Run `git pull` in the main branch to follow along today.

JS (Part 2), D3.js

DSC 106: Data Visualization

Sam Lau

UC San Diego

Announcements

Lab 4 due today.

Project 2 due on Tuesday.

FAQs:

1. I’m getting 404 errors when working on the lab! Check that you’re using relative URLs and they match your folder structure.

How to use GenAI in this course

Philosophy

ChatGPT

Whoa, you're not learning "real" programming!

```python
print('hello world')
```

Whoa, you're not learning "real" programming!

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

Sometimes details matter (e.g. high-perf computing, kernel drivers). But not always!

What you should know	What to use GenAI for
Visual encodings Perception, color How to design a good plot!	Brainstorming plot ideas, generating options to choose from.
JS syntax, functions, objects	Writing simple functions
The document object model	document API and functions
Event handlers/listeners	Event names
D3 core abstractions	Creating initial plot prototypes

You Try: Your favorite website interaction

Go to your favorite website, pick an interactive element, describe event listener and event handler.

tryclassbuzz.com
Code: **interaction**

JS (Part 2)

Async / await

js-lecture/weather01/

(demo)

```html
<head>
  <title>Temperature Converter</title>

  <link rel="stylesheet" href="main.css" />
  <script src="main.js" type="module"></script>
```

If main.js code is slow, then page will freeze (!) while waiting for JS to finish

What if the dataset just takes a while to download?

Idea: Allow functions to run in the background (asynchronously) so that page doesn't freeze

```
async function loadWeatherData() {
  try {
    const response = await fetch('./weather-data.json');
    const weatherData = await response.json();
    return weatherData;
  } catch (error) {
    console.error('Error loading weather data:', error);
  }
}
```

async = this function uses other async functions

```
async function loadWeatherData() {
  try {
    const response = await fetch('./weather-data.json');
    const weatherData = await response.json();
    return weatherData;
  } catch (error) {
    console.error('Error loading weather data:', error);
  }
}
```

async = this function uses other async functions

await = this function might take a while, so let the browser do other stuff while we wait

```
function loadStory() {
  return getJSON('story.json')
    .then(function (story) {
      addHtmlToPage(story.heading);

      return story.chapterURLs
        .map(getJSON)
        .reduce(function (chain, chapterPromise) {
          return chain
            .then(function () {
              return chapterPromise;
            })
            .then(function (chapter) {
              addHtmlToPage(chapter.html);
            });
        }, Promise.resolve());
    })
    .then(function () {
      addTextToPage('All done');
    })
    .catch(function (err) {
      addTextToPage('Argh, broken: ' + err.message);
    })
    .then(function () {
      document.querySelector('.spinner').style.display = 'none';
    });
}
```

Back in the day, we had to use JS Promises that had a .then() and .catch() syntax.

async/await is the modern version that makes writing this code a LOT easier

Half the code!

https://jakearchibald.com/2014/es7-async-functions/

```
async function loadStory() {
  try {
    let story = await getJSON('story.json');
    addHtmlToPage(story.heading);
    for (let chapter of story.chapterURLs.map(getJSON)) {
      addHtmlToPage((await chapter).html);
    }
    addTextToPage('All done');
  } catch (err) {
    addTextToPage('Argh, broken: ' + err.message);
  }
  document.querySelector('.spinner').style.display = 'none';
}
```

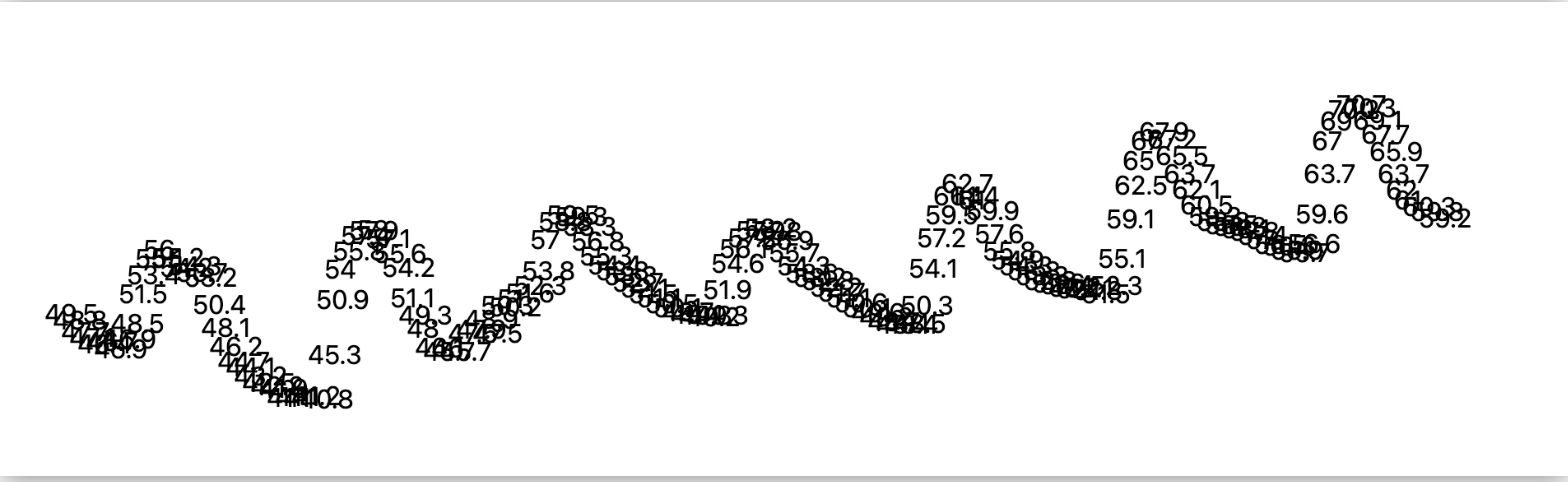
Now, let's make our very first data visualization in JS:

js-lecture/weather02/

(demo)

js-lecture/weather03/

(demo)

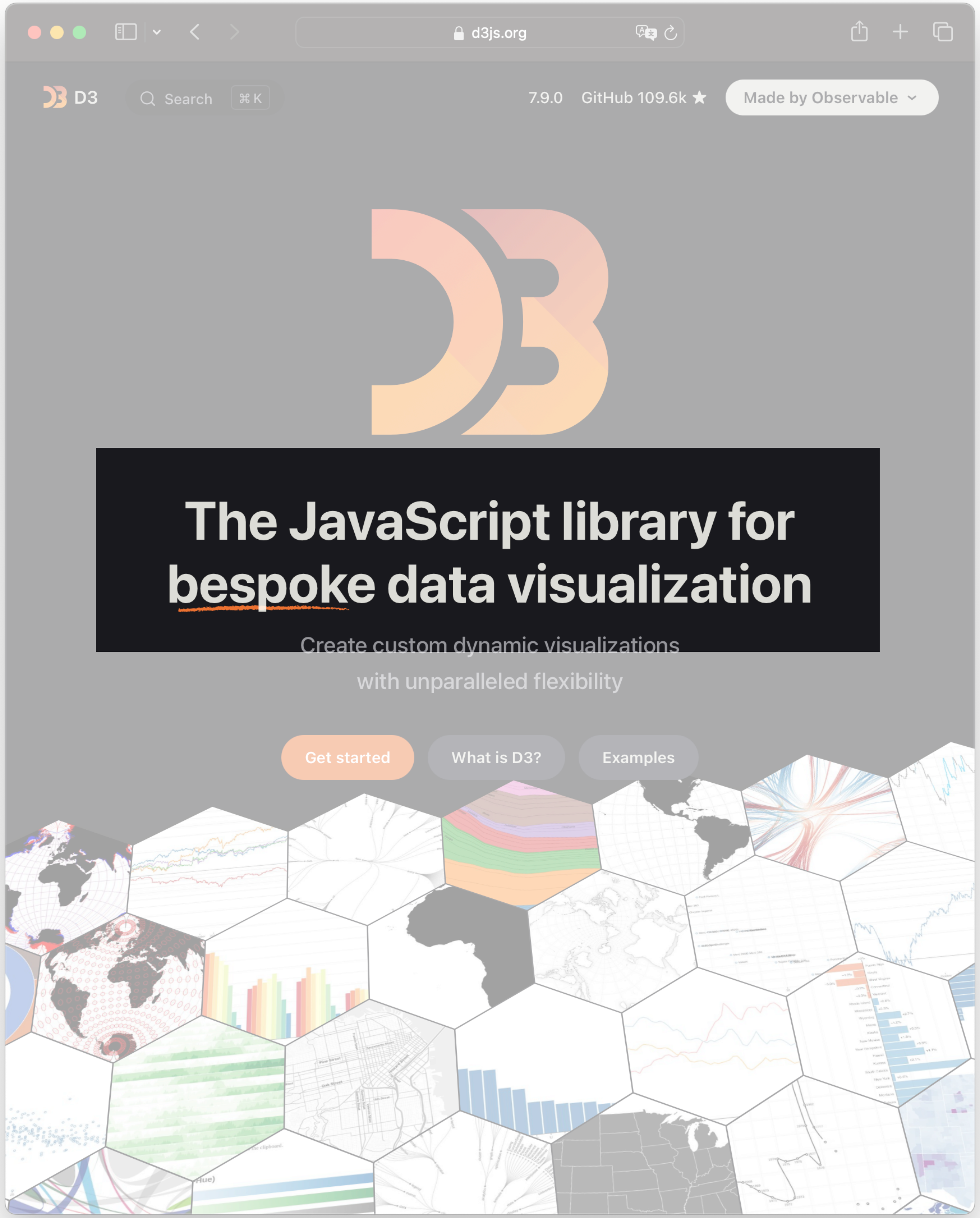


How would you add an x-axis and y-axis? Gridlines?

