To do:

* Interactief maken linechart
* Nons in data verwijderen
* Functies zetten
* Zorgen dat de linechart geupdate wordt

//if (d3.select("#beste") === ){

// d3.select("#beste").exit().remove();

//}

// verwijder oude grafiek (inclusief assen)

// var min =

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

// var minLE = Math.min.apply(null, lifeexpectancy)

// var maxLE = Math.max.apply(null, lifeexpectancy)

console.log(country);

console.log(typeof(country))

console.log(life[country]);

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

console.log(Object.values(life[country])[i]);

console.log(Object.values(life[country][i]));

console.log(testdata);

// console.log(Object.values(life)[country][i]);

console.log(life[country])

console.log(data);

console.log(Health);

console.log(life)

// console.log(d.properties.name);

// console.log(Health);

If (!) Varaabele naam

If not Nan

console.log(Object.keys(life))

console.log(life[country])

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

//this.attr('class', 'focus')

var line = d3.line()

.x(function(d, i) { return xScale(i); }) // set the x values for the line generator

.y(function(d) { return yScale(d.y); }) // set the y values for the line generator

.curve(d3.curveMonotoneX)

// var svg = d3.select("body").append("svg")

// .attr("width", width + margin.left + margin.right)

// .attr("height", height + margin.top + margin.bottom)

// .append("g")

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

// make line chart

svg.append("path")

.datum(Object.values(life)[0])

.attr("fill", "none")

.attr("stroke", "steelblue")

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

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

.attr("stroke-width", 1.5)

.attr("d", line);

// The number of datapoints

var n = 17;

// 7. d3's line generator

var line = d3.line()

.x(function(d, i) { return xScale(i); }) // set the x values for the line generator

.y(function(d) { return yScale(d.y); }) // set the y values for the line generator

.curve(d3.curveMonotoneX) // apply smoothing to the line

// 8. An array of objects of length N. Each object has key -> value pair, the key being "y" and the value is a random number

var dataset = d3.range(n).map(function(d) { return {"y": d3.randomUniform(1)() } })

// 1. Add the SVG to the page and employ #2

var svg = d3.select("body").append("svg")

.attr("width", width + margin.left + margin.right)

.attr("height", height + margin.top + margin.bottom)

.append("g")

.attr("transform", "translate(" + margin.left + "," + margin.top + ")");

// 3. Call the x axis in a group tag

svg.append("g")

.attr("class", "x axis")

.attr("transform", "translate(0," + height + ")")

.call(d3.axisBottom(xScale)); // Create an axis component with d3.axisBottom

// 4. Call the y axis in a group tag

svg.append("g")

.attr("class", "y axis")

.call(d3.axisLeft(yScale)); // Create an axis component with d3.axisLeft

// 9. Append the path, bind the data, and call the line generator

svg.append("path")

.datum(dataset) // 10. Binds data to the line

.attr("class", "line") // Assign a class for styling

.attr("d", line); // 11. Calls the line generator

// 12. Appends a circle for each datapoint

svg.selectAll(".dot")

.data(dataset)

.enter().append("circle") // Uses the enter().append() method

.attr("class", "dot") // Assign a class for styling

.attr("cx", function(d, i) { return xScale(i) })

.attr("cy", function(d) { return yScale(d.y) })

.attr("r", 5)

.on("mouseover", function(a, b, c) {

console.log(a)

this.attr('class', 'focus')

})

.on("mouseout", function() { })// make line chart

svg.append("path")

.datum(Object.values(life)[0])

.attr("fill", "none")

.attr("stroke", "steelblue")

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

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

.attr("stroke-width", 1.5)

.attr("d", line);

// The number of datapoints

var n = 17;

// 7. d3's line generator

var line = d3.line()

.x(function(d, i) { return xScale(i); }) // set the x values for the line generator

.y(function(d) { return yScale(d.y); }) // set the y values for the line generator

.curve(d3.curveMonotoneX) // apply smoothing to the line

// 8. An array of objects of length N. Each object has key -> value pair, the key being "y" and the value is a random number

var dataset = d3.range(n).map(function(d) { return {"y": d3.randomUniform(1)() } })

// 1. Add the SVG to the page and employ #2

var svg = d3.select("body").append("svg")

.attr("width", width + margin.left + margin.right)

.attr("height", height + margin.top + margin.bottom)

.append("g")

.attr("transform", "translate(" + margin.left + "," + margin.top + ")");

// 3. Call the x axis in a group tag

svg.append("g")

.attr("class", "x axis")

.attr("transform", "translate(0," + height + ")")

.call(d3.axisBottom(xScale)); // Create an axis component with d3.axisBottom

// 4. Call the y axis in a group tag

svg.append("g")

.attr("class", "y axis")

