rasters, all data about green areas and the district density.

**Sentinel Hub API**

Works in a similar way as geoserver.

Obsah obrázku text, monitor, snímek obrazovky, obrazovka

Popis byl vytvořen automaticky

The layer is called through first through a BaseURL and then trough urlProcessingApi. The rest is the same as when you call a layer from a geoserver. What is different though, that first you need to set up the layer in [sentinelHub](https://www.sentinel-hub.com/). I will not really go in details, because there is many already created manuals how to use this, for example [here](https://www.sentinel-hub.com/develop/dashboard/), but I’ll anyway post here the scripts I am using. Those and much more can be found in a github repository

**NDVI**

let ndvi = (B08 - B04) / (B08 + B04);

//Visualization, as used in EO Browser:

if (ndvi<-1.1) return [0,0,0];

else if (ndvi<-0.2) return [0.75,0.75,0.75];

else if (ndvi<-0.1) return [0.86,0.86,0.86];

else if (ndvi<0) return [1,1,0.88];

else if (ndvi<0.025) return [1,0.98,0.8];

else if (ndvi<0.05) return [0.93,0.91,0.71];

else if (ndvi<0.075) return [0.87,0.85,0.61];

else if (ndvi<0.1) return [0.8,0.78,0.51];

else if (ndvi<0.125) return [0.74,0.72,0.42];

else if (ndvi<0.15) return [0.69,0.76,0.38];

else if (ndvi<0.175) return [0.64,0.8,0.35];

else if (ndvi<0.2) return [0.57,0.75,0.32];

else if (ndvi<0.25) return [0.5,0.7,0.28];

else if (ndvi<0.3) return [0.44,0.64,0.25];

else if (ndvi<0.35) return [0.38,0.59,0.21];

else if (ndvi<0.4) return [0.31,0.54,0.18];

else if (ndvi<0.45) return [0.25,0.49,0.14];

else if (ndvi<0.5) return [0.19,0.43,0.11];

else if (ndvi<0.55) return [0.13,0.38,0.07];

else if (ndvi<0.6) return [0.06,0.33,0.04];

else return [0,0.27,0];

**NO2**

//VERSION=3

var minVal = 0.0;

var maxVal = 0.0001;

var diff = maxVal - minVal;

const map = [

[minVal, 0x11007f],

[minVal + 0.125 \* diff, 0x0000ff],

[minVal + 0.375 \* diff, 0x00ffff],

[minVal + 0.625 \* diff, 0xffff00],

[minVal + 0.875 \* diff, 0xff0000],

[maxVal, 0xFFFFFF]

];

const visualizer = new ColorRampVisualizer(map)

function setup() {

return {

input: ["NO2","dataMask"],

output: { bands: 4 }

};

}

function evaluatePixel(samples) {

const [r, g, b] = visualizer.process(samples.NO2);

return [r, g, b, samples.dataMask];

}

**SO2**

//VERSION=3

var minVal = 0.0;

var maxVal = 0.0001;

var diff = maxVal - minVal;

const map = [

[minVal, 0x11007f],

[minVal + 0.125 \* diff, 0x0000ff],

[minVal + 0.375 \* diff, 0x00ffff],

[minVal + 0.625 \* diff, 0xffff00],

[minVal + 0.875 \* diff, 0xff0000],

[maxVal, 0xFFFFFF]

];

const visualizer = new ColorRampVisualizer(map)

function setup() {

return {

input: ["SO2","dataMask"],

output: { bands: 4 }

};

}

function evaluatePixel(samples) {

const [r, g, b] = visualizer.process(samples.SO2);

return [r, g, b, samples.dataMask];

}

Two important notes regarding sentinelHub

* NDVI uses Sentinel 2, NO2 and SO2 Sentinel 5, there can’t be in the same configuration
* The free account always expires after one month – you need to either pay for the data or to recreate the account after each month passed

**SLIDERS**

Some functions (NDVI and air quality) have sliders to move within years.

Obsah obrázku stůl

Popis byl vytvořen automaticky

The slider is a simple html element styled with css, there is many living examples on the web

Obsah obrázku text

Popis byl vytvořen automaticky

There is an event listener on each slider, like thatObsah obrázku text

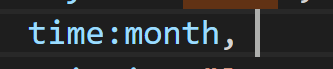
Popis byl vytvořen automaticky

If you move with the slider, his values is written in the variable (month) and passed into another function (ndvi2() in this case)

In the case of ndvi2, it is used for telling SentinelHub for which month to request data



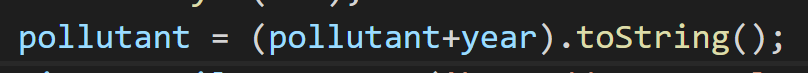
And then

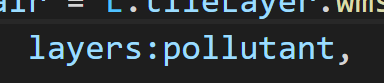


It is similar for the Air Quality layer, where the functions a

Obsah obrázku text

Popis byl vytvořen automaticky





The difference is that the specific layer of pollutant is mixed into that (chosen from another green buttons).

**LOCAL GEOJSON FEATURES**

For the Green areas layer, there are few given examples of how successful the method was

Obsah obrázku mapa

Popis byl vytvořen automaticky

This is done through a local js layer.

Obsah obrázku text

Popis byl vytvořen automaticky

For each point, there is coordinates, colour attribute showing weather it should be green or red and source of the picture

The point is then added through Leaflet geoJson and pointToLayer. See [documentation](https://leafletjs.com/examples/geojson/) for more details.

Obsah obrázku text

Popis byl vytvořen automaticky

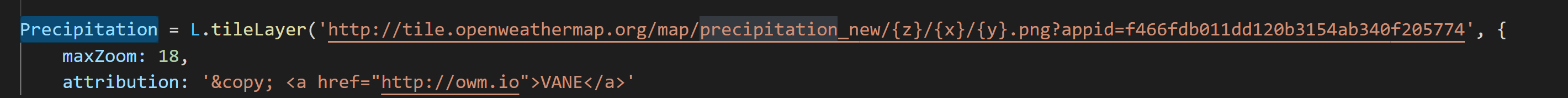
onEachFeature function is then adding the picture itself. Again, see more details in the documentation. Basically it adds the picture that is stored as the source attribute of the picture.

Obsah obrázku text

Popis byl vytvořen automaticky

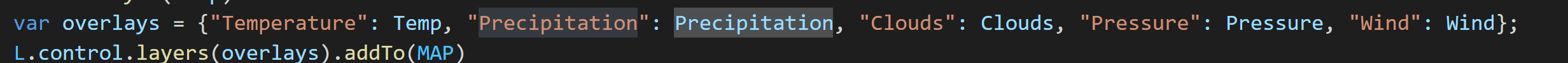
**CURRENT WEATHER**

An online WMS API from Open Weather Map is used. There are 5 layers – Temperature, Precipitation, Wind, Clouds and Pressure.

****

All of them are just simple WMS reques

All of them are then stored in overlay.



Which then creates this menu in the app

Obsah obrázku mapa

Popis byl vytvořen automaticky

This specific API however doesn’t provide legend, so there isn’t anything to get through GetLegendGraphic.

**GITHUB**

the whole project is currently transferred to GitHub – see the repository <https://github.com/borekj/GA_in_Yerevan> and the running version <https://borekj.github.io/GA_in_Yerevan/> here.

GitHub is – I guess – well known, so there is no need to explain it moe. However, there is one big problem. The Geoserver currently doesn’t support https requests (there is an issue with SSL certificate). I will leave the solution of this problem to you. The results is however, that none of the layers stored on Geoserver is possible to load.