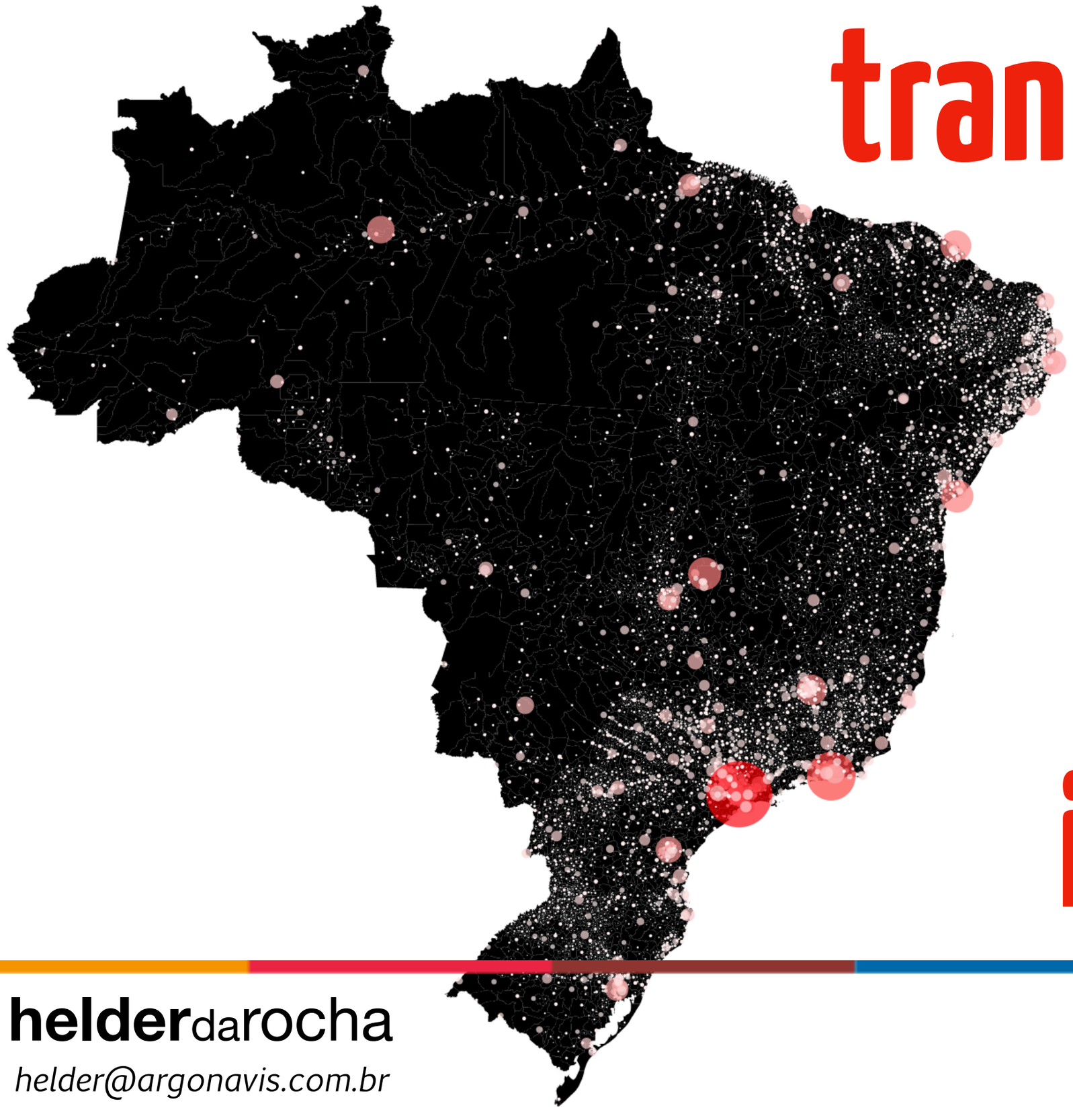
THE DEVELOPER'S CONFERENCE

são paulo, julho/2019

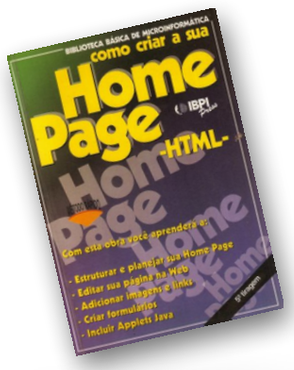


# transformando dados públicos em mapas interativos

**helder**da**rocha**

*helder@argonavis.com.br*

# Quem sou eu? Who am I? Кто я?






## **Helder** da **Rocha**

Tecnologia * Ciência * Arte

HTML & tecnologias Web desde 1995

Autor de cursos e livros sobre

Java, XML e tecnologias Web


*argonavis.com.br*

**helderdarocha.com.br**

# Tutorial prático

- Como usar dados abertos para construir um mapa temático iterativo usando: **HTML**, **CSS**, **JavaScript**, **SVG**, **D3.js**, **CSV**, **GeoJSON**.

    - Obter shapefile de naturalearthdata.com
    - Converter para GeoJSON usando mapshaper.org e filtrar dados
    - Obter dados temáticos de portais públicos: un.data.org e geonames.org
    - Gerar string para path **SVG** usando **D3.js**
    - Aplicar escalas, cores, projeções e recursos interativos

- Código: https://github.com/argonavisbr/tutorial-geojson
- Playground: **JSFiddle** (use links do Readme.md)

# Shapefile: naturalearthdata.com

https://www.naturalearthdata.com

Natural Earth

Free vector and raster map data at
1:10m, 1:50m, and 1:110m scales

Search

Home | Features | Downloads | Blog | Forums | Corrections | About


Rivers, Lake Centerlines

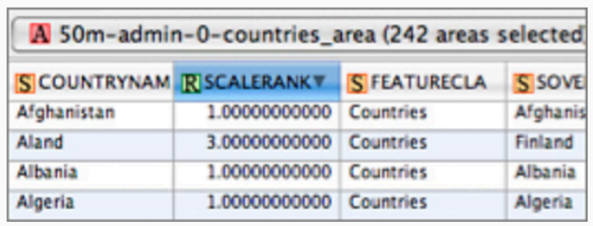
Natural Earth is a public domain map dataset available at 1:10m, 1:50m, and 1:110 million scales. Featuring tightly integrated vector and raster data, with Natural Earth you can make a variety of visually pleasing, well-crafted maps with cartography or GIS software.

Natural Earth was built through a collaboration of many volunteers and is supported by NACIS (North American Cartographic Information Society), and is free for use in any type of project (see our Terms of Use page for more information).

Get the Data

**Convenience**


**Neatness Counts**



A 50m-admin-0-countries_area (242 areas selected

| S COUNTRYNAM | R SCALERANK | S FEATURECLA | S SOVE |
|---|---|---|---|
| Afghanistan | 1.00000000000 | Countries | Afghanis |
| Aland | 3.00000000000 | Countries | Finland |
| Albania | 1.00000000000 | Countries | Albania |
| Algeria | 1.00000000000 | Countries | Algeria |

**GIS Attributes**

# www.naturalearthdata.com/downloads/110m-cultural-vectors/

https://www.naturalearthdata.com/downloads/110m-cultural-vectors/

Natural Earth

Free vector and raster map data at 1:10m, 1:50m, and 1:110m scales

Search

Home | Features | Downloads | Blog | Forums | Corrections | About

« Downloads

## 1:110m Cultural Vectors

Download all 110m cultural themes (1.31 MB) version 4.1.0

Files have been downloaded 475,243 times.

*NOTE: Version number indicates the update cycle when that theme was last updated. An older version number indicates updates have not been necessary since then.*

### Admin 0 – Countries

*There are 247 countries i… Greenland as separate from Denmark.*

Download countries (192.15 KB) version 4.1.0

Download without boundary lakes (194.37 KB) version 4.1.0

About | Issues | Version History »

### Admin 0 – Details

https://www.naturalearthdata.com/http//www.naturalearthdata.com/download/110m/cultural/ne_110m_admin_0_countries.zip

### Stay up to Date

Know when a new version of Natural Earth is released by subscribing to our announcement list.

