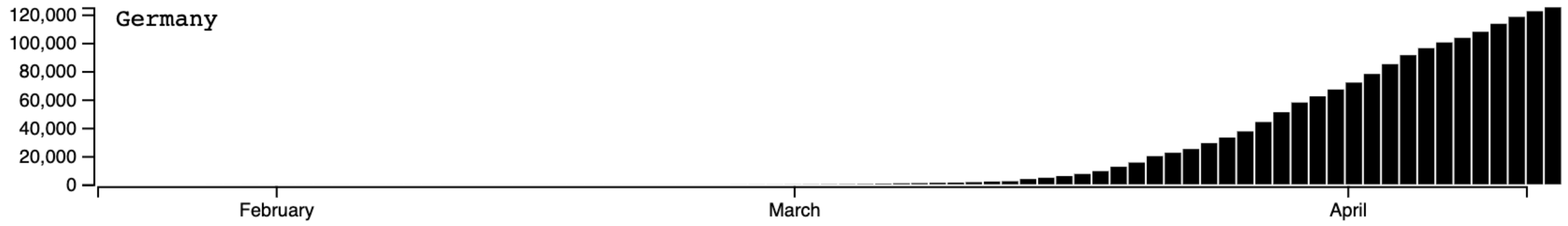
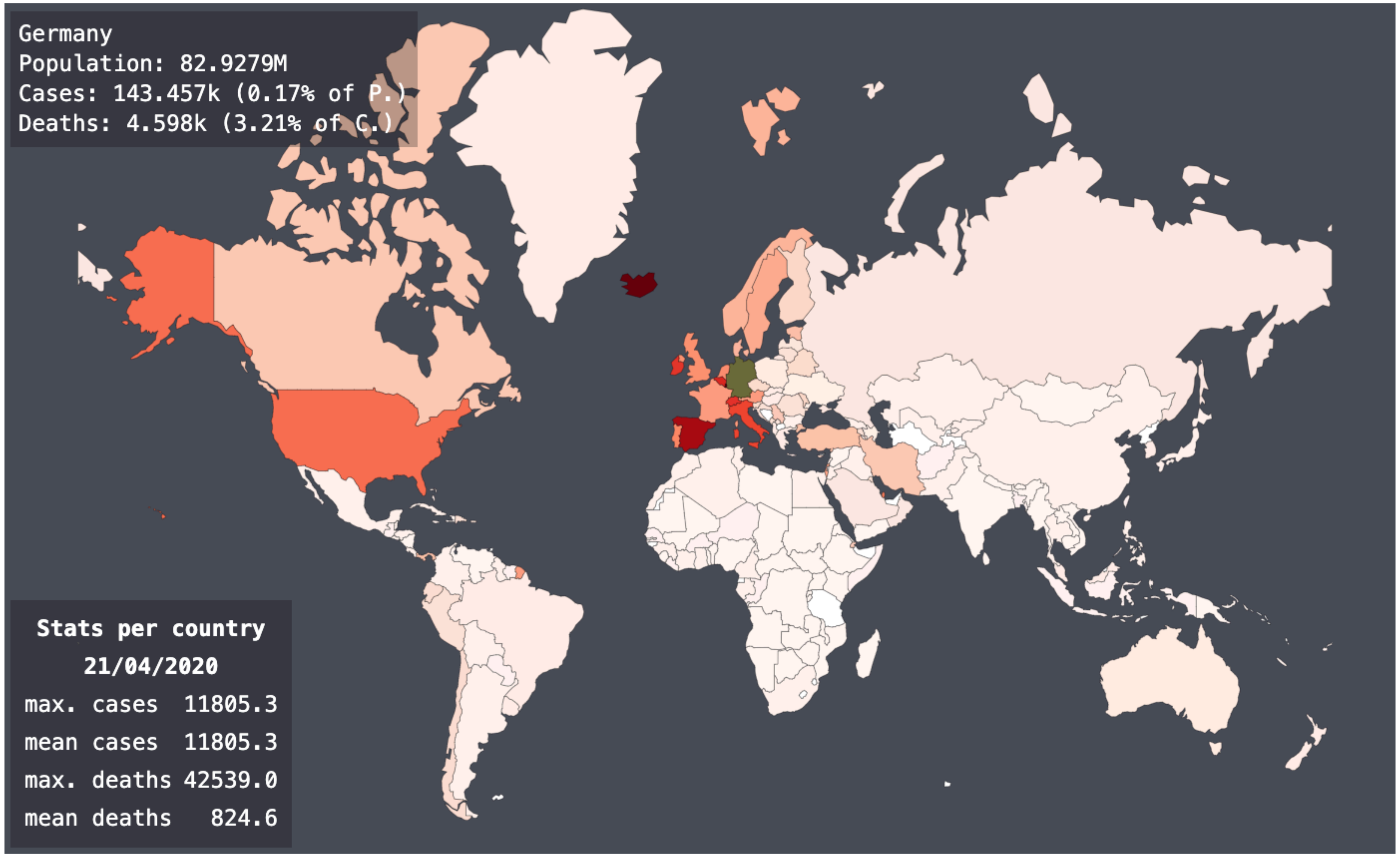
# DATENVISUALISIERUNG MIT D3

