Information Visualization 2021 | Computer Science & iCompetence
Marco Soldati

March 2021

D3 Introduction / Scale & Axis

02

Why D3.js

- D3 is popular 80M Downloads, 90K Stars
- D3 is flexible – as a developer you have full control
- D3 is renowned for animation and interaction
- D3 there is a huge community and lots of examples out there

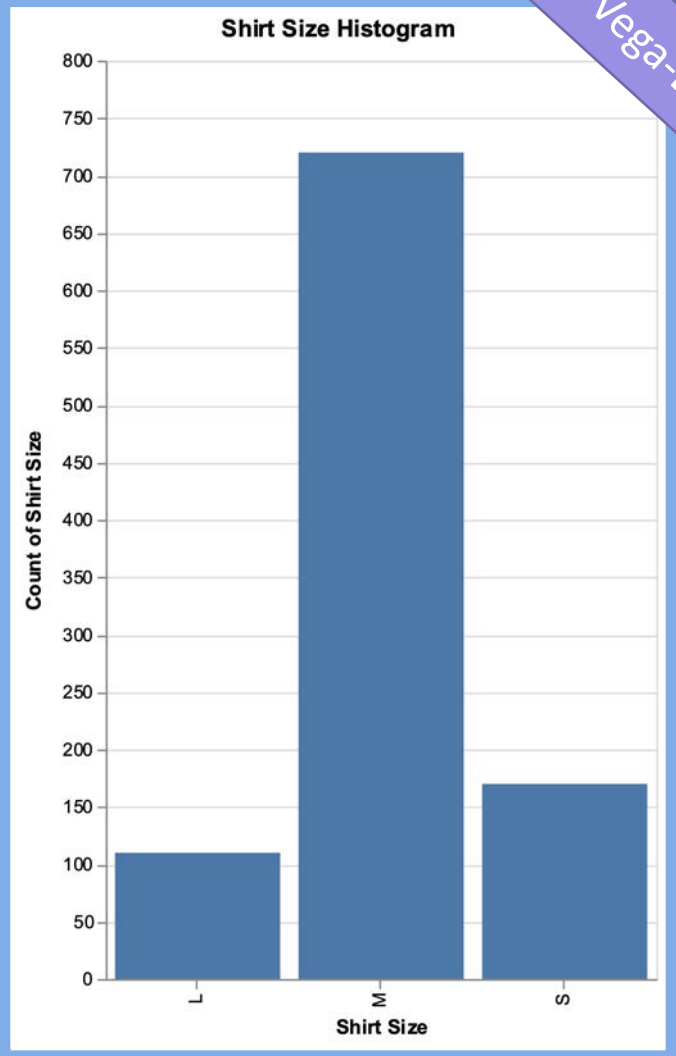
Why not d3.js

- There is quite a learning curve for d3.js
  → As a software engineer at bachelor level, it is achievable