### Find a Problem?

Submit suggestions and bug reports via our correction system and track the progress of your edits.

### Join Our Community

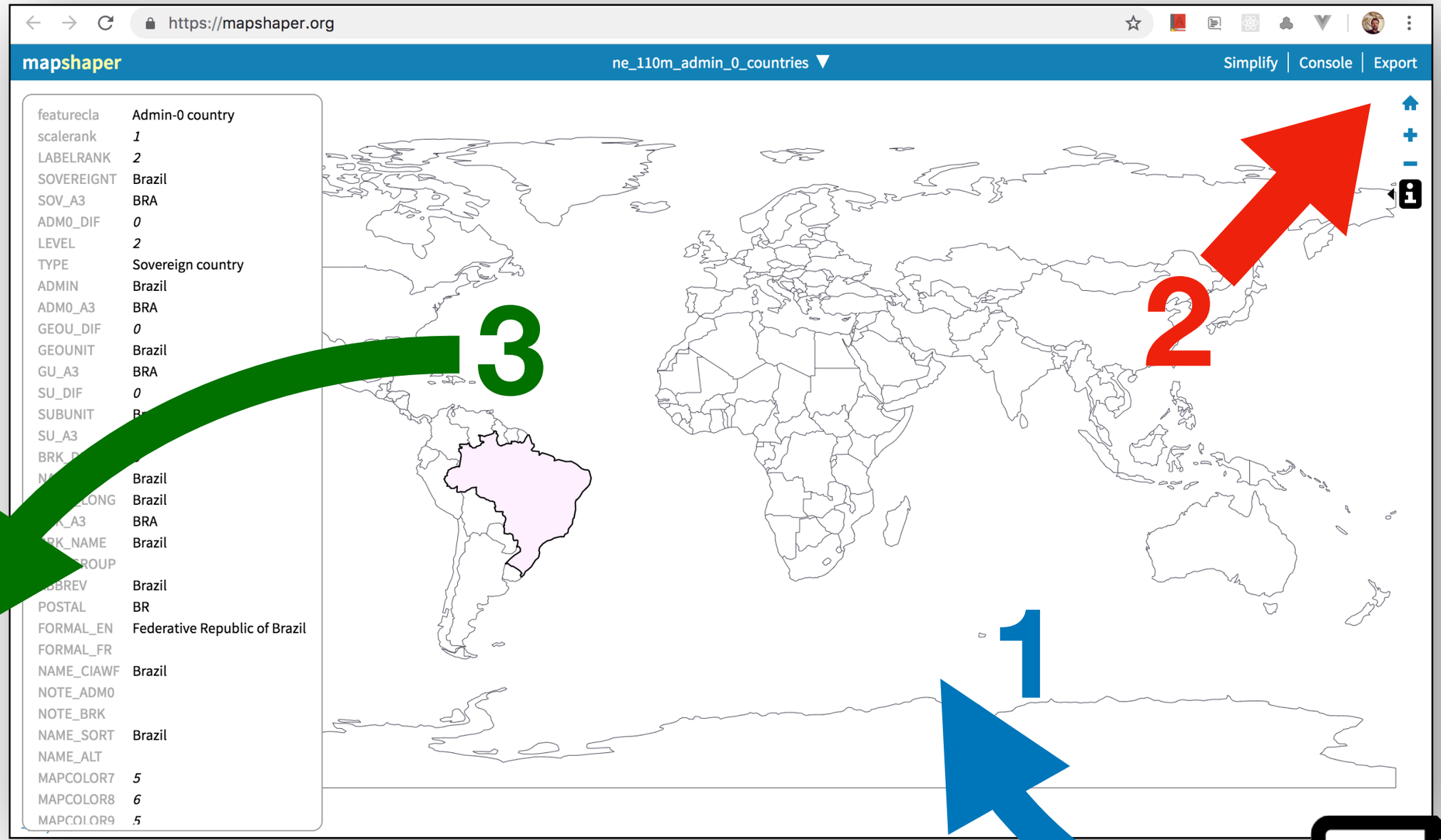
Talk back and discuss Natural Earth in the Forums.

### Thank You

Our data downloads are generously hosted by Florida State University.

…ecent Forum Topics

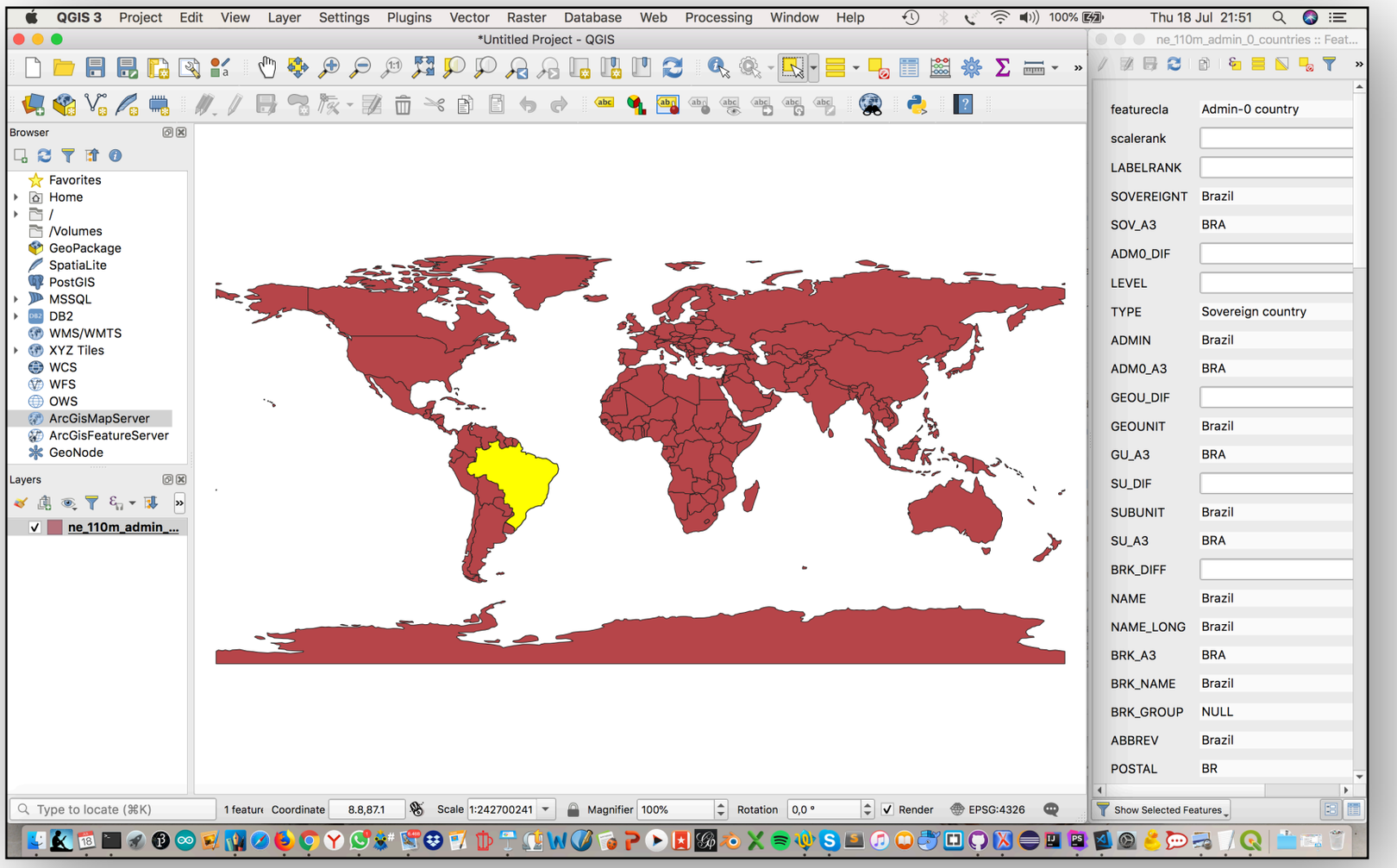
# Converter para GeoJSON: mapshaper.org




ne_110m_admin_0_countries.json

ne_110m_admin_0_countries.zip

# Alternativas: aplicativo **GIS** ou **GDAL**




QGIS

qgis.org

gdal.org

GDAL

```
$ ogr2ogr -f GeoJSON resultado.geojson ne_110m_admin_0_countries.json.shp
```

