

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

D3.js (Part 1)

DSC 106: Data Visualization

Sam Lau

UC San Diego

Announcements

Lab 4 due Friday.

Project 2 during Tuesday.

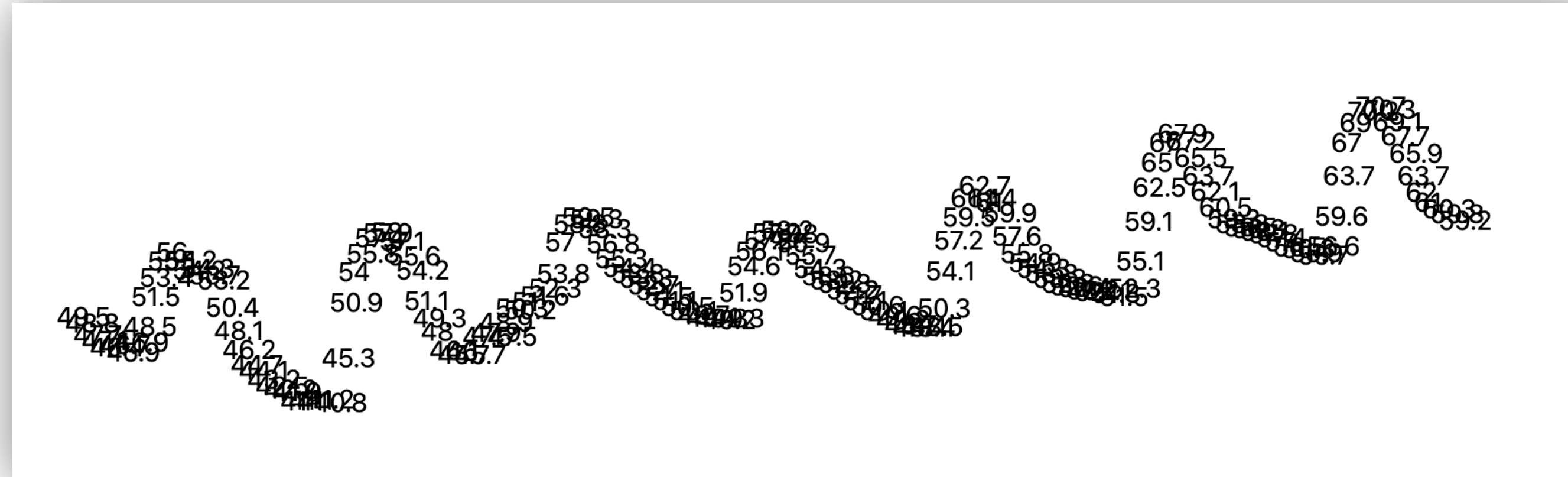
FAQs:

1. How are things going with the course?
Feedback on the Canvas videos?
tryclassbuzz.com

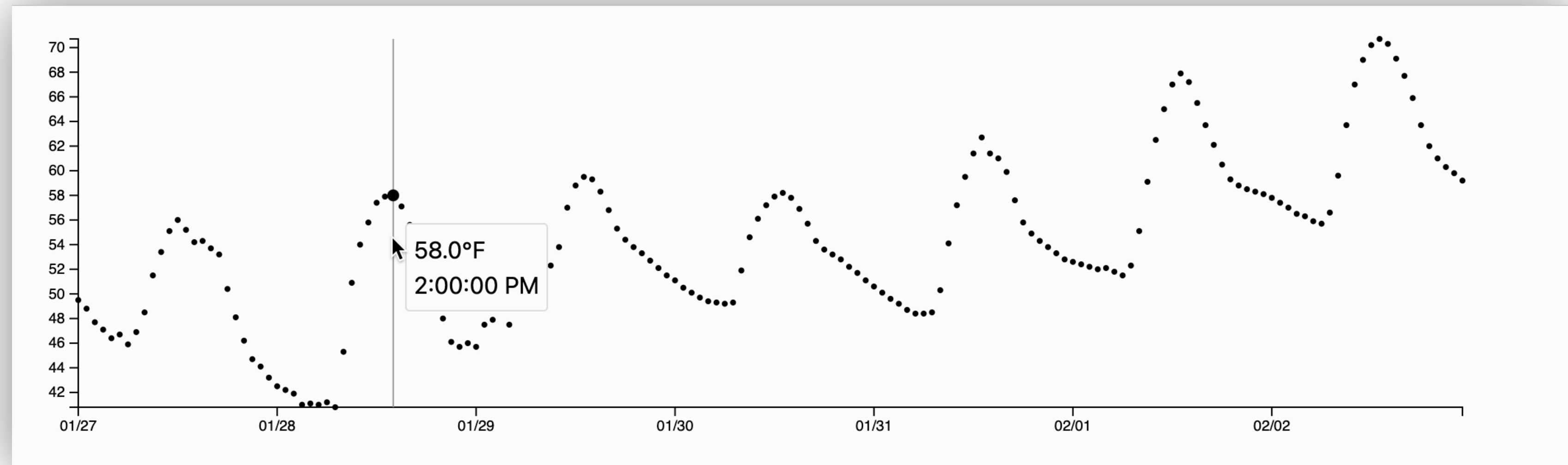
D3

Today: Making an interactive scatterplot

Before:

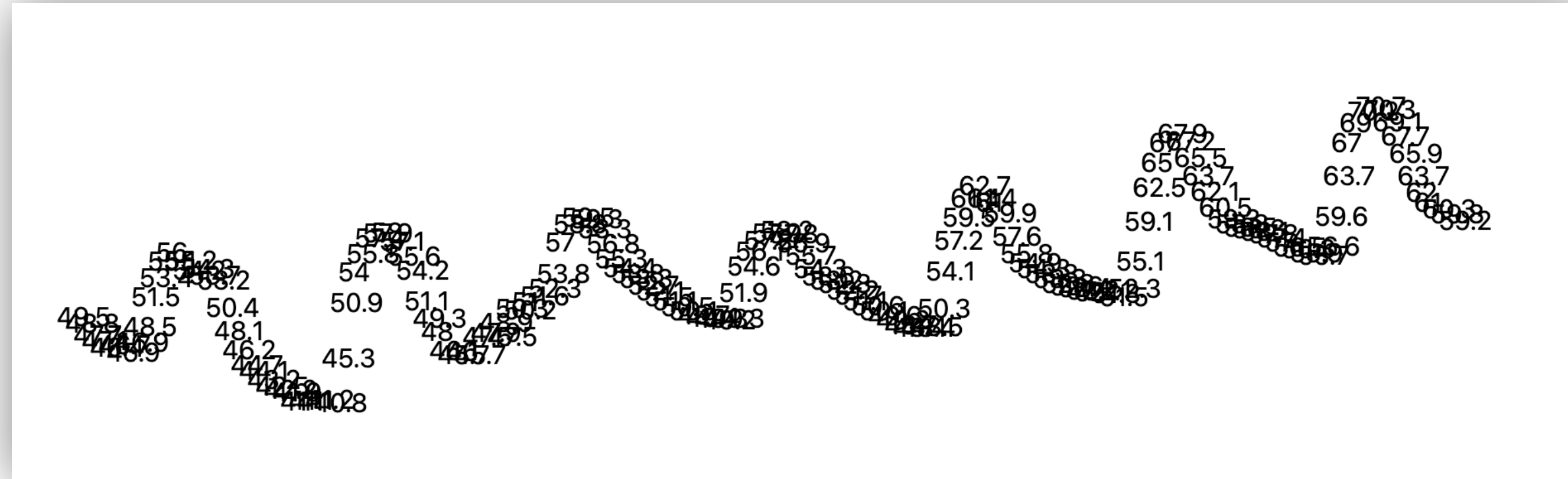


After:

