4de

13de

for (i = 0; i < Object.keys(life).length; i++){

lifeE = lifeAndTime[i]

}

.style("fill", d => color(d)) is hetzelfde als

.style("fill", function(d) {

return color(d)

})

Object.keys(life).forEach(function(datapoint, i) {

lifeExpectancy = Object.values(life)[i]

});

console.log(lifeExpectancy)

"LOCATION","Country","INDICATOR","Indicator","MEASURE","Measure","INEQUALITY","Inequality","Unit Code","Unit","PowerCode Code","PowerCode","Reference Period Code","Reference Period","Value","Flag Codes","Flags"

Health[d.properties.name]["Life expectancy"]

Value

33417 +

Anaconda prompt

python-m http.server8888

maties code bekijken voor de kaart.

<https://githib.com/MSVermet/Project>

# t = []

# for x in reader["Country"]:

# t.append(x)

#

# t = set(t)

# print(t)

<script type="text/javascript" src="heal.js"> </script>

<script src="http://d3js.org/queue.v1.min.js"></script>

<script src="http://d3js.org/topojson.v1.min.js"></script>

# print(reader["Country"])

#

# for x in reader["Household net adjusted disposable income"]:

# print(x)

# print(reader[i].index)

// var w = 1200;

// var h = 200;

//

// var barchart = d3.select("#chart").append("svg")

// .attr("width", w)

// .attr("height", h)

// .append('g')

// .attr('transform', 'translate('+margin.left+','+margin.top+')')

// .selectAll("rect")

// .enter()

// .append("rect")

// .attr("width", w / 177 - 1)

// .attr("height", function(d){

// return d \* 2;

// })

// .attr("x", function(d, i) {

// return i \* (w / 177);

// })

// .attr("y", function(d){

// return h - (d \* 2);

// })

// .attr("fill", function(d) {

// return "rgb(100, 200, " + ((Math.pow(d, 10))/10000000000000000) + ")";

// });

window.onload = function() {

var health = "health.json"

var data = "world\_countries.json"

var requests = [d3.json(health), d3.json(data)];

Promise.all(requests).then(function(response) {

var Health = response[0];

var data = response[1];

console.log(data);

console.log(Health);

var format = d3.format(",");

// Set tooltips

var tip = d3.tip()

.attr('class', 'd3-tip')

.offset([-10, 0])

.html(function(d) {

return "<strong>Country: </strong><span class='details'>" + d.properties.name + "<br></span>"+ "<strong>Income: </strong><span class='details'>" + d.income +"<br></span>" + "<strong>Life expectancy: </strong><span class='details'>" + format(d.life\_expectancy) +"</span>";

})

var margin = {top: 0, right: 0, bottom: 0, left: 50},

width = 960 - margin.left - margin.right,

height = 650 - margin.top - margin.bottom;

var padding = 20;

var color = d3.scaleThreshold()

.domain([10,15,20,25,30,35,40,45,50,55])

.range(["rgb(247,251,255)", "rgb(222,235,247)", "rgb(198,219,239)", "rgb(158,202,225)", "rgb(107,174,214)", "rgb(66,146,198)","rgb(33,113,181)","rgb(8,81,156)","rgb(8,48,107)","rgb(3,19,43)"]);

var path = d3.geoPath();

var svg = d3.select("#map").append("svg")

.attr("width", width)

.attr("height", height)

.append('g')

.attr('class', 'map');

var projection = d3.geoMercator()

.scale(130)

.translate( [width / 2, (height) / 1.5]);

var path = d3.geoPath().projection(projection);

var t = d3.transition()

.duration(750);

var click;

svg.call(tip);

ready(data, Health);

function ready(data, Health) {

var IndexbyCountry = {};

//var LifebyCountry = {};

//var WellbeingbyCountry = {};

//var FootprintbyCountry = {};

//var InequalitybyCountry = {};

//

// HappyPlanet.forEach(function(d) { IndexbyCountry[d.name] = +d.HappyPlanetIndex; });

// console.log(IndexbyCountry)

// data.features.forEach(function(d) { d.HappyPlanetIndex = IndexbyCountry[d.properties.name] });

// HappyPlanet.forEach(function(d) { LifebyCountry[d.name] = +d.Average\_life\_exp; });