.call(d3.axisLeft(yScale)); // Create an axis component with d3.axisLeft

// 9. Append the path, bind the data, and call the line generator

svg.append("path")

.datum(dataset) // 10. Binds data to the line

.attr("class", "line") // Assign a class for styling

.attr("d", line); // 11. Calls the line generator

// 12. Appends a circle for each datapoint

svg.selectAll(".dot")

.data(dataset)

.enter().append("circle") // Uses the enter().append() method

.attr("class", "dot") // Assign a class for styling

.attr("cx", function(d, i) { return xScale(i) })

.attr("cy", function(d) { return yScale(d.y) })

.attr("r", 5)

.on("mouseover", function(a, b, c) {

console.log(a)

this.attr('class', 'focus')

})

.on("mouseout", function() { })

// 2. Use the margin convention practice

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

, width = window.innerWidth - margin.left - margin.right // Use the window's width

, height = window.innerHeight - margin.top - margin.bottom; // Use the window's height

// The number of datapoints

var n = 21;

// 5. X scale will use the index of our data

var xScale = d3.scaleLinear()

.domain([0, n-1]) // input

.range([0, width]); // output

// 6. Y scale will use the randomly generate number

var yScale = d3.scaleLinear()

.domain([0, 1]) // input

.range([height, 0]); // output

// 7. d3's line generator

var line = d3.line()

.x(function(d, i) { return xScale(i); }) // set the x values for the line generator

.y(function(d) { return yScale(d.y); }) // set the y values for the line generator

.curve(d3.curveMonotoneX) // apply smoothing to the line

// 8. An array of objects of length N. Each object has key -> value pair, the key being "y" and the value is a random number

var dataset = d3.range(n).map(function(d) { return {"y": d3.randomUniform(1)() } })

// 1. Add the SVG to the page and employ #2

var svg = d3.select("body").append("svg")

.attr("width", width + margin.left + margin.right)

.attr("height", height + margin.top + margin.bottom)

.append("g")

.attr("transform", "translate(" + margin.left + "," + margin.top + ")");

// 3. Call the x axis in a group tag

svg.append("g")

.attr("class", "x axis")

.attr("transform", "translate(0," + height + ")")

.call(d3.axisBottom(xScale)); // Create an axis component with d3.axisBottom

// 4. Call the y axis in a group tag

svg.append("g")

.attr("class", "y axis")

.call(d3.axisLeft(yScale)); // Create an axis component with d3.axisLeft

// 9. Append the path, bind the data, and call the line generator

svg.append("path")

.datum(dataset) // 10. Binds data to the line

.attr("class", "line") // Assign a class for styling

.attr("d", line); // 11. Calls the line generator

// 12. Appends a circle for each datapoint

svg.selectAll(".dot")

.data(dataset)

.enter().append("circle") // Uses the enter().append() method

.attr("class", "dot") // Assign a class for styling

.attr("cx", function(d, i) { return xScale(i) })

.attr("cy", function(d) { return yScale(d.y) })

.attr("r", 5)

.on("mouseover", function(a, b, c) {

console.log(a)

this.attr('class', 'focus')

})

.on("mouseout", function() { })

var xScale = d3.scaleLinear()

.domain([2000, 2017])

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

var yScale = d3.scaleLinear()

.domain([0, 100])

.range([height - margin.bottom, margin.top])

.on mousover . on clink .mousout

// var consConf = "http://stats.oecd.org/SDMX-JSON/data/HH\_DASH/FRA+DEU+KOR+NLD+PRT+GBR.COCONF.A/all?startTime=2007&endTime=2015"

//var consumerConfidence = "https://data.mprog.nl/course/10%20Homework/100%20D3%20Scatterplot/datasets/consconf.json"

// var msti = <https://data.mprog.nl/course/10%20Homework/100%20D3%20Scatterplot/datasets/msti.json>

//time = consumerConf[0]["time"]

//console.log(time)

Counter bijhouden

If

Het loopt van 2017 – 2015 maar soms ontbreekt

Dictonary werken +

{2017: ConsC + WomenScience + country}

// is het een goede waarde en zit het in mijn array: nee: voeg toe + ja: sla over

//hoi = allYears.sort();

//console.log(hoi)

// twee vragen voor morgen: (1) hoe gaan we om dat de enige lijst langer is dan de ander? Hoe zorg ik ervoor dat de puntjes verdeeld zijn

// hier wil ik shit toevoegen

If allWomen[i][1] = ConsCOnf[j][1]

allWomen[i][2] = consConf[j][2]

verander consc

verander lijst van landen

var xAxis = d3.axisBottom(xScale);

var yAxis = d3.axisLeft(yScale);

// // create x-as

// xAxis.scale(xScale);

//

// var xAxis = d3.svg.axis()