# GeoJSON

Propriedade "id"
(opcional)

Propriedade "properties"
(conteúdo livre)

Propriedade "geometry"
(contém um tipo GeoJSON
ex: "Polygon", "Point", "MultiPolygon")

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "id": "FJI",
      "properties": {
        "name": "Fiji"
      },
      "geometry": {
        "type": "MultiPolygon",
        "coordinates": [
          [[  [180,-16.067132663642447],
              ...
              [179.4135093629971,-16.379054277547404],
              [180,-16.067132663642447]  ]],
          [[ ... ]], [[ ... ]]
        ]
      }
    },{
      "type": "Feature",
          ...
    },
    ...
  ]
}
```

# Remover propriedades desnecessárias

Welcome | filter-map.js

```
1   const fs = require('fs');
2   const source = "../geo/ne_110m_admin_0_countries.json";
3   const target = "../geo/world.geojson";
4
5   fs.readFile(source, 'utf8', (err, json) => {
6       if(err) throw err;
7       else process(JSON.parse(json));
8   });
9
10  function process(data) { // filter id (iso-a3) and name of country
11      data.features.forEach(function(d) {
12          d.id = d.properties.ISO_A3;
13          const name = d.properties.NAME;
14          d.properties = {name: name};
15      });
16      writeFile(JSON.stringify(data));
17  }
18
19  function writeFile(data) {
20      fs.writeFile(target, data, (err) => {
21          if(err) throw err;
22          console.log('Done')
23      });
24  }
```

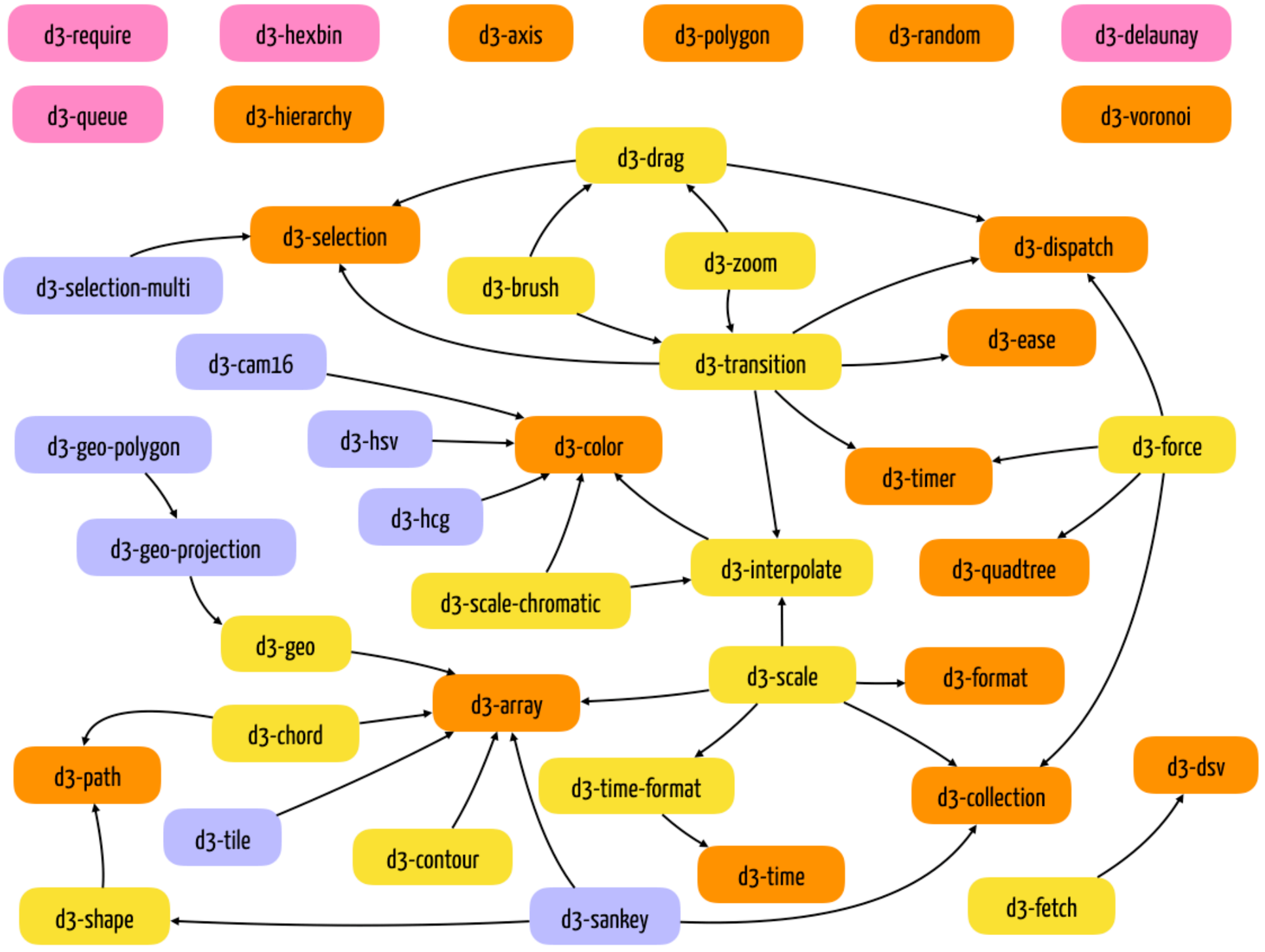
GitHub: tools/filter-map.js

## Para cada Feature:

```
{
  "type": "Feature",
  "properties": {
     "ISO_A3":"...",
     "NAME":"...",
     "ADMIN":"...",
     ...,
     "MAPCOLOR7":"..."
  },
  "geometry": {...}
}
```