tryclassbuzz.com
Code: **axes**

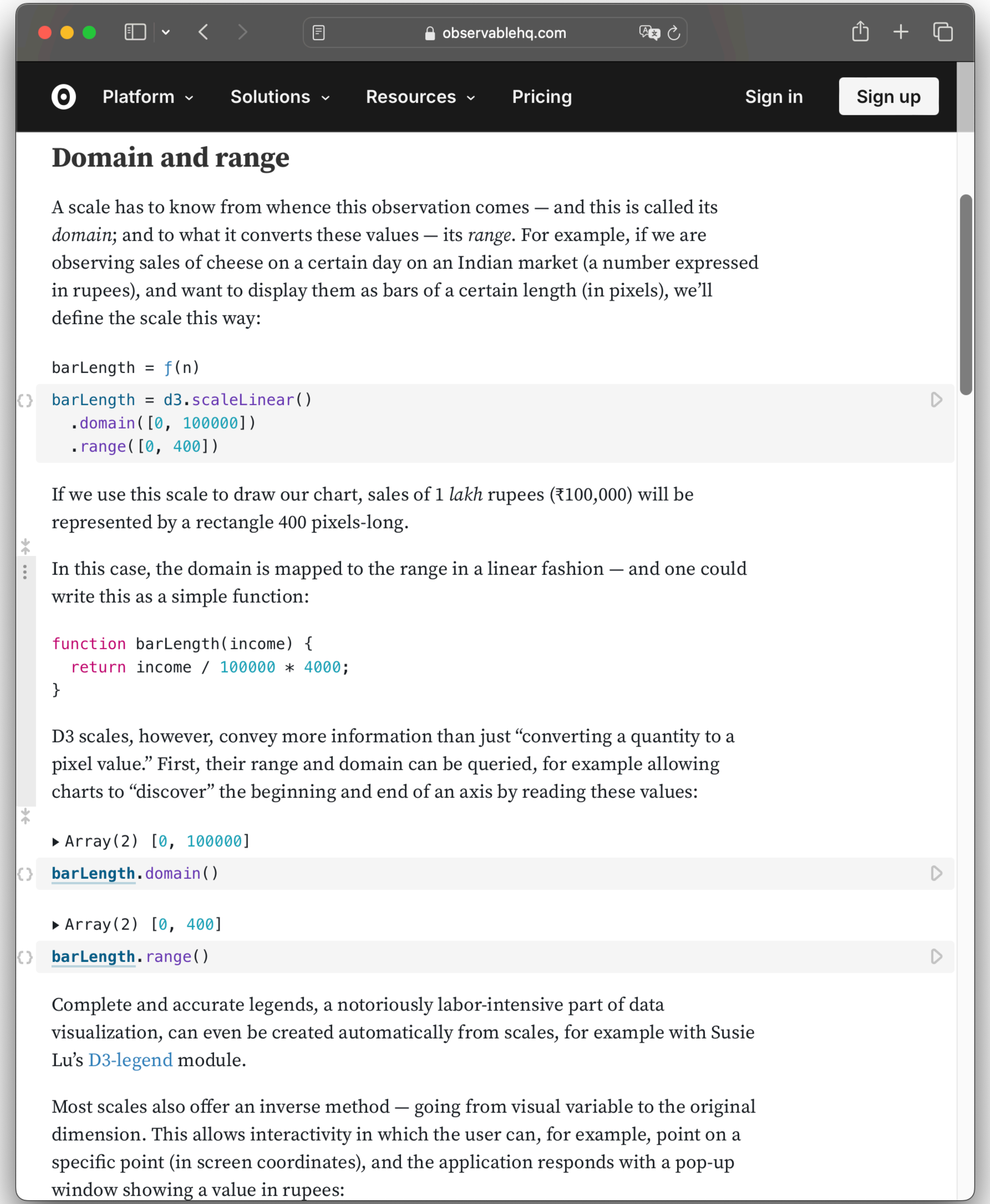
D3

https://d3js.org/



https://d3js.org/

Bespoke = fully custom


observablehq.com

Platform Solutions Resources Pricing Sign in Sign up

Domain and range

A scale has to know from whence this observation comes — and this is called its *domain*; and to what it converts these values — its *range*. For example, if we are observing sales of cheese on a certain day on an Indian market (a number expressed in rupees), and want to display them as bars of a certain length (in pixels), we'll define the scale this way:

```
barLength = f(n)
```

```
barLength = d3.scaleLinear()
  .domain([0, 100000])
  .range([0, 400])
```

If we use this scale to draw our chart, sales of 1 *lakh* rupees (₹100,000) will be represented by a rectangle 400 pixels-long.

In this case, the domain is mapped to the range in a linear fashion — and one could write this as a simple function:

```
function barLength(income) {
  return income / 100000 * 4000;
}
```

D3 scales, however, convey more information than just "converting a quantity to a pixel value." First, their range and domain can be queried, for example allowing charts to "discover" the beginning and end of an axis by reading these values:

```
▸Array(2) [0, 100000]
barLength.domain()
```

```
▸Array(2) [0, 400]
barLength.range()
```

Complete and accurate legends, a notoriously labor-intensive part of data visualization, can even be created automatically from scales, for example with Susie Lu's D3-legend module.

Most scales also offer an inverse method — going from visual variable to the original dimension. This allows interactivity in which the user can, for example, point on a specific point (in screen coordinates), and the application responds with a pop-up window showing a value in rupees:

D3 has a parent company called Observable…

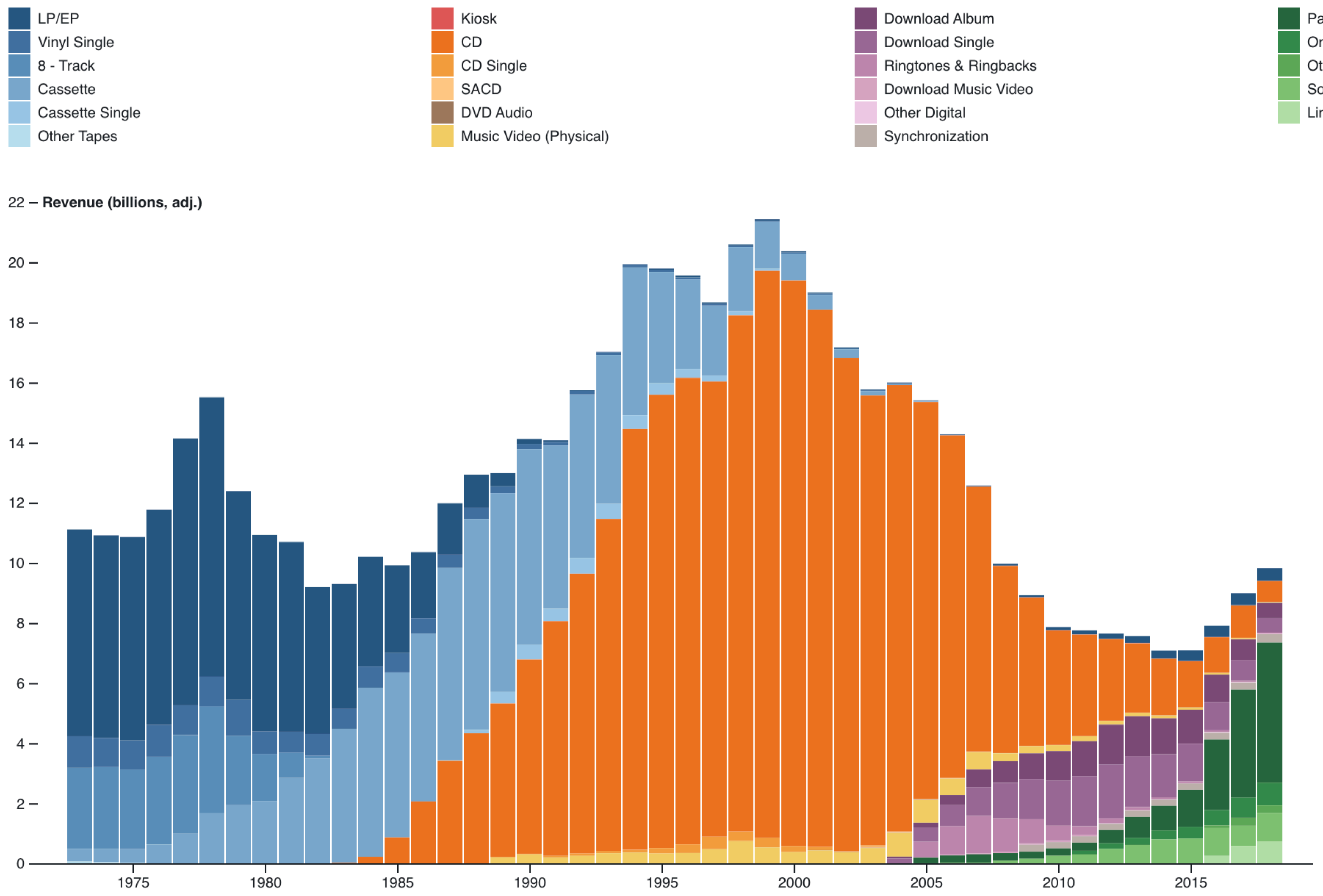
…which has a modified JS language!