- SVG Aufbau
- Daten einlesen und filtern
- Achsen und Skalierung
- Säulengrafik
- Tortendiagramm
- Enter, Exit, Update und Merge
- Interaktion


Germany
120,000
100,000
80,000
60,000
40,000
20,000
0
February
March
April


Germany
Population: 82.9279M
Cases: 143.457k (0.17% of P.)
Deaths: 4.598k (3.21% of C.)

**Stats per country**
**21/04/2020**

```
max. cases  11805.3
mean cases  11805.3
max. deaths 42539.0
mean deaths   824.6
```

## THE HTML SCAFFOLD

## SCAFFOLD

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Covid Bars and Pie</title>
    <style>
        html,
        body,
        svg {
            margin: 0;
            width: 100%;
            height: 100%;
        }
    </style>
</head>

<body>

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

</html>
```

## THE SVG CONSTRUCTION

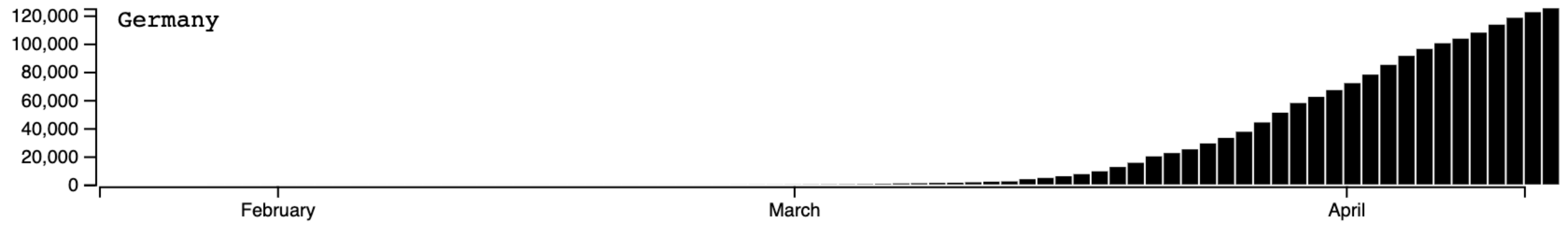
# SVG WIDTH, HEIGHT (FULLSCREEN)

```
let svg = d3.select('body')
    .append('svg')
    .attr('id', 'svg-bars')
    .attr('width', '100%')
    .attr('height', '100%');

let width = document.querySelector('#svg-bars').clientwidth;
let height = document.querySelector('#svg-bars').clientHeight;
let padding = 40;
```

## GLOBAL SCOPED

```
let
    data,       // The dataset
    enter,      // The enter selection
    group,      // A svg group
    bars,       // The bars rectangles
    gap = 1,    // Gap between bars
    xScale,     // The data based x scale
    yScale;     // The data based y scale

let parseDate = d3.timeParse('%m/%d/%Y');
```

# DATA BASED BAR CHART



## ADD A GROUP

```
function setGroup() {
    group = d3.select('#svg-bars')
        .append('g')
        .attr('id', 'bars');
}
```

## DATA ENTER SELECTION

