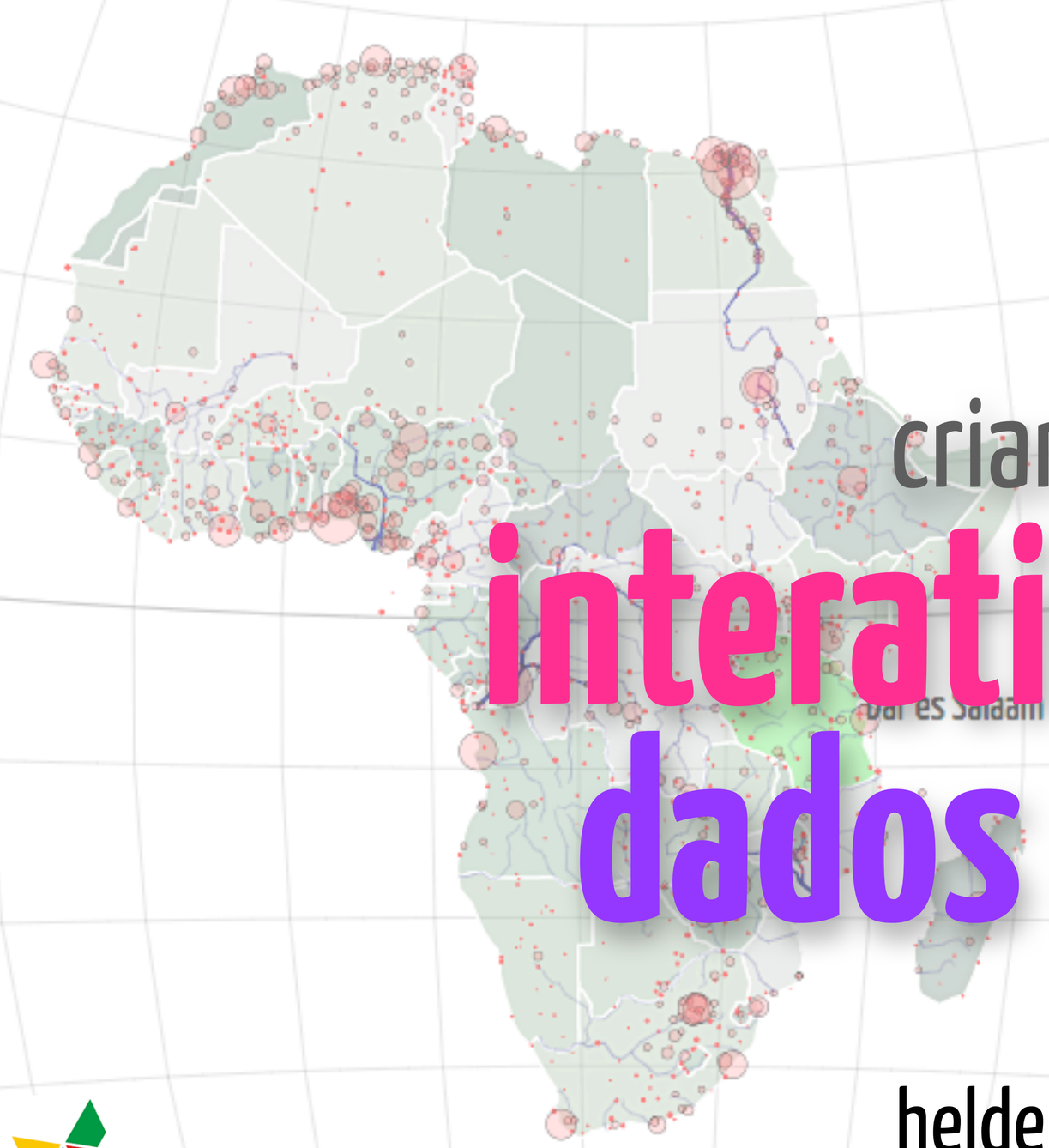


CODA·BR

criando mapas temáticos interativos para a web com dados abertos e D3.js

helder da rocha
helder@argonavis.com.br

SÃO PAULO, 24 de novembro de 2019

Tutorial prático