```
{
  "type": "Feature",
  "id":"...",
  "properties": {
     "name":"...",
  },
  "geometry": {...}
}
```

# Data-Driven Documents




## D3.js v5 modules

Not included in default bundle
Included in default bundle

Has no dependencies
Has dependencies

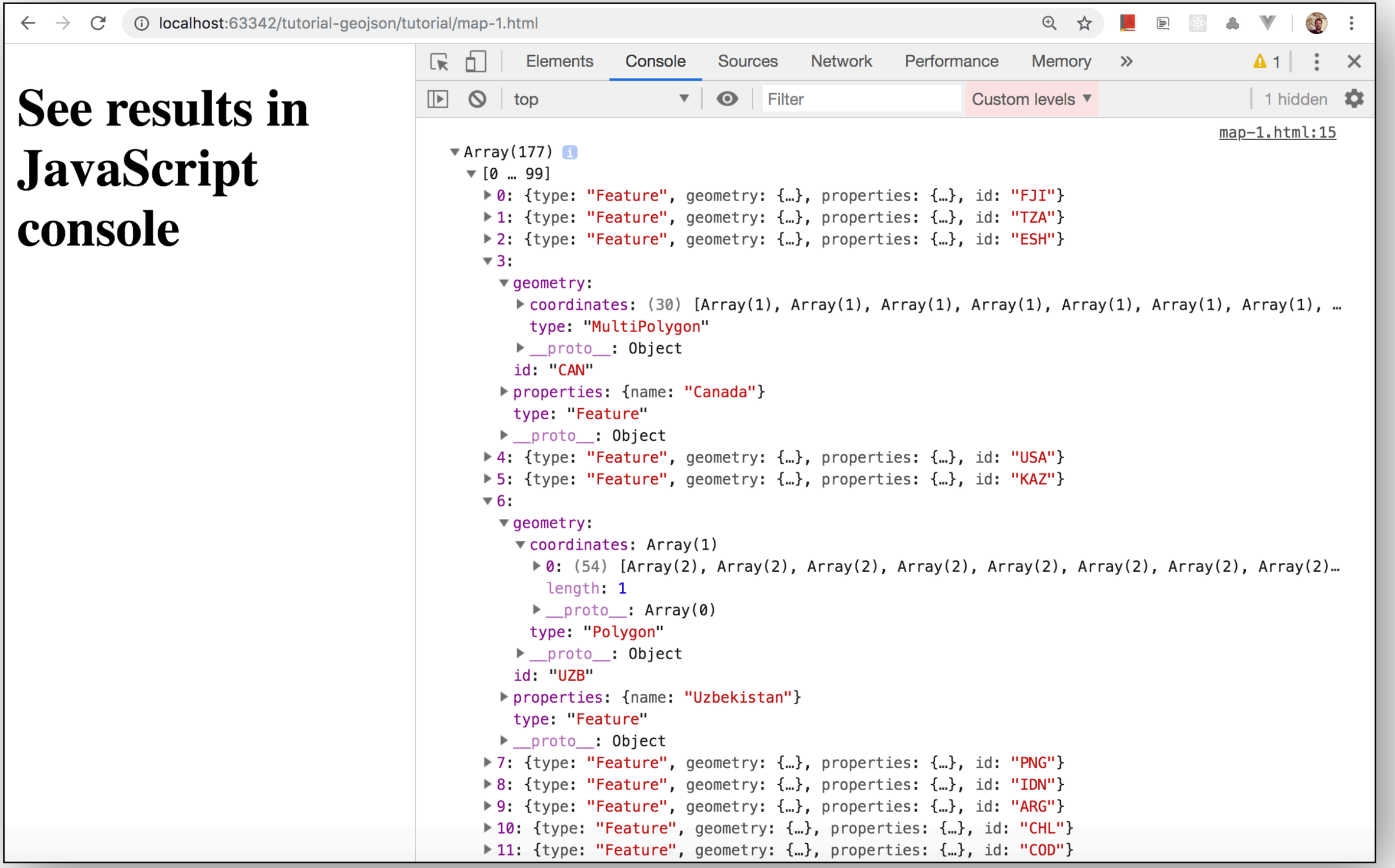
# Data-Driven Documents

```
<script src="https://d3js.org/d3.v5.min.js"></script>

<script>
    d3.json('../geo/world.geojson')
        .then(function(data) {
            console.log(data.features);
    });
</script>
```

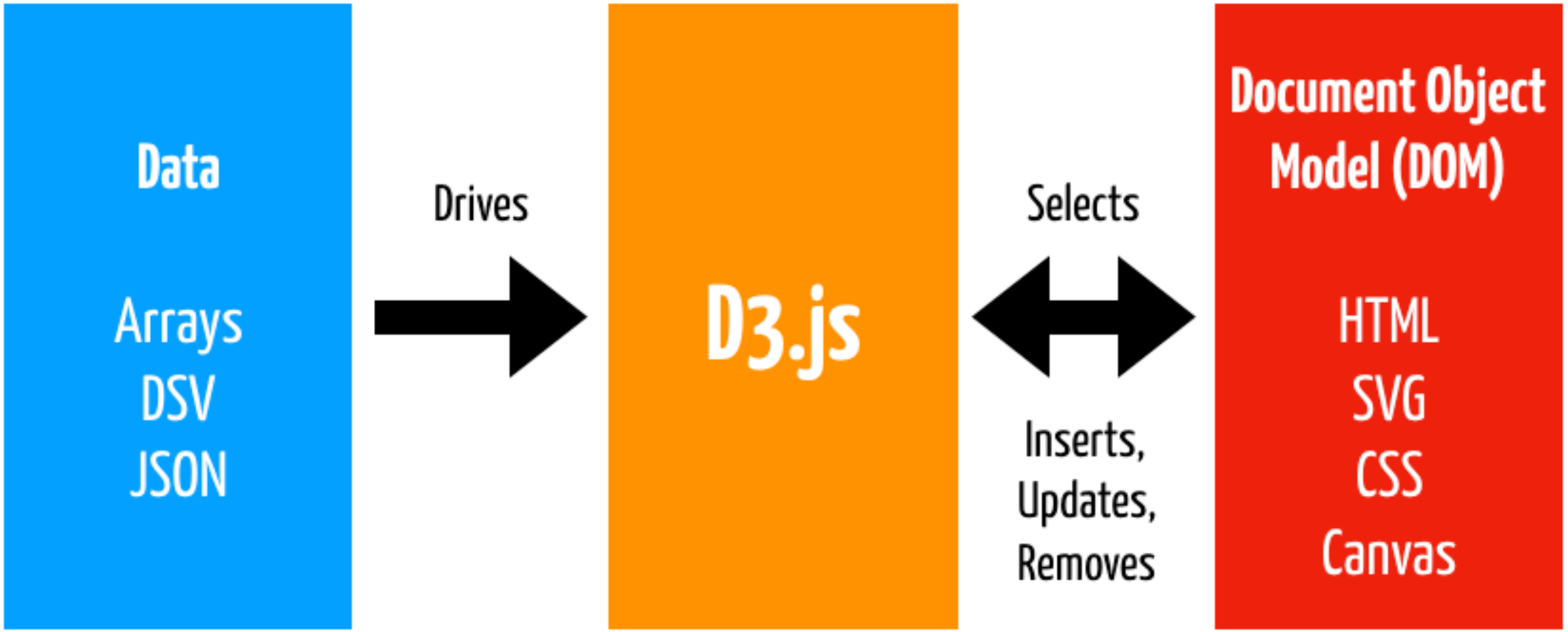
```
{
  "type": "FeatureCollection",
  "features": [ ... ]
}
```

# Passo 1: Carregar o GeoJSON

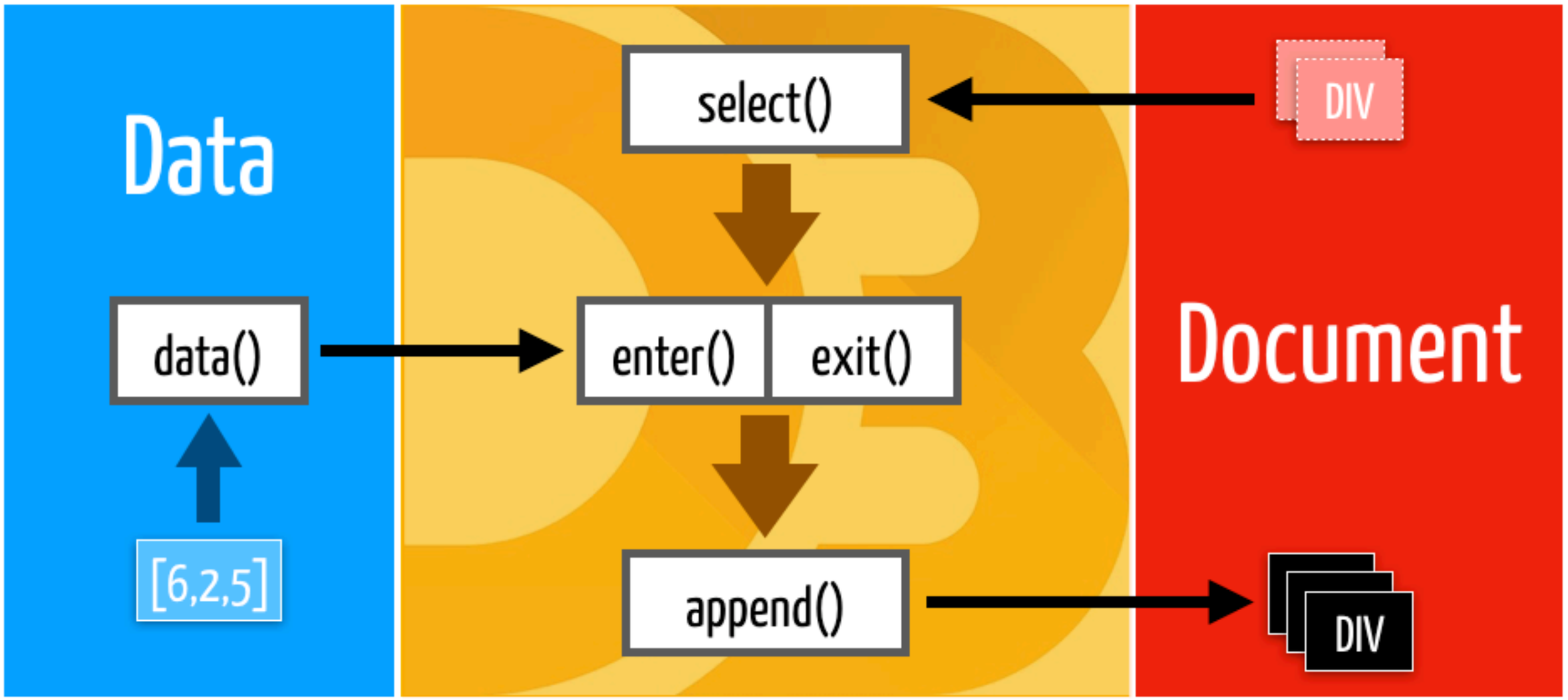
## GitHub: tutorial/map-1.html




# Data-Driven Documents



# Data-Driven Documents



# Passo 2: Listar propriedades

## GitHub: tutorial/map-2.html