```
function enterData() {

    data = [100, 208, 112, 79, 50];

    enter = group
        .selectAll()
        .data(data)
        .enter();
}
```

## DRAW BARS

```
function drawBars() {
   enter
        .append('rect')
        .classed('bar', true)
        .attr('x', (d, i) => padding+i*10)
        .attr('y', (d, i) => height-padding-d)
        .attr('width', (d, i) => 8)
        .attr('height', (d, i) => d)
}
```

## RICH DATA

```
data = [{
    date: '06/21/2020',
    value: 100
}, {
    date: '06/21/2020',
    value: 208
}, {
    date: '06/21/2020',
    value: 112
}, {
    date: '06/21/2020',
    value: 79
}, {
    date: '06/21/2020',
    value: 50
}];
```

## DRAW BARS

```
function drawBars() {
    enter
        .append('rect')
        .classed('bar', true)
        .attr('x', (d, i) => padding + i * 10)
        .attr('y', (d, i) => height - padding - d.value)
        .attr('width', (d, i) => 8)
        .attr('height', (d, i) => d.value)
}
```

## X- AND Y-SCALING

```
function setXScale() {
  xScale = d3.scaleTime()
    .domain(d3.extent(data, (d, i) => parseDate(d['date'])))
    .range([0, 2 * padding - width]);
}


function setYScale() {
  yScale = d3.scaleLinear()
    .domain([0, d3.max(data, (d, i) => +d['value'])])
    .range([2 * padding - height, 0]);
}
```

# CALCULATE THE BAR WIDTH, USING NUMBER OF DAYS

```
function setBarWidth(d, i) {

    let minDate = d3.min(data, (d) => d.date);
    let maxDate = d3.max(data, (d) => d.date);

    const diffTime = new Date(maxDate) - new Date(minDate);
    const diffDays = diffTime / (1000 * 60 * 60 * 24) + 1;

    return (width - 2 * padding) / diffDays;
}
```

## CALCULATE THE BAR HEIGHT

```
function setBarHeight(d, i) {
    return height - 2 * padding - yScale(d.value)
}
```

## SHOW THE X-AXIS

```
function drawXAxis() {
  group
    .append('g')
    .classed('axis', true)

    .attr('transform',
      `translate(${width-padding},${height-padding+1})`)

    .call(d3.axisBottom(xScale)
            .ticks(d3.timeMonth.every(1))
    );
}
```

## SHOW THE Y-AXIS

```
function drawYAxis() {
  group
    .append("g")
    .classed('axis', true)

    .attr('transform',
        `translate(${padding-2},${height-padding})`)

    .call(d3.axisLeft(yScale).ticks(5));
}
```

## WORKING WITH CSV FILES

## DATA AS CSV FILE

```
country,date,value,value1, value2
Germany,06/19/2020,100,8,48
Germany,06/20/2020,208,14,79
Germany,06/21/2020,112,5,32
Germany,06/22/2020,79,7,12
Germany,06/23/2020,50,9,21
Germany,06/24/2020,0,0,0
Germany,06/25/2020,90,5,42
```

## READING A CSV FILE

```
d3.csv('data/data-1.csv').then((_data) => {

    data = _data;

    setGroup();
    enterData();
    setXScale();
    setYScale();
    drawXAxis();
    drawYAxis();
    drawBars();
});
```

## FILTER DATA

```
function enterData() {

    data = data .filter((d, i) => d.country === 'Germany');

    enter = group
        .selectAll()
        .data(data)
        .enter();
}
```

## EVENTS ON BARS

```
function drawBars() {
  enter
    .append('rect')
    .classed('bar', true)
    .attr('x', (d, i) => padding + setBarWidth(d,i) * i)
    .attr('y', (d, i) => height - padding - setBarHeight(d,i))
    .attr('width', (d, i) => setBarWidth(d, i) - gap)
    .attr('height', (d, i) => setBarHeight(d,i))

    .on('mouseenter', onMouseEnter)
    .on('mouseleave', onMouseLeave)
}
```

## SHOWING A DATA LABEL

## ADDING AN OVER COLOR