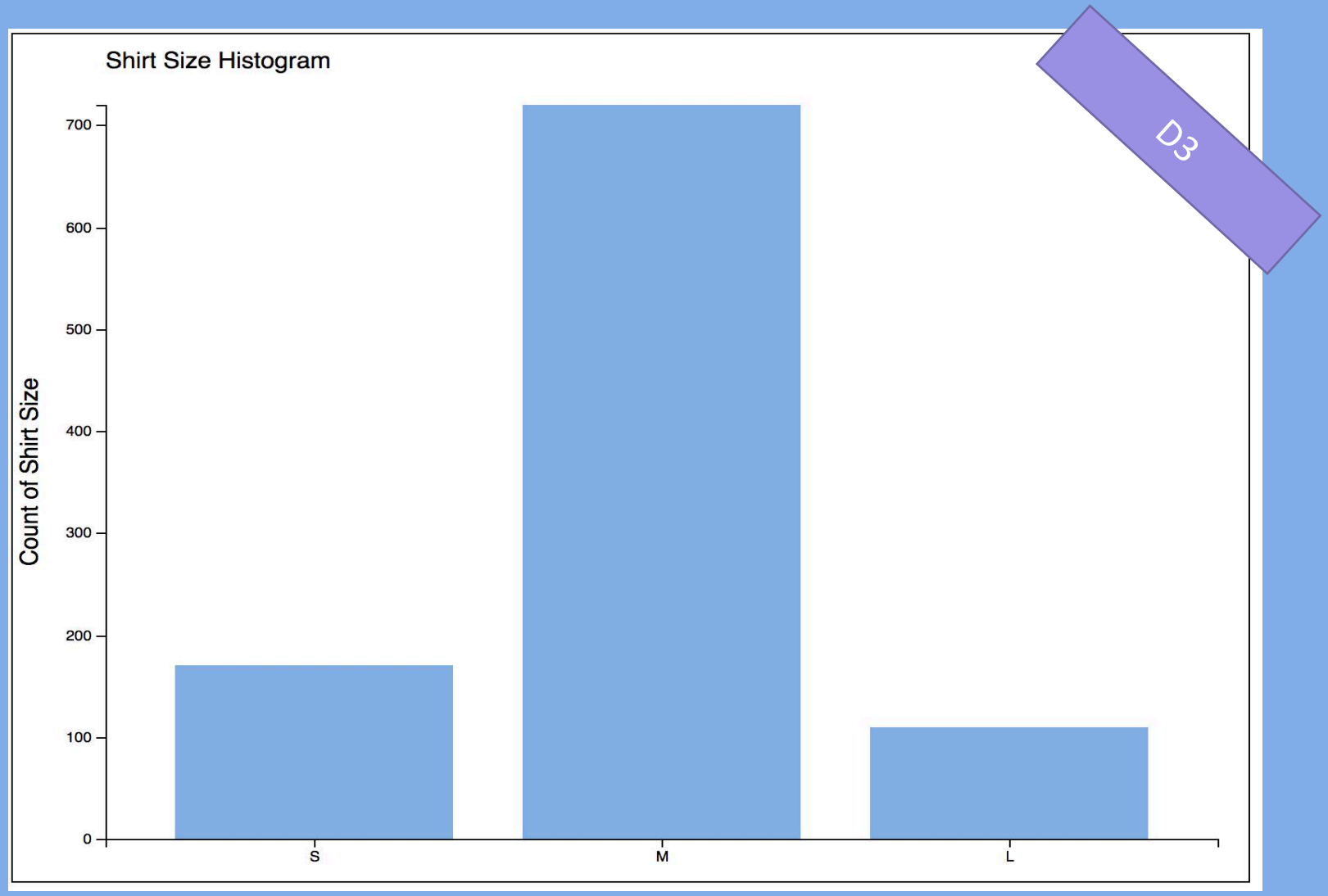
Some comments

- This tutorial is meant as a quick start for some of the important and slightly more complex features of d3.js
- Thus, this tutorial does not cover all aspects of d3.js
- Check the excellent online resources for more details
- Best is to start your project from an existing drawing and adapt it towards your needs