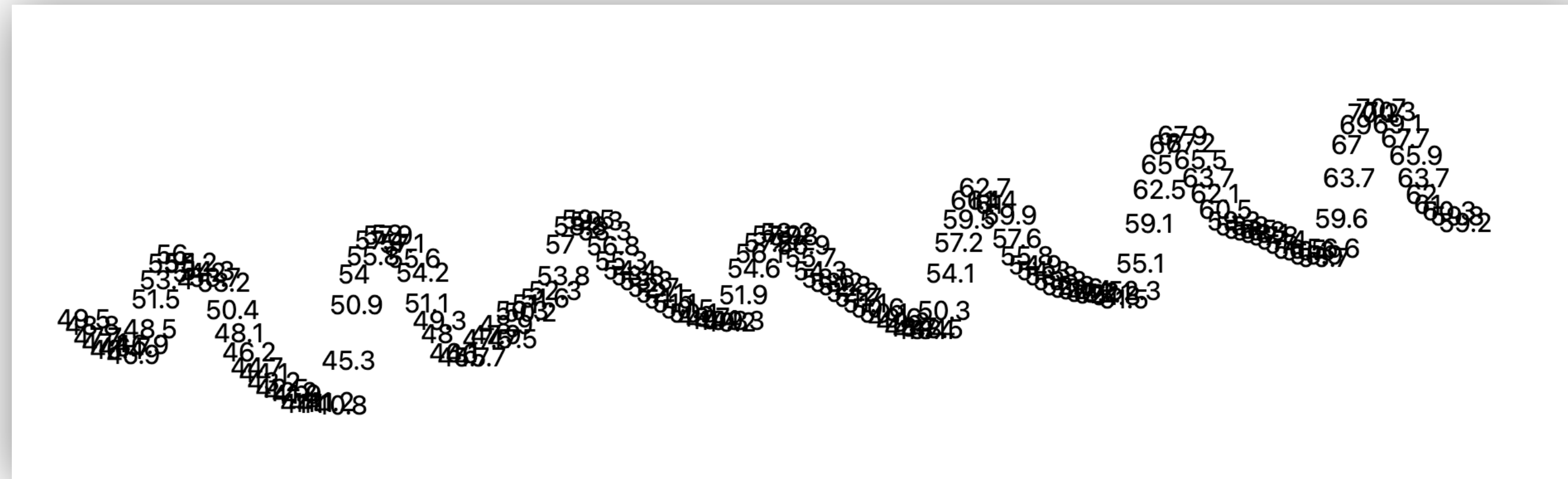


Step 1: Using D3 instead of plain JS

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');
```

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

HTML element method

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

After:

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

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

D3 equivalent

But ALWAYS know when you
have a D3 vs. native object!

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

No explicit for loop, but there is one internally!

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

What's not intuitive: You use `selectAll()` for elements that you want to **CREATE**, but it sounds like you're looking for text elements that already exist.

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

Does more than just create new elements, but good enough analogy for now.

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

Notice that each `.attr()` method takes in a **function**. Function gets called with 2 arguments: actual data element and index.

(Actually, called with 3 arguments. But JS functions ignore extra arguments!)

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?

What happens if we make the magic numbers in the d3 code bigger (5, 6, 500)? Your answer should look like:

5: {your response}

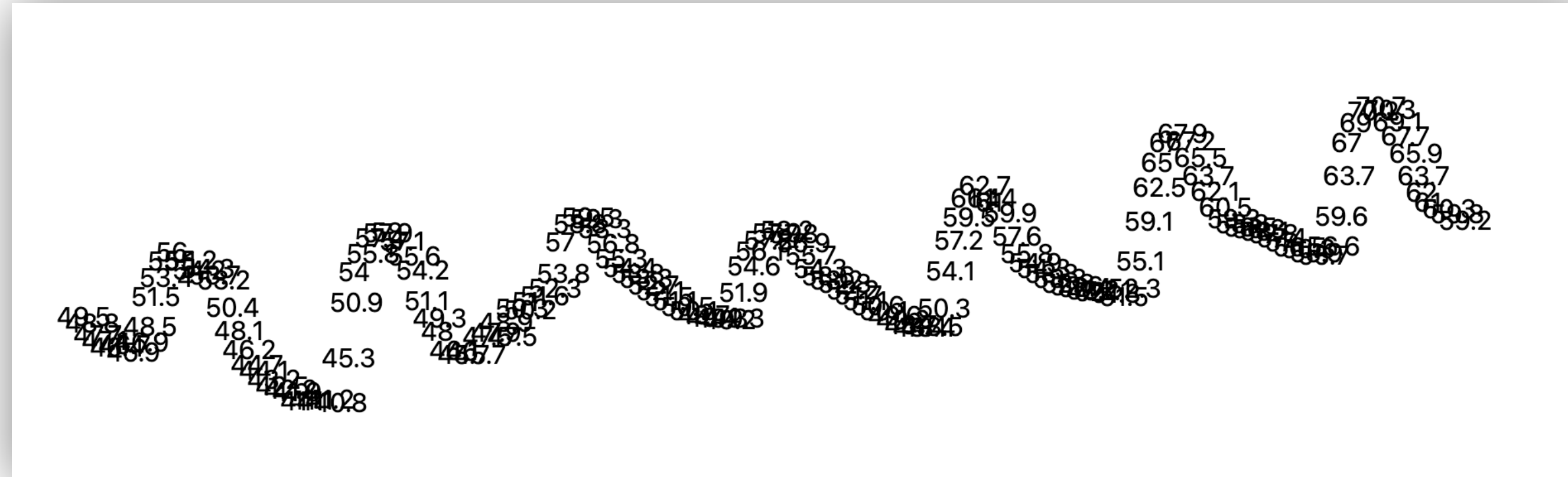
6: {your response}

500: {your response}

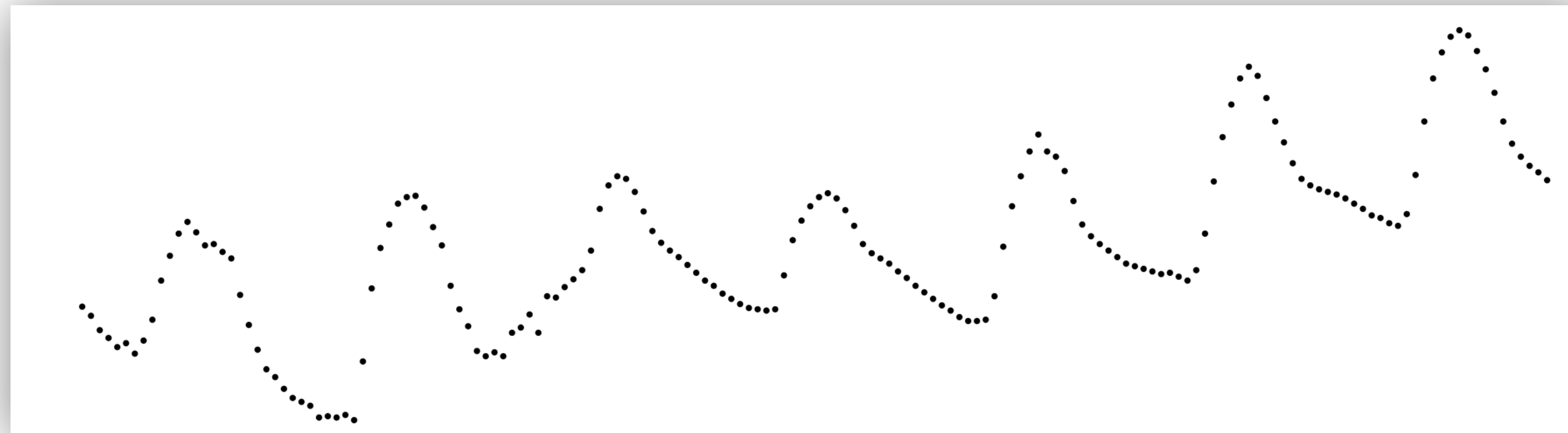
tryclassbuzz.com

Step 2: Making circles and using d3 scales

Before:



After:



Demo: [d3-lecture/weather02](https://d3js.org/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);
```