```
const map = {};
d3.json('../geo/world.geojson')
   .then(function(data) {
      map.features   = data.features;
      draw();
});


function draw() {
     const ol = d3.select("body").append("ol");
     ol.selectAll("li")
        .data(map.features)
        .enter().append("li")
        .text(d => d.properties.name +
                   " ("+ d.id + ")");
}
```

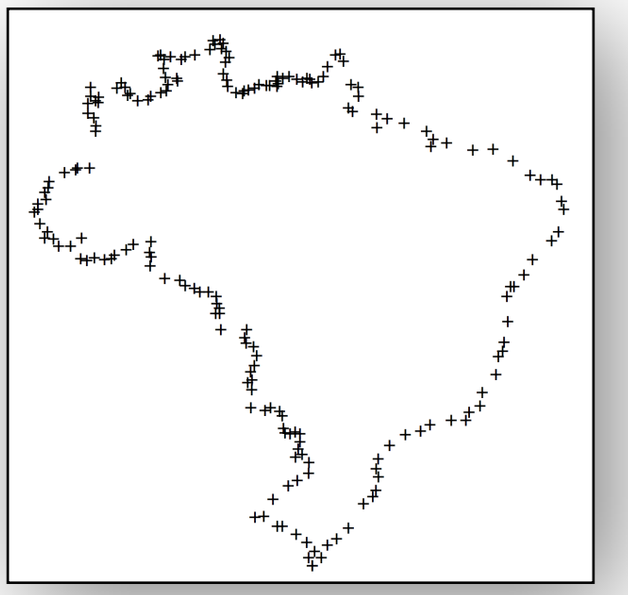
map.features[n]

```
{ "type": "Feature",
  "id":"...",
  "properties": {
     "name":"...",
   },
   "geometry": {...} }
```

# Passo 3: Listar coordenadas de objeto

## GitHub: tutorial/map-3.html

```
const country =
  map.features.filter(k => k.id === "BRA")[0];

const body = d3.select("body");
body.append("h1")
    .text(country.properties.name);

body.append("ul").selectAll("li.coord")
   .data(country.geometry.coordinates[0])
   .enter()
   .append("li").attr("class", "coord")
   .text(d => "Lon: "+d[0]+", Lat: "+d[1]);
```



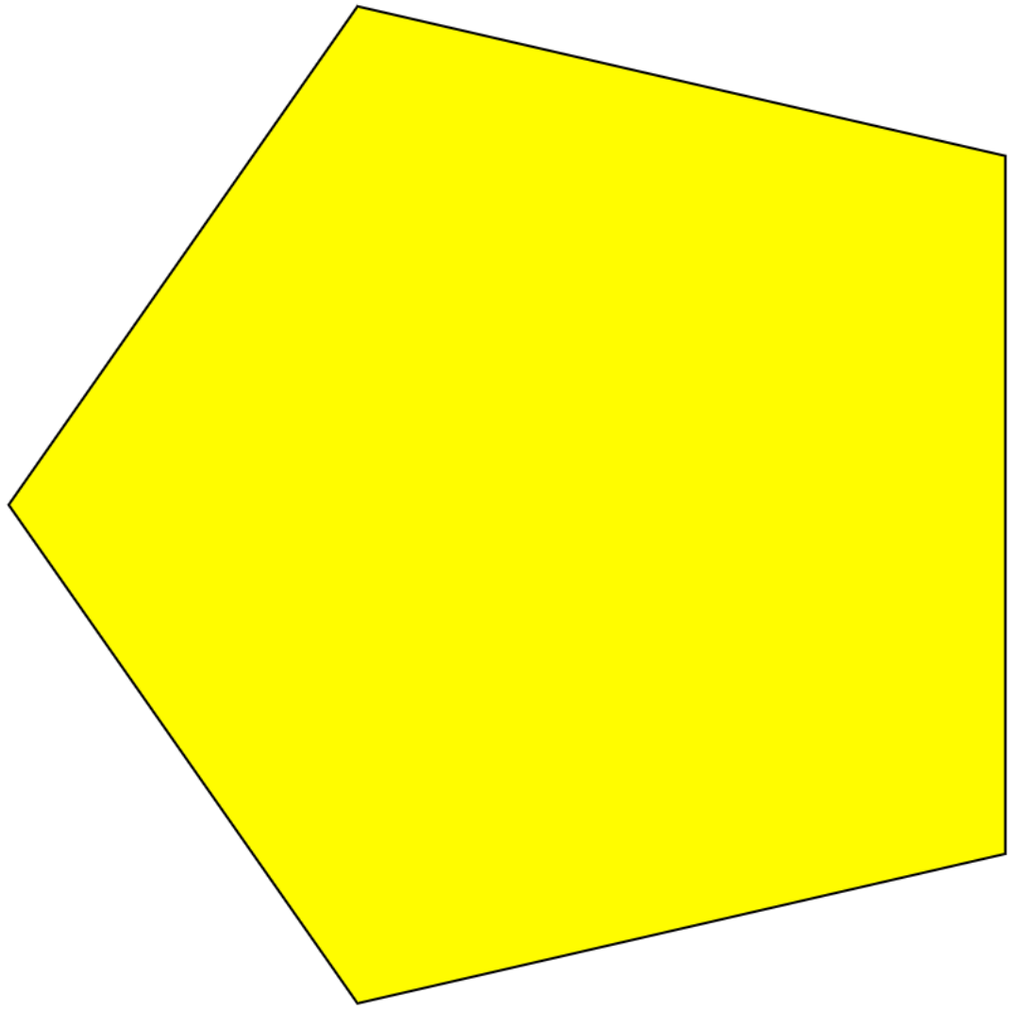
localhost:63342/tutorial-geojson/tutorial/map-3.html

# Brazil

- Lon: -53.373661668498244, Lat: -33.768377780900764
- Lon: -53.6505439927181, Lat: -33.20200408298183
- Lon: -53.209588995971544, Lat: -32.727666110974724
- Lon: -53.78795162618219, Lat: -32.047242526987624
- Lon: -54.57245154480512, Lat: -31.494511407193748
- Lon: -55.601510179249345, Lat: -30.853878676071393
- Lon: -55.97324459494094, Lat: -30.883075860316303
- Lon: -56.976025763564735, Lat: -30.109686374636127
- Lon: -57.62513342958296, Lat: -30.21629485445426
- Lon: -56.29089962423908, Lat: -28.852760512000895
- Lon: -55.16228634298457, Lat: -27.881915378533463
- Lon: -54.490725267135524, Lat: -27.47475676850579
- Lon: -53.64873531758789, Lat: -26.92347258881609
- Lon: -53.628348965048744, Lat: -26.124865004177472
- Lon: -54.13004960795439, Lat: -25.547639255477254
- Lon: -54.625290696823576, Lat: -25.739255466415514
- Lon: -54.42894609233059, Lat: -25.162184747012166
- Lon: -54.29347632507745, Lat: -24.570799655863965
- Lon: -54.29295956075452, Lat: -24.02101409271073
- Lon: -54.65283423523513, Lat: -23.83957813893396
- Lon: -55.02790178080955, Lat: -24.00127369557523
- Lon: -55.40074723979542, Lat: -23.956935316668805
- Lon: -55.517639329639636, Lat: -23.571997572526637

# Passo 4: SVG <path>

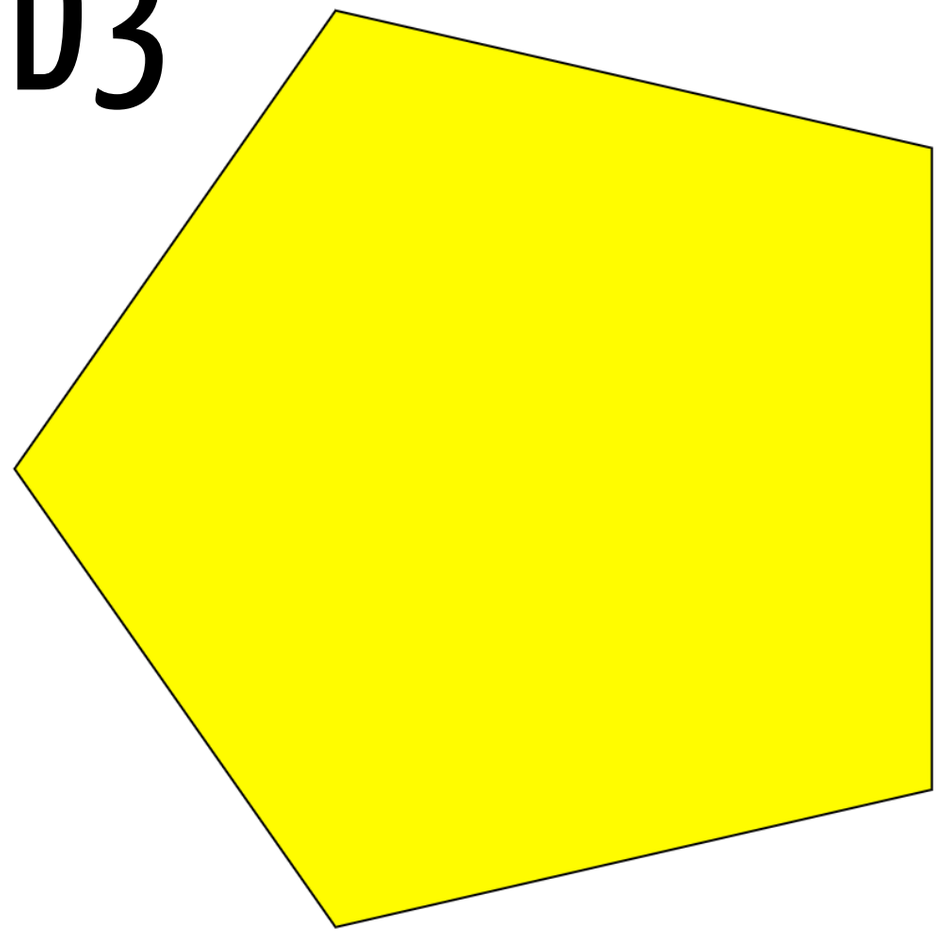
**GitHub: tutorial/map-6a.html**