```
.over {
    fill: gold;
}
```

## SOME LABEL CSS

```
#label {
    display: flex;
    justify-content: center;
    align-items: center;
    position: absolute;
    padding: 0.25rem;
    min-width: 2rem;
    min-height: 2rem;
    background-color: white;
    font-family: monospace;
    font-size: 0.8rem;
    box-shadow: 1px 1px 9px hsla(0, 0%, 0%, 0.25);
    opacity: 0;
}

#label.visible {
    opacity: 1;
}
```

## ADDING A HTML DATA LABEL

```
function onMouseEnter(d, i) {
    d3.select(d3.event.target).classed('over', true);

    let label = document.querySelector('#label');
    label.classList.add('visible');
    label.innerHTML = `
        ${d.country}<br>
        ${d.date}<br>
        ${d.value}
    `;

    label.setAttribute('style', `
    left:${padding + setBarWidth(d, i) * i - 5}px;
    top:${height - padding - setBarHeight(d, i)-5}px
    `);
}
```

## REMOVE THE LABEL

```
function onMouseLeave(d, i) {
    d3.select(d3.event.target).classed('over', false);

    let label = document.querySelector('#label');
    label.classList.remove('visible');
}
```

## ADDING A TITLE

## ADDING AN INPUT ELEMENT

```
function setTitle() {
    let titleGroup = d3.select(`#svg`)
        .append('g')
        .attr('id', 'title')
        .classed('title', true)
        .attr('transform', `translate(${padding+10},$
{padding})`);

    titleGroup
        .append('text')
        .attr('x', 0)
        .attr('y', 0)
        .text(localStorage.getItem('country'));
}
```

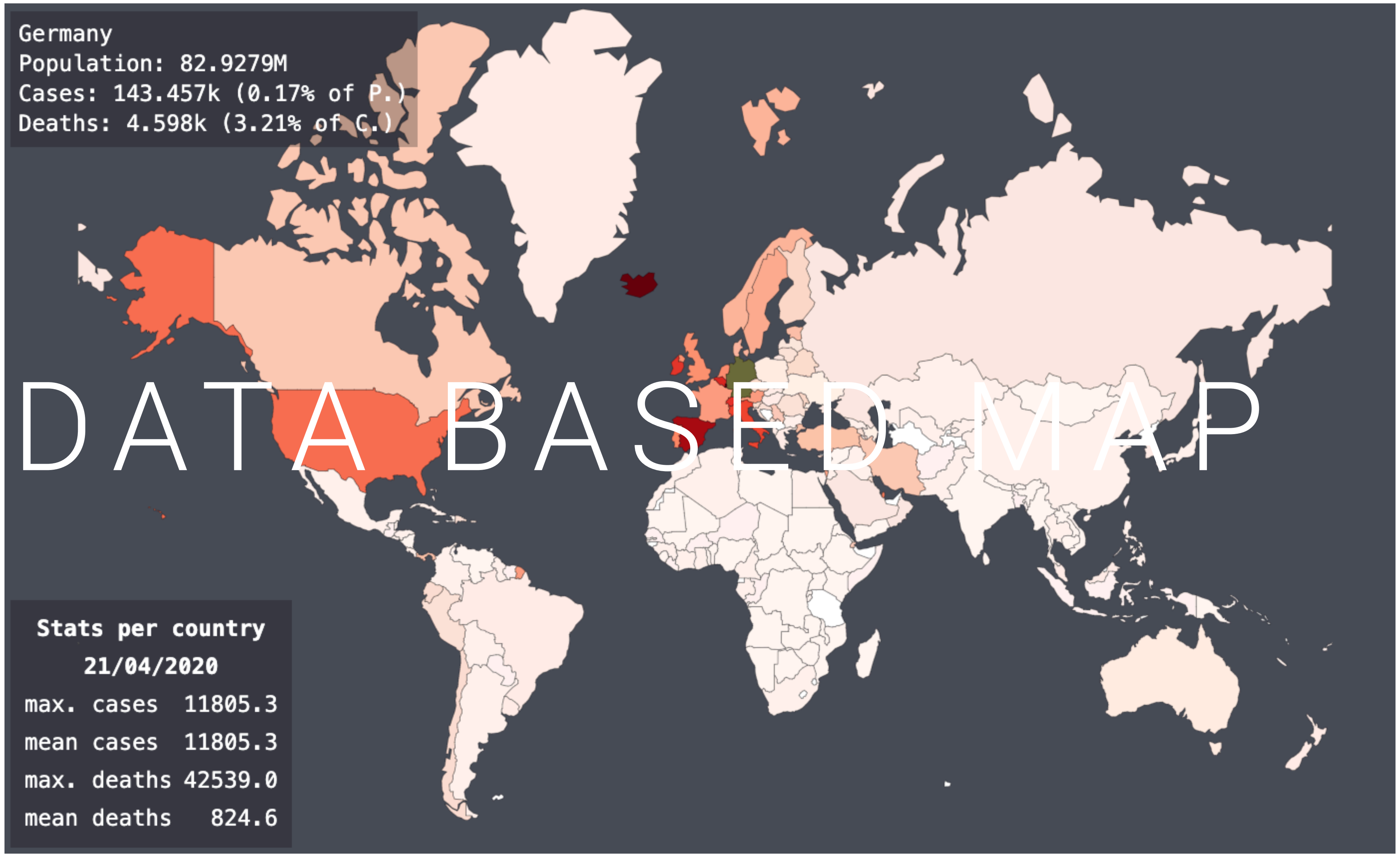
## BUILDING A SELECT MENU FOR THE COUNTRY

## ADDING AN INPUT ELEMENT