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

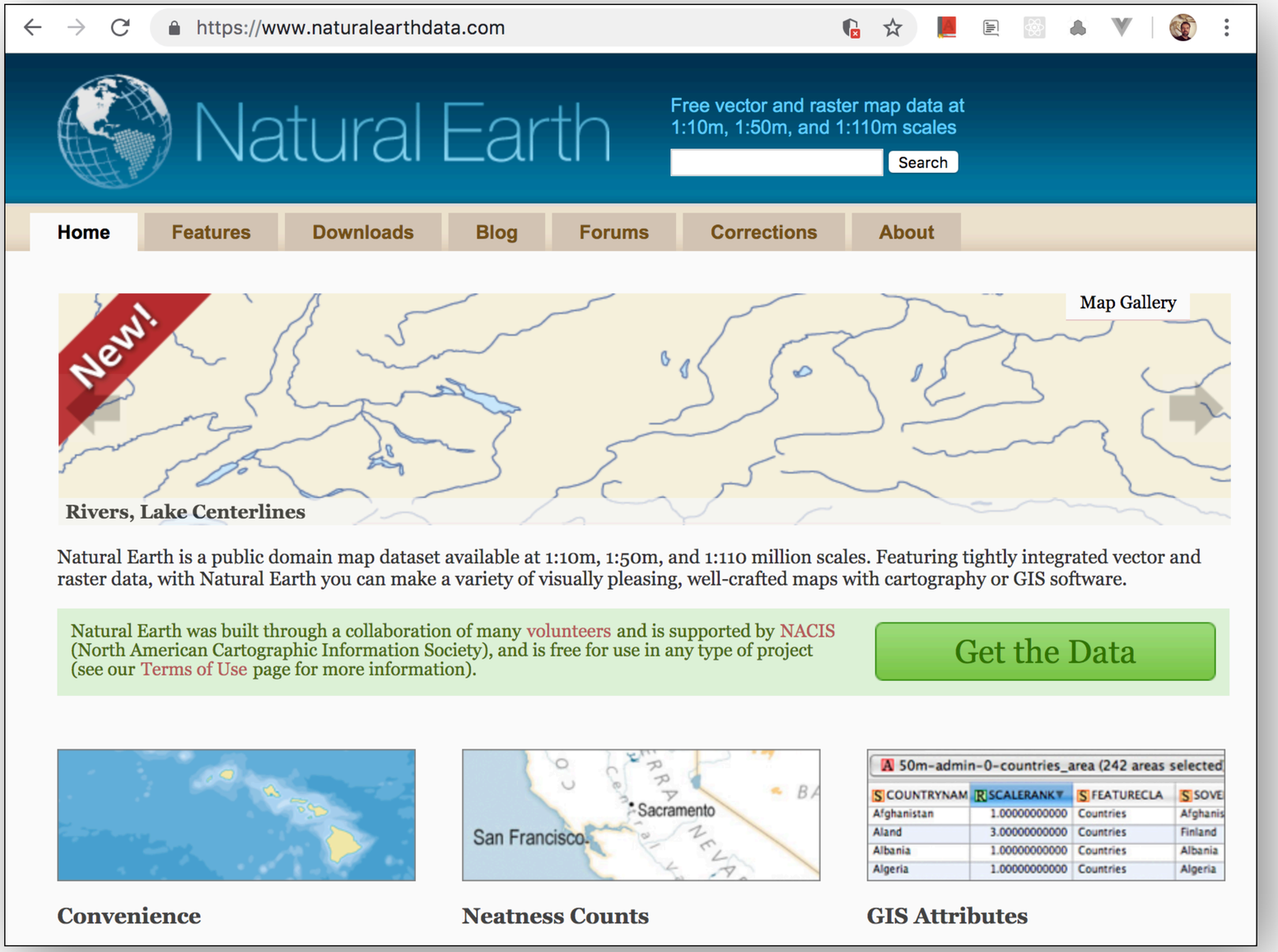
Código

https://github.com/argonavisbr/tutorial-geojson

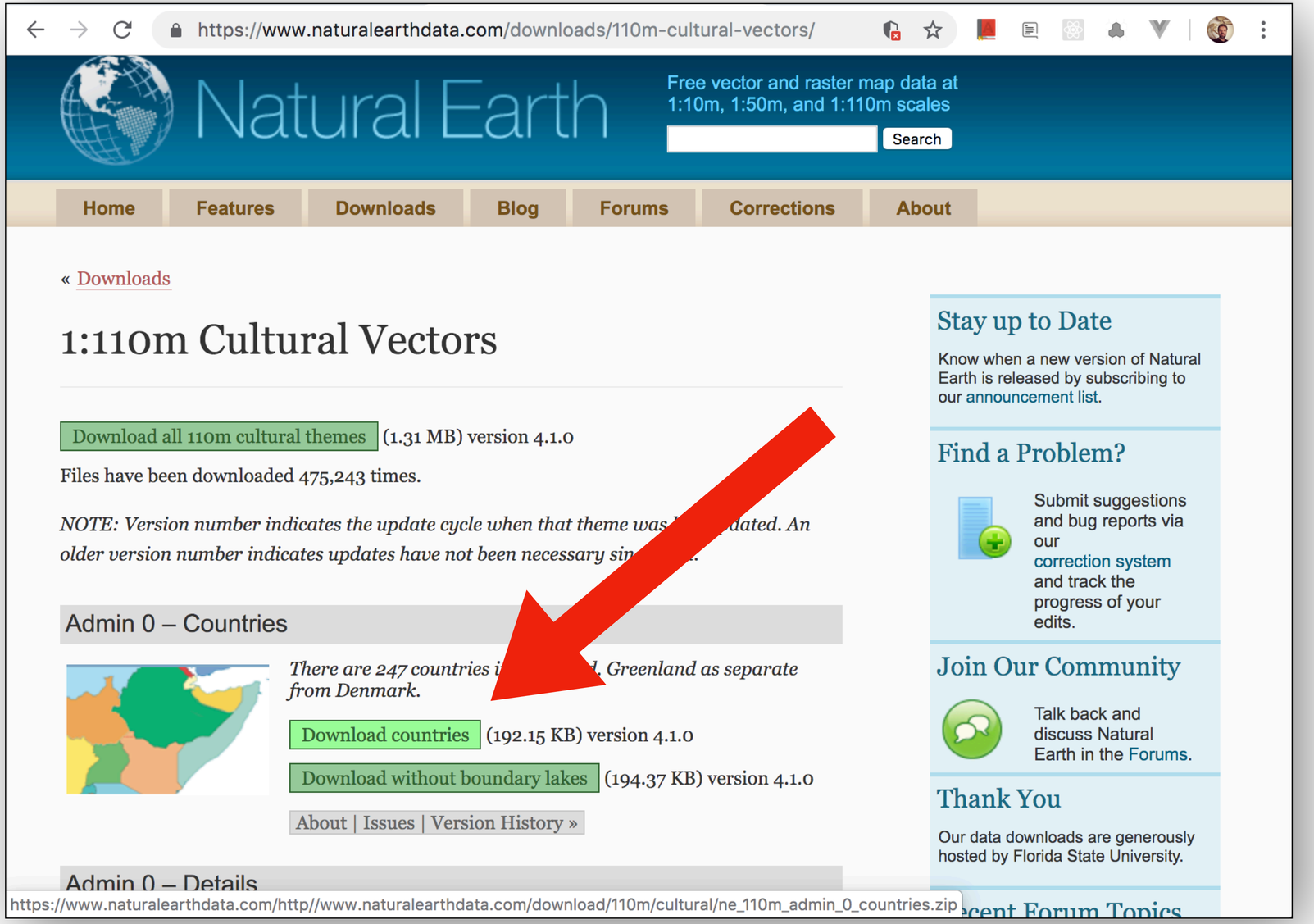
Playground

CodePen (links no Readme.md)