```
<html lang="en">
<head>
    <style>
        .poly {
            stroke: black;
            stroke-width: .05;
            fill: yellow;
        }
    </style>
</head>
<body>
  <svg height="500" width="960" viewBox="7 -13 14 28">
    <path d="M-10,0 L-3,10 L10,7 L10,-7 L-3,-10 L-10,0" class="poly" />
  </svg>
</body>
</html>
```

# Passo 5: SVG <path> em D3

**GitHub: tutorial/map-8.html**



```
<svg height="500" width="960"></svg>

<script>
    const svg = d3.select("svg");
    const pathData = "M-10,0 L-3,10 L10,7 L10,-7 L-3,-10 Z";

    svg.append("path").attr("class", "poly")
                      .attr("d", pathData);
</script>
```

# Passo 6: Gerando um <path> com D3

**GitHub: tutorial/map-9.html**

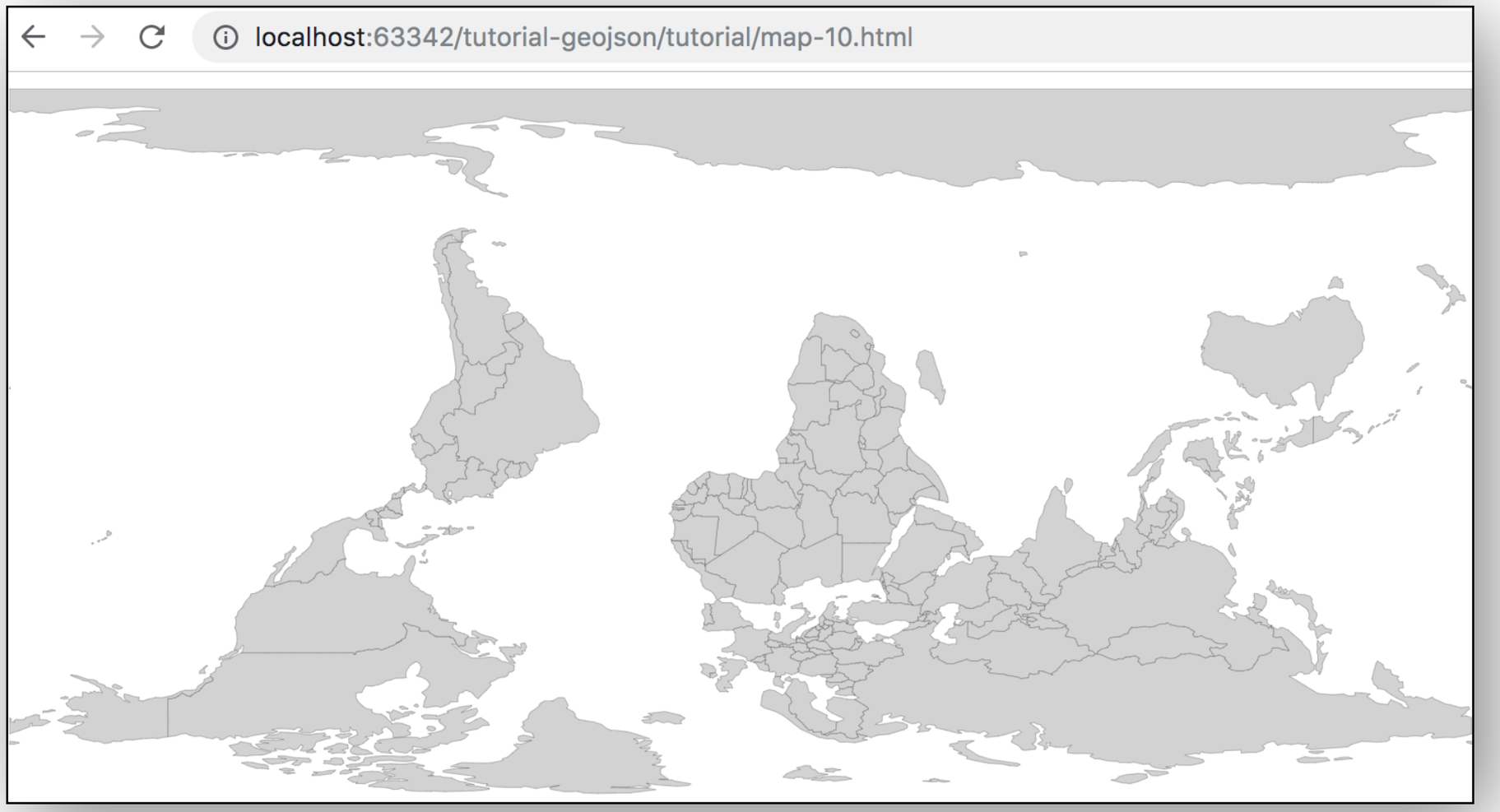