```
<input type="text" id="funnel" placeholder="enter country"
    onchange="bars.switchData();" list="countries">

<datalist id="countries"></datalist>
```

## THE DATA LIST

```
function setDatalist() {
    d3.csv('data/data-2.csv').then((data) => {
        let c = d3
            .nest()
            .key(function (d,i) {
                // console.log(d.country);
                let value = d['country'];
                return value;
            })
            .entries(data);

        for (let o of c) {
            let option = document.createElement('option');
            option.setAttribute('value', o.key);

document.querySelector('#countries').appendChild(option);
        }
    });
}
```


Germany
Population: 82.9279M
Cases: 143.457k (0.17% of P.)
Deaths: 4.598k (3.21% of C.)

```
Stats per country
   21/04/2020
max. cases   11805.3
mean cases   11805.3
max. deaths 42539.0
mean deaths   824.6
```

## SVG WIDTH, HEIGHT (FULLSCREEN)

```
let svg = d3.select('body')
    .append('svg')
    .attr('id', 'svg-map')
    .attr('width', '100%')
    .attr('height', '100%');

let width = document.querySelector('#svg-map').clientwidth;
let height = document.querySelector('#svg-map').clientHeight;
let padding = 40;
```

## GLOBAL SCOPED

```
let
    data,       // The Covid dataset
    world,      // The Topojson
    enter,      // The enter selection
    group,      // A svg group
    map;        // The map paths

let path = d3.geoPath().projection(
    d3.geoMercator()
    .scale(100)
    .translate([704 / 2, 504 / 2 + 40]));
```

## ADD A GROUP

```
function setGroup() {
    group = d3.select('#svg-map')
        .append('g')
        .attr('id', 'map');
}
```

## LOAD TWO DATASETS, THEN DO SO FAR …

```
Promise.all([
    d3.json('../assets/data/countries-110m.json'),
    d3.csv('data/covid-19-worldwide.csv'),
]).then(function (files) {

    world = files[0];
    data = files[1];


    // Map:
    setGroup();
    enterData(world);
    drawMap();
})
```

## DATA ENTER SELECTION

```
function enterData(world) {
    enter = group.selectAll("path")
        .data(
            topojson.feature(world,
            world.objects.countries).features
        )
        .enter();
}
```

## DRAW MAP

```
function drawMap() {
    enter.append("path")
        .attr('id', (
            d, i) => d.properties.name.replace(/[ ]/, '_')
        )
        .attr("d", path)
        .on('mouseenter', (d, i) => {
            let name = d.properties.name;
            d3.select('label').html(name);
        })
        .on('mouseleave', (d, i) => {
            d3.select('label').html('');
        });
}
```

## THE TOPOJSON

## TOPOJSON COUNTRY LIST