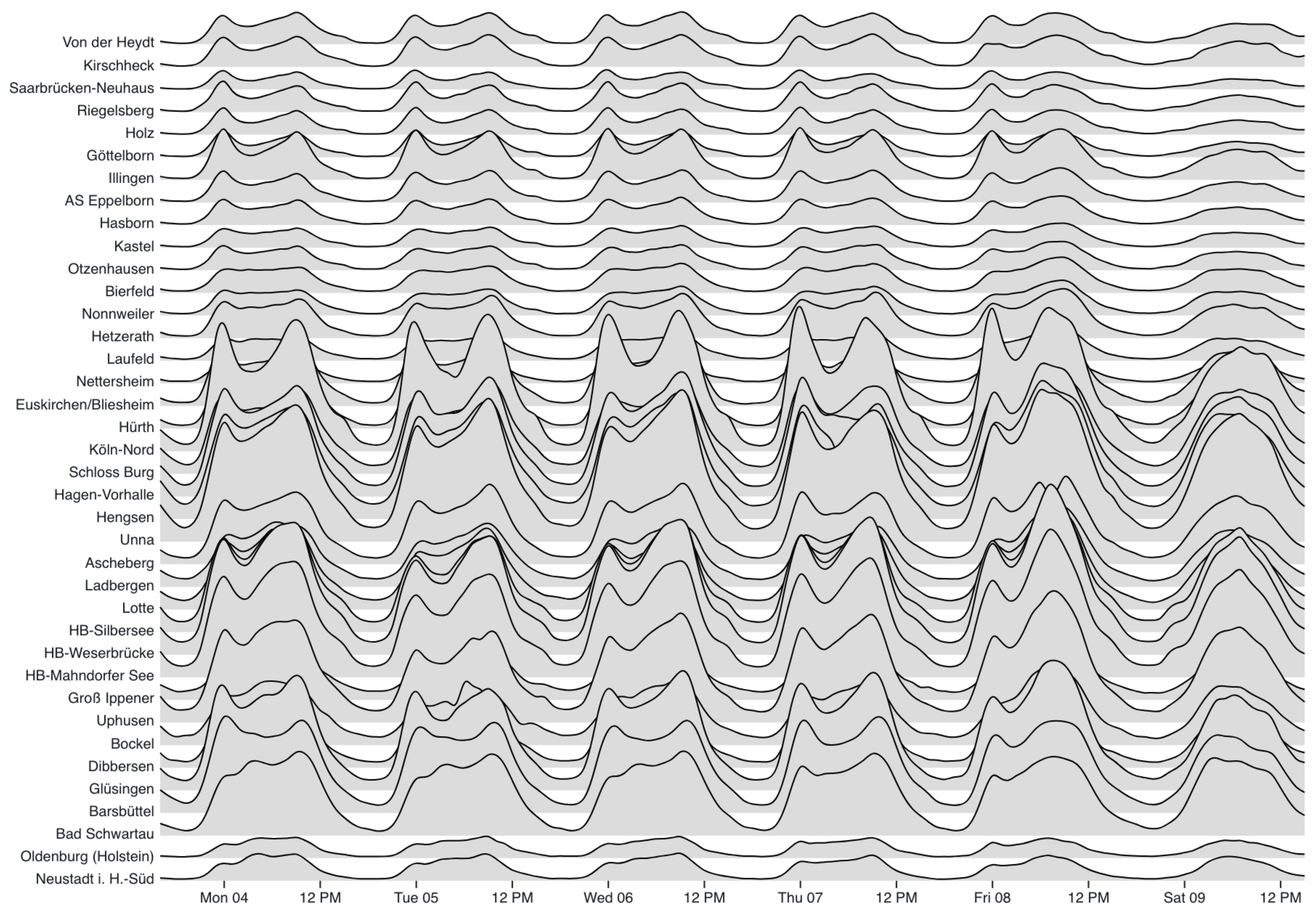
We won't use Observable for this class, but keep that in mind

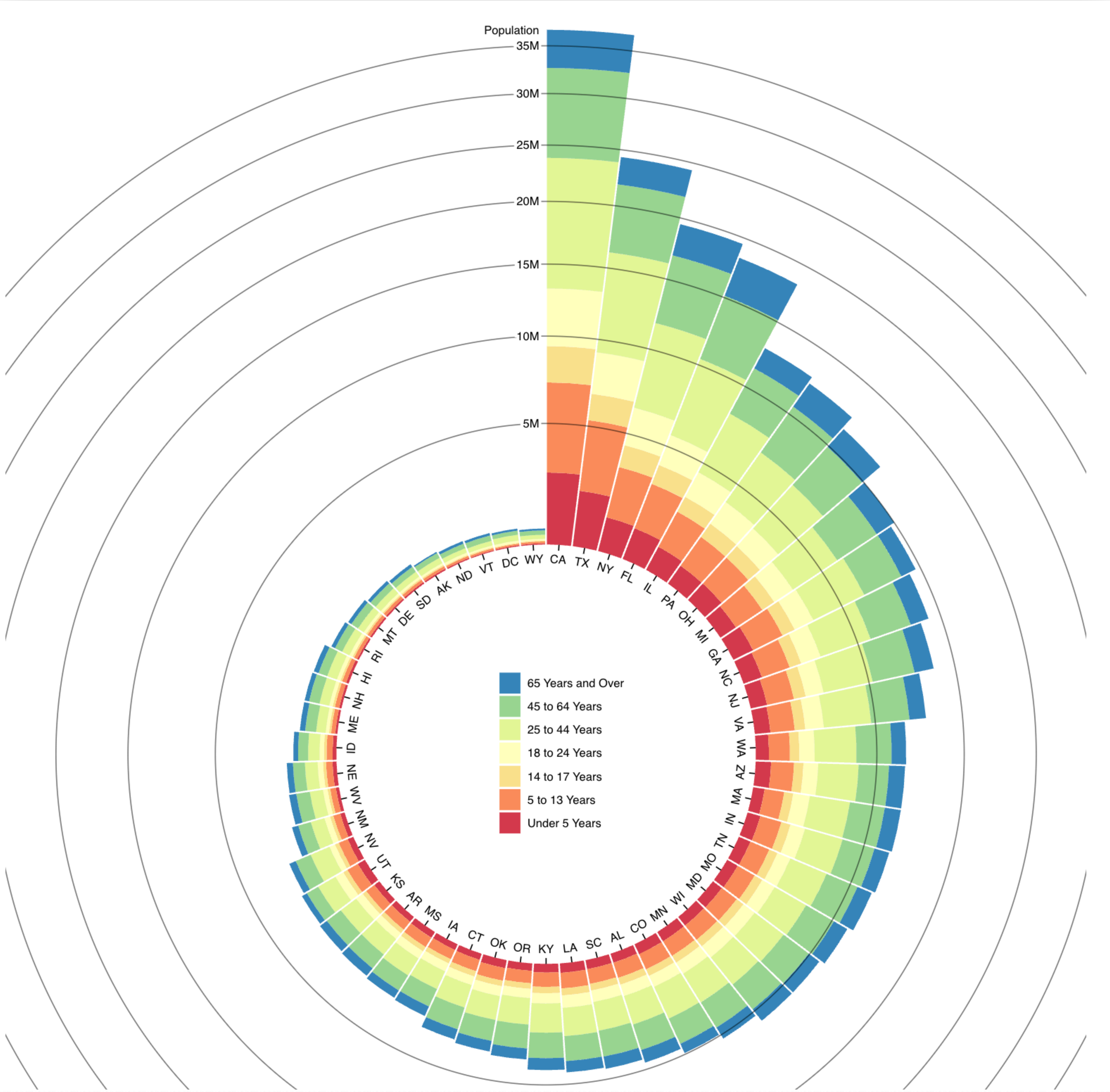
D3 › GALLERY

Revenue by music format, 1973–2018

Data: RIAA

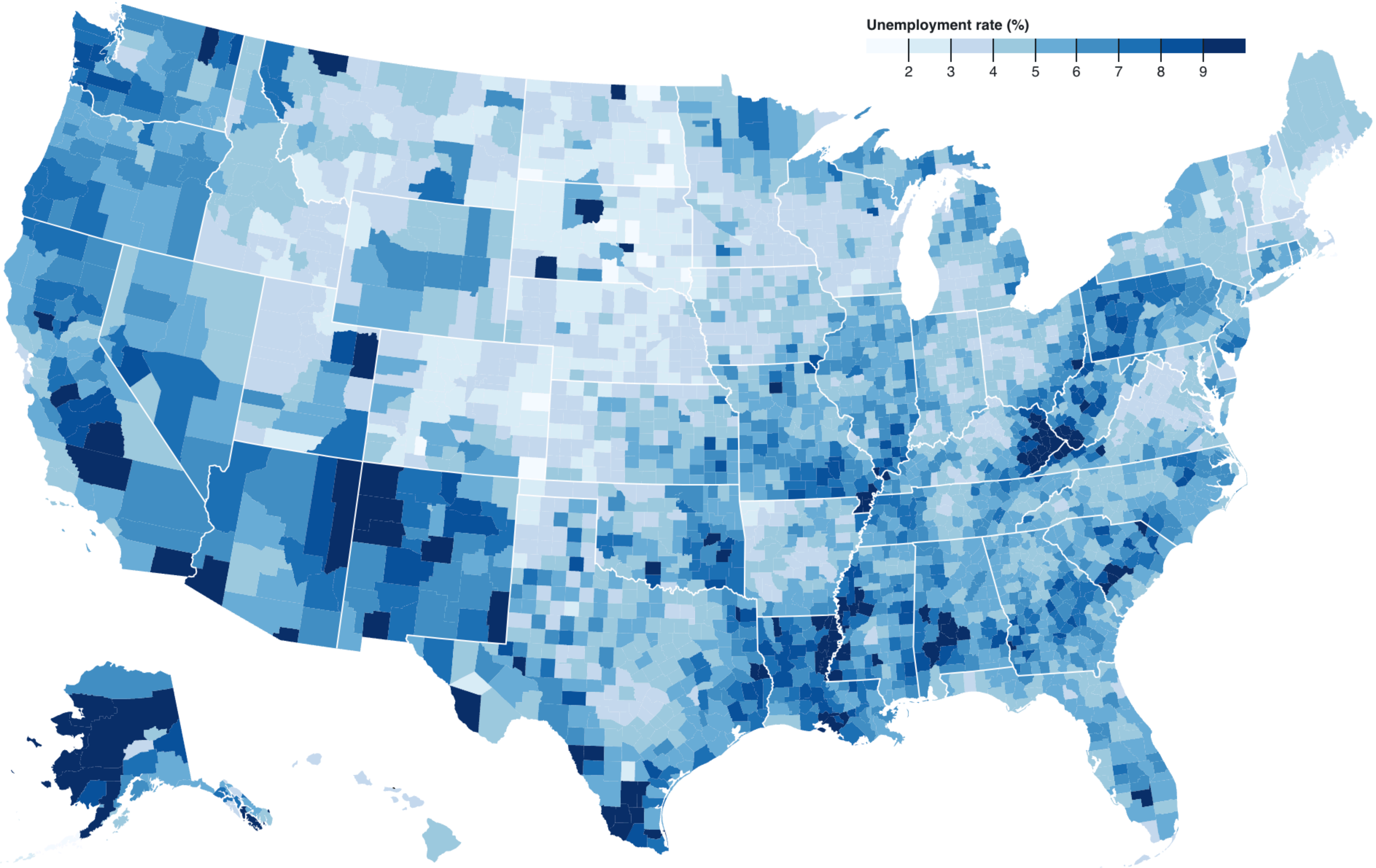



Von der Heydt
Kirschheck
Saarbrücken-Neuhaus
Riegelsberg
Holz
Göttelborn
Illingen
AS Eppelborn
Hasborn
Kastel
Otzenhausen
Bierfeld
Nonnweiler
Hetzerath
Laufeld
Nettersheim
Euskirchen/Bliesheim
Hürth
Köln-Nord
Schloss Burg
Hagen-Vorhalle
Hengsen
Unna
Ascheberg
Ladbergen
Lotte
HB-Silbersee
HB-Weserbrücke
HB-Mahndorfer See
Groß Ippener
Uphusen
Bockel
Dibbersen
Glüsingen
Barsbüttel
Bad Schwartau
Oldenburg (Holstein)
Neustadt i. H.-Süd
Mon 04
12 PM
Tue 05
12 PM
Wed 06
12 PM
Thu 07
12 PM
Fri 08
12 PM
Sat 09
12 PM


Population
35M
30M
25M
20M
15M
10M
5M
65 Years and Over
45 to 64 Years
25 to 44 Years
18 to 24 Years
14 to 17 Years
5 to 13 Years
Under 5 Years
CA TX NY FL IL PA OH MI GA NC NJ VA WA AZ MA IN TN MO MD WI MN CO AL SC LA KY OR OK CT IA MS AR KS UT NV NM WV NE ID ME NH HI RI MT DE SD AK ND VT DC WY

Choropleth

Unemployment rate by U.S. county, August 2016. Data: Bureau of Labor Statistics.