```
const geoPath = d3.geoPath();

const map = {};
d3.json('../geo/world.geojson')
   .then(function(data) {
        map.features   = data.features;
        draw();
});

function draw() {
     const country = map.features.filter(k => k.id === "BRA")[0];
     const pathData = geoPath(country);
     d3.select("svg").append("path")
                     .attr("d", pathData);
}
```

# Passo 7: Mapeando dados ao DOM SVG

GitHub: tutorial/map-10.html

Cada feature mapeada
a um objeto <path>

```
const geoPath = d3.geoPath();
svg.selectAll("path")
        .data(map.features)
        .enter().append("path")
        .attr("d", geoPath)
```



# Passo 8: Projeções geográficas

## GitHub: tutorial/map-12.html

```
const geoPath = d3.geoPath();
const projection = d3.geoMercator(); //d3.geoOrthographic(), d3.geoMollweide()...
geoPath.projection(projection);
```

# Coordenadas de cidades: geonames.org

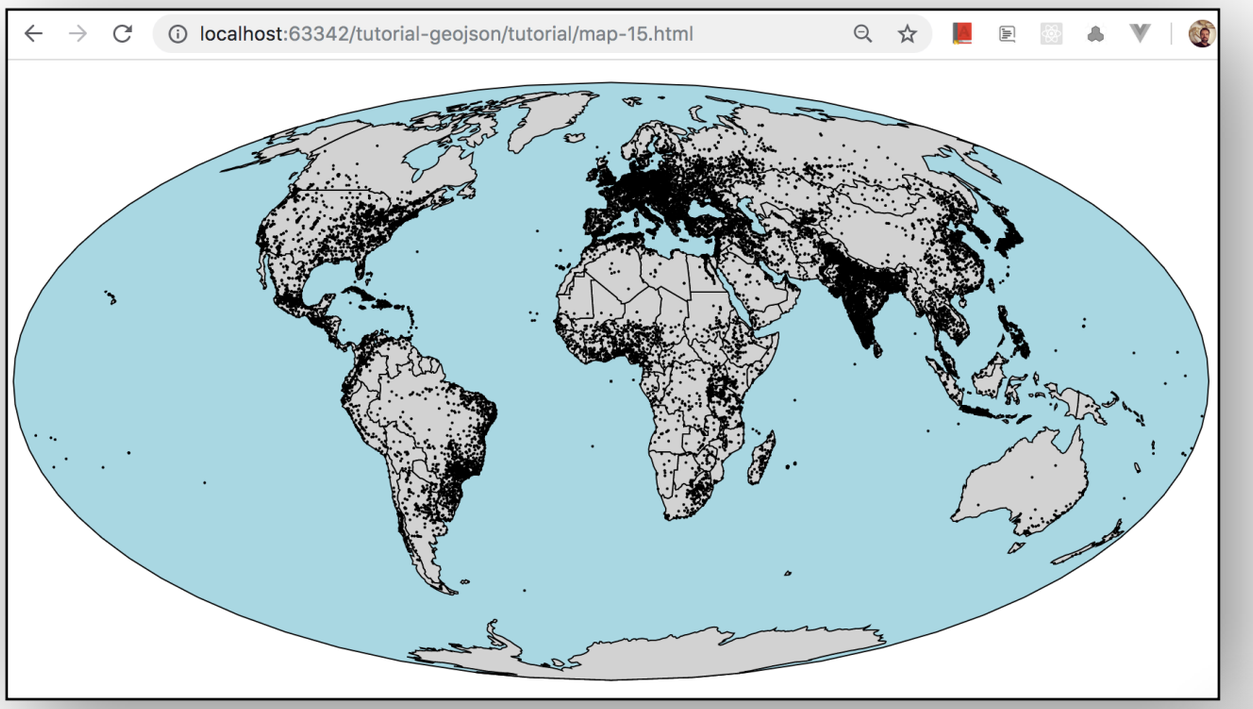
```
geonameid;asciiname;latitude;longitude;population;timezone
3040051;les Escaldes;42.50729;1.53414;15853;Europe/Andorra
3041563;Andorra la Vella;42.50779;1.52109;20430;Europe/Andorra
290594;Umm al Qaywayn;25.56473;55.55517;44411;Asia/Dubai
291074;Ras al-Khaimah;25.78953;55.9432;115949;Asia/Dubai
291696;Khawr Fakkan;25.33132;56.34199;33575;Asia/Dubai
292223;Dubai;25.0657;55.17128;1137347;Asia/Dubai
292231;Dibba Al-Fujairah;25.59246;56.26176;30000;Asia/Dubai
292239;Dibba Al-Hisn;25.61955;56.27291;26395;Asia/Dubai
292672;Sharjah;25.33737;55.41206;543733;Asia/Dubai
292688;Ar Ruways;24.11028;52.73056;16000;Asia/Dubai
292878;Al Fujayrah;25.11641;56.34141;62415;Asia/Dubai
292913;Al Ain;24.19167;55.76056;408733;Asia/Dubai
```

## GitHub: csv/cities15000.csv

(23505 localidades)

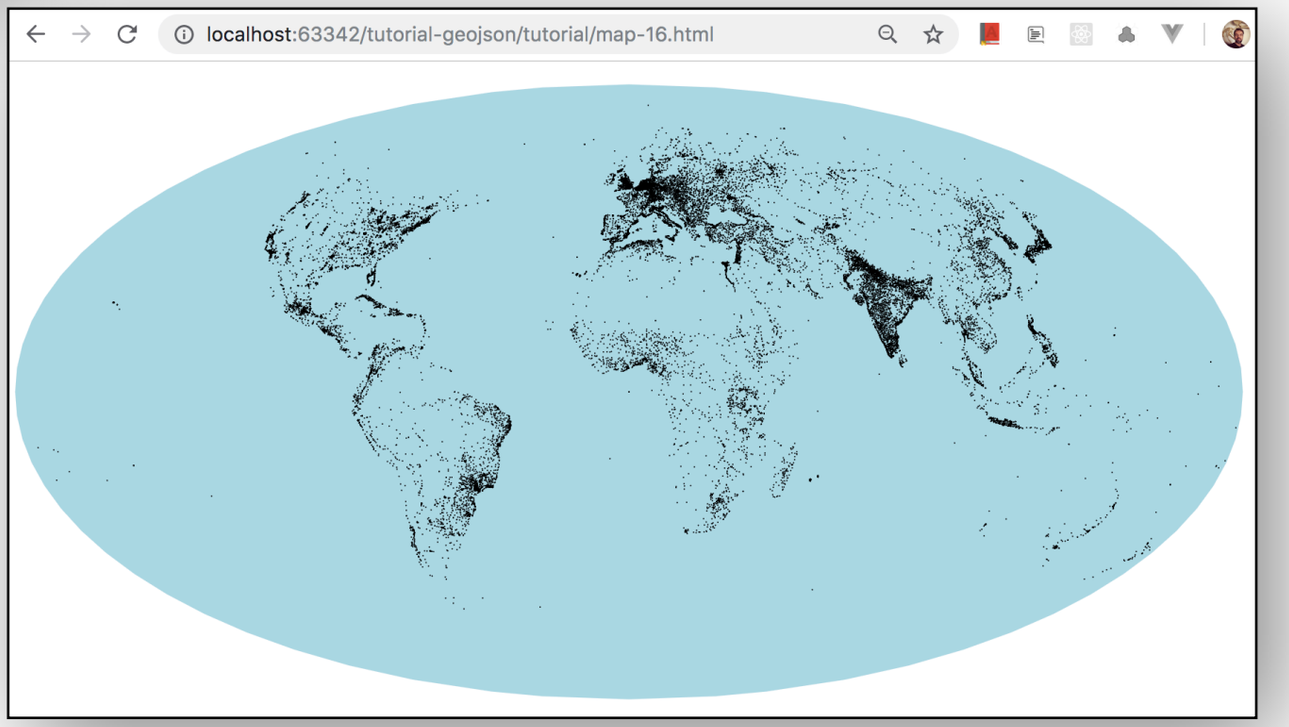
# Passo 9: Dados temáticos: cidades

## GitHub: tutorial/map-15.html



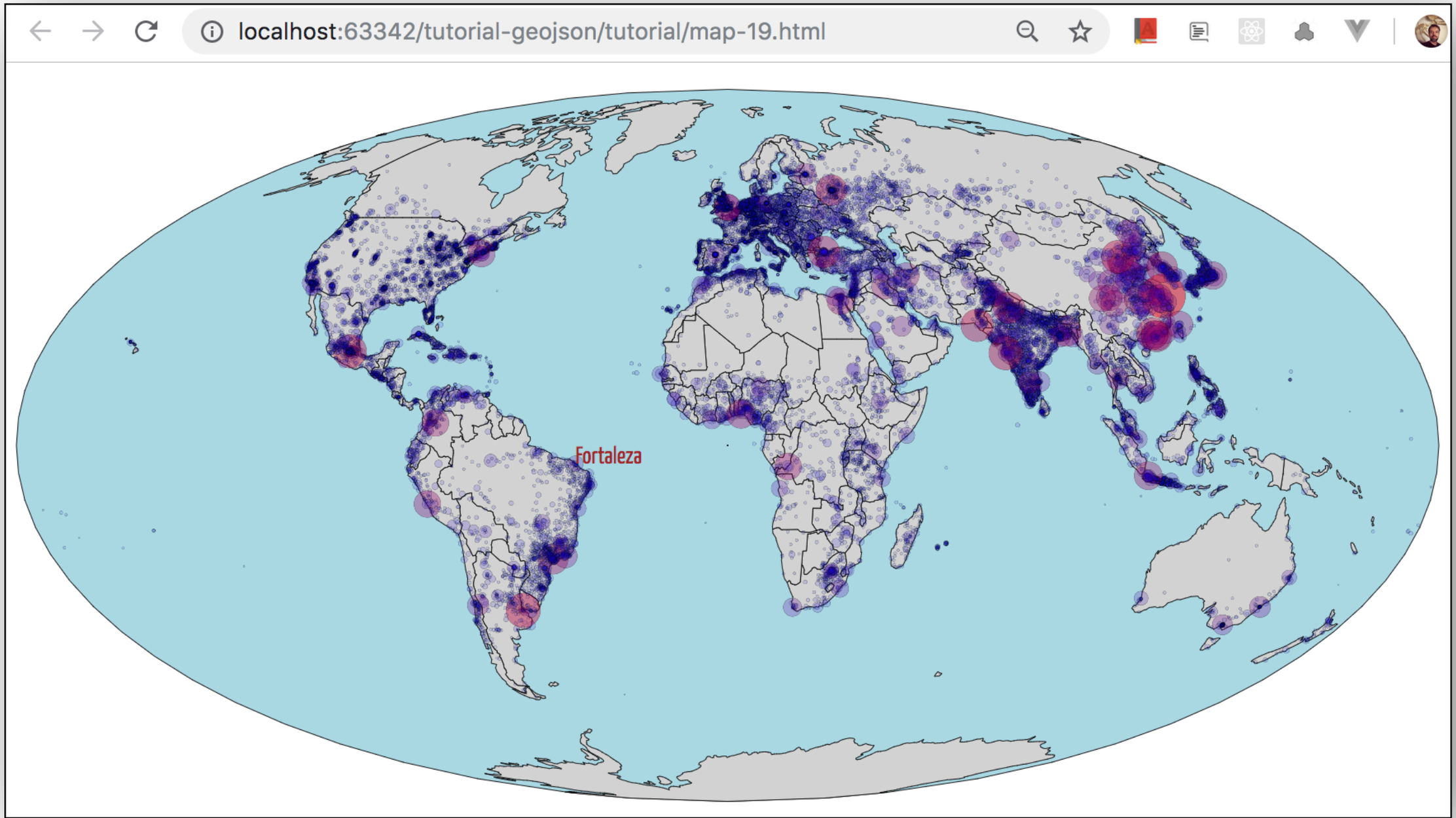