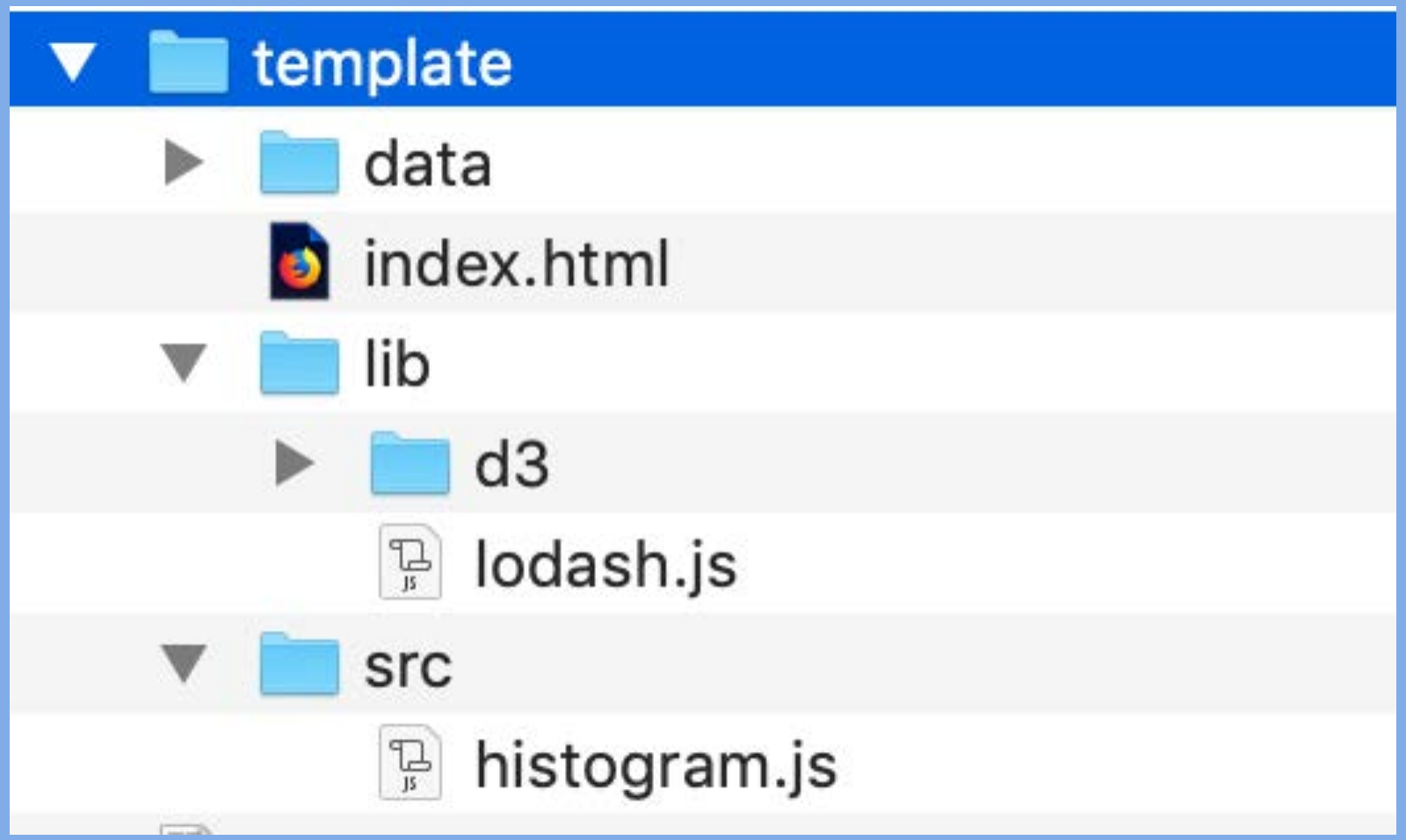
Goal – D3 Shirt Size Histogram



Goal – D3 Shirt Size Histogram



Template folder



Your turn

1. Copy template folder from active directory to your computer
2. Open Terminal and cd to your directory
3. Run a local server:

```
# For Python version >3
python3 -m http.server # On windows try "python"
# For Python version 2
python -m SimpleHTTPServer
```

4. Open index.html in Browser
5. Open index.html in your favourite Editor

Alternatively: use your web development IDE with an integrated server

HTML Template

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8"/>
</head>
<body>

  <!-- put your html stuff here -->

  <script src='lib/d3/d3.js'></script>
  <script src='lib/lodash.js'></script>
  <script src='src/histogram.js'></script>
</body>
</html>
```

histogram.js – sample data

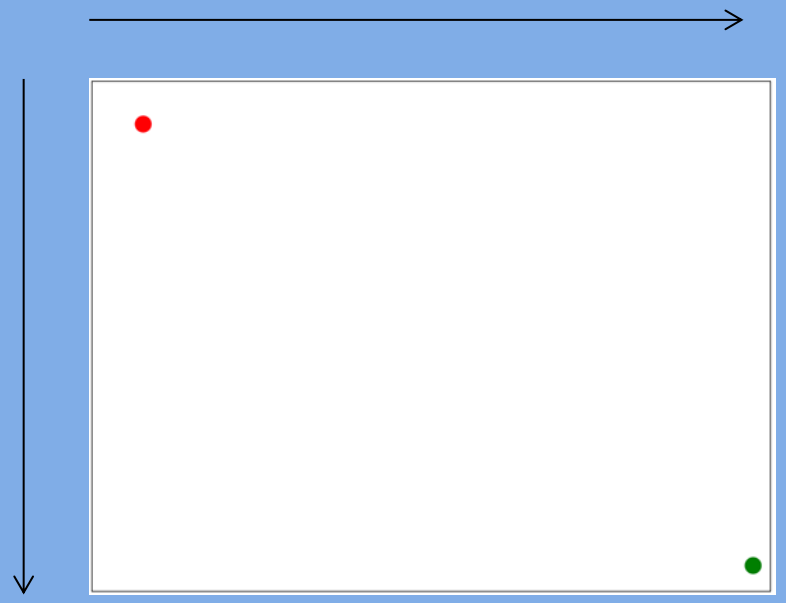
```
var shirt_data = [
   {"size": "S", "count" : 170},
   {"size": "M", "count" : 720},
   {"size": "L", "count" : 110}
];
```

histogram.js – drawing area

```
// create svg canvas
const canvHeight = 600, canvWidth = 800;
const svg = d3.select("body").append("svg")
  .attr("width", canvWidth)
  .attr("height", canvHeight)
  .style("border", "1px solid");

// calc the width and height depending on margins.
const margin = {top: 50, right: 20, bottom: 30, left: 60};
height = canvHeight - margin.top - margin.bottom;
width = canvWidth - margin.left - margin.right;
```

histogram.js – drawing area



```
svg.append("circle")
.attr("cx", margin.left)
.attr("cy", margin.top)
.attr("r", 10)
.style("fill", "red");
```