// data.features.forEach(function(d) { d.Average\_life\_exp = LifebyCountry[d.properties.name]});

// HappyPlanet.forEach(function(d) { WellbeingbyCountry[d.name] = +d.Average\_wellbeing; });

// data.features.forEach(function(d) { d.Average\_wellbeing = WellbeingbyCountry[d.properties.name]});

// HappyPlanet.forEach(function(d) { FootprintbyCountry[d.name] = +d.Footprint; });

// data.features.forEach(function(d) { d.Footprint = FootprintbyCountry[d.properties.name]});

// HappyPlanet.forEach(function(d) { InequalitybyCountry[d.name] = +d.Inequality\_of\_outcomes; });

// data.features.forEach(function(d) { d.Inequality\_of\_outcomes = InequalitybyCountry[d.properties.name]});

// console.log(data)

svg.append("g")

.attr("class", "countries")

.selectAll("path")

.data(data.features)

.enter().append("path")

.attr("d", path)

.style("fill", function(d) { return color(IndexbyCountry[d.properties.name]); })

.style('stroke', 'white')

.style('stroke-width', 1.5)

.style("opacity",0.8)

// tooltips

.style("stroke","white")

.style('stroke-width', 0.3)

.on('mouseover',function(d){

tip.show(d);

d3.select(this)

.style("opacity", 1)

.style("stroke","white")

.style("stroke-width",3);

})

.on('mouseout', function(d){

tip.hide(d);

d3.select(this)

.style("opacity", 0.8)

.style("stroke","white")

.style("stroke-width",0.3);

});

svg.append("path")

.datum(topojson.mesh(data.features, function(a, b) { return a.name !== b.name; }))

// .datum(topojson.mesh(data.features, function(a, b) { return a !== b; }))

.attr("class", "names")

.attr("d", path);

};

//var LifebyCountry = {};

//Health.forEach(function(d) { LifebyCountry[d.name] = +d.Average\_life\_exp; });

//data.features.forEach(function(d) { d.Average\_life\_exp = LifebyCountry[d.properties.name]});

var w = 1200;

var h = 200;

var barchart = d3.select("#chart").append("svg")

.attr("width", w)

.attr("height", h)

.append('g')

.attr('transform', 'translate('+margin.left+','+margin.top+')')

.selectAll("rect")

//.data(Object.values(LifebyCountry))

.enter()

.append("rect")

.attr("width", w / 177 - 1)

.attr("height", function(d){

return d \* 2;

})

.attr("x", function(d, i) {

return i \* (w / 177);

})

.attr("y", function(d){

return h - (d \* 2);

})

.attr("fill", function(d) {

return "rgb(100, 200, " + ((Math.pow(d, 10))/10000000000000000) + ")";

});

}).catch(function(e){

throw(e);

});

};

//var income = 400;

//var life\_expectancy = 3;

// (Object.values(Health)).forEach(function(datapoint) {

// income = datapoint['Household net adjusted disposable income']

// life\_expectancy = datapoint['Life expectancy']

// });

//console.log(color(Health[d.properties.name]["Life expectancy"]));

//return color(IndexbyCountry[d.properties.name]);

//var LifebyCountry = {};

//var WellbeingbyCountry = {};

//var FootprintbyCountry = {};

//var InequalitybyCountry = {};

// Health.forEach(function(d) { IndexbyCountry[d.name] =+d.Life\_expectancy; })

// console.log(IndexbyCountry)

// data.features.forEach(function(d) { d.Life\_expectancy = IndexbyCountry[d.properties.name] });

//

// HappyPlanet.forEach(function(d) { IndexbyCountry[d.name] = +d.HappyPlanetIndex; });

// console.log(IndexbyCountry)

// data.features.forEach(function(d) { d.HappyPlanetIndex = IndexbyCountry[d.properties.name] });

// HappyPlanet.forEach(function(d) { LifebyCountry[d.name] = +d.Average\_life\_exp; });

// data.features.forEach(function(d) { d.Average\_life\_exp = LifebyCountry[d.properties.name]});

// HappyPlanet.forEach(function(d) { WellbeingbyCountry[d.name] = +d.Average\_wellbeing; });

// data.features.forEach(function(d) { d.Average\_wellbeing = WellbeingbyCountry[d.properties.name]});

