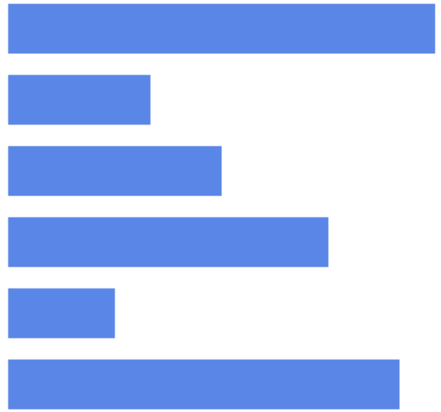
Scales

So far...

data

```
const bardata = [300, 100, 150, 225, 75, 275];
```



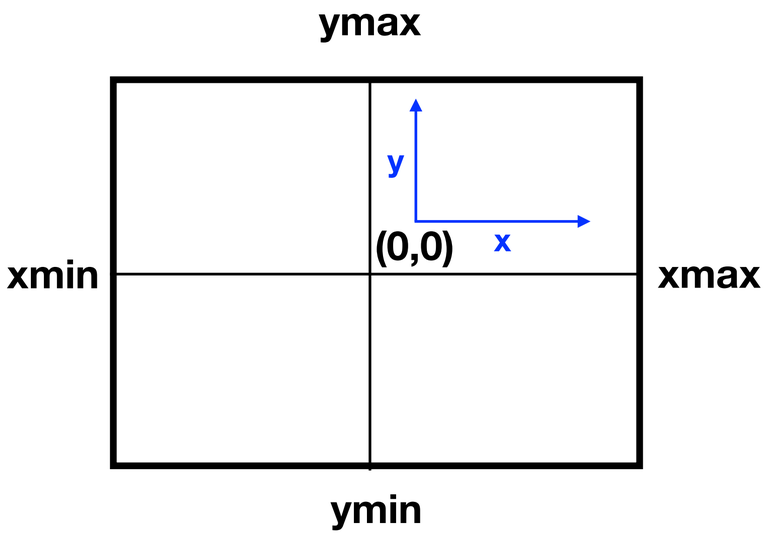
pixels

```
<svg width="700" height="500">
  <rect x="20" y="50" width="300" height="30" fill="cornflowerblue"></rect>
  <rect x="20" y="100" width="100" height="30" fill="cornflowerblue"></rect>
  <rect x="20" y="150" width="150" height="30" fill="cornflowerblue"></rect>
  <rect x="20" y="200" width="225" height="30" fill="cornflowerblue"></rect>
  <rect x="20" y="250" width="75" height="30" fill="cornflowerblue"></rect>
  <rect x="20" y="300" width="275" height="30" fill="cornflowerblue"></rect>
</svg>
```

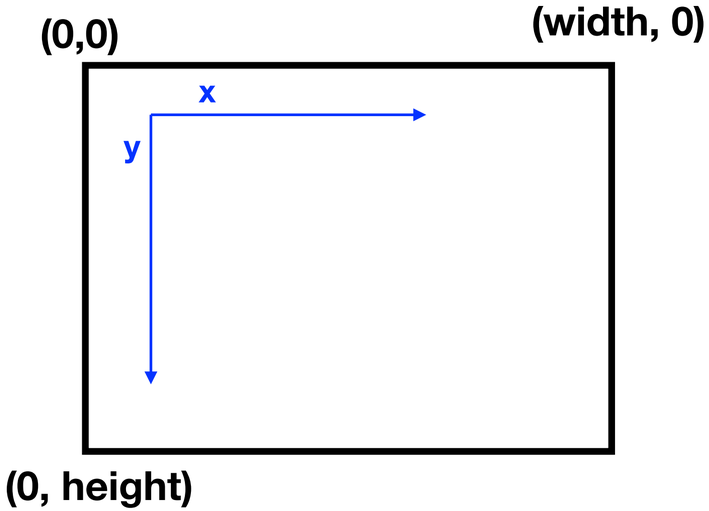
Scales

Convert data <--> pixels

Data



SVG (pixels)