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

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);
```

Circles only have cx and cy, not x and y

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

Let's work out how a scale works by hand.

	time	temperature_2m
0	2025-04-24T00:00	55.6
1	2025-04-24T01:00	55.6
2	2025-04-24T02:00	55.2
3	2025-04-24T03:00	55.9
4	2025-04-24T04:00	56.7
...
163	2025-04-30T19:00	63.4
164	2025-04-30T20:00	61.8
165	2025-04-30T21:00	61.0
166	2025-04-30T22:00	60.9
167	2025-04-30T23:00	60.7



1000px wide

Scales

Let's work out how a scale works by hand.

	time	temperature_2m
0	2025-04-24T00:00	55.6
1	2025-04-24T01:00	55.6
2	2025-04-24T02:00	55.2
3	2025-04-24T03:00	55.9
4	2025-04-24T04:00	56.7
...
163	2025-04-30T19:00	63.4
164	2025-04-30T20:00	61.8
165	2025-04-30T21:00	61.0
166	2025-04-30T22:00	60.9
167	2025-04-30T23:00	60.7



Index=0 → x=0

1000px wide

Scales

Let's work out how a scale works by hand.

	time	temperature_2m
0	2025-04-24T00:00	55.6
1	2025-04-24T01:00	55.6
2	2025-04-24T02:00	55.2
3	2025-04-24T03:00	55.9