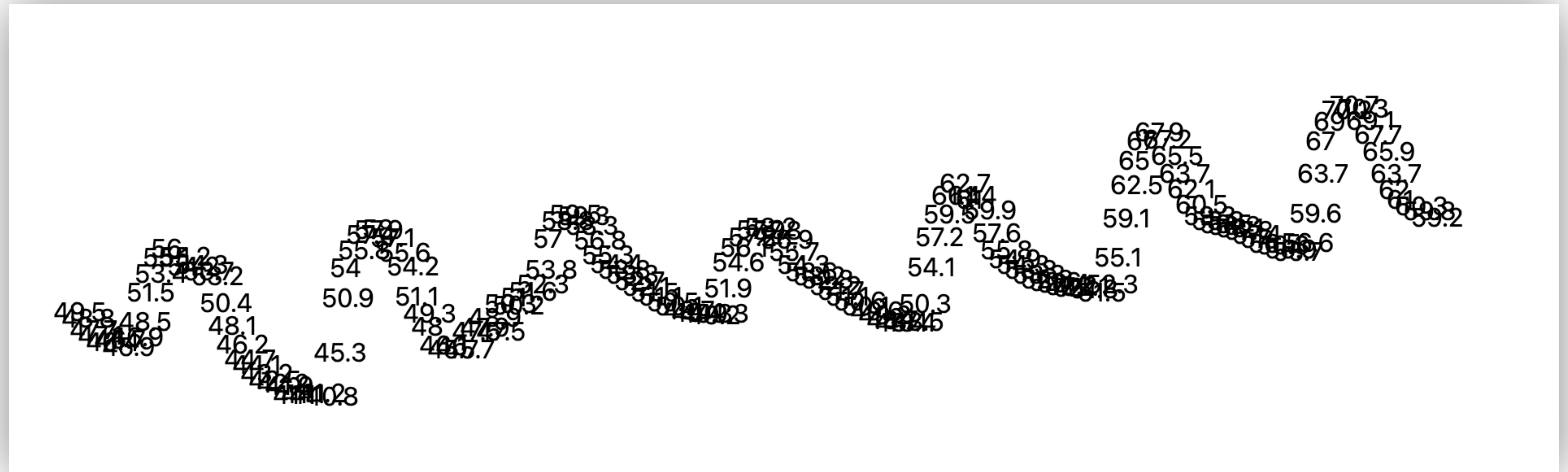

Pause
1964
Sub-Saharan Africa
South Asia
Middle East & North Africa
America
Europe & Central Asia
East Asia & Pacific
↑ Life expectancy (years)
80
70
60
50
40
30
20
200
300
400
500
1,000
2,000
3,000
4,000
5,000
10,000
20,000
30,000
40,000
50,000
100,000
Income per capita (dollars) →

https://observablehq.com/@mbostock/the-wealth-health-of-nations

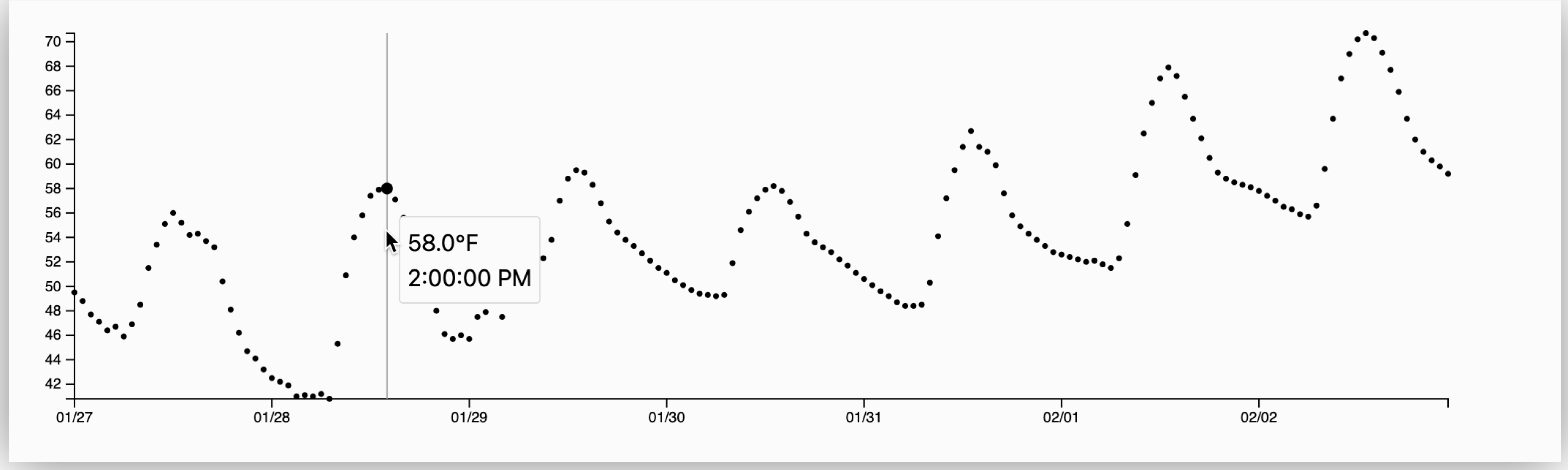
200+ lines of code!

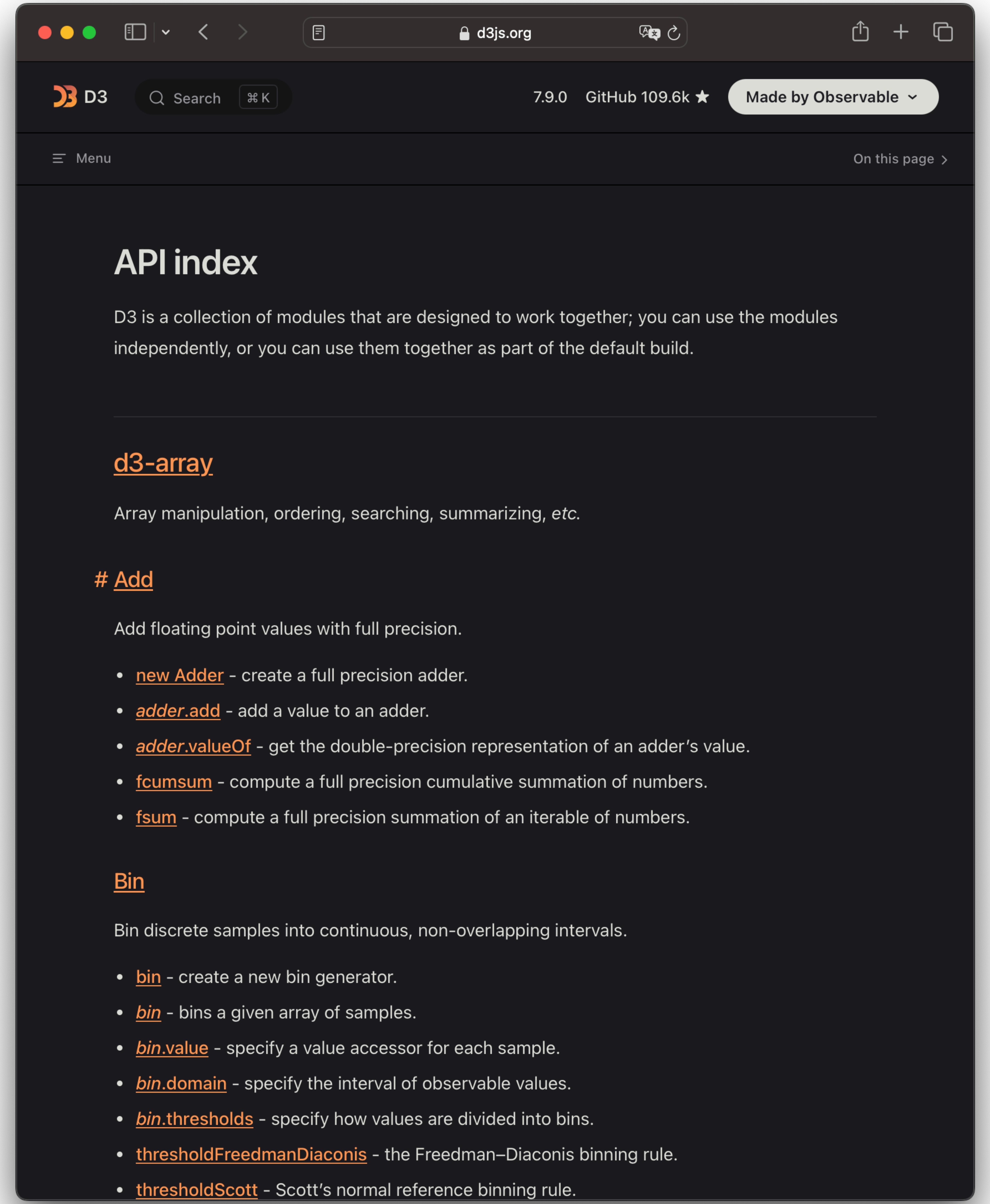
Goal: Making an interactive scatterplot

Before:



After:



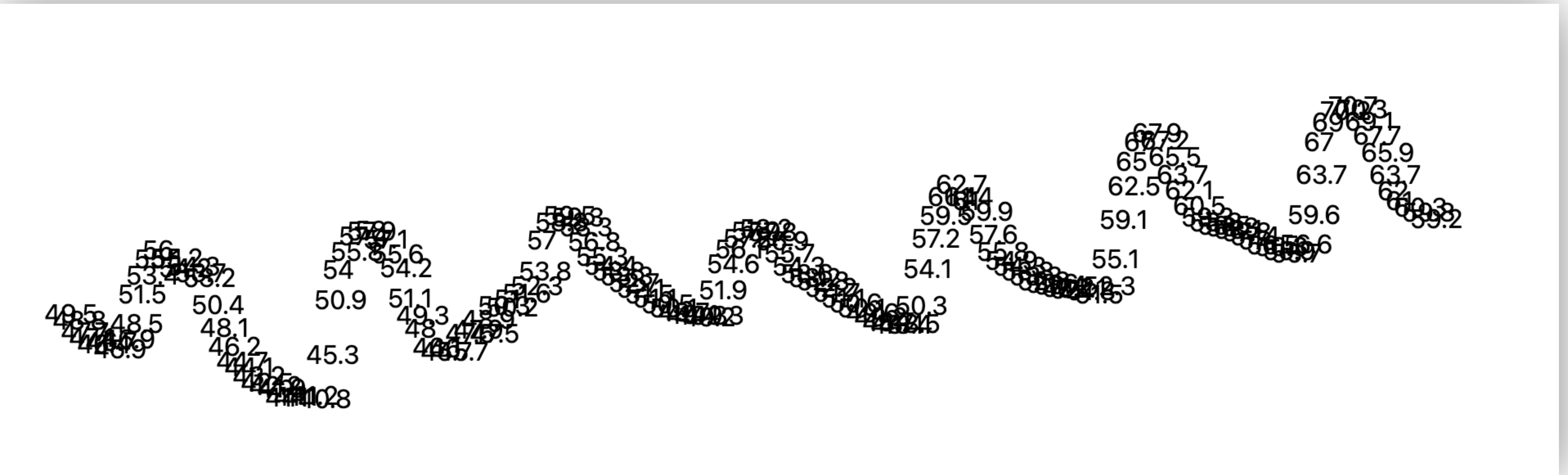


d3 is a huge library, 30 submodules, >1k methods!

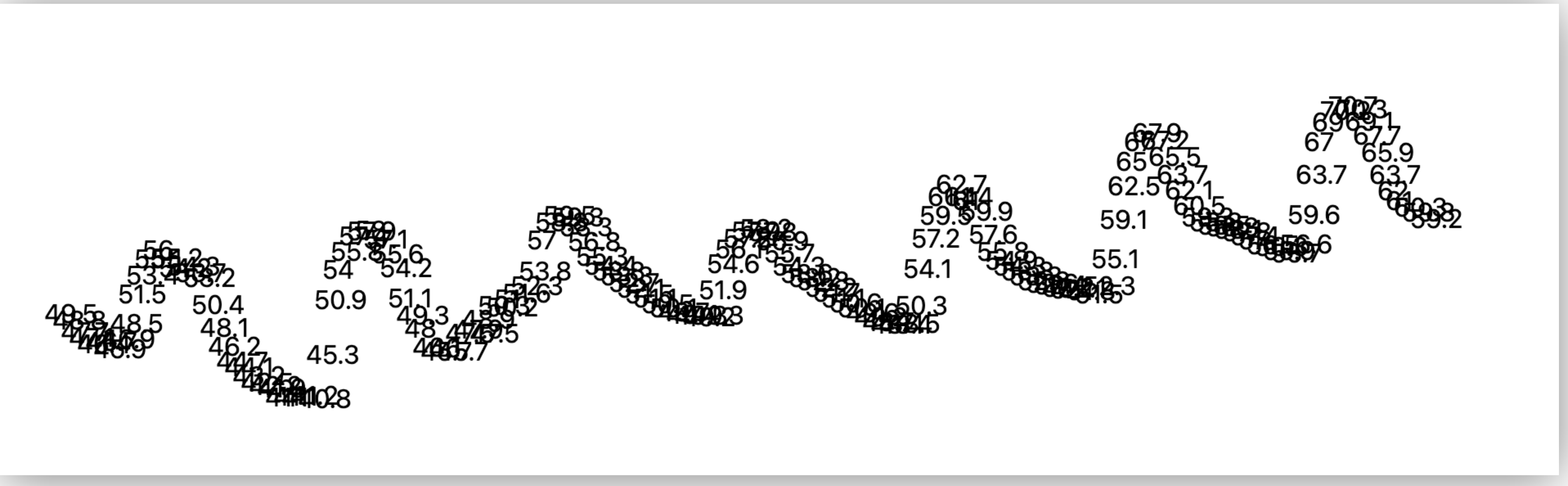
80/20 rule: Learn 20% and you'll be able to do 80% of what d3 is capable of.

Step 1: Using D3 instead of plain JS

Before:



After:



But in D3!

Demo: `d3-lecture/weather01`

D3 Selections

Before:

```
const svg = document.querySelector('#weather-plot');
```

After:

```
const svg = d3.select('#weather-plot');
```

This is a D3 selection object, so it has D3 methods (not HTML methods!)

D3 Selections

Before:

```
const svg = document.querySelector('#weather-plot');
```

After:

```
const svg = d3.select('#weather-plot');
```

```
svg.setAttribute('width', 1000);
svg.setAttribute('height', 500);
```

HTML element method

```
svg.attr('width', 1000);
svg.attr('height', 500);
```

D3 equivalent

Don’t memorize method names,
just use Copilot / ChatGPT

Data Joins

Before:

```
weatherData.hourly.temperature_2m.forEach((temp, index) => {
  const text = document.createElementNS('http://www.w3.org/2000/svg', 'text');
  text.setAttribute('x', index * 5);
  text.setAttribute('y', 500 - temp * 6);

  text.textContent = temp;
  svg.appendChild(text);
});
```

HTML methods

After:

```
svg
  .selectAll('text')
  .data(weatherData.hourly.temperature_2m)
  .join('text')
  .attr('x', (d, i) => i * 5)
  .attr('y', (d) => 500 - d * 6)
  .text((d) => d);
```

The D3 way

Data Joins

```
svg
  .selectAll('text')
  .data(weatherData.hourly.temperature_2m)
  .join('text')
  .attr('x', (d, i) => i * 5)
  .attr('y', (d) => 500 - d * 6)
  .text((d) => d);
```

Match data with text elements

Data Joins

```
svg
  .selectAll('text')
  .data(weatherData.hourly.temperature_2m)
  .join('text')
  .attr('x', (d, i) => i * 5)
  .attr('y', (d) => 500 - d * 6)
  .text((d) => d);
```

Create one new text element for each datum

Data Joins

```
svg
  .selectAll('text')
  .data(weatherData.hourly.temperature_2m)
  .join('text')
  .attr('x', (d, i) => i * 5)
  .attr('y', (d) => 500 - d * 6)
  .text((d) => d);
```

Set the x, y, and text content of each text element

Data Joins

```
svg
  .selectAll('text')
  .data(weatherData.hourly.temperature_2m)
  .join('text')
  .attr('x', (d, i) => i * 5)
  .attr('y', (d) => 500 - d * 6)
  .text((d) => d);
```

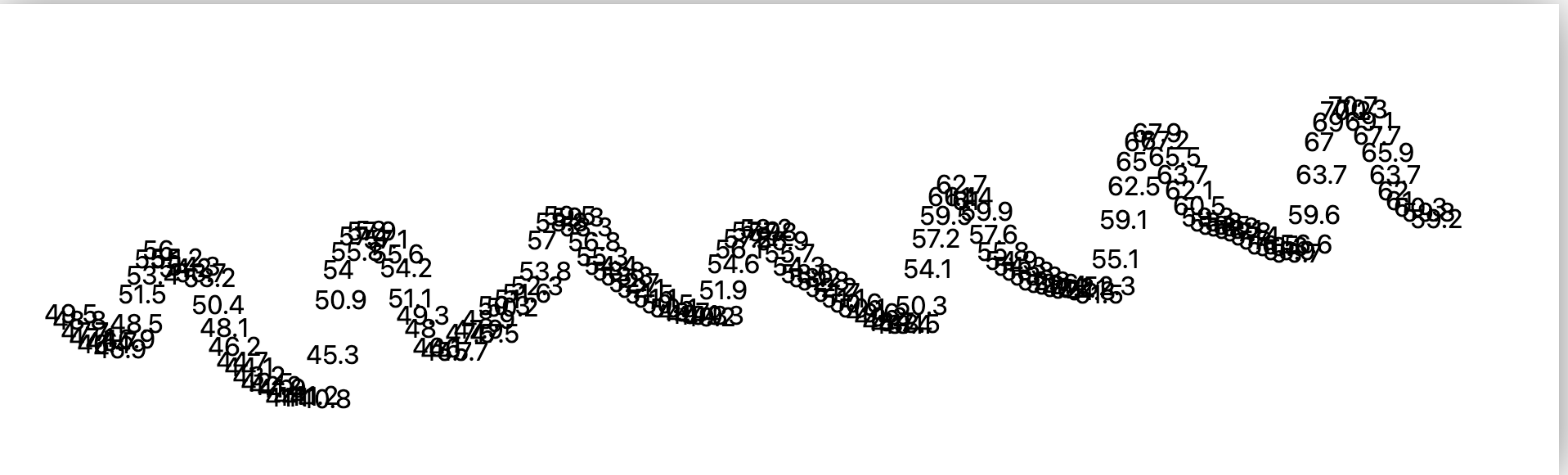
Set the x, y, and text content of each text element

What do the numbers 5, 6, and 500 mean?