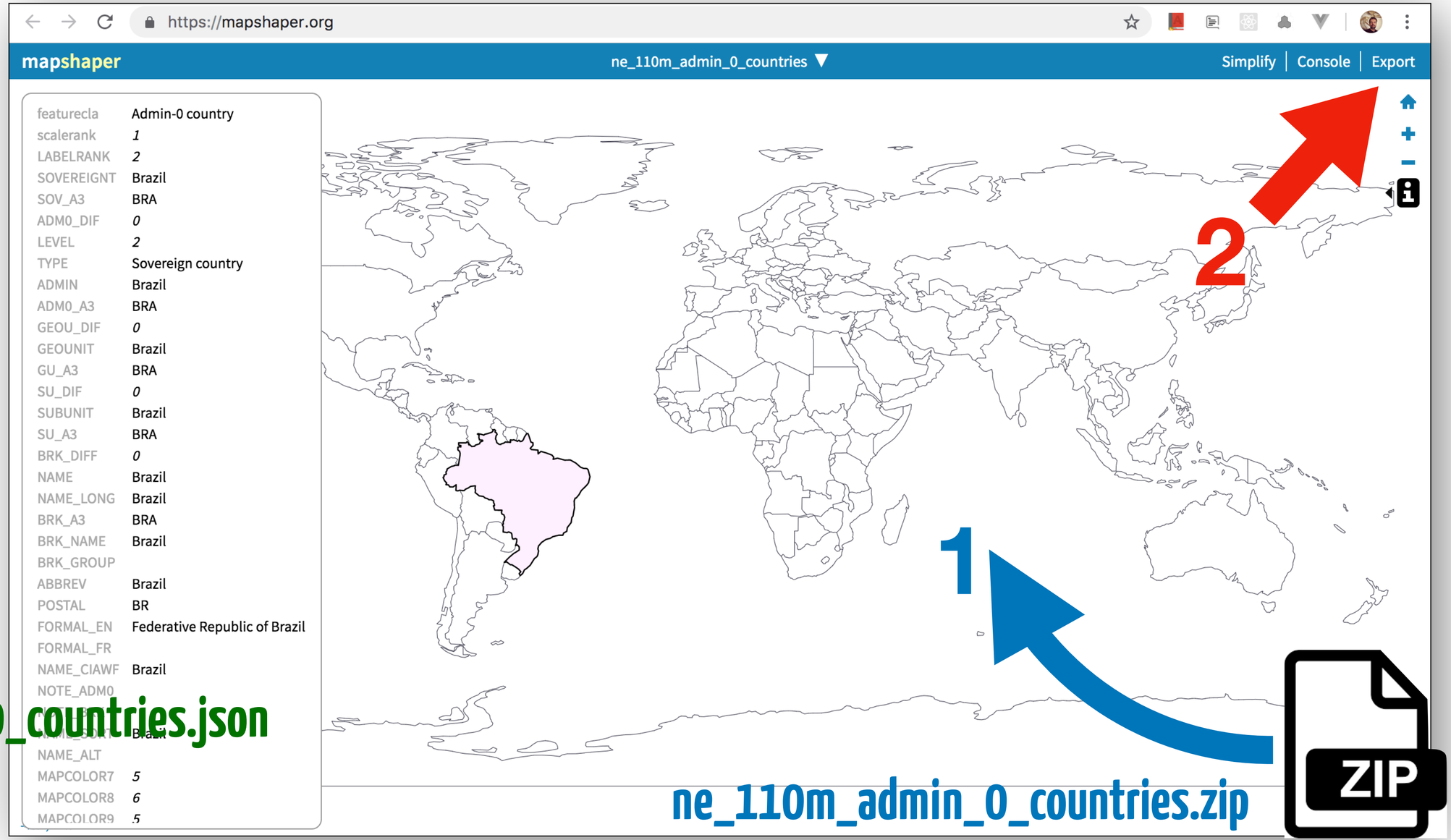
Shapefile: naturalearthdata.com



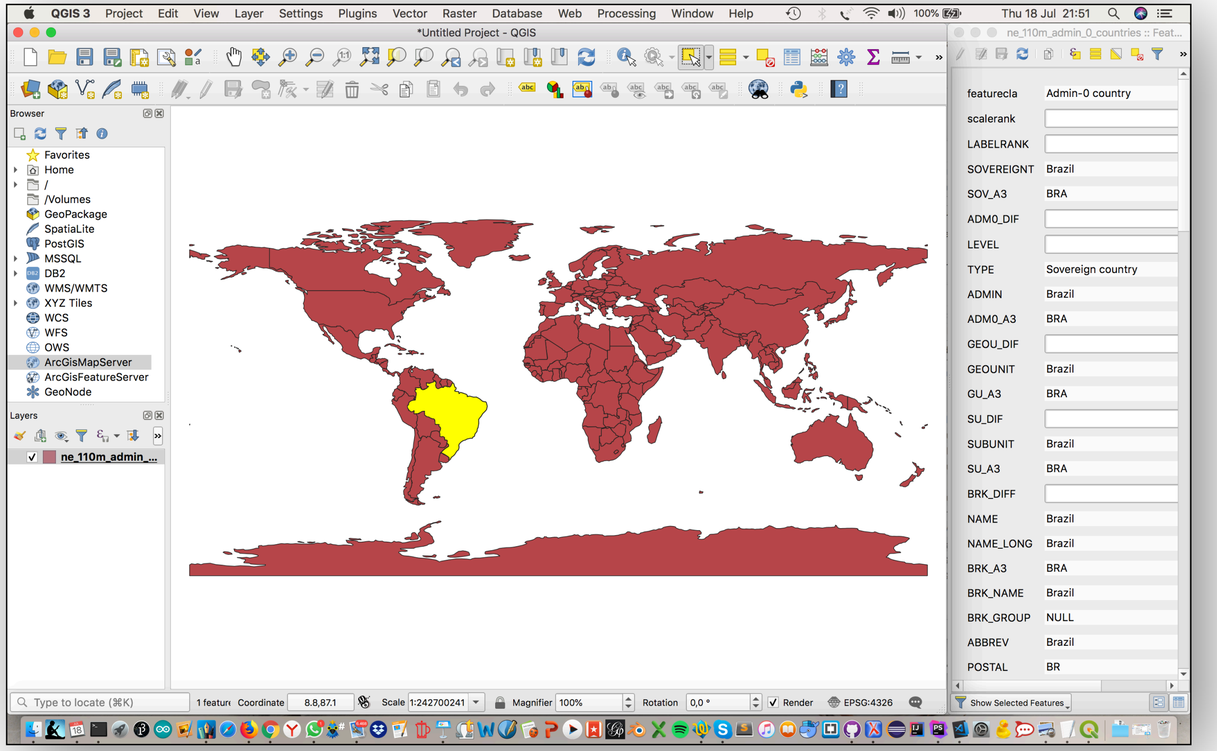
www.naturalearthdata.com/downloads/110m-cultural-vectors/



Converter para **GeoJSON**: mapshaper.org



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



qgis.org

gdal.org

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

GeoJSON

```
{
  "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", ... },
    { "type": "Feature", ... },
    ...
  ]
}
```

GeoJSON

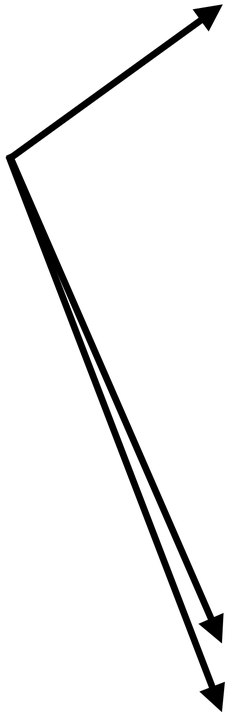
FeatureCollection

Propriedade **feature** contém uma lista (array) de elementos do tipo **Feature**

```
{
  "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", ... },
    { "type": "Feature", ... },
    ...
  ]
}
```

GeoJSON

Cada elemento **Feature** contém propriedades **properties**, **geometry**, **id**



```
{
  "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", ... },
    { "type": "Feature", ... },
    ...
  ]
}
```

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", ... },
    { "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:

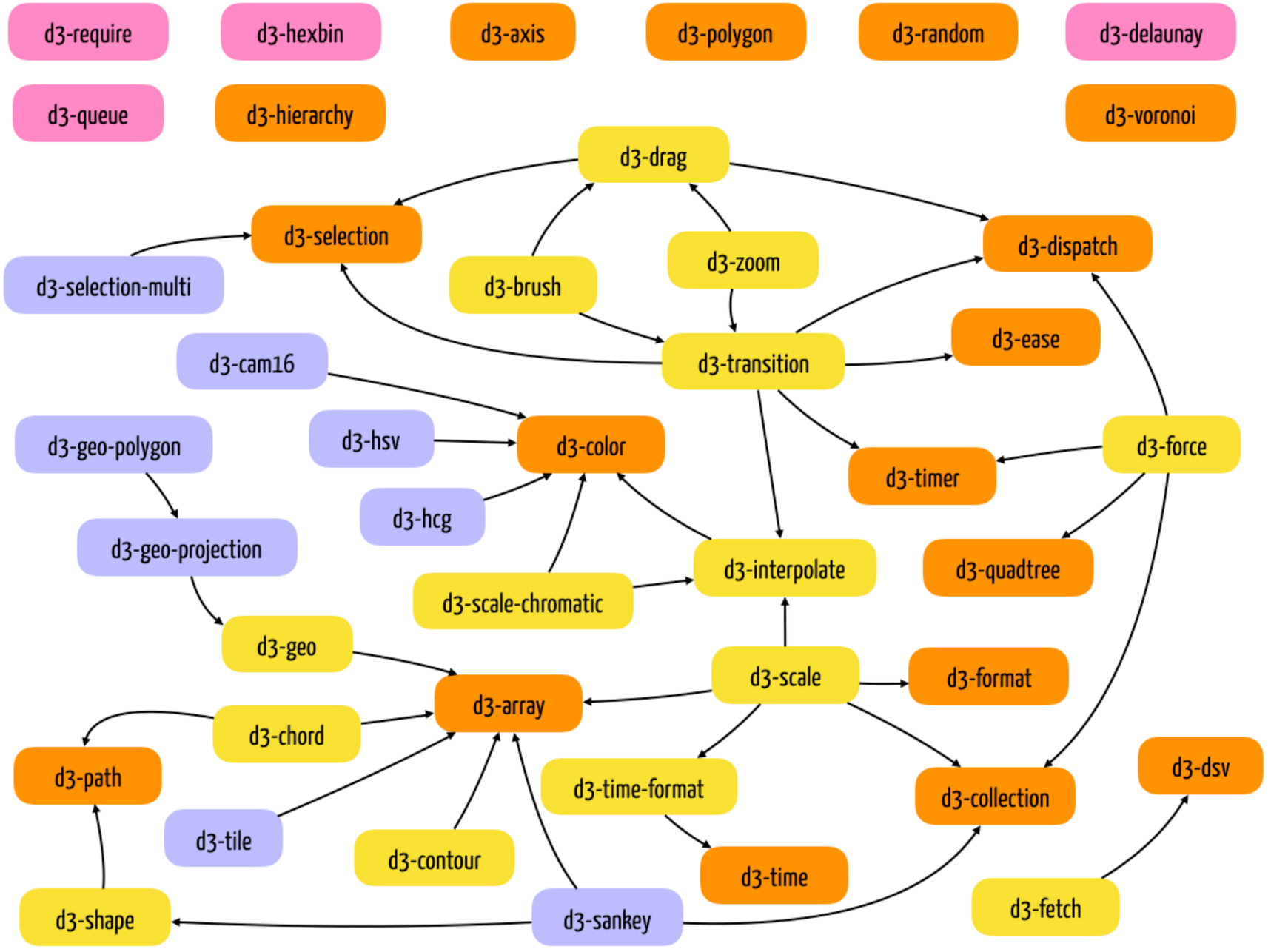
IN

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

OUT

```
{
  "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>
```

1. Parse JSON file
2. Then receive data Object

data

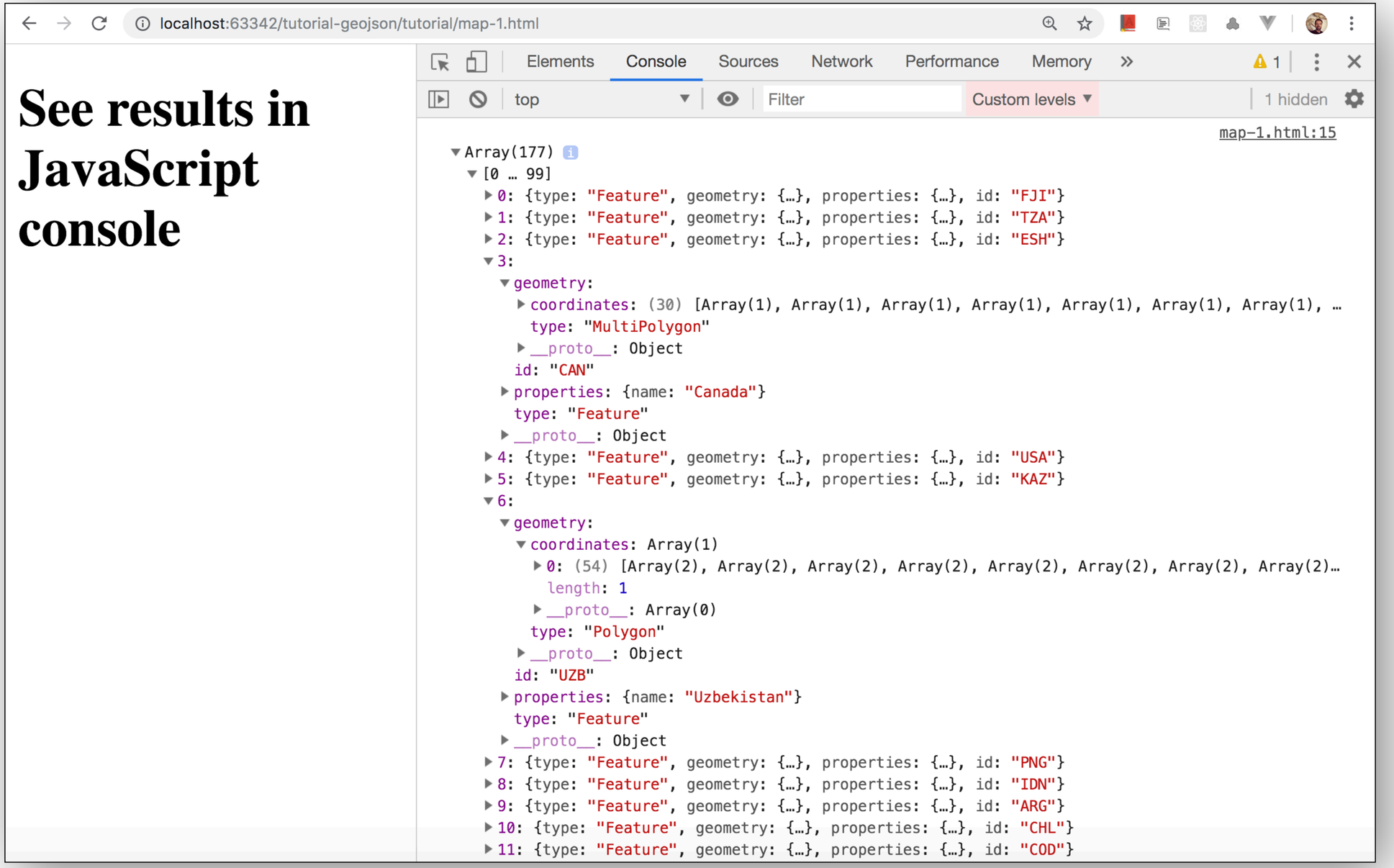
```
{
  "type": "FeatureCollection",
  "features": [ {...},{...},... ]
}
```

