Visualisation de données

09 - Cartographie web

31 mars 2025

Noemi Romano
noemi.romano@heig-vd.ch

Formats de données

- `geojson` Extension `.json` avec les données géographiques (`features`)
- `OSM (xml)` OpenStreetMap
- `shp` Shapefile
- `WKT` Well Known Text
- `GeoParquet` Extension Parquet pour les données géographiques (cloud based)

↳ Observable | Données cartographiques

Sources de données

- Mondiales: OpenStreetMap (OSM), Natural Earth
- Suisse: opendata.swiss, swisstopo, Viageo

OpenStreetMap (OSM)

↳ OpenStreetMap

OSM

vs

Google Map

“Google Map is a closed system, and every information is property of Google. OpenStreetMap is an open data source, and its information is available to every organisation and user.”

↳ How are Google Maps different from OpenStreetMap?, Medium

Récupérer les données

- Par région : `curl "https://api.openstreetmap.org/api/0.6/map?bbox=6.645,46.779,6.65,46.783" heig.osm`
- Par types d'objets : Overpass Turbo API
- Types d' Objets: https://wiki.openstreetmap.org/wiki/Map_features
- Géocodage: Nominatim `https://nominatim.openstreetmap.org/search?city=yverdon&format=json`

Overpass API

↳ Exemple : Overpass Turbo

- Cherchez la ville, village, région qui vous intéresse
- Cliquez sur le bouton **Wizard**
- Cherchez un mot clef comme *pub*, *bar*, *tree* etc.
- Cliquez sur **Export** et choisissez le format que vous souhaitez

Librairies JS

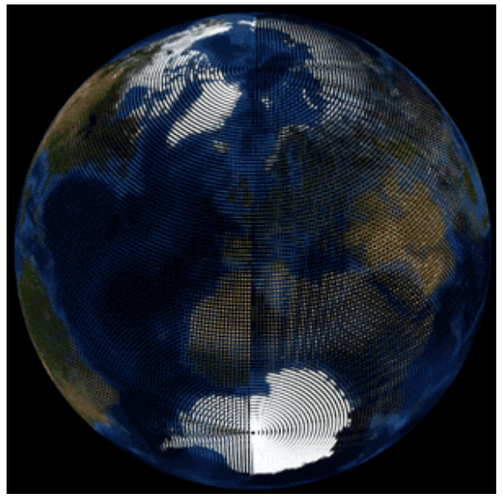
- d3-geo
- Leaflet.js
- OpenLayers.js
- Deck.gl
- MapLibre GL JS
- Kepler.gl
- Giro3D

d3-geo

Installation

```
npm install d3-geo
```

Projections



```
let projection = nom_projection()
```

↳ d3 | Projections

Projections

Transformations

`projection.fonction(arguments)`

Manuelles

`.scale(facteur_échelle)`
Changer l'échelle de la carte

`.translate(X, Y)`
Appliquer une translation de X et Y

Selon les données

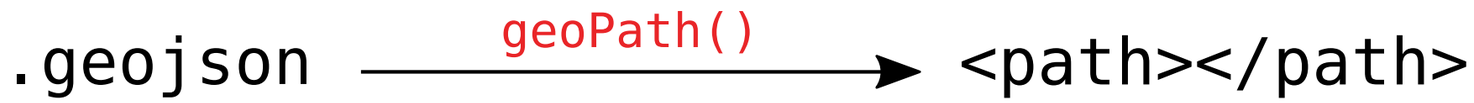
`.fitSize([width, height], geojson)`
Adapter selon largeur/hauteur du SVG

`.fitExtent([X0, Y0, X1, Y1], geojson)`
Adapter selon une certaine étendue

Geopath

GeoPath

Syntaxe



1. Définition de la projection
2. Générateur de *path*
3. Dessiner *path* selon les données

```
// Définition projection (p. ex. geoMercator, geoOrtographic etc.)
let projection = geoMercator().fitSize([width, height], geojson);

// Générateur de <path></path>
const pathGenerator = geoPath()
  .projection(projection)

// <path></path> selon les données
groupe.selectAll("path")
    .data(geojson.features)
    .join(enter => enter.append("path")
                        .attr("d", pathGenerator)
```

Exemple

↳ Observable | d3-geo

Leaflet

Installation

```
npm install leaflet
```


Syntaxe

1. Création division
2. Générer un élément *map*
3. Rajouter un fond de plans
4. Ajouter des éléments à la carte
5. Écouter des événements

↳ Leaflet | Tutoriels

```
// index.html
// 1. Définition du containeur
<div id="mapid" style="width: 600px; height: 400px;"></div>

// index.js
// 2. Rajouter la carte au containeur
let map = L.map('mapid').setView([cx, cy], zoom);

// 3. Fond de carte
L.tileLayer(fondDeCarte).addTo(map);

// 4. Rajouter des éléments
L.Marker([cx,cy]).addTo(map);
L.geoJSON(objetGeojson).addTo(map);
L.bindPopup([cx, cy], "<h1>My popup</h1>").openPopup();

// 5. Event
map.on('click', fonction);
```

Exemple

↳ Observable | Leaflet