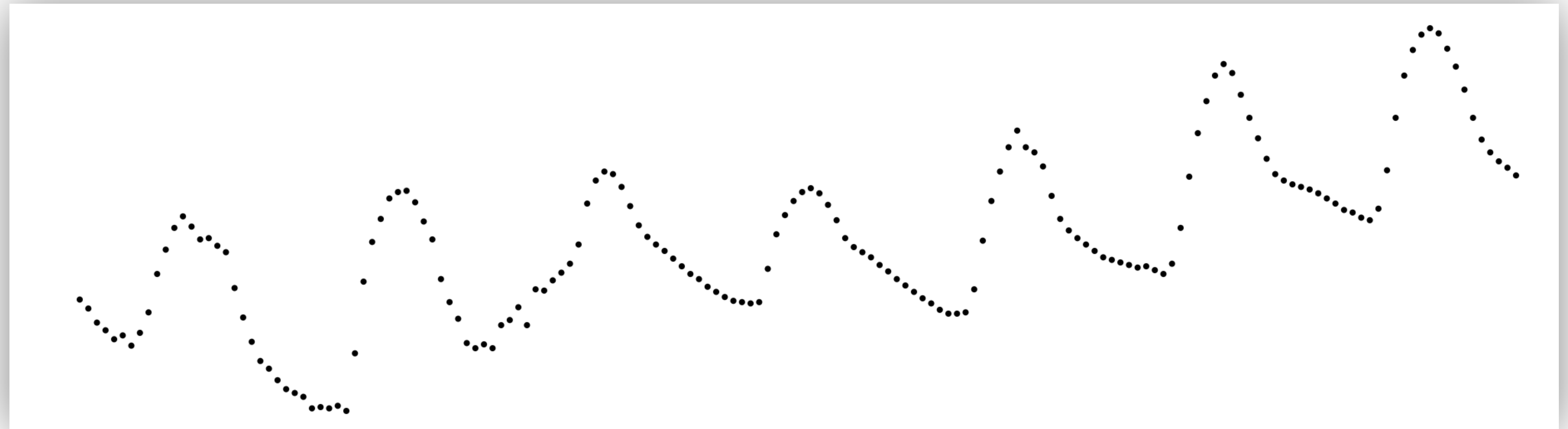
Nothing really, why not do that automatically?

Step 2: Making circles and using d3 scales

Before:



After:



Demo: `d3-lecture/weather02`

Making circles

Before:

```
svg
  .selectAll('text')
  .data(weatherData.hourly.temperature_2m)
  .join('text')
  .attr('x', (d, i) => i * 5)
  .attr('y', (d) => 500 - d * 6)
  .text((d) => d);
```

After:

```
svg
  .selectAll('circle')
  .data(weatherData.hourly.temperature_2m)
  .join('circle')
  .attr('cx', (d, i) => xScale(i))
  .attr('cy', (d) => yScale(d))
  .attr('r', 2);
```

Just needed to swap out text with circle + set the right attributes.

Scales

Before:

```
.attr('cx', (d, i) => i * 5)
.attr('cy', (d) => 500 - d * 6)
```

Magic numbers!

After:

```
.attr('cx', (d, i) => xScale(i))
.attr('cy', (d) => yScale(d))
```

D3 scales

```
const xScale = d3
  .scaleLinear()
  .domain([0, weatherData.hourly.temperature_2m.length - 1])
  .range([margin.left, width - margin.right]);
```

Domain = possible inputs

Range = possible outputs

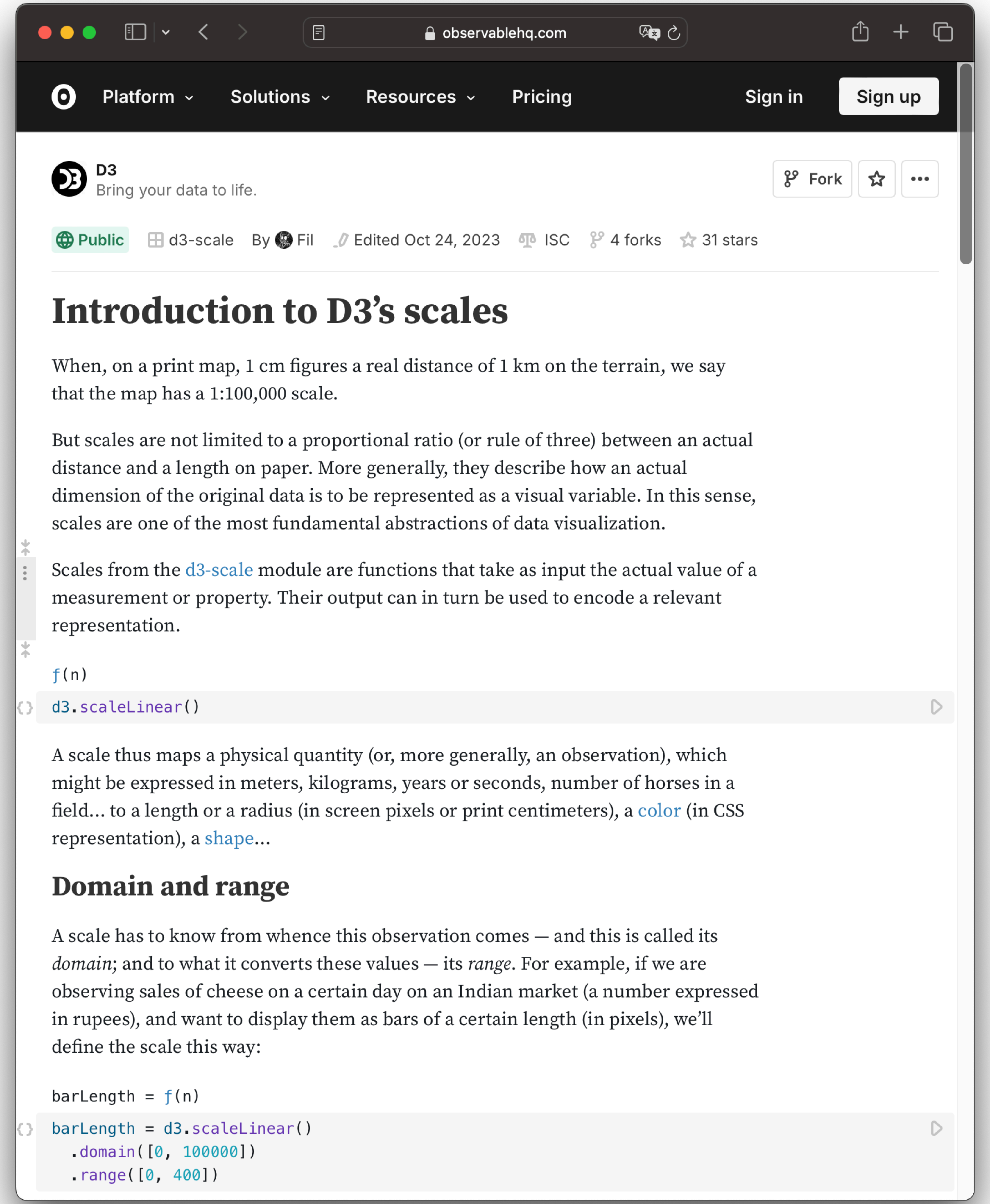
D3 scales will automatically make plot fit the space.

Scales

Input type	Output type	Example scales
Number	Number	`d3.scaleLinear([10, 130], [0, 960])` `d3.scaleLog([1, 10], [0, 960])`

Datetime	Number	`d3.scaleUtc([`
		`  new Date(“2000-01-01"), new Date(“2000-01-02”),`
		`], [0, 960]);`

Category	Category	`d3.scaleOrdinal(["a", "b", “c"],` `                [“red", "green", "blue"])`
Number	Color	`d3.scaleSequential([0, 100], d3.interpolateBlues)` `d3.scaleDiverging([-1, 0, 1], d3.interpolateRdBu)`
Number	Quantized color	`d3.scaleQuantize([0, 100], d3.schemeBlues[9])`


observablehq.com

Platform Solutions Resources Pricing Sign in Sign up

D3
Bring your data to life.

Fork

Public d3-scale By Fil Edited Oct 24, 2023 ISC 4 forks 31 stars

Introduction to D3's scales

When, on a print map, 1 cm figures a real distance of 1 km on the terrain, we say that the map has a 1:100,000 scale.

But scales are not limited to a proportional ratio (or rule of three) between an actual distance and a length on paper. More generally, they describe how an actual dimension of the original data is to be represented as a visual variable. In this sense, scales are one of the most fundamental abstractions of data visualization.

Scales from the d3-scale module are functions that take as input the actual value of a measurement or property. Their output can in turn be used to encode a relevant representation.

```
ƒ(n)
d3.scaleLinear()
```

A scale thus maps a physical quantity (or, more generally, an observation), which might be expressed in meters, kilograms, years or seconds, number of horses in a field… to a length or a radius (in screen pixels or print centimeters), a color (in CSS representation), a shape…