```
svg.append("circle")
.attr("cx", margin.left + width)
.attr("cy", margin.top + height)
.attr("r", 10)
.style("fill", "green");
```

Your turn

- Draw red and green point

Your turn

- Append a <g>-tag for the chart area, set its top/left point to the red dot (hint: use `transform="translate(x,y)"`)
- Append a `<rect id="chart-area-box" ...>` to mark the chart area

Cascading Style Sheets (CSS)

- Describes the presentation of HTML and SVG elements
- CSS
  - https://www.cheatography.com/davechild/cheat-sheets/css2/

- CSS Selectors – exhaustively used by d3.select() and d3.selectAll()
  - https://www.cheatography.com/dimitrios/cheat-sheets/the-30-css-selectors-you-must-memorize/
  - https://wiki.selfhtml.org/wiki/CSS/Selektoren

CSS – index.html

```
<head>
<meta charset="utf-8"/>
  <style>
    #chart-area-box { }
    .bar { fill: #82AEE5; }
  </style>
</head>
<body>
...
```

Your turn

- Add CSS <style> section to index.html
- Create a style for the outer box (fill: none; stroke: black)

D3.js

- D3 modules
  - https://github.com/d3
- D3 Gallery
  - https://github.com/d3/d3/wiki/Gallery
- D3 API Reference
  - https://github.com/d3/d3/blob/master/API.md

Scale

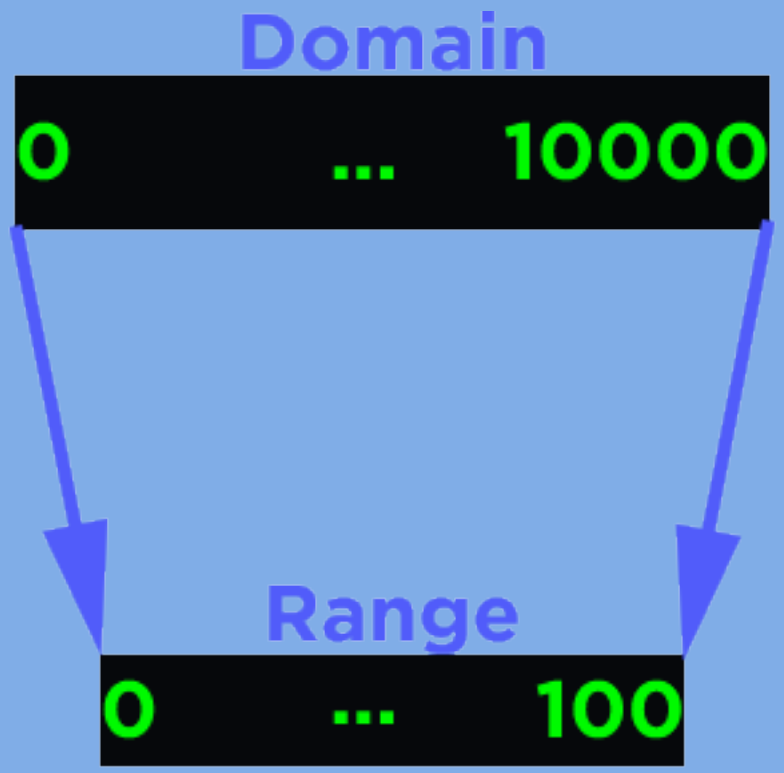
A **mapping** from values to values ("Math")

Axis

A **visual representation** of a scale ("SVG Generator")

Scale

A mapping from values to values – aka a **function:**




```
var x = d3.scaleLinear()
.domain([0, 10000])
.range([0, 100]);