// .scale(xScale)

// .orient("bottom");

//

// svg.append("g")

// .call(d3.svg.axis()

// .scale(xScale)

// .orient("bottom"));

//

// svg.append("g")

// .attr("class", "axis") //Assign "axis" class

// .call(xAxis);

//

// .axis path,

// .axis line {

// fill: none;

// stroke: black;

// shape-rendering: crispEdges;

// }

//

// .axis text {

// font-family: sans-serif;

// font-size: 11px;

// }

// // hier check ik of de landen overeenkomen + voeg ik het aan een grote lijst toe

// for (i = 0; i < allWomen.length; i++) {

// for (j = 0; j < allConsConf.length; j++){

// //if land is gelijk aan land en jaar is gelijk aan jaar

// if (allWomen[i][1] == allConsConf[i][1] && allWomen[i][2] == allConsConf[i][2]){

// grandList.push([allConsConf[i][0], allWomen[i][0], allWomen[i][1], allWomen[i][2]])

//

// }

//

// }

//

// };

//

// console.log(grandList);

Ik heb twee kleinere vraagjes: over het domain en over een remove functie.

**Misschien ga ik zo ergens anders zitten**

**//create x and yaxis**

**// var xAxis = d3.svg.axis()**

**// .scale(xScale)**

**// .orient("bottom");**

**//**

**// svg.append("g")**

**// .call(d3.svg.axis()**

**// .scale(xScale)**

**// .orient("bottom"));**

**//**

**// svg.append("g")**

**// .attr("class", "axis") //Assign "axis" class**

**// .call(xAxis);**

**//**

**// .axis path,**

**// .axis line {**

**// fill: none;**

**// stroke: black;**

**// shape-rendering: crispEdges;**

**//}**

**// .axis text {**

**// font-family: sans-serif;**

**// font-size: 11px;**

**// }**

**//Object.keys(dict).forEach(function(datapoint) {**

**// return Object.values(dict)**

**//}))**

**console.log(Object.values(dict))**

**//})**

**//function(d){**

**//})**

**// legend = svg.append("g")**

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

**// .attr("tranform", translate(50,30)")**

**// .attr("data-legend", function(d) { return d.name})**

**//**

**// .box {**

**// display: inline-block;**

**// height: 20px;**

**// width: 20px;**

**// border: 2px solid;**

**// }**

**// legenda toevoegen**

**// legend = svg.append("g")**

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

**// .attr("transform","translate(50,30)")**

**// .style("font-size","12px")**

**// .call(d3.legend)**

**//console.log(color)**

**//console.log(countries)**

**//console.log(dict)**

**//allWomen.push([women, year, country])**

**//if (year == "2015"){**

**// y++;**

**//}**

**//var country = x[y];**

**//var x = ["France", "Germany", "Korea", "Netherlands", "Portugal", "United Kingdom"];**

**//var y = 0;**

**// naam: Liora Rosenberg**

**// Student number: 11036435**

**// this file draws a scatterplot in a website that is based of the amount of women in science and consumer confidence.**

**// dimensions chart**

**var width = screen.width - 30;**

**var height = 250;**

**var margin = 50;**

**var barheight = 0;**

**var barWidth;**

**window.onload = function() {**

**// distract jasons**

**var womenInScience = "http://stats.oecd.org/SDMX-JSON/data/MSTI\_PUB/TH\_WRXRS.FRA+DEU+KOR+NLD+PRT+GBR/all?startTime=2007&endTime=2015"**

**var consConf = "http://stats.oecd.org/SDMX-JSON/data/HH\_DASH/FRA+DEU+KOR+NLD+PRT+GBR.COCONF.A/all?startTime=2007&endTime=2015"**