// HappyPlanet.forEach(function(d) { FootprintbyCountry[d.name] = +d.Footprint; });

// data.features.forEach(function(d) { d.Footprint = FootprintbyCountry[d.properties.name]});

// HappyPlanet.forEach(function(d) { InequalitybyCountry[d.name] = +d.Inequality\_of\_outcomes; });

// data.features.forEach(function(d) { d.Inequality\_of\_outcomes = InequalitybyCountry[d.properties.name]});

// console.log(data)

window.onload = function() {

var health = "health.json"

var data = "world\_countries.json"

var requests = [d3.json(health), d3.json(data)];

Promise.all(requests).then(function(response) {

var Health = response[0];

var data = response[1];

console.log(data);

console.log(Health);

var format = d3.format(",");

// Set tooltips

var tip = d3.tip()

.attr('class', 'd3-tip')

.offset([-10, 0])

.html(function(d) {

// als landen niet undifined zijn

if (Health[d.properties.name] !== undefined){

console.log(d.properties.name);

console.log(Health);

income = Health[d.properties.name]["Household net adjusted disposable income"]

life\_expectancy = Health[d.properties.name]["Life expectancy"]

return "<strong>Country: </strong><span class='details'>" + d.properties.name + "<br></span>"+ "<strong>Income: </strong><span class='details'>" + income +"<br></span>" + "<strong>Life expectancy: </strong><span class='details'>" + life\_expectancy +"</span>";

}

})

var margin = {top: 0, right: 0, bottom: 0, left: 50},

width = 960 - margin.left - margin.right,

height = 650 - margin.top - margin.bottom;

var padding = 20;

var color = d3.scaleThreshold()

.domain([10,20,30,40,50,60,70,80,90,100])

.range(["rgb(247,251,255)", "rgb(222,235,247)", "rgb(198,219,239)", "rgb(158,202,225)", "rgb(107,174,214)", "rgb(66,146,198)","rgb(33,113,181)","rgb(8,81,156)","rgb(8,48,107)","rgb(3,19,43)"]);

var path = d3.geoPath();

var svg = d3.select("#map").append("svg")

.attr("width", width)

.attr("height", height)

.append('g')

.attr('class', 'map');

var projection = d3.geoMercator()

.scale(130)

.translate( [width / 2, (height) / 1.5]);

var path = d3.geoPath().projection(projection);

var t = d3.transition()

.duration(750);

var click;

svg.call(tip);

ready(data, Health);

function ready(data, Health) {

var IndexbyCountry = {};

//var Life\_expectancy = {}

//var Income = {}

svg.append("g")

.attr("class", "countries")

.selectAll("path")

.data(data.features)

.enter().append("path")

.attr("d", path)

.style("fill", function(d) {

console.log(d.properties.name)

// als een land niet in de data voorkomt, maak het wit

console.log(Health)

if (Health[d.properties.name] !== undefined){

return (color(Health[d.properties.name]["Life expectancy"]));

}

return "#FFFFFF"

})

.style('stroke', 'white')

.style('stroke-width', 1.5)

.style("opacity",0.8)

// tooltips

.style("stroke","white")

.style('stroke-width', 0.3)

.on('mouseover',function(d){

tip.show(d);

d3.select(this)

.style("opacity", 1)

.style("stroke","white")

.style("stroke-width",3);

})

.on('mouseout', function(d){

tip.hide(d);

d3.select(this)

.style("opacity", 0.8)

.style("stroke","white")

.style("stroke-width",0.3);

});

svg.append("path")

.datum(topojson.mesh(data.features, function(a, b) { return a.name !== b.name; }))

.attr("class", "names")

.attr("d", path);

};

var w = 1200;

var h = 200;

var barchart = d3.select("#chart").append("svg")

.attr("width", w)

.attr("height", h)

.append('g')

.attr('transform', 'translate('+margin.left+','+margin.top+')')

.selectAll("rect")

//.data(Object.values(LifebyCountry))

.enter()

.append("rect")

.attr("width", w / 177 - 1)

.attr("height", function(d){

return d \* 2;

})

.attr("x", function(d, i) {

return i \* (w / 177);

})

.attr("y", function(d){

return h - (d \* 2);

})