4	2025-04-24T04:00	56.7
...
163	2025-04-30T19:00	63.4
164	2025-04-30T20:00	61.8
165	2025-04-30T21:00	61.0
166	2025-04-30T22:00	60.9
167	2025-04-30T23:00	60.7

Index=167 → x=1000

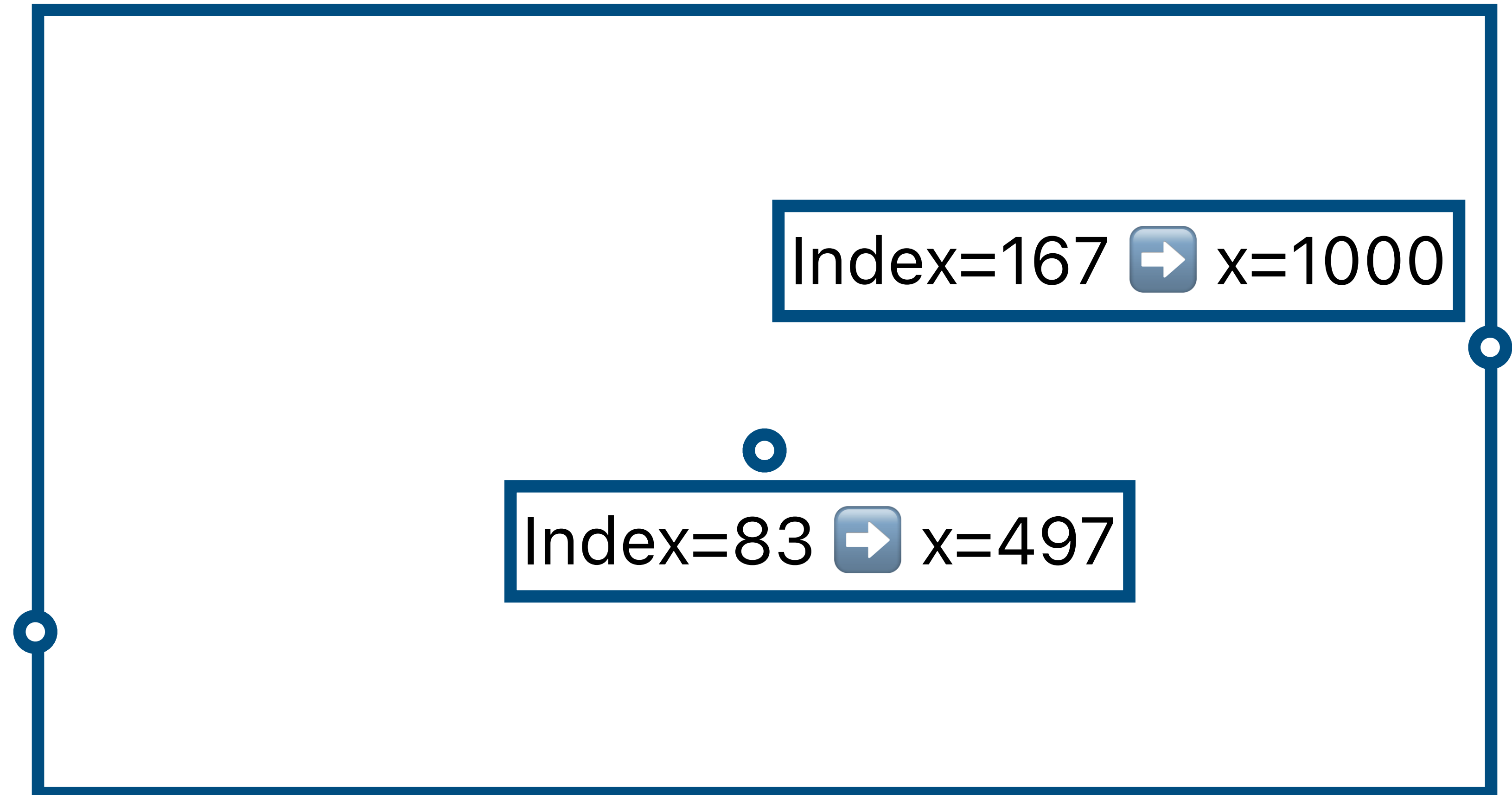
Index=0 → x=0

1000px wide

Scales

Let's work out how a scale works by hand.

	time	temperature_2m
0	2025-04-24T00:00	55.6
1	2025-04-24T01:00	55.6
2	2025-04-24T02:00	55.2
3	2025-04-24T03:00	55.9
4	2025-04-24T04:00	56.7
...
163	2025-04-30T19:00	63.4
164	2025-04-30T20:00	61.8
165	2025-04-30T21:00	61.0
166	2025-04-30T22:00	60.9
167	2025-04-30T23:00	60.7



Index=0 → x=0

Index=83 → x=497

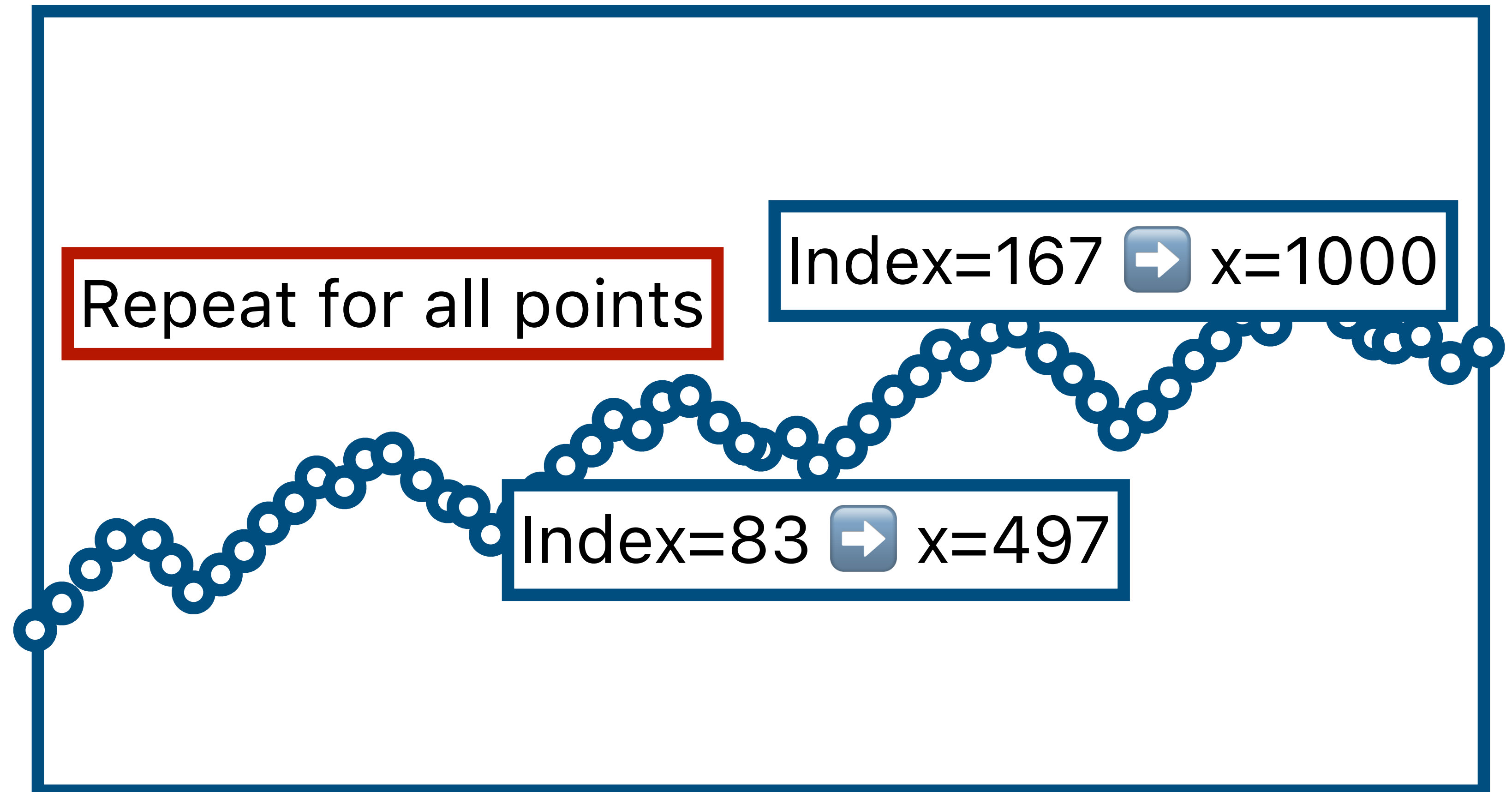
Index=167 → x=1000

1000px wide

Scales

Let's work out how a scale works by hand.

	time	temperature_2m
0	2025-04-24T00:00	55.6
1	2025-04-24T01:00	55.6
2	2025-04-24T02:00	55.2
3	2025-04-24T03:00	55.9
4	2025-04-24T04:00	56.7
...
163	2025-04-30T19:00	63.4
164	2025-04-30T20:00	61.8
165	2025-04-30T21:00	61.0
166	2025-04-30T22:00	60.9
167	2025-04-30T23:00	60.7



Index=0 → x=0

1000px wide

Scales

Let's work out how a scale works by hand.

