

Reflection: It has helped me fix the errors related to legends placement, value placement and also helped with the boilerplate code of the problem which I corrected based on my need.

Question: The above code don't actually represent the legend in the side of the visualization. Can you add that in the above code?

Looking at the reference image, the legend is a **vertical gradient bar** placed to the right of the matrix, with:

- "0 Celsius" label at the top (cool/blue end)
- "40 Celsius" label at the bottom (hot/dark red end)
- A continuous color gradient bar in between

Here is the updated `buildLegend()` function and the required layout changes:

Changes Needed in `index.html`

Replace the `#legend` div and update the main layout to a side-by-side flex container:

```
xml
<!-- REMOVE the old #legend div and replace with this wrapper -->
<div id="viz-wrapper" style="display: flex; align-items: flex-start; justify-content: center;">
  <div id="chart"></div>
  <div id="legend-container" style="margin-left: 16px; margin-top: 40px;"></div>
</div>

<!-- Keep the tooltip div as-is -->
<div class="tooltip" id="tooltip"></div>
```

Also remove `<div id="legend"></div>` — it's no longer needed.

Updated `buildLegend()` in `main.js`

Replace the entire old `buildLegend()` function with this:

```
js
//-----
// SECTION 9 — VERTICAL LEGEND (RIGHT SIDE)
// Draws a vertical gradient bar to the right of
// the matrix, exactly like the reference image.
// "0 Celsius" sits at the top (cool end) and
```

```

// "40 Celsius" at the bottom (hot end).
// Re-built whenever mode changes to reflect the
// updated color domain.
//-----
function buildLegend(colorScale) {
  const container = d3.select("#legend-container");
  container.selectAll("*").remove(); // clear on re-render

  const LH  = 300; // total height of the gradient bar
  const LW  = 20; // width of the gradient bar
  const steps = 200; // number of gradient slices (smoother = more steps)
  const [lo, hi] = colorScale.domain();

  const legendSvg = container
    .append("svg")
    .attr("width", LW + 60) // extra space for labels
    .attr("height", LH + 40); // extra space for top/bottom labels

  //— Define a <linearGradient> in <defs> ———
  // Using an SVG gradient is cleaner than stacking
  // many tiny rects, and looks identical to the
  // reference. The gradient runs top → bottom
  // (cold → hot, matching the inverted color scale).
  const defs = legendSvg.append("defs");
  const gradientId = "legend-gradient";

  const linearGrad = defs.append("linearGradient")
    .attr("id", gradientId)
    .attr("x1", "0%").attr("y1", "0%") // top
    .attr("x2", "0%").attr("y2", "100%"); // bottom

  // Add color stops evenly across the gradient
  // We map t=0 (top) → lo temperature (cool color)
  // and t=1 (bottom) → hi temperature (hot color)
  d3.range(steps + 1).forEach(i => {
    const t = i / steps;
    const val = lo + t * (hi - lo);
    linearGrad.append("stop")
      .attr("offset", `${t * 100}%`)
      .attr("stop-color", colorScale(val));
  });

  //— Draw the gradient rectangle ———
  legendSvg.append("rect")
    .attr("x", 0)
    .attr("y", 20) // offset from top to leave room for top label
    .attr("width", LW)
    .attr("height", LH)
    .style("fill", `url(${gradientId})`);

  //— Top label: cool end ———
  // Shows the minimum temperature value (e.g. "0 Celsius")
  legendSvg.append("text")
    .attr("x", LW + 6)
    .attr("y", 20 + 6) // vertically aligned with top of bar

```

```

.attr("font-size", "11px")
.attr("dominant-baseline", "middle")
.text(`${Math.round(lo)} Celsius`);

//—— Bottom label: hot end ————
// Shows the maximum temperature value (e.g. "40 Celsius")
legendSvg.append("text")
  .attr("x", LW + 6)
  .attr("y", 20 + LH) // aligned with bottom of bar
  .attr("font-size", "11px")
  .attr("dominant-baseline", "middle")
  .text(`${Math.round(hi)} Celsius`);

//—— Optional: add a scale axis on the left ————
// Uncomment this block if you want tick marks
// like a proper axis (e.g. 0, 10, 20, 30, 40)
/*
const legendScale = d3.scaleLinear()
  .domain([lo, hi])
  .range([20, 20 + LH]);

legendSvg.append("g")
  .attr("transform", `translate(${LW}, 0)`)
  .call(
    d3.axisRight(legendScale)
      .ticks(5)
      .tickFormat(d => `${d}°C`)
  );
*/
}