.attr("fill", function(d) {

return "rgb(100, 200, " + ((Math.pow(d, 10))/10000000000000000) + ")";

});

}).catch(function(e){

throw(e);

});

};

# Liora Rosenberg

# Student number: 11036435

import pandas

import csv

import json

INPUT\_CSV = "health.csv"

reader = pandas.read\_csv(INPUT\_CSV)

# remove data

reader = reader[(reader['Indicator'] == 'Household net adjusted disposable income') | (reader['Indicator'] == 'Life expectancy')]

reader = reader.drop(["LOCATION","INDICATOR","MEASURE","Measure","INEQUALITY","Inequality","Unit Code", "Unit", "PowerCode Code", "PowerCode", "Reference Period Code", "Reference Period","Flag Codes","Flags"],

axis=1)

#countries = reader

# het zijn niet de juiste waarde. Vraag om hulp!

reader = reader.pivot\_table(values='Value', index='Country', columns='Indicator', aggfunc='mean')

countries = []

household\_income = []

life\_expectancy = []

for i in reader:

if (i == "Life expectancy"):

life\_expectancy = reader[i].values

else:

household\_income = reader[i].values

#print(reader[i].values)

index = reader[i].index

for t in reader[i].index:

#print(reader[i])

#print(t)

countries.append(t)

print(countries)

# print(LE)

# print(household\_income)

dict = {}

for i in range(len(household\_income)):

dict[countries[i]] = {"life expectancy": life\_expectancy[i], "household\_income": household\_income[i]}

# print(countries[i], LE[i], HHNADI[i])

#print(eindproduct)

#reader = reader.set\_index('Indicator')

reader = reader.to\_json('health.json', orient="index")

<!DOCTYPE html>

<!DOCTYPE html>

<meta charset="utf-8">

<style>

.names {

fill: none;

stroke: #fff;

stroke-linejoin: round;

}

/\* Tooltip CSS \*/

.d3-tip {

line-height: 1.5;

font-weight: 400;

font-family:"avenir next", Arial, sans-serif;

padding: 6px;

background: rgba(0, 0, 0, 0.6);

color: #FFA500;

border-radius: 1px;

pointer-events: none;

}

/\* Creates a small triangle extender for the tooltip \*/

.d3-tip:after {

box-sizing: border-box;

display: inline;

font-size: 8px;

width: 100%;

line-height: 1.5;

color: rgba(0, 0, 0, 0.6);

position: absolute;

pointer-events: none;

}

/\* Northward tooltips \*/

.d3-tip.n:after {

content: "\25BC";

margin: -1px 0 0 0;

top: 100%;

left: 0;

text-align: center;

}

/\* Eastward tooltips \*/

.d3-tip.e:after {

content: "\25C0";

margin: -4px 0 0 0;

top: 50%;

left: -8px;

}

/\* Southward tooltips \*/

.d3-tip.s:after {

content: "\25B2";

margin: 0 0 1px 0;

top: -8px;

left: 0;

text-align: center;

}

/\* Westward tooltips \*/

.d3-tip.w:after {

content: "\25B6";

margin: -4px 0 0 -1px;

top: 50%;

left: 100%;

}

/\* text{

pointer-events:none;

}\*/

.details{

color:white;

}

</style>

<body>

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

<script src="http://d3js.org/queue.v1.min.js"></script>

<script src="http://d3js.org/topojson.v1.min.js"></script>

<script src="d3-tip.js"></script>

<script type="text/javascript" src="heal.js"> </script>

<div id="map"></div>

<div id="chart"></div>

</body>

</html>

// naam: Liora Rosenberg

// Student number: 11036435

// this file

// dimensions

var margin = {top: 0, right: 0, bottom: 0, left: 50},

width = screen.width - margin.left - margin.right,

height = 650 - margin.top - margin.bottom;

var padding = 20;

window.onload = function() {

var life = "life.json"

var health = "health.json"

var data = "world\_countries.json"

var requests = [d3.json(health), d3.json(data), d3.json(life)];

Promise.all(requests).then(function(response) {

var Health = response[0];

var data = response[1];

var life = response[2];

console.log(data);

console.log(Health);

console.log(life)

var format = d3.format(",");

// Set tooltips

var tip = d3.tip()

.attr('class', 'd3-tip')

.offset([-10, 0])