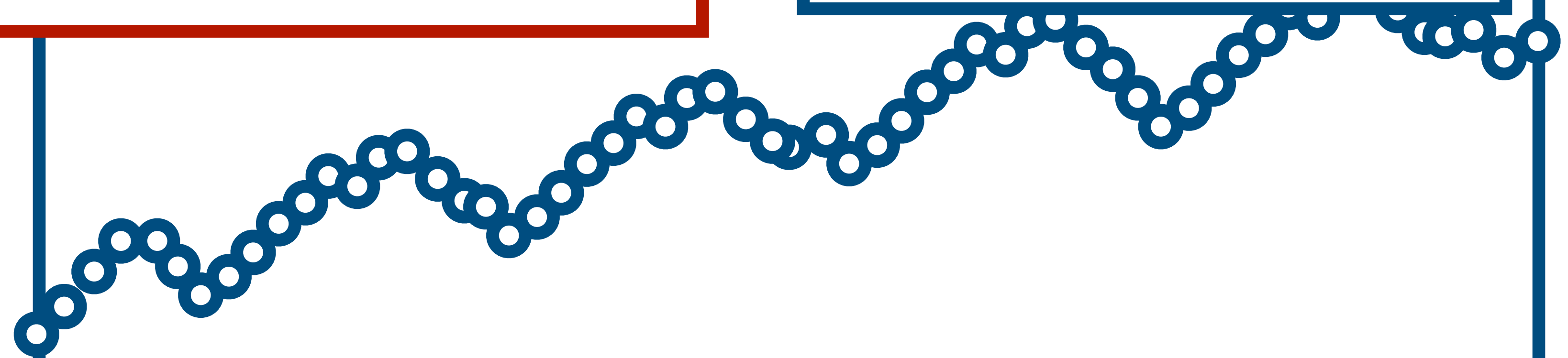
Want a function that converts between:

Input: index \rightarrow Output: x-coordinate

Index=167 \rightarrow x=1000

Index=0 \rightarrow x=0

1000px wide



Scales

Let's work out how a scale works by hand.

Want a function that converts between:

Input: index → Output: x-coordinate

Index=167 → x=1000

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

Domain = possible inputs

Range = possible outputs

Include margin for axes

Index=0 → x=0

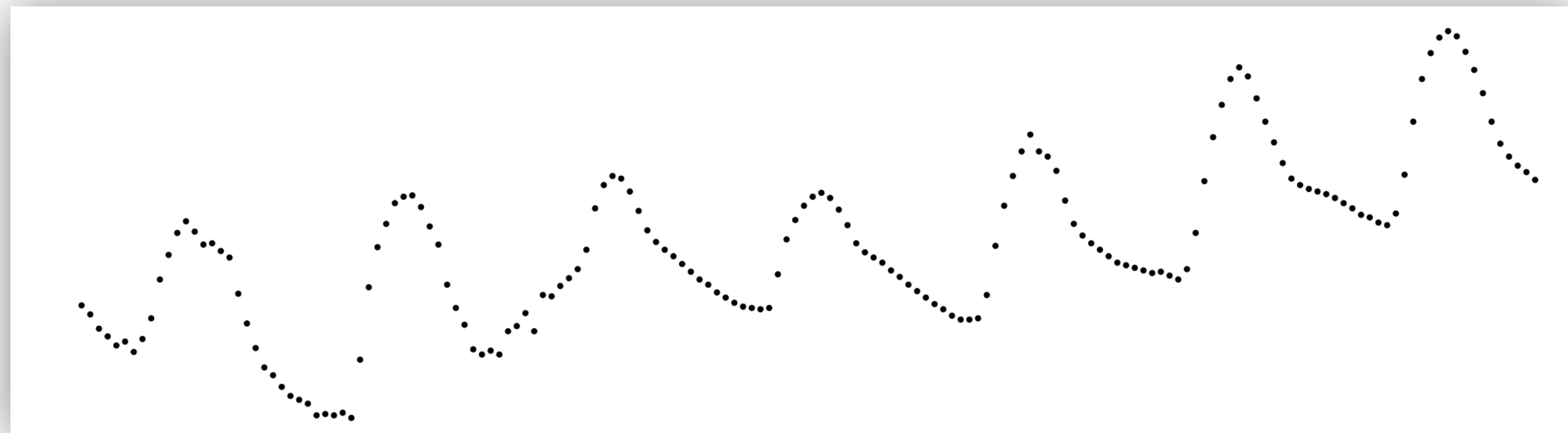
1000px wide

What happens if we swap the arguments for `.range()`?

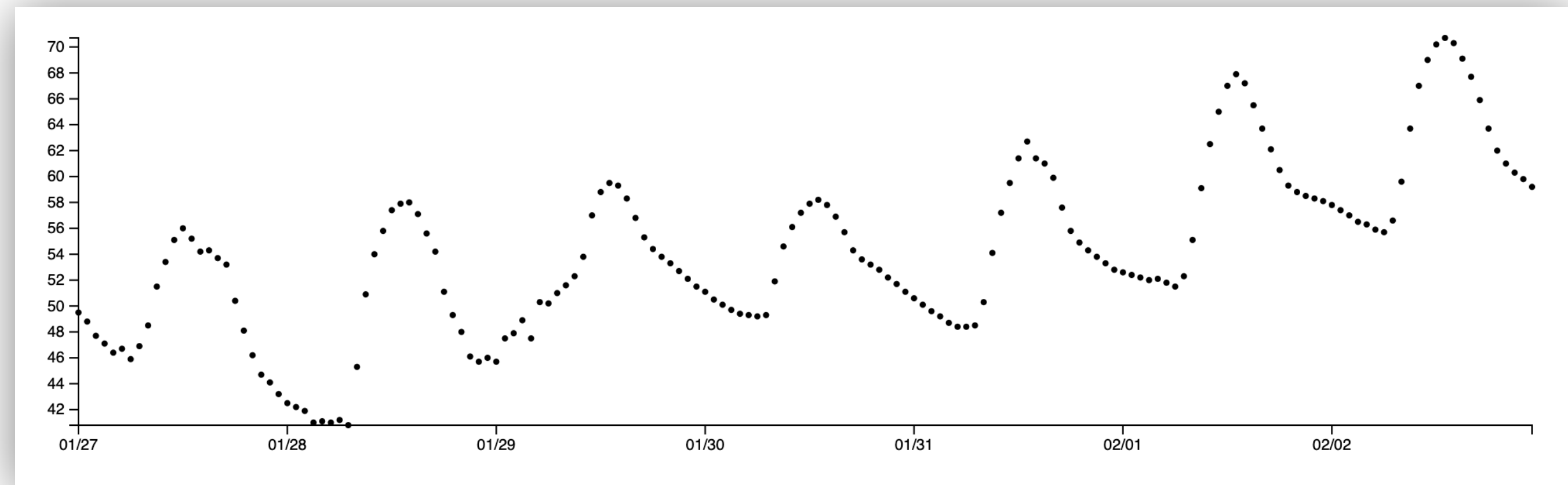
tryclassbuzz.com

Step 3: Adding axes

Before:



After:



Demo: [d3-lecture/weather03](#)

Using a Time Scale

Old:

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

scaleLinear: number input

New:

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

scaleTime: Date() input

Using a scaleTime lets us get date labels on the x-axis for free!

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);
```