Linear scales

```
d3.scaleLinear()
  .domain([min, max])
  .range([min, max]);
```

- Domain and range are *continuous*
- D3 scales are used to create your own scale functions

```
> const myscale = d3.scaleLinear()
    .domain([0, 1])
    .range([0, 500]);
< undefined
> myscale(.2);
< 100
```

Linear scale for x-axis

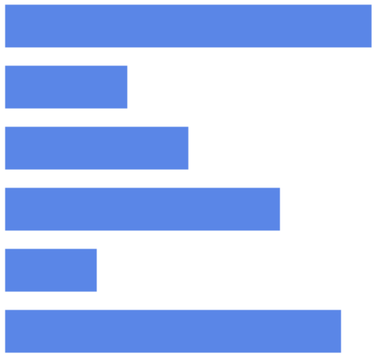
for continuous data

```
d3.scaleLinear()
  .domain([min, max])
  .range([min, max]);
```

xScale

```
const xScale = d3.scaleLinear()
  .domain([-100, 100])   // data world
  .range([0, 500]);      // pixel world
```

```
.attr("width", d => d);
```
becomes
```
.attr("width", d => xScale(d));
```



https://d3js.org/d3-scale/linear#scaleLinear

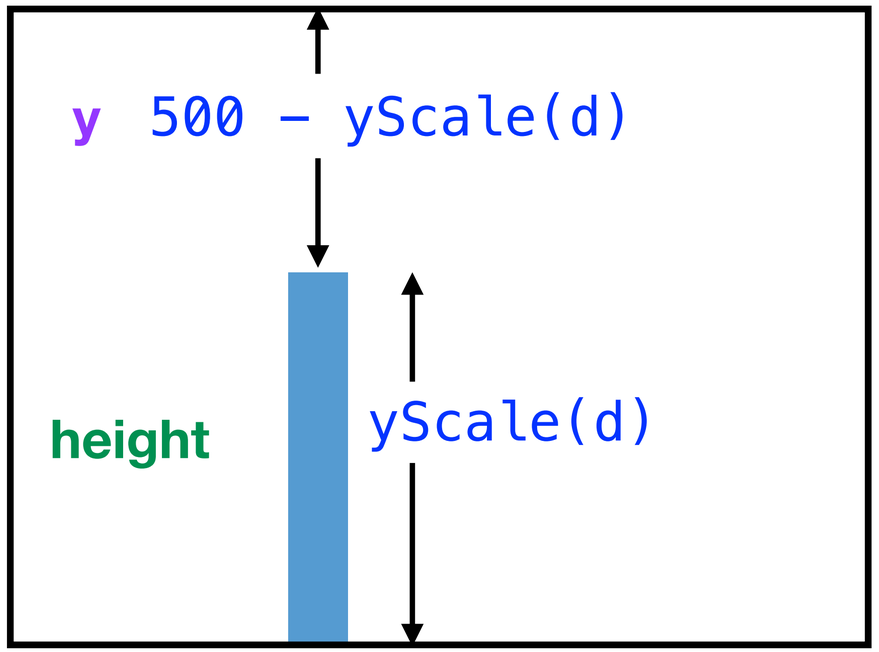
Linear scale for y-axis (one option)

yScale

```
const yScale = d3.scaleLinear()
  .domain([-100, 100])   // data world
  .range([0, 500]);       // pixel world
```

```
.attr("y", d => 500 - d)
```
becomes
```
.attr("y", d => 500 - yScale(d))
```

```
.attr("height", d => d)
```
becomes
```
.attr("height", d => yScale(d))
```