x(2000); // 20
x(32000); // 320
```

Scale

A mapping from values to values – aka a **function:**

Input	▶	**Output**
Domain	▶	Range
Attribute	▶	Channel
Data (Values)	▶	Screen (Pixels)

Examples:

Person Height	▶	Pixel Position
Person Shirt Size	▶	Pixel Color

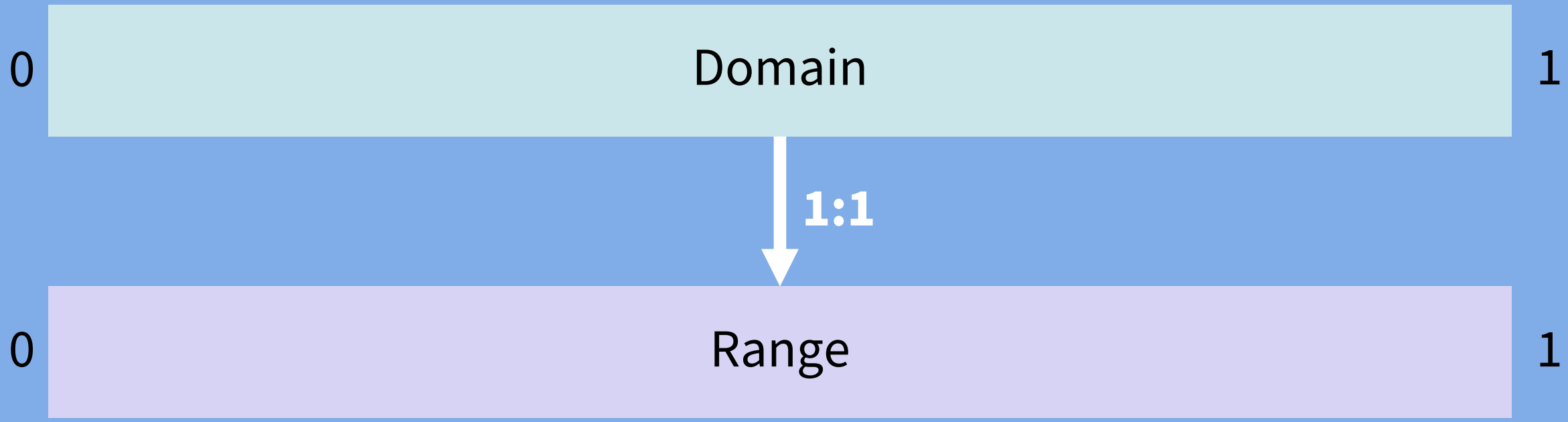
Linear Scales

Continuous Input ▸ Continuous Output

≤ v3 `d3.scale.linear();`

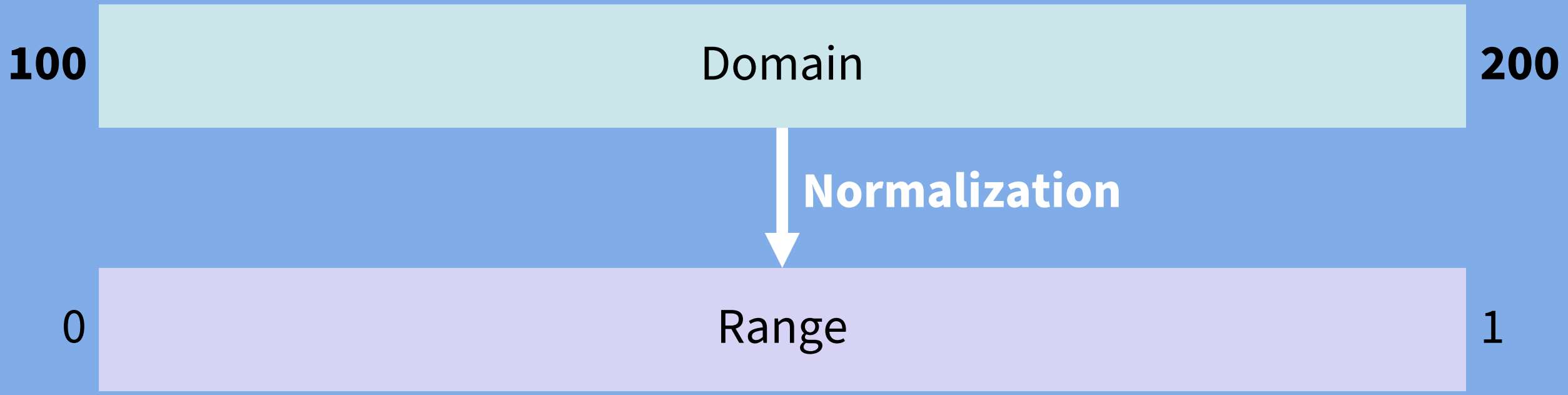
≥ v4 `d3.scaleLinear();`

```
const scale = d3.scaleLinear();
```



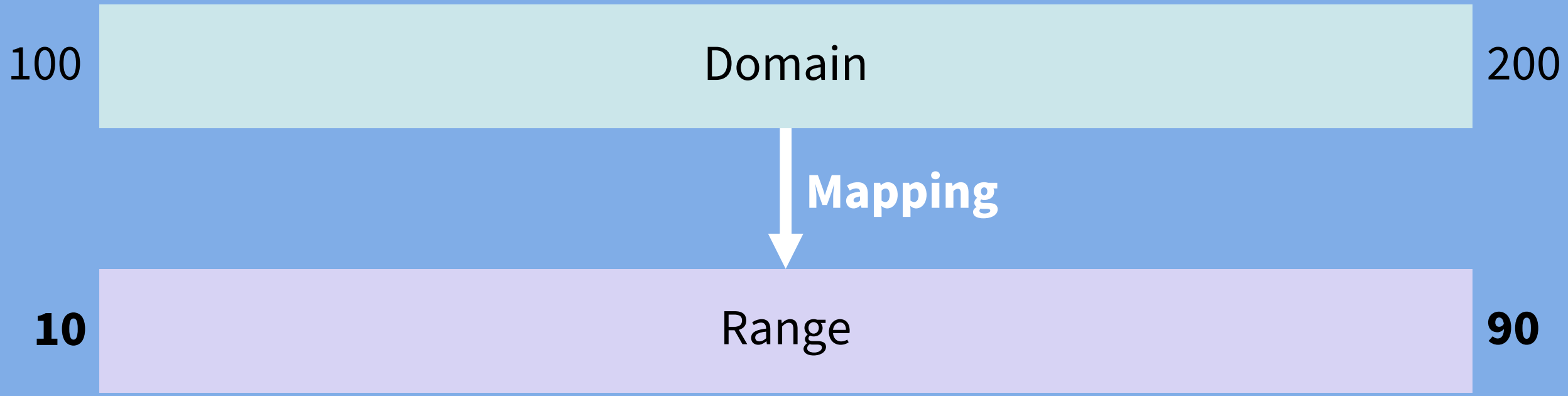