```
{
    "type": "Topology",
    "objects": {
        "countries": {
              "type": "GeometryCollection",
              "geometries": [
                   ...
                   {
                       "type": "Polygon",
                       "arcs": [
                            [2, 3, 4, 5, 6, 7, 8, 9, 10]
                       ],
                      "id": "834",
                      "properties": {
                      "name": "Tanzania"
                   }
              },
              ...
```

# TOPOJSON ARCS

```
"land": {
    "type": "GeometryCollection",
    "geometries": [{
    "type": "MultiPolygon",
    "arcs": [
        [ [0] ],
        [ [1] ],
        [ [3, 320, 184, 255, 323, 104, 322, 311, 313, 315, 289,
          284, 273, 290, 293, 297, 305, 307, 302, 304, 263, 336,
          258, 272, 13, 573, 341, 338, 577, 575, 329, 332, 423,
          487, 535, 153, 435, 164, 436, 461, 477, 489, 484, 490,
          592, 497, 548, 525, 240, 507, 505, 506, 242, 501, 503,
          ... ],
          [421, 416, 64, 150, 533]
        ],
    ...
```

## TOPOJSON BOX

```
,
    "bbox": [-180, -85.60903777459771, 180, 
83.64513000000001],
    "transform": {
        "scale": [0.0036000360003600037, 
0.0016925586033320105],
        "translate": [-180, -85.60903777459771]
    }
```

https://github.com/topojson/topojson-specification/blob/master/README.md

MIKE BOSTOCK, CALVIN METCALF

## COVID DATA

# COVID-19-WORLDWIDE.CSV

```
dateRep,day,month,year,cases,deaths,countriesAndTerritories,
geoId,countryterritoryCode,popData2018

17/04/2020,17,4,2020,10,4,Afghanistan,AF,AFG,37172386
16/04/2020,16,4,2020,70,2,Afghanistan,AF,AFG,37172386
15/04/2020,15,4,2020,49,2,Afghanistan,AF,AFG,37172386
14/04/2020,14,4,2020,58,3,Afghanistan,AF,AFG,37172386
13/04/2020,13,4,2020,52,0,Afghanistan,AF,AFG,37172386
12/04/2020,12,4,2020,34,3,Afghanistan,AF,AFG,37172386
11/04/2020,11,4,2020,37,0,Afghanistan,AF,AFG,37172386
...
```

## GROUPING DATA BY COUNTRY

```
function nestByCountry(data) {
    return d3.nest()
        .key((d) => d.countriesAndTerritories)
        .entries(data);
}
```

## NESTED BY COUNTRY

```
[{
    key:    "Afghanistan"
    values: [
        {dateRep: "17/04/2020", ..., cases: "10", ...},
        {dateRep: "16/04/2020", ..., cases: "70", ...},
        {dateRep: "15/04/2020", ..., cases: "49", ...},
        {dateRep: "14/04/2020", ..., cases: "58", ...},
        {dateRep: "13/04/2020", ..., cases: "52", ...},
        ...
    ]
},{
    key:    "Albania"
    values: [
        {dateRep: "17/04/2020", ..., cases: "24", ...},
        ...
```

# CALCULATE SUMS

```
function summarizeCases(data) {
    return d3.nest().key((d) => d.countriesAndTerritories)
        .rollup((leaves) => d3.sum(leaves, (d) => d.cases))
        .entries(data)
        .map((d) => {
            return {
                country: d.key,
                value: d.value
            };
        });
}

function summarizeDeaths(data) {
    return d3.nest().key((d) => d.countriesAndTerritories)
        .rollup((leaves) => d3.sum(leaves, (d) => d.deaths))
        .entries(data)
        .map((d) => {
            return {
                country: d.key,
                value: d.value
            };
        });
}
```

## STATISTIC

```
let sumCases  = summarizeCases(data);
let sumDeaths = summarizeDeaths(data);
let maxCases  = d3.max(sumCases, (d, i) => d.value);
let maxDeaths = d3.max(sumDeaths, (d, i) => d.value);
```

https://github.com/d3/d3-array

MIKE BOSTOCK