https://d3js.org/d3-scale/linear#scaleLinear

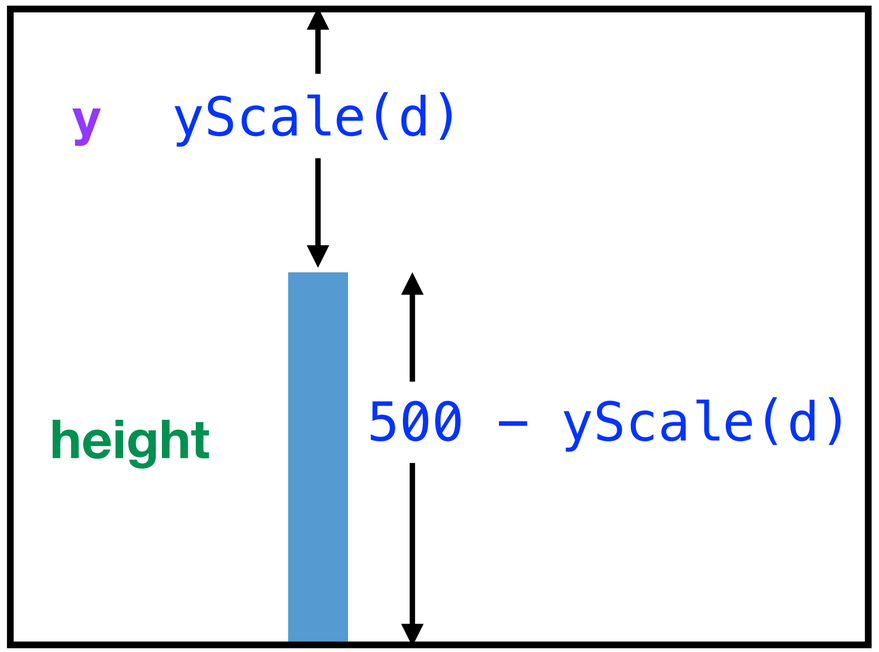
Linear scale for y-axis (better option)

yScale

```
const yScale = d3.scaleLinear()
  .domain([-100, 100])   // data world
  .range([500, 0])       // pixel world
```

```
.attr("y", d => 500 - d)
```
becomes
```
.attr("y", d => yScale(d))
```

```
.attr("height", d => d)
```
becomes
```
.attr("height", d => 500 - yScale(d))
```



https://d3js.org/d3-scale/linear#scaleLinear

Band scales

```
d3.scaleBand()
  .domain([min, max])
  .range([min, max]);
```

- Domain is *categorical* and range is *continuous*

```
> const abandscale = d3.scaleBand()
    .domain(["cold", "warm", "hot"])
    .range([0, 600]);
<· undefined
> abandscale("cold");
<· 0
```

```
> abandscale("warm");
<· 200
> abandscale("hot");
<· 400
```

https://d3js.org/d3-scale/band

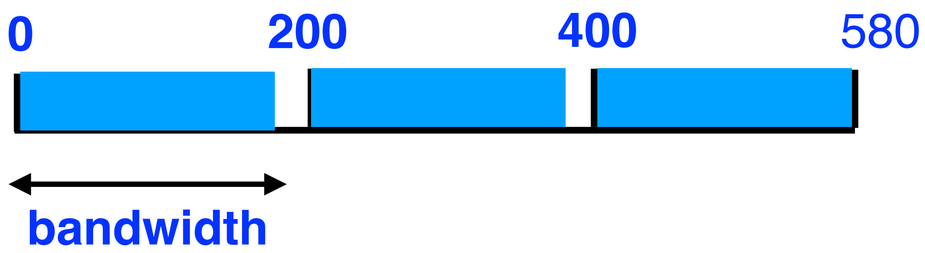
Scale function properties

```
> const abandscale = d3.scaleBand()
    .domain(["cold", "warm", "hot"])
    .range([0, 600]);
< undefined
> abandscale.domain();
< ▶ (3) ['cold', 'warm', 'hot']
> abandscale.bandwidth();
< 200
> abandscale.paddingInner();
< 0
```

https://d3js.org/d3-scale/band

Band scale function with padding

```
> const newscale = d3.scaleBand()
    .domain(["cold", "warm", "hot"])
    .range([0, 580])
    .paddingInner(.1);
<· undefined
> newscale("cold");
<· 0
> newscale("warm");
<· 200
> newscale("hot");
<· 400
> newscale.bandwidth();
<· 180
> newscale.paddingInner();
<· 0.1
```



https://d3js.org/d3-scale/band