Workshop MapLibre GL JS

FOSS4GNL 2021, 19th Octobre, Enschede

Edward Mac Gillavry
Frederieke Ridder

Contents

Emergency communication

Setting the scene

An earthquake in the municipality of Apeldoorn makes the national headlines. After Limburg and Groningen a third region hit by earthquakes? The National Meteorological Institute KNMI offers the most recent data. We're expecting a lot of attention from the national press, but also from foreign press agencies. Let's create our own map application at a new website as not be dependent on the KNMI.

Data exploration

The National Meteorological Institute KNMI provides the recent earthquake data.
https://www.dropbox.com/s/62j4vzfndgokuo9/earthquakes.gpkg

Statistics Netherlands provides data for municipalities:
https://www.dropbox.com/s/7vlj2gq6dg2nfrp/netherlands.geojson

1. Open the data in QGIS and view the information on a map:
    - Use the OpenStreetMap from the QuickMapServices plugin to view the data in their geographic context (Plugins > Manage Plugins > Search "QuickMapServices")
    - Where are the earthquakes located?
    - Where is the municipality of Apeldoorn located in relation to the other clusters?
    - How can we map this data?

2. How can we visualise the information?
    - Which attributes do the earthquakes have?
    - Which attributes do the municipalities have?
    - Which attributes are interesting to show in this context?
    - Are these attributes:
        - qualitative: nominal, ordinal?
        - quantitavive: interval or ratio?
    - What graphic variables can best be applied for these measurement levels?
    - Which area do you want to show on the map?

Visualisation

Make a map of the Netherlands, showing the force of each earth quake and the population density per municipality.

- Please note, that the Richter scale is logarithmic!
- Use the "Size Assistant" to control the size of the markers?
- What's the difference between Radius, Surface, or Flannery?
- Can you put the larger circles behind the smaller circles?
- How can you show the population density for each municipality?

Publication

Using the qgis2web-plugin (https://github.com/tomchadwin/qgis2web/wiki), you can easily prepare your map for online publication.

Exit QGIS and load the earthquake data and municipality data again, but don't add any styling. We'll do that later on.

1. Install the qgis2web plugin in QGIS (Plugins > Manage and install plugins > Search "qgis2web")
2. Start the qgis2web-plugin (Web > qgis2web > Create web map)
3. In the "Layers and Groups" section, check the layers and attributes to show
4. In the "Appearance" section:
   - Select "Add layer list": Collapsed
   - Check "Highlight on hover"
   - Set "Extent" to your canvas
   - Select "Max zoom level"
   - Select "Min zoom level"
   - Check "Restrict to extent"
5. In the "Export" section:
   - Exporter: select a folder on your computer
   - Check "Minify GeoJSON files"
   - Set "Precision" to 6 decimal places

Select "Mapbox GL JS" as the JavaScript map library and click on the "Export" button. This may take a while. Don't get nervous: it's a whole lot of data!

Finally, open the "index.html" file in your browser to the view the map application you have just created.

Should you run into Python errors, it's probably due to an outdated version of the GeoCat bridge-style library (https://github.com/GeoCat/bridge-style):

- Download and deflate the ZIP-file
- Locate the "qgis" folder in the bridge-style directory, copy the folder, and overwrite the qgis2web/bridge-style/qgis folder in the plugins directory:
  - Windows: C:\Users\USER\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins
  - Linux: /home/USER/.local/share/QGIS/QGIS3/profiles/default/python/plugins
  - MacOS: ~/Library/Application\ Support/QGIS/QGIS3/profiles/default/python/plugins

Now, that's the strength of Open Source software. Thank you very much GeoCat!

From Mapbox to MapLibre

In the file "index.html" there's a reference to the local JavaScript library "mapbox-gl.js":

```
<script src="./mapbox/mapbox-gl.js"></script>
```

and the local Cascading Style Sheet "mapbox-gl.css":

```
<link rel="stylesheet" href="mapbox/mapbox-gl.css">
```

We'll change these to the MapLibre equivalents that are hosted online. For the JavaScript library we'll change this to:

```
<script src="https://unpkg.com/maplibre-gl@1.15.2/dist/maplibre-gl.js"></script>
```

For the Cascading Style Sheet we'll change this to:

```
<link rel="stylesheet" href="https://unpkg.com/maplibre-gl@1.15.2/dist/maplibre-gl.css">
```

You can also download these to a new folder "maplibre" in the folder created by "qgis2web" and refer to these files from "index.html":

For the JavaScript library we'll change this to:

```
<script src="./maplibre/maplibre-gl.js"></script>
```

For the Cascading Style Sheet we'll change this to:

```
<link rel="stylesheet" href="./maplibre/maplibre-gl.css">
```

In the JavaSript in "index.html", we'll replace every occurrence of

```
new mapboxgl.
```

with

```
new maplibregl.
```

In a final step, replace the reference to Mapbox from the "customAttribution". Reload the webpage and see how "nothing" has changed. For reference, see https://maplibre.org/maplibre-gl-js-docs/example/simple-map/

Improvements

Time to cross the t's and dot the i's. There are for sure some issues that can be improved. Here's a few suggestions:

1. Check out the file "index.html" and change the name of the webpage created using the plug-in.
2. Change the attribute names being used in the popups. Can somebody without prior knowledge of the sources understand the information? This can be changed in QGIS and publish the data again.
3. Adjust the viewport of the map. The easiest way is to take the values from the URL hash. Add the "hash" property to the "Map" object and set its value to "true". Adjust the "center" and "zoom" properties of the "Map" object according to the URL hash (https://maplibre.org/maplibre-gl-js-docs/api/map/).

```
var map = new maplibregl.Map({
      container: 'map',
      style: styleJSON,
      center: [ADJUST_LONGITUDE, ADJUST_LATITUDE],
      zoom: ADJUST_ZOOM_LEVEL,
      hash: true,
      bearing: 0,
      attributionControl: false
});
```

Add a vector-based map from MapTiler

Setting the scene

Typically, we'll use general reference map to display information. More and more, we see OpenStreetMap (https://www.openstreetmap.org/) being used. For the Netherlands, the BRT-Achtergrondkaart (https://www.pdok.nl/introductie/-/article/basisregistratie-topografie-achtergrondkaarten-brt-a-) is an example of a general reference map from Kadaster, the National Mapping Agency in the Netherlands.

To really make our data speak for themselves, we'll be creating our own reference map style! We'll be creating an account with MapTiler. They're hosting vector tiles based on OpenStreetMap data that can be easily customised.

Create a MapTiler account

1. Go to the MapTiler website
2. Create a "Free" plan
3. View the "Basic" map using the MapLibre GL JS-based viewer: https://cloud.maptiler.com/maps/basic/maplibre-gl-js

Add your tiles and "Basic" style to the project

The "Basic Map" is a map style from using the OpenMapTiles vector tiles (https://cloud.maptiler.com/tiles/v3-openmaptiles/). Halfway down the page, there's the heading "Vector tiles".

1. Copy and paste the TileJSON URL into your preferred code editor: https://api.maptiler.com/tiles/v3-openmaptiles/tiles.json?key=**********
2. Open "style.json" in the "mapbox" folder on your desktop and replace the OpenStreetMap source in the JSON-object with the following source:

```
"sources": {
      "openmaptiles": {
            "type": "vector",
            "url": ["https://api.maptiler.com/tiles/v3/tiles.json?
key==********************"]
      },
      …
},
```

3. Replace the URL in “glyphs” from https://glfonts.lukasmartinelli.ch/fonts/{fontstack}/{range}.pbf

   to https://api.maptiler.com/fonts/{fontstack}/{range}.pbf?key=**********

4. Replace the “OpenStreetMap” layer in the in the JSON-object with the layers from the “Basic” style at https://api.maptiler.com/maps/basic/style.json?key=**********

Did you notice the MapTiler and OpenStreetMap attribution automagically added to your viewer? That’s the power of the TileJSON spec (https://github.com/mapbox/tilejson-spec).

Add your styled map

You can start styling the “Basic” map to your liking. MapTiler provides two online style editors:

- Customize Tool: https://cloud.maptiler.com/customize/#basic/7/52/5
- Maputnik:
  - Save the file in the Customize Tool
  - Select “Advanced Editing” in the Customize Tool
  - Maputnik interface opens your custom map style

Please note the map style ID in the URL: https://cloud.maptiler.com/maps/style-editor/?style=**********#7/52/5.

If you like to use the style in your project, export your custom map style from MapTiler:

https://cloud.maptiler.com/maps/**********/export and replace the layers in the file “style.json” in the “mapbox” folder on your laptop as we showed in step 4 in the previous exercise.

Thematic mapping: earthquakes and municipalities

Setting the scene

What’s the societal impact of earthquakes in the Netherlands? Let’s size the earthquakes by their magnitude and colour the municipalities by their population density. The “qgis2web”-plugin has converted the Geopackage to a GeoJSON object. We have prepared a GeoJSON of the municipalities using QGIS and MapShaper (https://mapshaper.org/).

Size the earth quakes by their magnitude

First, we look for the earthquakes in the “style.json” as it’s been added by the “qgis2web”-plugin. Now we adjust the actual cartographic style it too:

```
{
      'id': 'earthquakes',
      'type': 'circle',
      'source': 'earthquakes',
      'paint': {'circle-stroke-color': 'white',
            'circle-stroke-width': 1,
            'circle-radius': {
                  property: 'magnitude',
                  type: 'exponential',
                  stops: [[0, 0], [6, 20]]
            },
            'circle-color': '#0074D9'}}
      )
})
```

Check out what the values of the stops should be for the magnitude. Also, be aware the Richter scale is logarithmic.

Colour the municipalities by their population density

First we add the municipalities imperatively to the “index.html” file instead of declaratively the “style.json”.

```
map.addSource('municipalities', {
      type: 'geojson',
      data: 'PUT YOUR GEOJSON OBJECT HERE'
});
```

Then we add the layer and style too. Use QGIS to check what should be the class boundaries you can reuse in the stops.

```
map.addLayer({
      'id': 'municipalities',
      'type': 'fill',
      'source': 'municipalities',
      'layout': {'visibility': 'visible'},
      'paint': {'fill-outline-color': '#000000',
```

```
     'fill-color': {
          property: 'bev_dichth',
          stops: [[20, '#4d9221'],[50, '#a1d76a'],[100, '#e6f5d0'],
               [200, '#fde0ef'],[500, '#e9a3c9'], [1000,
'#c51b7d'],]
     },
});
```

Can you make them appear below the names, rivers and water areas of the reference map? Check out https://maplibre.org/maplibre-gl-js-docs/api/map/#map#addlayer.