```

Update the `render()` call site

In `render()`, change the two `buildLegend()` calls to pass the correct target:

```

js
// Inside render(), near the bottom — replace old calls:
buildLegend(colorScale); // initial

// Inside the toggle button handler:
d3.select("#toggle-btn").on("click", () => {
  mode = mode === "max" ? "min" : "max";
  d3.select("#toggle-btn")
    .text(`Showing: ${mode.toUpperCase()} Temperature (click to switch}`);
  updateColors(cellGroups, monthlyData, colorScale, years);
  buildLegend(colorScale); // ← refreshes vertical legend on toggle
});

```

What Changed and Why

Part	What it does
display: flex wrapper	Places #chart and #legend-container side-by-side, matching the reference layout
<linearGradient> in <defs>	SVG-native gradient — smoother than stacking 100 tiny rectangles and matches the smooth bar in the reference image
y1=0% → y2=100%	Makes the gradient flow top to bottom (cool → hot)
Top/bottom text labels	Replicate "0 Celsius" at top and "40 Celsius" at bottom from the reference
container.selectAll("*").remove()	Clears and redraws legend when mode toggles, keeping the domain in sync
Optional axis block	Uncomment if you want intermediate tick marks (5, 10, 15...°C) along the bar

Question: There are few issues in my current code. First of all, the legend is supposed to change (color scheme) after we change between maximum and minimum temperature. Rectify this issue and give me the solution for that please. Tell me how can i do that.

The issue is straightforward: when you toggle between max and min, `updateColors()` updates the `colorScale.domain()` correctly, but the legend's `<linearGradient>` color stops are built **once** using the old domain and the SVG `<defs>` element caches the old gradient by `id`. Even though `buildLegend()` is called again, the `gradientId` is the same string "`legend-gradient`" every time, so the browser reuses the stale gradient.

There are **two fixes** needed:

Fix 1 — Give each rebuilt gradient a unique ID

This is the root cause. Change `buildLegend()` to generate a fresh gradient ID on every call so the browser never reuses a stale one.

```
js
function buildLegend(colorScale) {
  const container = d3.select("#legend-container");
  container.selectAll("*").remove();

  const LH = 300;
  const LW = 20;
  const steps = 200;
  const [lo, hi] = colorScale.domain();
```

```

const legendSvg = container
.append("svg")
.attr("width", LW + 80)
.attr("height", LH + 50);

// — FIX: use a unique gradient ID each call —
// Without this, the browser's SVG <defs> cache
// keeps serving the OLD gradient even after
// container.selectAll("*").remove() clears the DOM.
const gradientId = 'legend-gradient-${mode}'; // "legend-gradient-max" or "legend-gradient-min"

const defs = legendSvg.append("defs");
const linearGrad = defs.append("linearGradient")
.attr("id", gradientId)
.attr("x1", "0%").attr("y1", "0%")
.attr("x2", "0%").attr("y2", "100%");

d3.range(steps + 1).forEach(i => {
  const t = i / steps;
  const val = lo + t * (hi - lo);
  linearGrad.append("stop")
    .attr("offset", `${t * 100}%`)
    .attr("stop-color", colorScale(val));
});

legendSvg.append("rect")
.attr("x", 0)
.attr("y", 20)
.attr("width", LW)
.attr("height", LH)
.style("fill", `url(#${gradientId})`); // ← now points to the fresh gradient

legendSvg.append("text")
.attr("x", LW + 6)
.attr("y", 20)
.attr("font-size", "11px")
.attr("dominant-baseline", "middle")
.text(`${Math.round(lo)} Celsius`);

legendSvg.append("text")
.attr("x", LW + 6)
.attr("y", 20 + LH)
.attr("font-size", "11px")
.attr("dominant-baseline", "middle")
.text(`${Math.round(hi)} Celsius`);
}