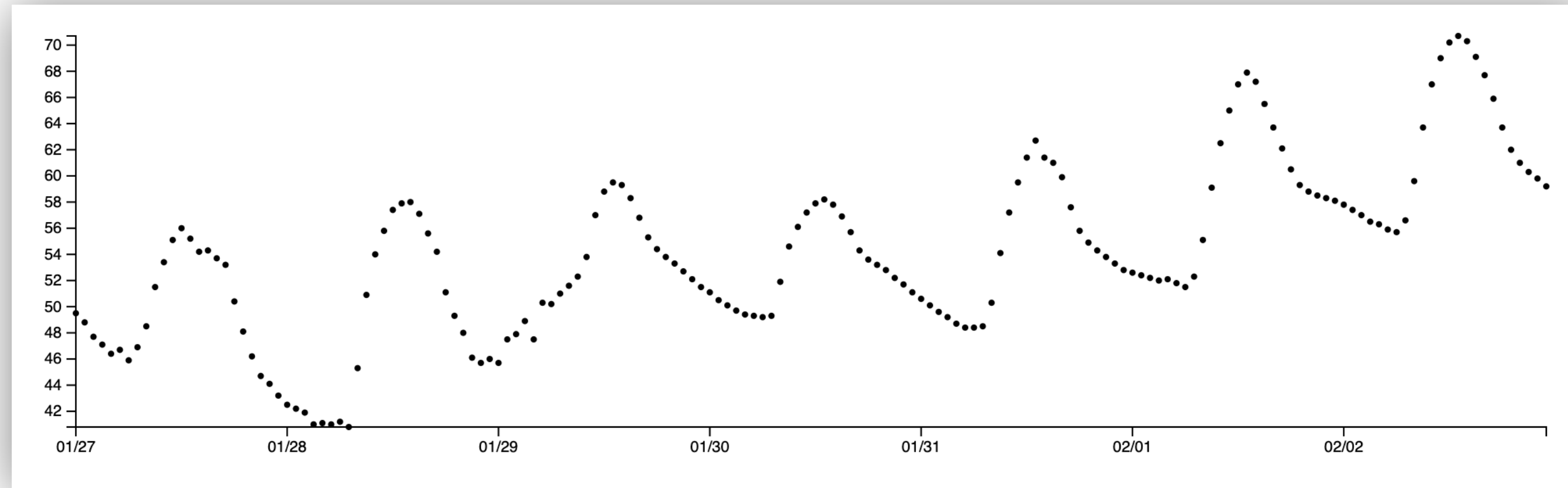
Creates an SVG <g> object, then draws axis into it