```
scale(0.5); // returns 0.5
scale(99);  // returns 99
```

```
const scale = d3.scaleLinear()
                .domain([100, 200]);
```




```
scale(100); // 0
scale(150); // 0.5
scale(200); // 1
scale(300); // 2
```

```
const scale = d3.scaleLinear()
              .domain([100, 200])
              .range([10, 90]);
```




```
scale(100); //  10
scale(150); //  50
scale(200); //  90
scale(250); // 130
```

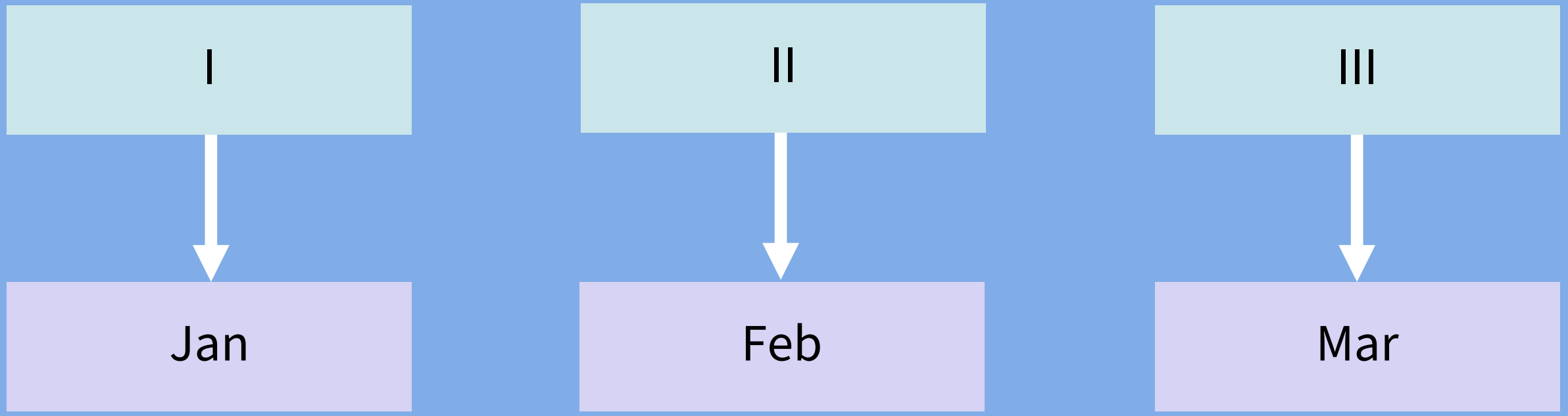
Ordinal Scales

Discrete Input ▸ Discrete Output

≤ v3 `d3.scale.ordinal();`

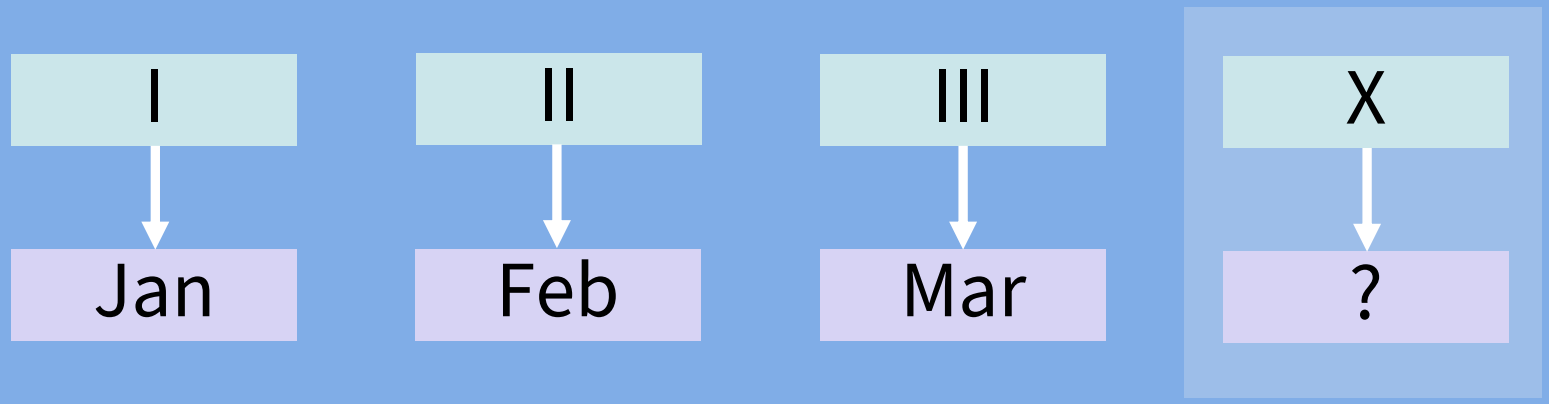
≥ v4 `d3.scaleOrdinal();`