GeoJSON

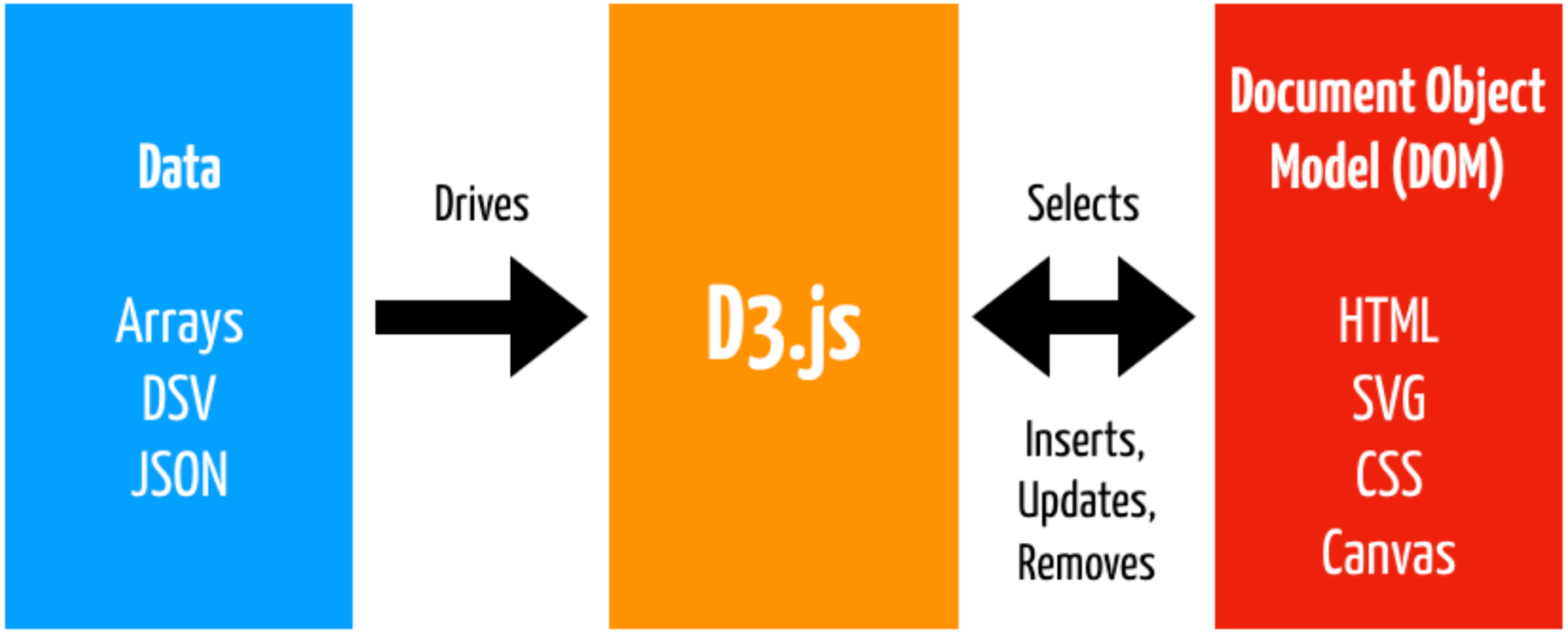
← Coleção (array) de Features

Passo 1: Carregar o GeoJSON

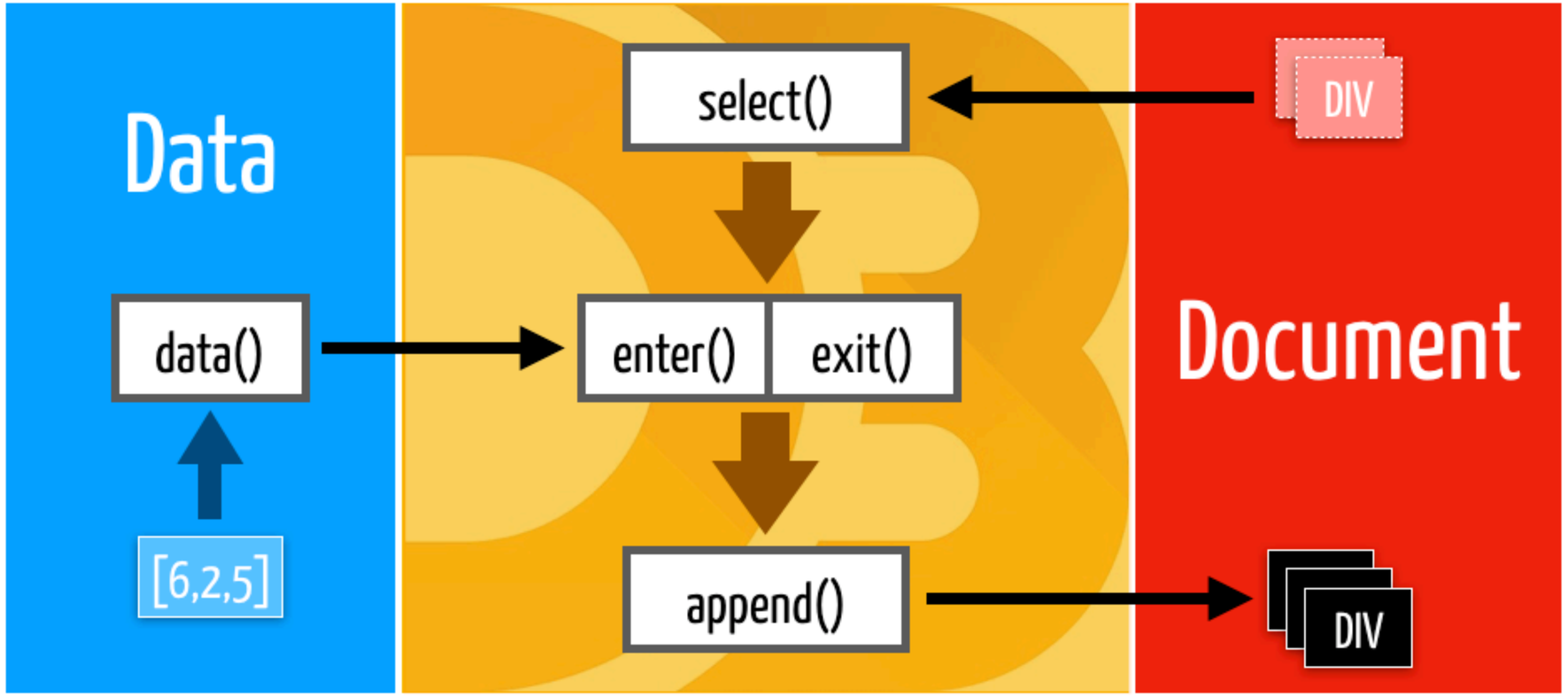
GitHub: tutorial/map-1.html



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 + ")");
}
```

Lista (array) de Features

Um Feature

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

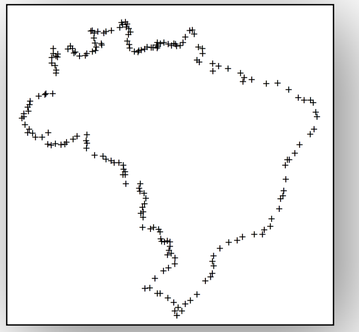
map.features[n]

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]);
```