Domain and range

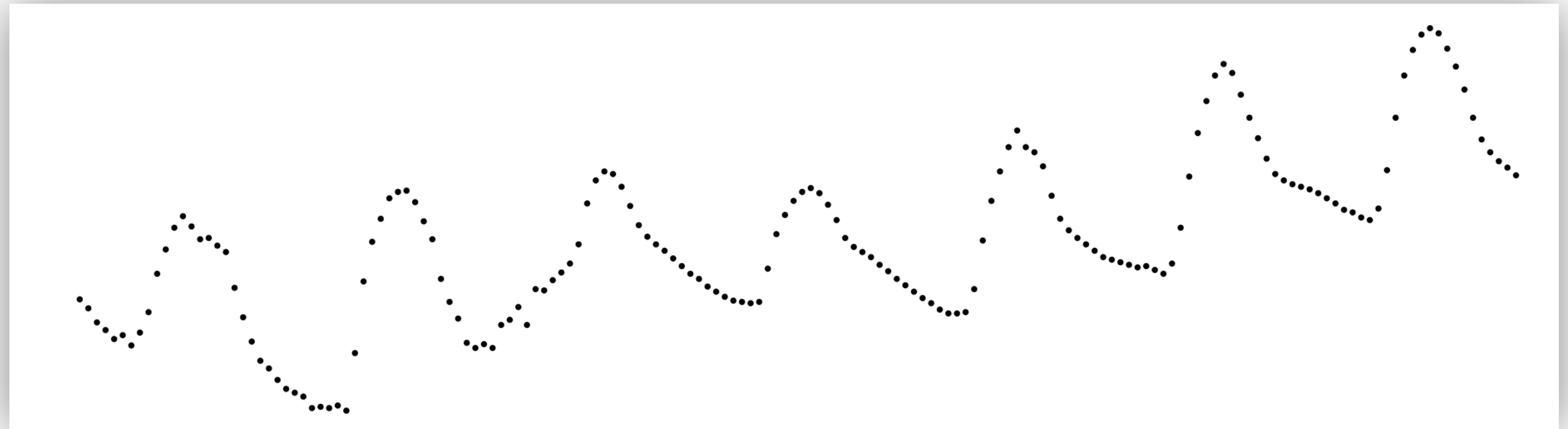
A scale has to know from whence this observation comes — and this is called its *domain*; and to what it converts these values — its *range*. For example, if we are observing sales of cheese on a certain day on an Indian market (a number expressed in rupees), and want to display them as bars of a certain length (in pixels), we'll define the scale this way:

```
barLength = ƒ(n)
barLength = d3.scaleLinear()
  .domain([0, 100000])
  .range([0, 400])
```

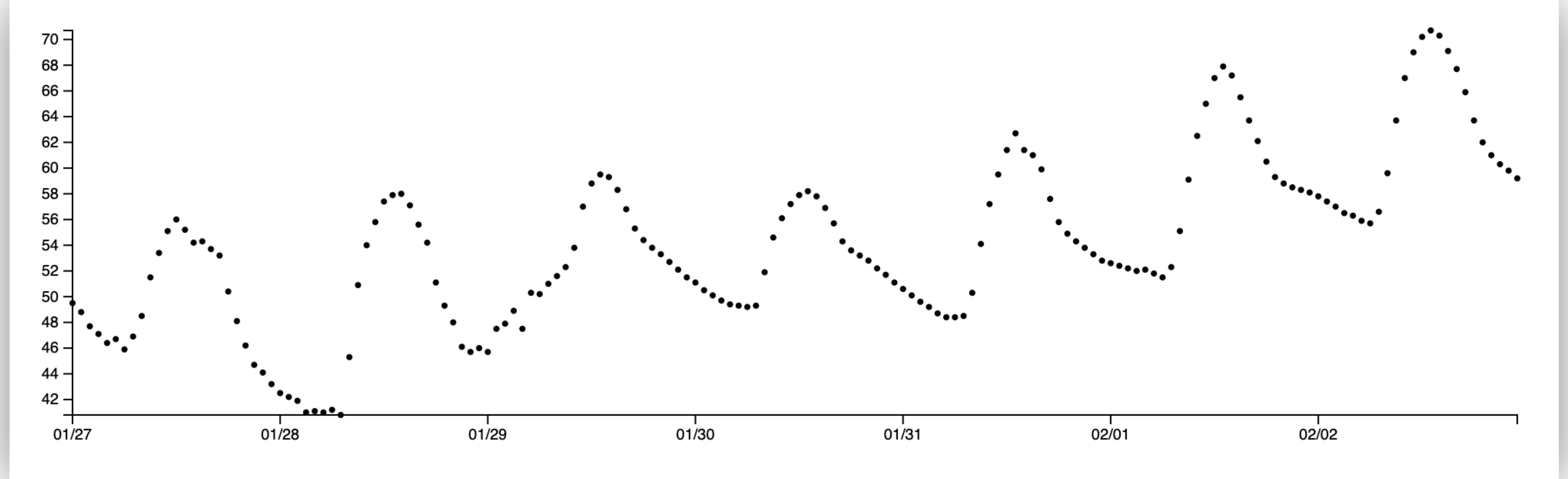
https://observablehq.com/@d3/introduction-to-d3s-scales?collection=@d3/d3-scale

Step 3: Adding axes

Before:



After:



Demo: `d3-lecture/weather03`

Axes

```
const yAxis = d3.axisLeft(yScale);
```

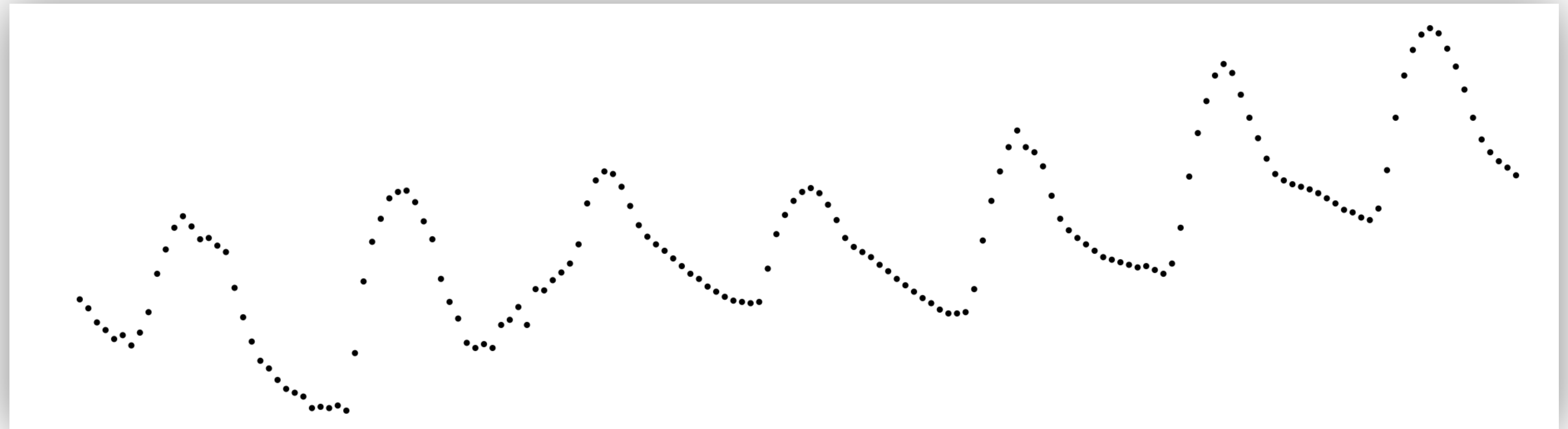
Creates a D3 axis object

```
svg
  .append('g')
  .attr('class', 'y axis')
  .attr('transform', `translate(${margin.left}, 0)`)
  .call(yAxis);
```

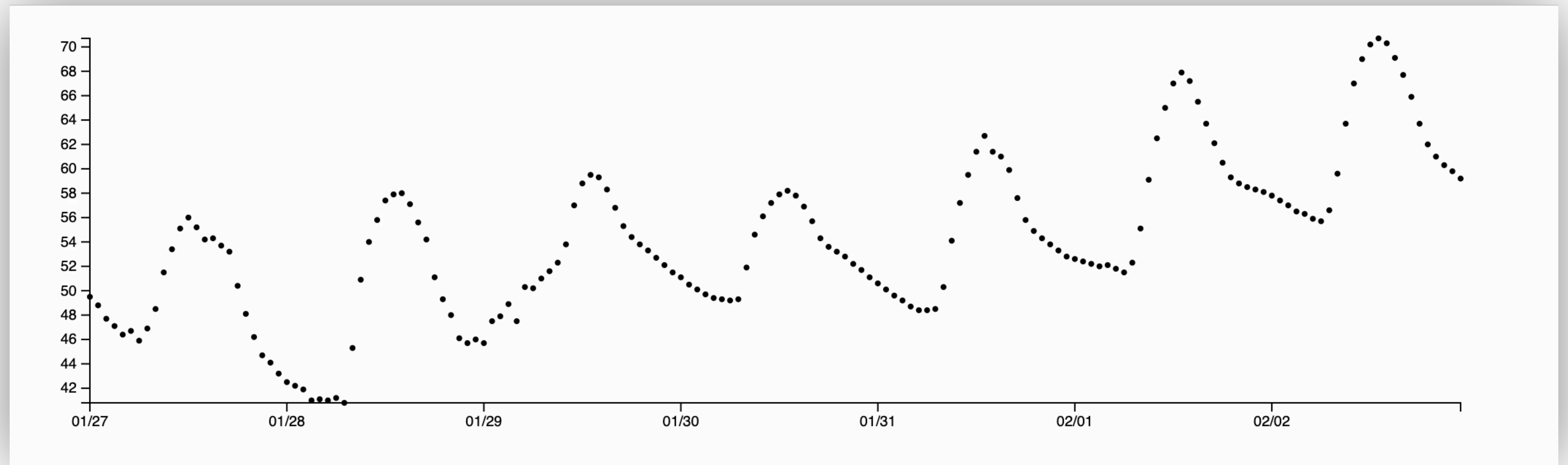
Creates an SVG <g> object, then draws axis into it

Step 4: Adding a basic tooltip

Before:



After:



Demo: `d3-lecture/weather04`

Making a tooltip

```
const tooltip = d3
  .select('body')
  .append('div')
  .attr('class', 'tooltip')
  .style('position', 'absolute')
  .style('visibility', 'hidden')
  .style('background-color', 'white')
  .style('border', '1px solid #ddd')
  .style('padding', '5px')
  .style('border-radius', '3px');
```

Creates a <div>, styles it, and hides it so that it’ll only show up with interaction

Adding interaction

```
.on('mouseover', function (event, d) {
  d3.select(this).attr('r', 4); // Increase circle size on hover

  tooltip.style('visibility', 'visible').text(`${d.toFixed(1)}°F`);
})
```

D3 version of event listener + handler

Adding interaction

```
.on('mouseover', function (event, d) {
                                                     cle size on hover

  tooltip.style('visibility', 'visible').text(`${d.toFixed(1)}°F`);
})
```

When a circle is moused over…

D3 version of event listener + handler

Adding interaction

```
.on('mouseover', function (event, d) {
  d3.select(this).attr('r', 4); // Increase circle size on hover
                                                        ${d.toFixed(1)}°F`);
})
```