```
const scale = d3.scaleOrdinal()
              .domain(["I", "II", "III"])
              .range(["Jan", "Feb", "Mar"]);
```



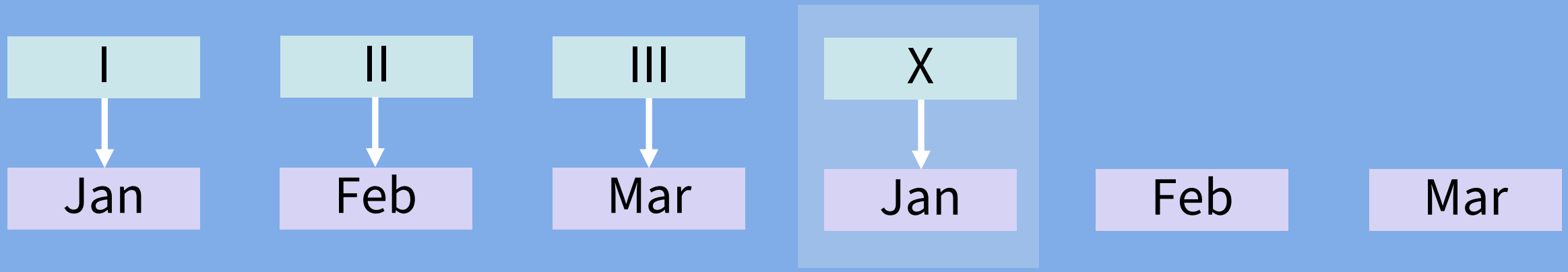

```
scale("II"); // "Feb"
```

```
const scale = d3.scaleOrdinal()
              .domain(["I", "II", "III"])
              .range(["Jan", "Feb", "Mar"]);
```




```
scale("X");    // ???
```

```
const scale = d3.scaleOrdinal()
              .domain(["I", "II", "III"])
              .range(["Jan", "Feb", "Mar"]);
```



Domain grows dynamically (cyclic)

```
scale("X");     // "Jan”
scale.domain(); // ["I", "II", "III", "X"]
```

```
const color = d3.scaleOrdinal()
                    .range(d3.schemeCategory10);
```

```
color("one");  // "#1f77b4" (Blue)
color("next"); // "#ff7f0e" (Orange)
```

```
const color = d3.scaleOrdinal()
  .range(d3.schemeCategory10);
```

Equivalent

Pass range to constructor