Geocoords to pixel coords

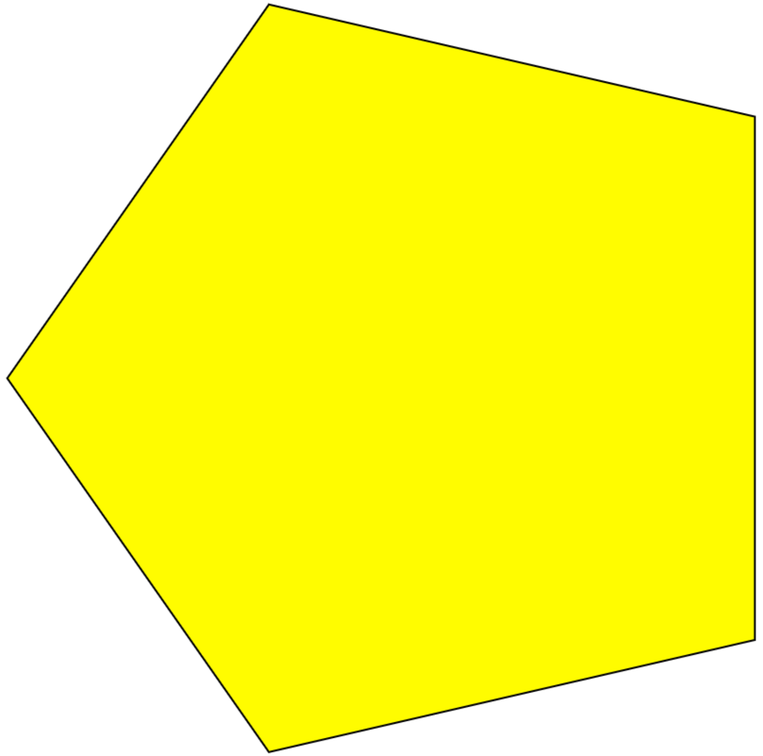
localhost:63342/tutorial-geojson/tutorial/map-3.html

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>

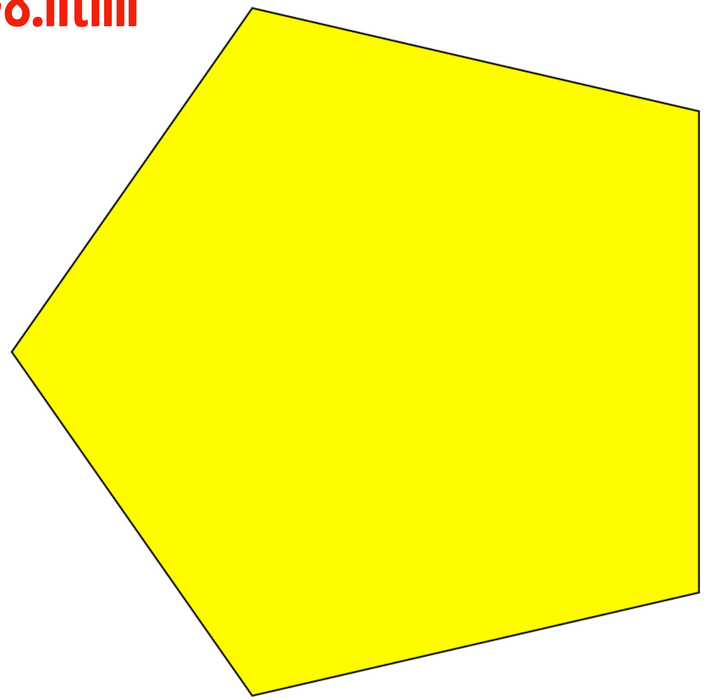
GitHub: tutorial/map-6a.html



```
<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>
```

Estrutura SVG gerada

```
<svg height="500" width="960">
   <path class="poly" d="M-10,0 L-3,10 L10,7 L10,-7 L-3,-10 Z">
</svg>
```

Passo 6: Gerando um <path> com D3

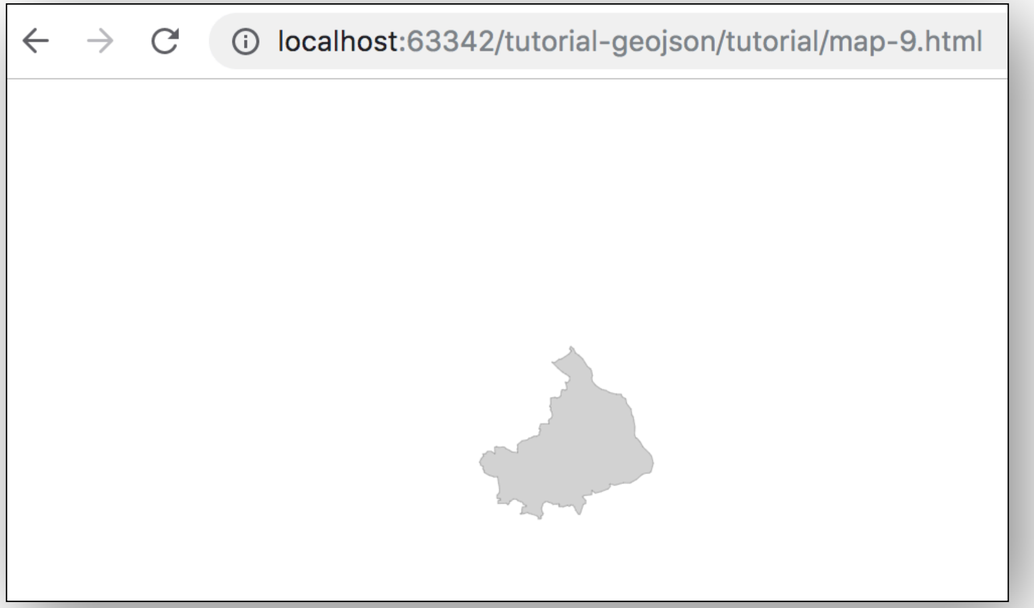
GitHub: tutorial/map-9.html

```
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);
}
```

Cria uma função geradora de string de dados para <path>)



Usa a função para gerar um string de dados <path> a partir de um Feature GeoJSON)

```
<svg height="500" width="960">
   <path d="conteúdo de pathData">
</svg>
```