What would happen if we remove the `.class()` line in the previous code?

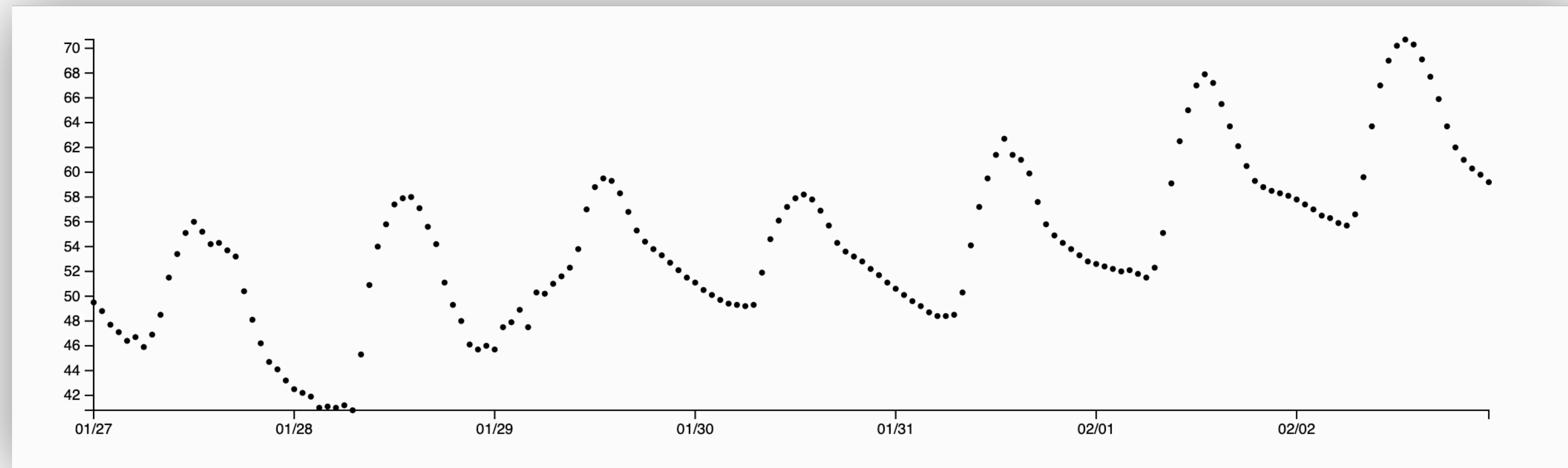
tryclassbuzz.com

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) {  
  When a circle is moused over... 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) {  
  d3.select(this).attr('r', 4); // Increase circle size on hover  
  Make the circle's radius larger  
  d.attr('r', d3.select(this).attr('r') * 2);  
  d.attr('stroke-width', d3.select(this).attr('stroke-width') * 2);  
  d.attr('stroke', `#${d.toFixed(1)}°F`);  
})
```

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

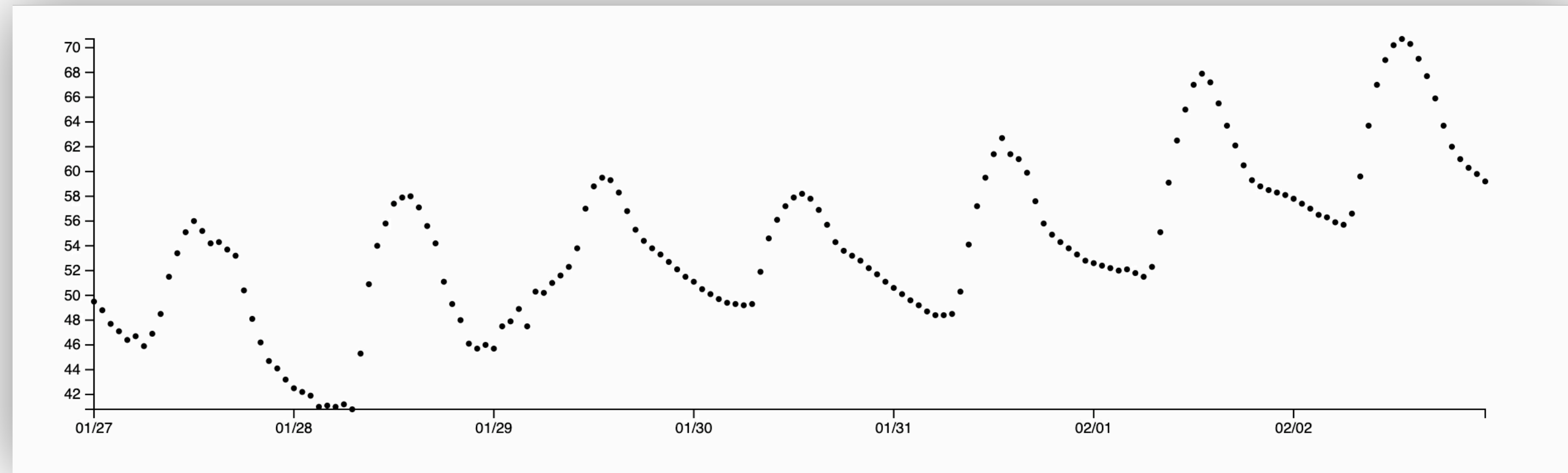
D3 version of event listener + handler

Why is our tooltip not ideal?

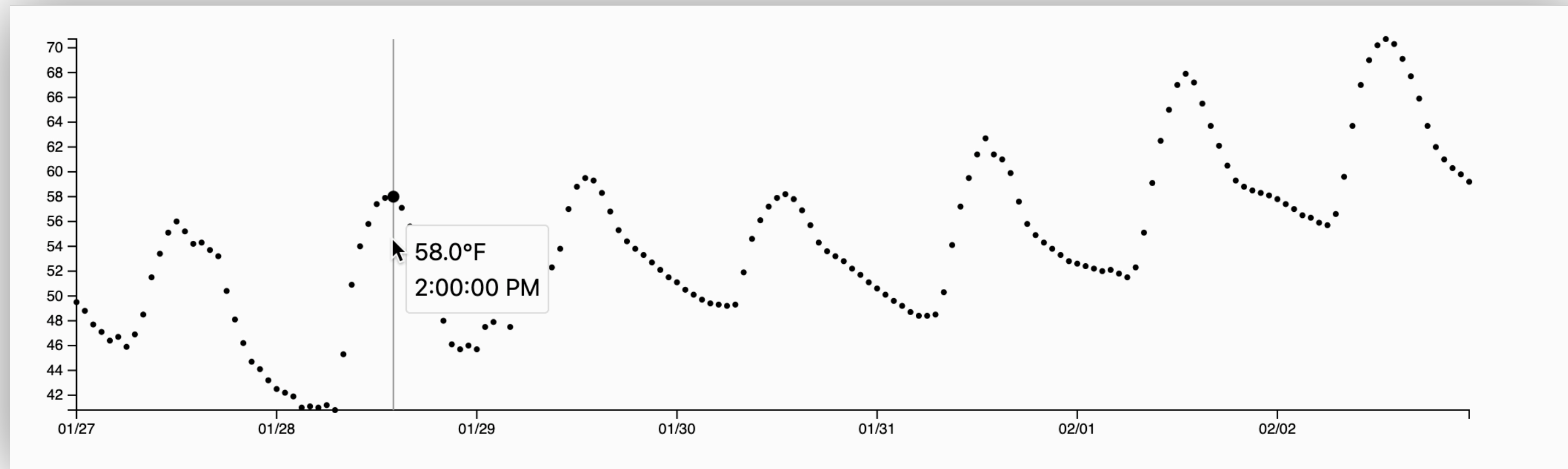
tryclassbuzz.com

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

Submit a question about Step 5

tryclassbuzz.com

You Try: Explain D3 code

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

D3
Bring your data to life.

Public 2 collections By Mike Bostock Edited Nov 23 Paused ISC 203 forks
Importers 951 stars

D3 gallery

Looking for a good D3 example? Here's a few (okay, 173...) to peruse.

Animation

D3's [data join](#), [interpolators](#), and [easings](#) enable flexible [animated transitions](#) between views while preserving [object constancy](#).

Animated treemap Temporal force-directed graph Connected scatterplot The wealth & health of nations
Scatterplot tour Bar chart race Stacked-to-grouped bars Streamgraph transitions
Smooth zooming Zoom to bounding box Orthographic to equirectangu... World tour

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