```
const color =
  d3.scaleOrdinal(d3.schemeCategory10);
```

Band Scales

Discrete Input ▸ Continuous Output Bands

≤ v3

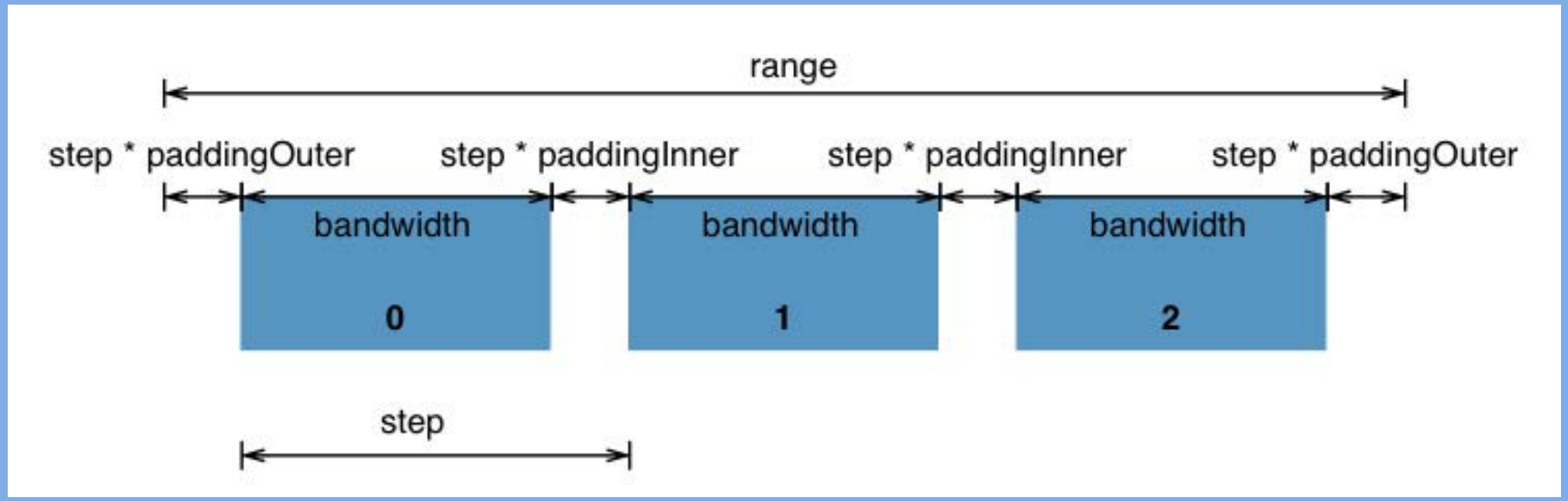
```
d3.scale.ordinal().rangeBands()
```

≥ v4

```
d3.scaleBand();
```

Full Padding Control

Perfect for Bar Charts / Histograms / Grids



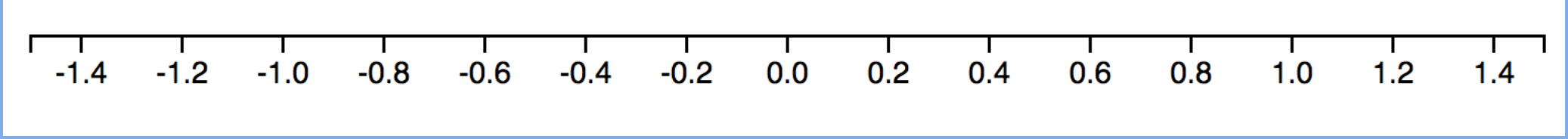
Axis

A **visual representation** of a scale

aka

A **function** which generates SVG

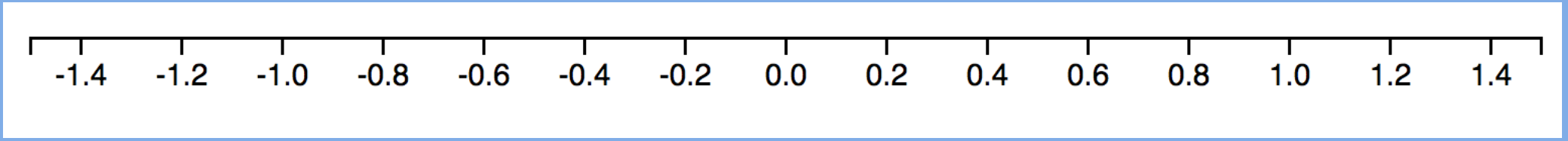
Scale ▸ SVG



```
const xAxis = d3.axisBottom(xScale);

svg.call(xAxis);
```

```
▼<svg class="chart" width="500" height="300">
  ▼<g fill="none" font-size="10" font-family="sans-serif" text-anchor="middle">
     <path class="domain" stroke="#000" d="M0.5,6V0.5H500.5V6"></path>
    ▼<g class="tick" opacity="1" transform="translate(16.666666666666647,0)">
       <line stroke="#000" y2="6" x1="0.5" x2="0.5"></line>
       <text fill="#000" y="9" x="0.5" dy="0.71em">-1.4</text>
     </g>
    ▶<g class="tick" opacity="1" transform="translate(49.99999999999997,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(83.33333333333333,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(116.66666666666666,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(150,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(183.33333333333331,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(216.66666666666669,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(250,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(283.3333333333333,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(316.66666666666663,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(349.99999999999994,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(383.3333333333333,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(416.6666666666667,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(450,0)">…</g>
    ▶<g class="tick" opacity="1" transform="translate(483.33333333333337,0)">…</g>
   </g>
 </svg>
```



Homework

- Create the TShirt-Size Histogram
  - Create boxes
  - Create titles
  - Create scales for x- and y
  - Create axis for xScale and yScale

- Use Chrome Console to check it.