```
Promise.all([
  d3.json('../geo/world.geojson'),
  d3.dsv(';', '../csv/cities15000.csv',
    function(row) {
      return {
        name: row.asciiname,
        coords: [+row.longitude, +row.latitude]
      }
    })
]).then(function([shapes, points]) {
    map.features   = shapes.features;
    map.points     = points;
    draw();
});
```

```
svg.selectAll("circle.city")
  .data(map.points)
  .enter()
  .append("circle").attr("class", "city")
    .attr("cx", d => projection(d.coords)[0])
    .attr("cy", d => projection(d.coords)[1])
    .attr("r", d => 1)
```



# Passo 10: Interatividade

## GitHub: tutorial/map-19.html




# Conclusões

- **D3.js** é uma poderosa biblioteca para visualização de dados baseada em **padrões abertos**

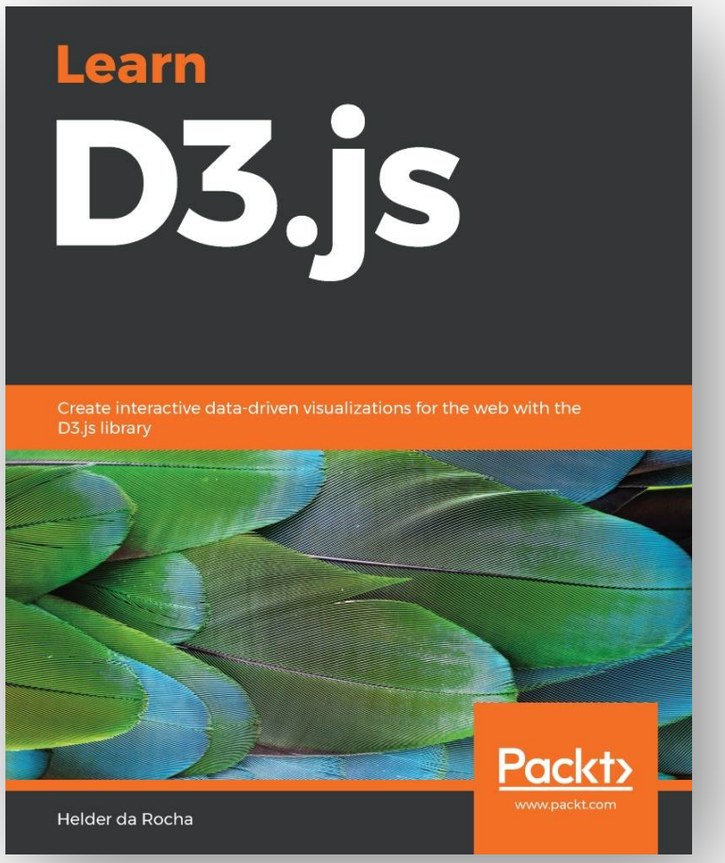
- Usada em grandes portais (NY Times, The Guardian, etc.) para visualizações e mapas interativos e **base** para outros frameworks de visualização

- Neste tutorial, foi mostrado como D3 pode ser usada para criar **mapas temáticos** usando **dados abertos**: uma ferramenta essencial em jornalismo de dados

helder da rocha

helder@argonavis.com.br

www.amazon.com/dp/B07RFBV4PC




Slides, +código-fonte e +demonstrações

**github.com/argonavisbr/tutorial-geojson**

**www.argonavis.com.br/download/tdc_2019_js.html**

www.amazon.com/dp/B07PDBBHLL