Estrutura SVG gerada

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)
```

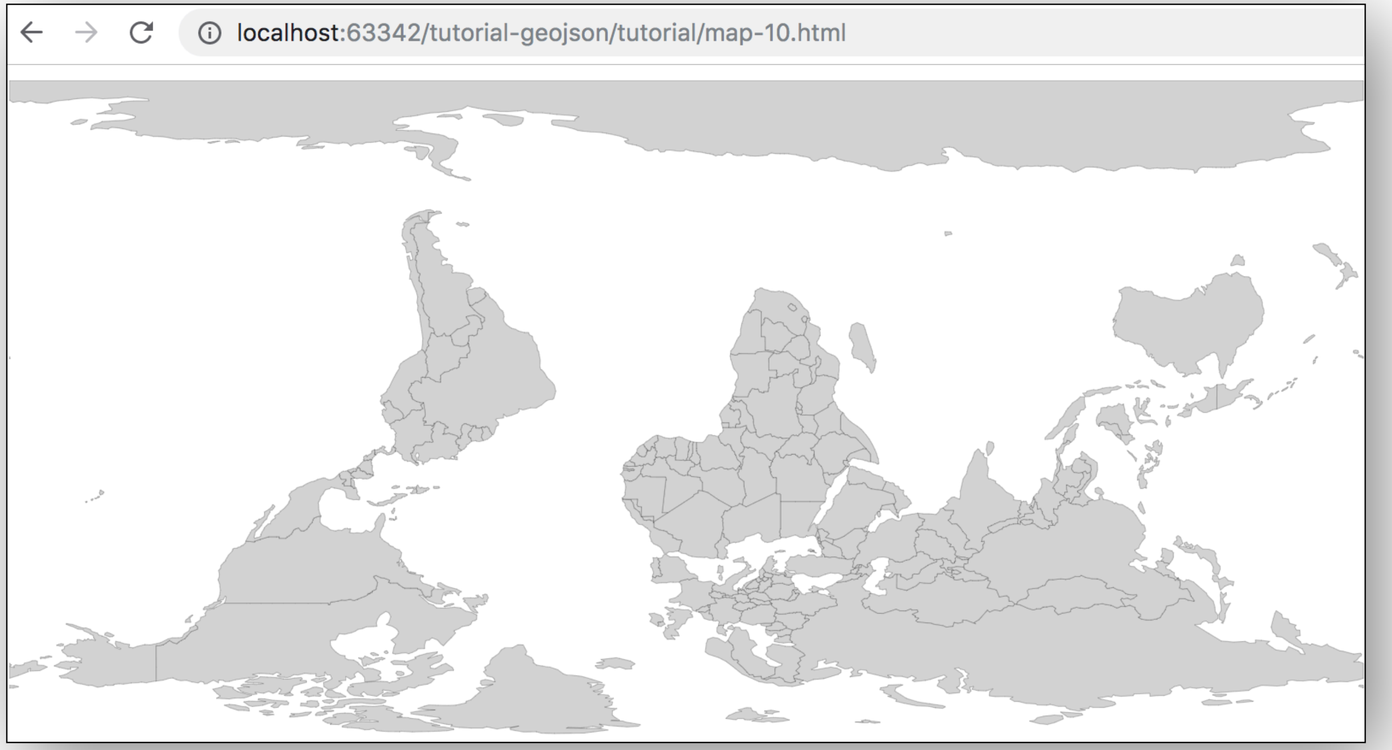
Cria função

Selecione objetos para mapear (podem não existir)

Mapeia cada Feature a um elemento ou placeholder

Cria um novo <path> para cada elemento

Chama função para cada Feature e põe resultado no atrributo "d"



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

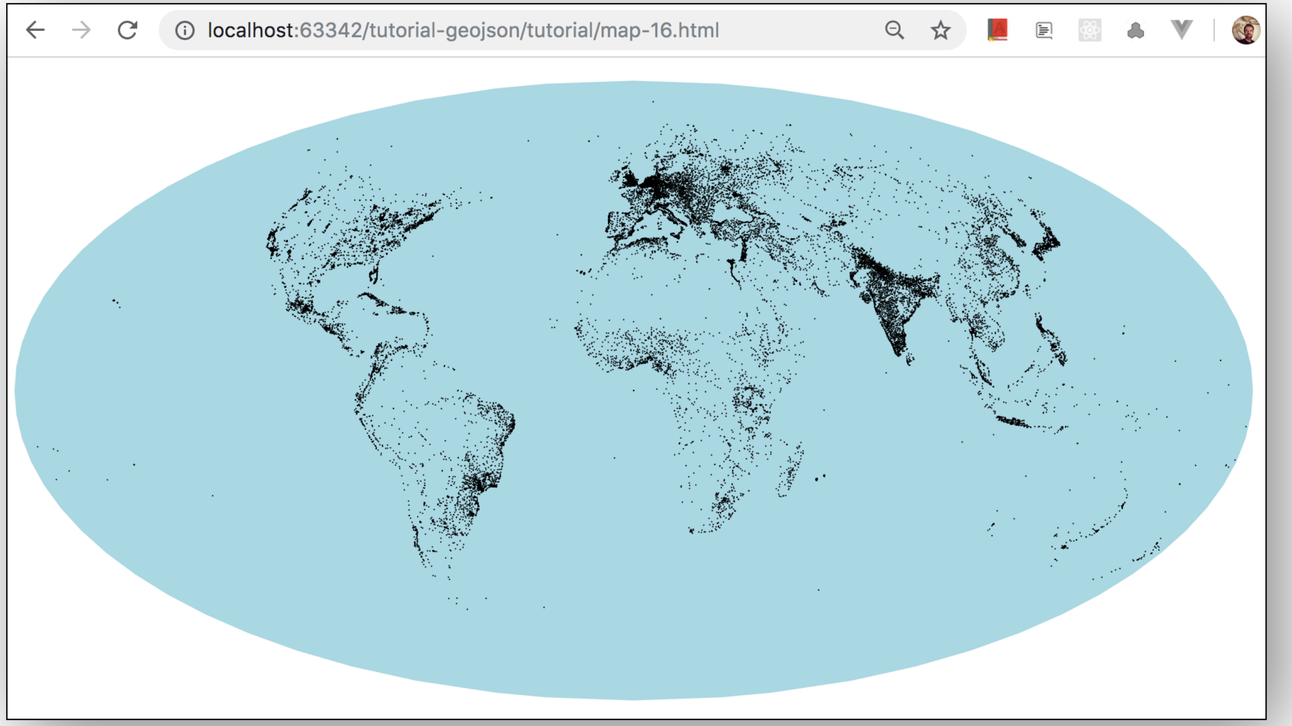
```
geonameid;asciiname;latitude;longitude;population;timezone
3040051;les Escaldes;42.50729;1.53414;15853;Europe/Andorra
3041563;Andorra la Vella;42.50779;1.52109;20430;Europe/Andorra
290594;Umm al Qaywayn;25.56473;55.55517;44411;Asia/Dubai
291074;Ras al-Khaimah;25.78953;55.9432;115949;Asia/Dubai
291696;Khawr 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