**var requests = [d3.json(consConf), d3.json(womenInScience)];**

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

**main(response)**

**}).catch(function(e){**

**throw(e);**

**});**

**};**

**function main(response){**

**var consumerConf = transformResponse(response[0]);**

**var womInScien = transformResponse(response[1]);**

**console.log(consumerConf)**

**console.log(womInScien)**

**//allYears = []**

**countries = []**

**allConsConf = []**

**allWomen = []**

**grandList = []**

**// alle jaren die we tegenkomen:**

**consumerConf.forEach(function(datapoint) {**

**var year = datapoint["time"]**

**//allYears.push(year)**

**//console.log(allYears)**

**var country = datapoint["Country"]**

**countries.push(country)**

**console.log(countries)**

**var consConfP = datapoint["datapoint"]**

**allConsConf.push([consConfP])**

**grandList.push([consConfP, year, country]);**

**});**

**womInScien.forEach(function(datapoint) {**

**var women = datapoint["datapoint"]**

**var year = datapoint["time"]**

**var country = datapoint['Country']**

**allWomen.push([women])**

**grandList.forEach(function(point) {**

**if (point[1] === year && point[2] === country) {**

**point.unshift(women);**

**return;**

**}**

**});**

**});**

**// remove the row in list that is not complete**

**grandList.forEach(function(point, i) {**

**if (point.length !== 4) {**

**grandList.splice(i, 1)**

**}**

**});**

**console.log(grandList);**

**// scaling**

**//var min = Math.min.apply(null, allWomen)**

**var max = Math.max.apply(null, allWomen)**

**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])**

**// kleur toevoegen**

**dict ={}**

**color = ["#ffffb2", "#fed976", "#feb24c", "#fd8d3c", "#f03b20", "#bd0026"]**

**count = 0**

**countries.forEach(function(datapoint, i) {**

**if (!(countries[i] in dict)){**

**dict[countries[i]] = color[count]**

**count++**

**}**

**})**

**// create SVG element + body**

**var svg = d3.select("body")**

**.append("svg")**

**.attr("width", width)**

**.attr("height", height)**

**.style('background', 'wit');**

**svg.selectAll("circle")**

**.data(grandList)**

**.enter()**

**.append("circle")**

**// .attr("data-legend",function(d) { return d[3]})**

**.attr("cx", function(d) {**

**return xScale(d[0]);**

**})**

**.attr("cy", function(d) {**

**return yScale(d[1]);**

**})**

**.attr("r", 5)**

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

**return dict[d[3]];**

**});**

**// make y-as**

**var yAxis = d3.axisLeft(yScale);**

**svg.append("g")**

**.attr("class", "axis")**

**.attr("transform", "translate(" + [margin, 0] + ")")**

**.call(yAxis)**

**// make x-as**

**var xAxis = d3.axisBottom(xScale);**

**svg.append("g")**

**.attr("class", "axis")**

**.attr("transform", "translate(" + [0, height - margin] + ")")**

**.call(xAxis)**

**// add label year**

**svg.append("text")**

**.attr("transform", "translate(" + [width / 2, height - margin / 3] + ")")**

**.text("percentage women researchers")**

**// add label percentage**

**svg.append("text")**

**.attr("text-anchor", "middle")**

**.attr("transform", "translate(" + [margin / 3, height / 3 \* 2 - margin] + ") rotate(-90)")**

**.text("consumer confidence")**

**// make legend**

**console.log(Object.values(dict))**

**legend = svg.selectAll(".legend")**

**.data(Object.keys(dict))**

**.enter()**

**.append("g")**

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

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

**legend.append("rect")**

**.attr("x", width - 145)**

**.attr("y", 0)**

**.attr("width", 32)**

**.attr("height", 20)**

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

**return Object.values(dict)[i]**

**})**

**// add text to legend**

**legend.append("text")**

**.attr("x", width - 110)**

**.attr("y", 20)**

**.text(function(d){**

**return d;**

**})**

**}**

**function transformResponse(data){**

**// access data property of the response**

**let dataHere = data.dataSets[0].series;**

**// access variables in the response and save length for later**

**let series = data.structure.dimensions.series;**

**let seriesLength = series.length;**

**// set up array of variables and array of lengths**

**let varArray = [];**

**let lenArray = [];**

**series.forEach(function(serie){**

**varArray.push(serie);**

**lenArray.push(serie.values.length);**

**});**

**// get the time periods in the dataset**

**let observation = data.structure.dimensions.observation[0];**

**// add time periods to the variables, but since it's not included in the**

**// 0:0:0 format it's not included in the array of lengths**

**varArray.push(observation);**

**// create array with all possible combinations of the 0:0:0 format**

**let strings = Object.keys(dataHere);**

**// set up output array, an array of objects, each containing a single datapoint**

**// and the descriptors for that datapoint**

**let dataArray = [];**

**// for each string that we created**

**strings.forEach(function(string){**

**// for each observation and its index**

**observation.values.forEach(function(obs, index){**

**let data = dataHere[string].observations[index];**

**if (data != undefined){**

**// set up temporary object**

**let tempObj = {};**

**let tempString = string.split(":");**

**tempString.forEach(function(s, indexi){**

**tempObj[varArray[indexi].name] = varArray[indexi].values[s].name;**

**});**

**// every datapoint has a time and ofcourse a datapoint**

**tempObj["time"] = obs.name;**

**tempObj["datapoint"] = data[0];**

**dataArray.push(tempObj);**

**}**

**});**

**});**

**// return the finished product!**

**return dataArray;**

**}**

**console.log(consumerConf)**

**console.log(womInScien)**

**//allYears.push(year)**

**//console.log(allYears)**

**//allYears = []**

**//var min = Math.min.apply(null, allWomen)**

**// .attr("data-legend",function(d) { return d[3]})**