```

Fix 2 — Ensure `updateColors()` is called BEFORE `buildLegend()`

The legend reads `colorScale.domain()` to build its gradient stops. The domain is updated inside `updateColors()`, so the order of calls in the toggle handler matters:

```

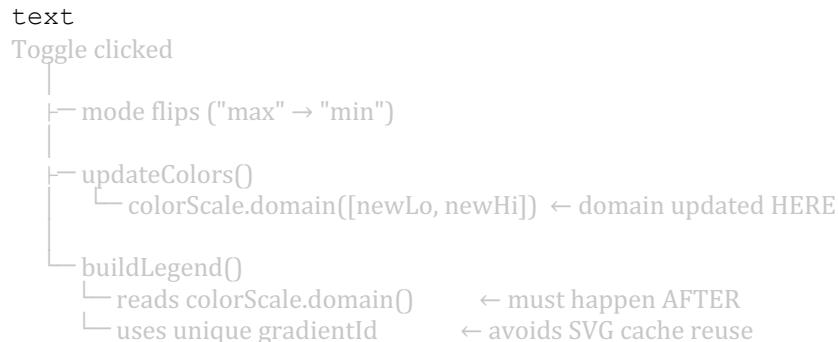
js
d3.select("#toggle-btn").on("click", () => {
  mode = mode === "max" ? "min" : "max";
  d3.select("#toggle-btn")
    .text(`Showing: ${mode.toUpperCase()} Temperature (click to switch)`);

// --- ORDER MATTERS ---
// Step 1: update colorScale.domain() first
updateColors(cellGroups, monthlyData, colorScale, years);
// Step 2: now buildLegend reads the correct domain
buildLegend(colorScale);
});

```

If you call `buildLegend()` before `updateColors()`, the legend will always show the **previous** mode's color range.

Why This Happens — Quick Mental Model



The two fixes together ensure:

1. The **color stops** in the gradient reflect the new temperature range
2. The SVG renderer doesn't **reuse a stale cached gradient** from `<defs>`

Question: Few errors in the above code, first of all, the range of legend is being changed when changing between maximum and minimum. It has to be fixed. Can you help me fix this problem?

The problem is in `updateColors()` — the `colorScale.domain()` is being set to `d3.extent(allVals)`, which computes the **min/max of only the current mode's values**. So when you toggle, the domain shrinks or expands to fit just that mode's range, causing the legend colors to shift.

The fix is to **compute the domain once across both max and min temperatures combined**, and keep it fixed for the entire lifetime of the visualization.

The Fix — Compute a Fixed Domain Once

Step 1: Compute the global domain right after data loads in `render()`

```
js
// After monthlyData and years are ready, inside render():

// —— FIXED DOMAIN ——
// Collect ALL temperature values (both max AND min)
// across every year and month so the domain never
// changes when toggling. This keeps the legend stable.
const allMaxVals = [];
const allMinVals = [];

years.forEach(year => {
  d3.range(12).forEach(month => {
    const cd = monthlyData.get(year)?.get(month);
    if (!cd) return;
    allMaxVals.push(cd.maxTemp);
    allMinVals.push(cd.minTemp);
  });
});

const globalDomain = [
  d3.min(allMinVals), // absolute coldest across all months/years
  d3.max(allMaxVals) // absolute hottest across all months/years
];

// Set the domain once — never change it again
colorScale.domain(globalDomain);
```

Step 2: Remove domain-setting from `updateColors()`

`updateColors()` should **only repaint cells** — never touch the domain:

```
js
function updateColors(cellGroups, monthlyData, colorScale, years) {
  // —— Domain is NOT reset here anymore ——
  // It was set once in render() and stays fixed.

  cellGroups.forEach(g => {
    const d = g.datum();
    const val = mode === "max"
      ? d.cellData.maxTemp
      : d.cellData.minTemp;
    g.select("rect.bg").attr("fill", colorScale(val));
  });
}
```

Step 3: Pass `globalDomain` to `buildLegend()`

Since the domain is now fixed, the legend never needs to re-read a changing domain. Pass it explicitly so it always uses the same range:

```
js
// In render(), change calls to:
buildLegend(colorScale, globalDomain);

// In toggle button:
updateColors(cellGroups, monthlyData, colorScale, years);
buildLegend(colorScale, globalDomain);
```

And update `buildLegend()` signature:

```
js
// — Accept explicit domain so it never drifts —
function buildLegend(colorScale, domain) {
  const container = d3.select("#legend-container");
  container.selectAll("*").remove();

  const LH = 300;
  const LW = 20;
  const steps = 200;
  const [lo, hi] = domain; // ← use passed-in domain, not colorScale.domain()

  // ... rest of the function stays exactly the same
}
```

Why This Fixes It

Before (broken)

`colorScale.domain()` set to `d3.extent` of
current mode only

Legend range jumps between ~11–34°C (min)
and ~16–36°C (max)

Colors are not comparable across max/min toggle

After (fixed)

Domain set **once** from the full range of both
max and min values

Legend always shows the same global range,
e.g. 11–36°C

Same color always means the same
temperature regardless of mode