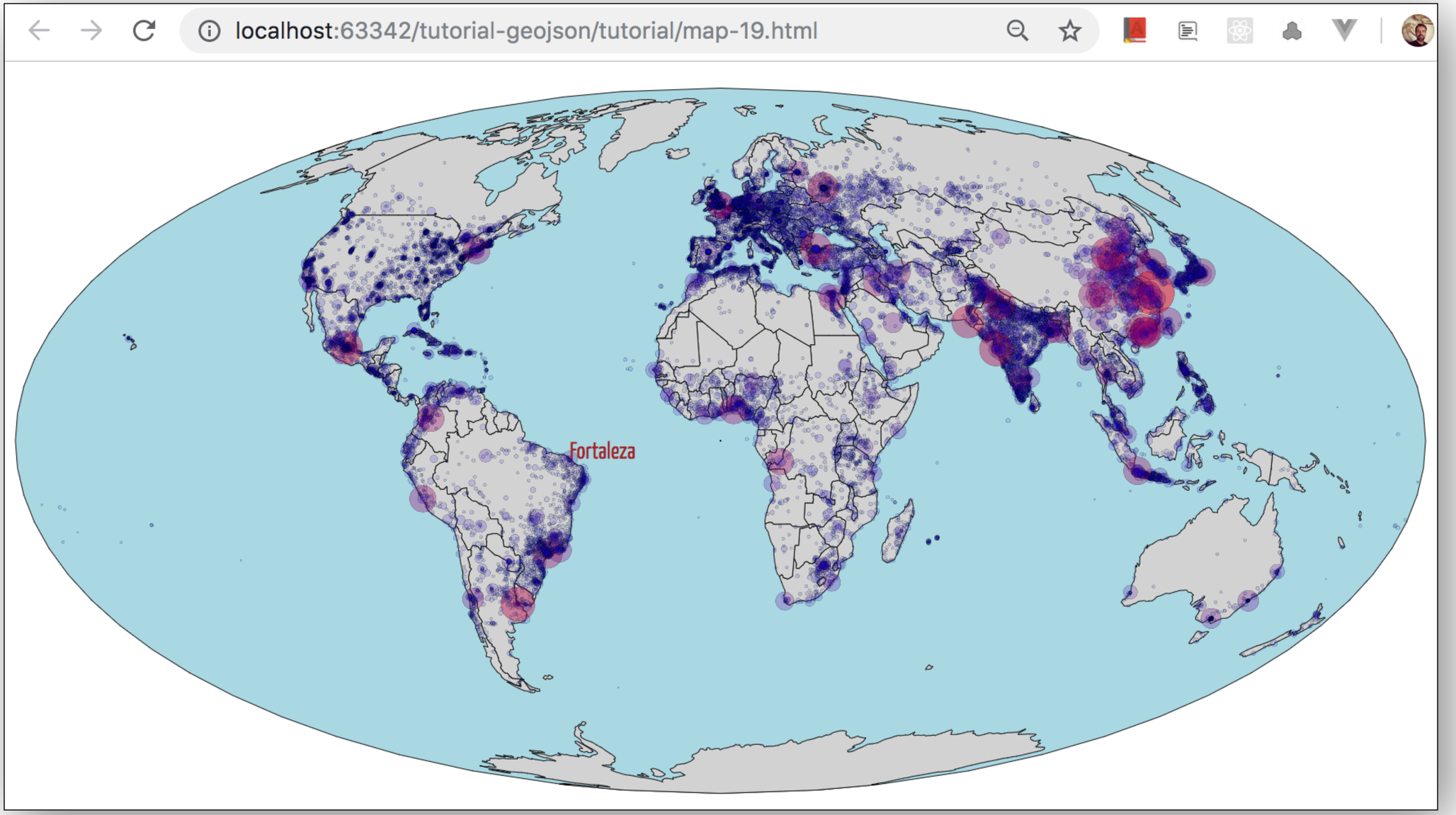



```
Promise.all([
  d3.json('../geo/world.geojson'),
  d3.dsv(';', '../csv/cities15000.csv',
    function(row) {
      return {
        name: row.asciiname,
        coords: [+row.longitude, +row.latitude]
      }
    })
]).then(function([shapes, points]) {
    map.features   = shapes.features;
    map.points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



Explore mais!

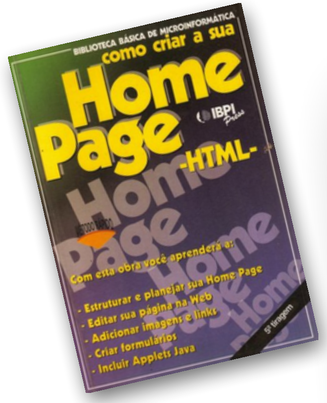
- Outros exemplos deste tutorial em
  github.com/argonavisbr/tutorial-geojson

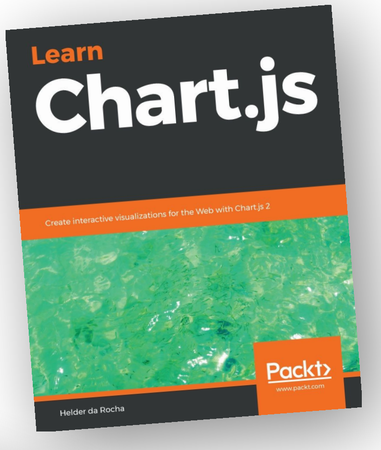
- Muitos e muitos exemplos (do livro Learn D3.js) em
  github.com/argonavisbr/DataVizWithJavaScriptBook/tree/master/D3Book

- Exemplos em **d3js.org** em **bl.ocks.org** e **observablehq.com**

- Entre em contato: 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

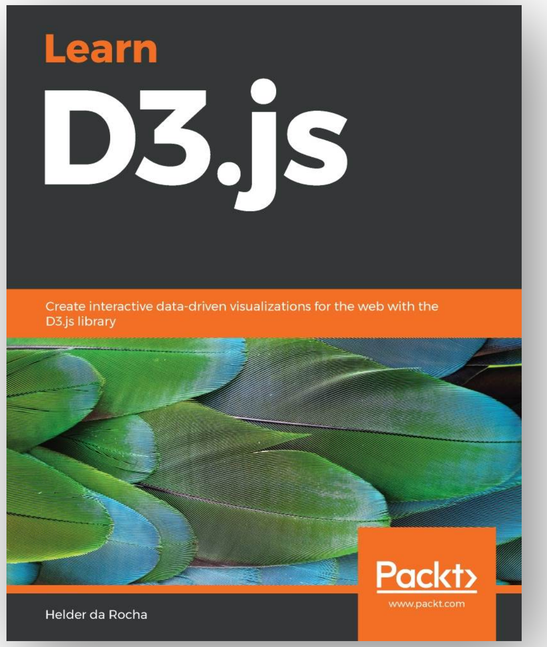
Java, XML e tecnologias Web

argonavis.com.br

helderdarocha.art.br

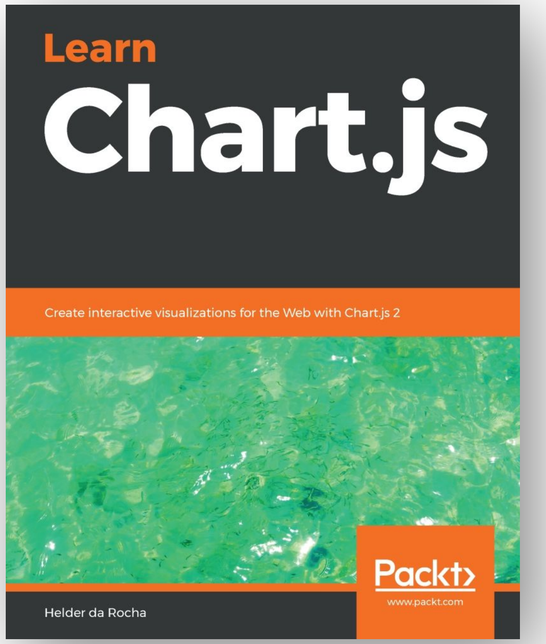
helder da rocha
helder@argonavis.com.br

www.amazon.com/dp/B07RFBV4PC



Slides, +código-fonte e +demonstrações
github.com/argonavisbr/tutorial-geojson

www.amazon.com/dp/B07PDBBHLL



Sorteio

SCAN ME

http://tiny.cc/h1rrgz