.html(function(d) {

// als landen niet undifined zijn

if (Health[d.properties.name] !== undefined){

console.log(d.properties.name);

console.log(Health);

income = Health[d.properties.name]["Household net adjusted disposable income"]

life\_expectancy = Health[d.properties.name]["Life expectancy"]

return "<strong>Country: </strong><span class='details'>" + d.properties.name + "<br></span>"+ "<strong>Income: </strong><span class='details'>" + income +"<br></span>" + "<strong>Life expectancy: </strong><span class='details'>" + life\_expectancy +"</span>";

}

})

var color = d3.scaleThreshold()

.domain([10,20,30,40,50,60,70,80,90,100])

.range(["rgb(247,251,255)", "rgb(222,235,247)", "rgb(198,219,239)", "rgb(158,202,225)", "rgb(107,174,214)", "rgb(66,146,198)","rgb(33,113,181)","rgb(8,81,156)","rgb(8,48,107)","rgb(3,19,43)"]);

var path = d3.geoPath();

var svg = d3.select("#map").append("svg")

.attr("width", width)

.attr("height", height)

.append('g')

.attr('class', 'map');

var projection = d3.geoMercator()

.scale(130)

.translate( [width / 2, (height) / 1.5]);

var path = d3.geoPath().projection(projection);

var t = d3.transition()

.duration(750);

var click;

svg.call(tip);

ready(data, Health);

function ready(data, Health) {

var IndexbyCountry = {};

svg.append("g")

.attr("class", "countries")

.selectAll("path")

.data(data.features)

.enter().append("path")

.attr("d", path)

.style("fill", function(d) {

console.log(d.properties.name)

// als een land niet in de data voorkomt, maak het wit

console.log(Health)

if (Health[d.properties.name] !== undefined){

return (color(Health[d.properties.name]["Life expectancy"]));

}

return "#FFFFFF"

})

.style('stroke', 'white')

.style('stroke-width', 1.5)

.style("opacity",0.8)

// tooltips

.style("stroke","white")

.style('stroke-width', 0.3)

.on('mouseover',function(d){

tip.show(d);

d3.select(this)

.style("opacity", 1)

.style("stroke","white")

.style("stroke-width",3);

})

.on('mouseout', function(d){

tip.hide(d);

d3.select(this)

.style("opacity", 0.8)

.style("stroke","white")

.style("stroke-width",0.3);

});

svg.append("path")

.datum(topojson.mesh(data.features, function(a, b) { return a.name !== b.name; }))

.attr("class", "names")

.attr("d", path);

};

// make legend

legend = svg.selectAll("#map")

.data([10,20,30,40,50,60,70,80,90,100])

.enter()

.append("g")

.attr("class", ".legend")

.attr("transform", function(d, i) { return "translate(0," + i \* 20 + ")"; });

legend.append("rect")

.attr("x", width - 35)

.attr("y", 0)

.attr("width", 32)

.attr("height", 20)

.style("fill", d => color(d))

// add text to legend

legend.append("text")

.attr("x", width - 65)

.attr("y", 20)

.text(function(d){

return d;

})

// alles wat hieronder staat werkt nog niet!!!!!

//(life["Afghanistan"]) = (Object.values(life)[0])

console.log(life)

console.log(Object.keys(life).length)

console.log(Object.values(life))

console.log(Object.values(Object.values(life)))

//landen = object.keys(life)

console.log(Object.keys(life))

for (i = 0; i < Object.keys(life).length; i++) {

lifeAndTime = Object.values(life)[i]

}

console.log(lifeAndTime)

// scaling

var max = Math.max.apply(null, Object.values(life))

var minCC = Math.min.apply(null, allConsConf)

var maxCC = Math.max.apply(null, allConsConf)

var xScale = d3.scaleLinear()

.domain([0, max])

.range([margin, width - margin - 120])

var yScale = d3.scaleLinear()

.domain([minCC, maxCC])

.range([height - margin, margin])

// make line chart

g.append("path")

.datum(data)

.attr("fill", "none")

.attr("stroke", "steelblue")

.attr("stroke-linejoin", "round")

.attr("stroke-linecap", "round")

.attr("stroke-width", 1.5)

.attr("d", line);

}).catch(function(e){

throw(e);

});

};