Make the circle's radius larger

D3 version of event listener + handler

Adding interaction

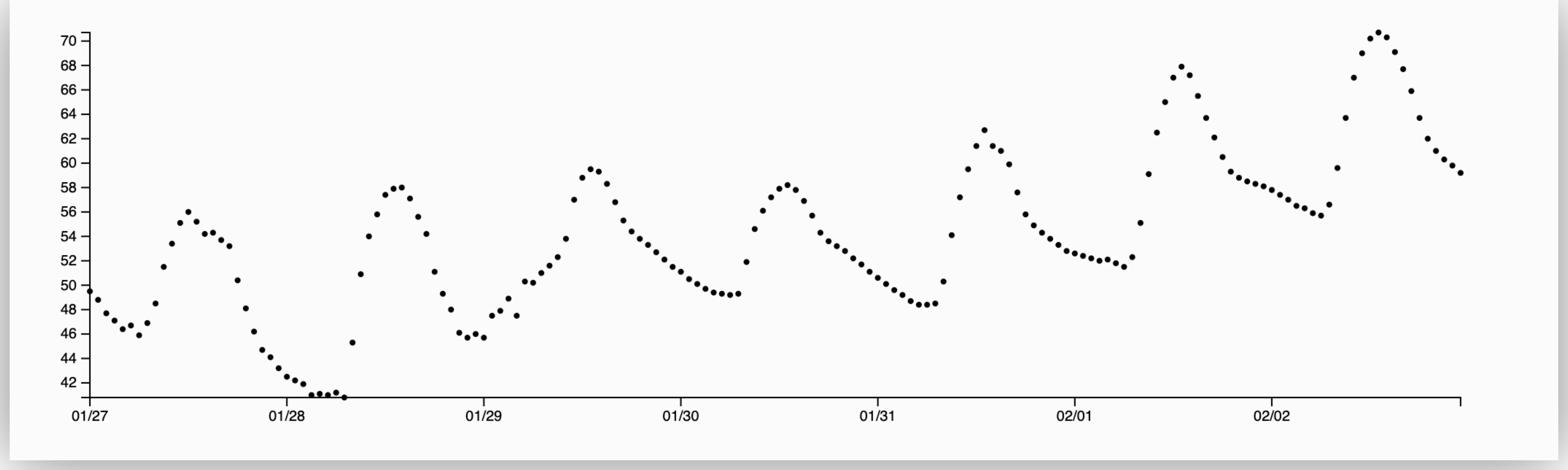
```
.on('mouseover', function (event, d) {
  d3.select(this).attr('r', 4); // Increase circle size on hover

  tooltip.style('visibility', 'visible').text(`${d.toFixed(1)}°F`);
})
```

Make tooltip visible and set its text

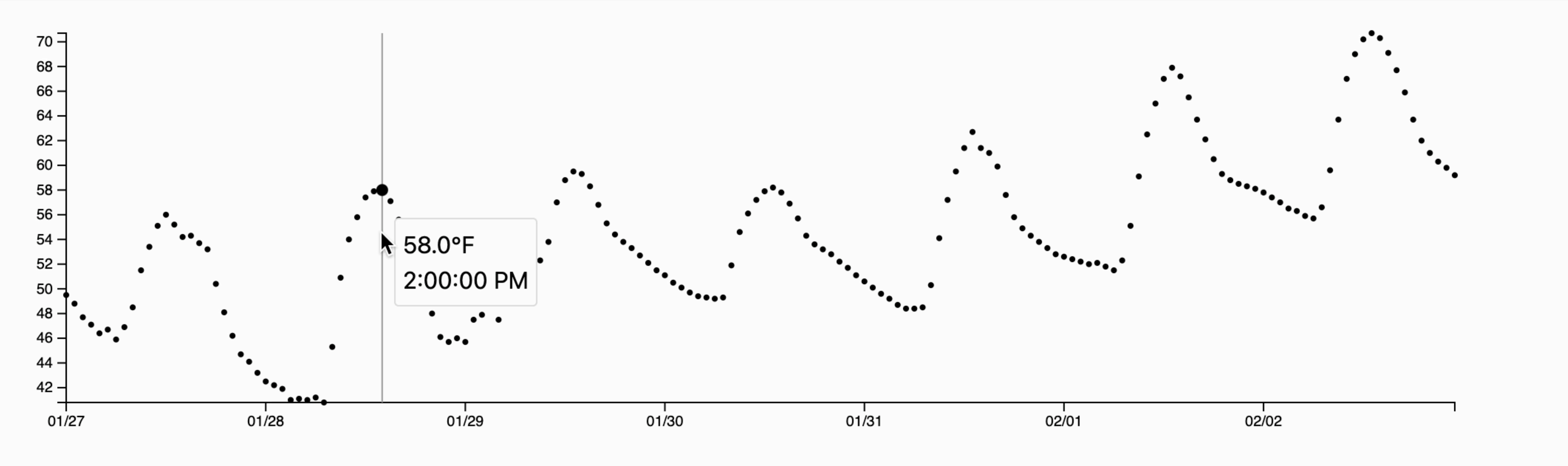
D3 version of event listener + handler

Step 5: Improving our tooltip

Before:



After:



Demo: `d3-lecture/weather05`

Interacting with the plot, not just points

```
// Create a rect overlay for mouse tracking
const overlay = svg
  .append('rect')
  .attr('class', 'overlay')
  .attr('x', margin.left)
  .attr('y', margin.top)
  .attr('width', width - margin.left - margin.right)
  .attr('height', height - margin.top - margin.bottom)
  .style('fill', 'none')
  .style('pointer-events', 'all');
```

Interaction trick:
Add an invisible rectangle just to capture mouse events

Listening for mouse events on the parent <svg> tag also ok

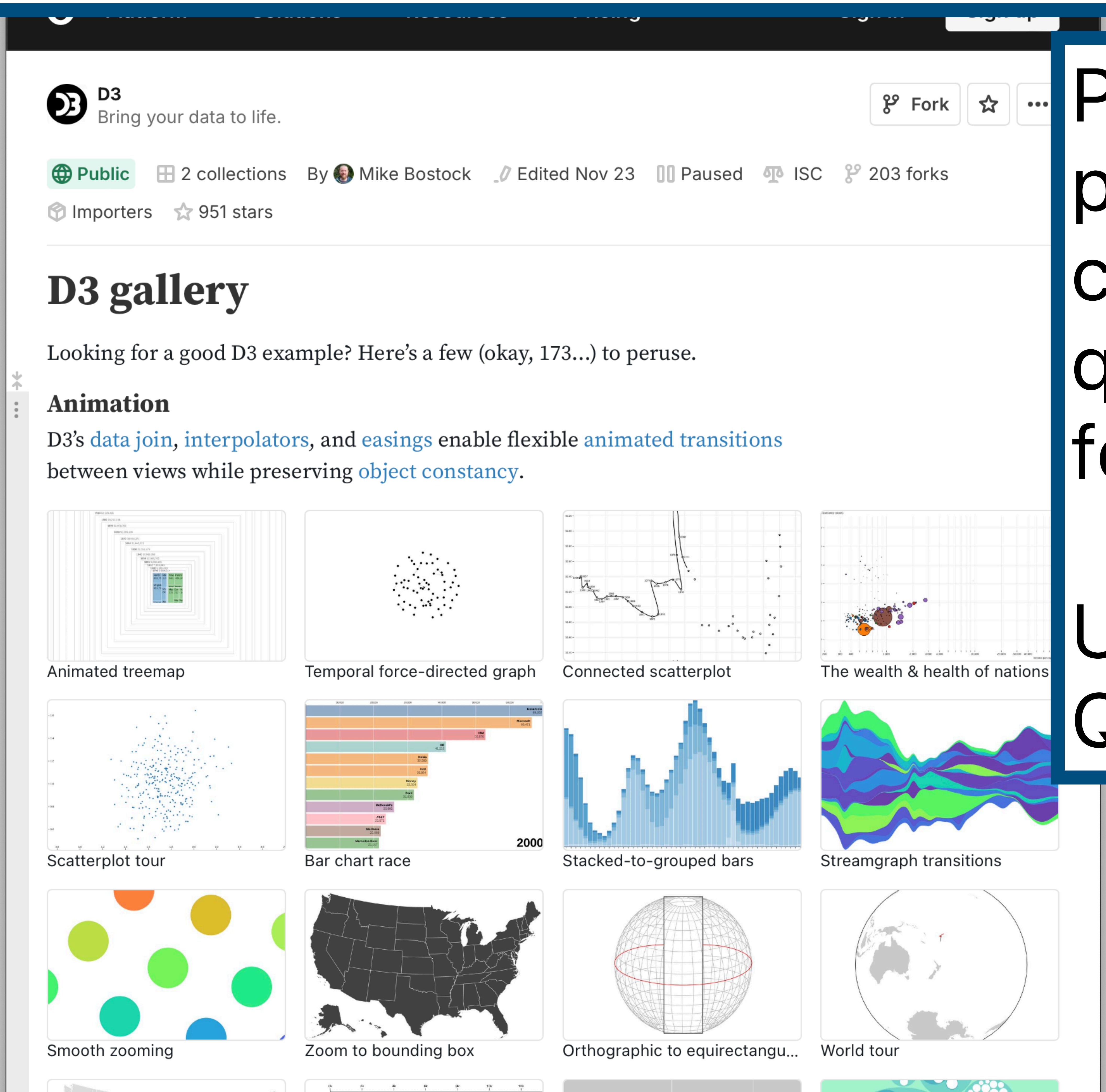
Improving interaction

```
.on('mousemove', function (event) {
  const mouseX = d3.pointer(event)[0];
  const xDate = xScale.invert(mouseX);

  // Find the closest data point
  const bisect = d3.bisector((d) => new Date(d)).left;
  const index = bisect(weatherData.hourly.time, xDate);
  const temp = weatherData.hourly.temperature_2m[index];
  const time = new Date(weatherData.hourly.time[index]);
```

Challenge: since we're not hovering directly over points, we have to use the mouse position to find nearest point

You Try: Explain D3 code

https://observablehq.com/@d3/gallery



Pick a simple visualization (scatter plot, line plot, bar chart). Explain the code to your neighbor, then write a question about the code using this format:

URL: ...
Question: ...

tryclassbuzz.com
Code: **explain-d3**