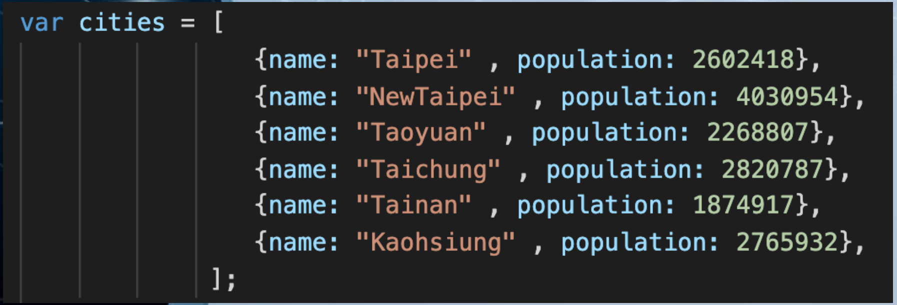
Scale & Axis

Data Visualization

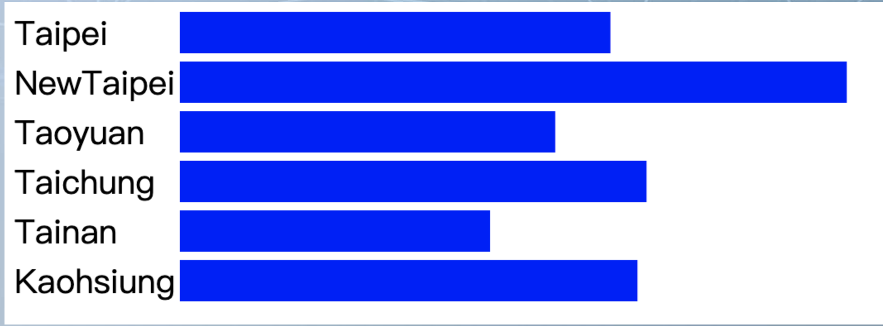
Ex04-1

- The data is the populations of cities in Taiwan
- We would like to show the following bar charts

- Files
  - index.html
  - main.js

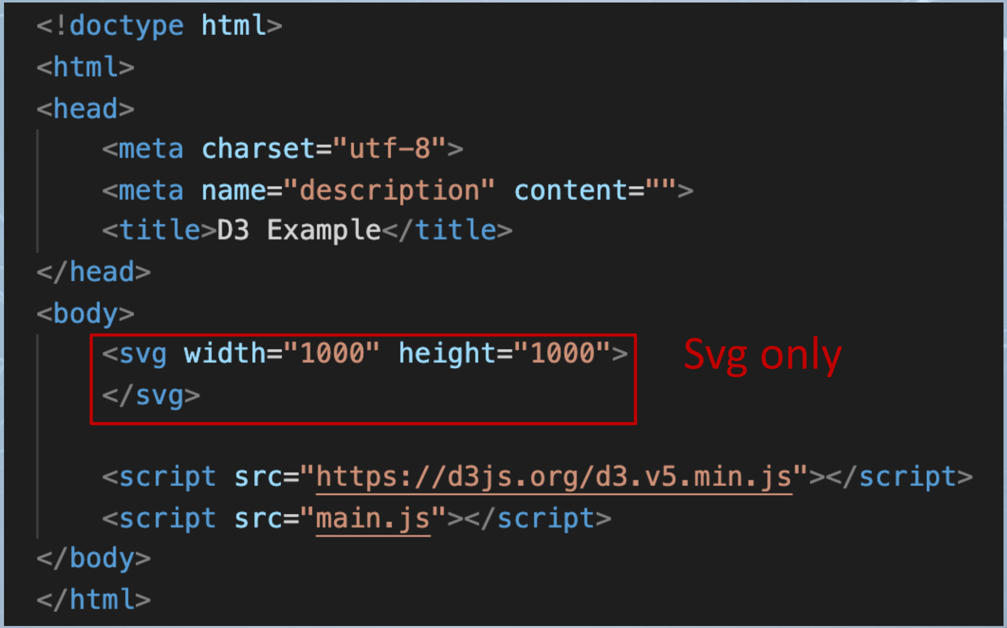


```
var cities = [
            {name: "Taipei" , population: 2602418},
            {name: "NewTaipei" , population: 4030954},
            {name: "Taoyuan" , population: 2268807},
            {name: "Taichung" , population: 2820787},
            {name: "Tainan" , population: 1874917},
            {name: "Kaohsiung" , population: 2765932},
        ];
```



Ex04-1

- Index.html

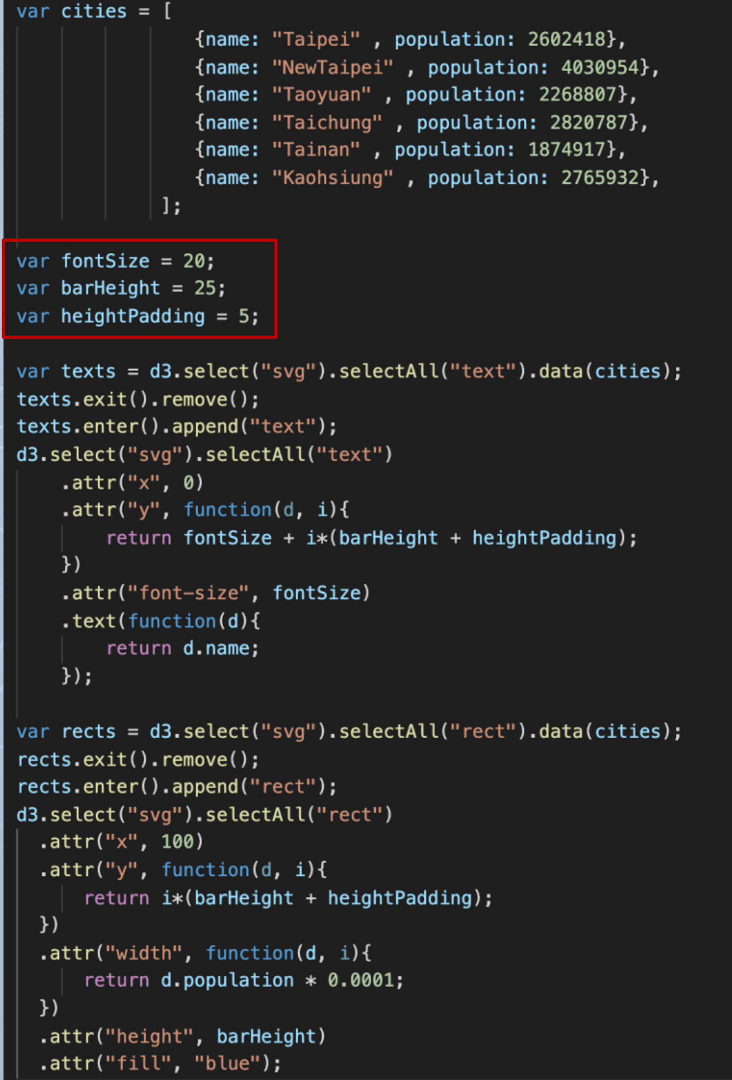


```html
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="description" content="">
    <title>D3 Example</title>
</head>
<body>
    <svg width="1000" height="1000">
    </svg>

    <script src="https://d3js.org/d3.v5.min.js"></script>
    <script src="main.js"></script>
</body>
</html>
```

Svg only

Ex04-1

- main.js
- Set fontSize, barHeight, heightPadding in varaibles
  - Easy to change later

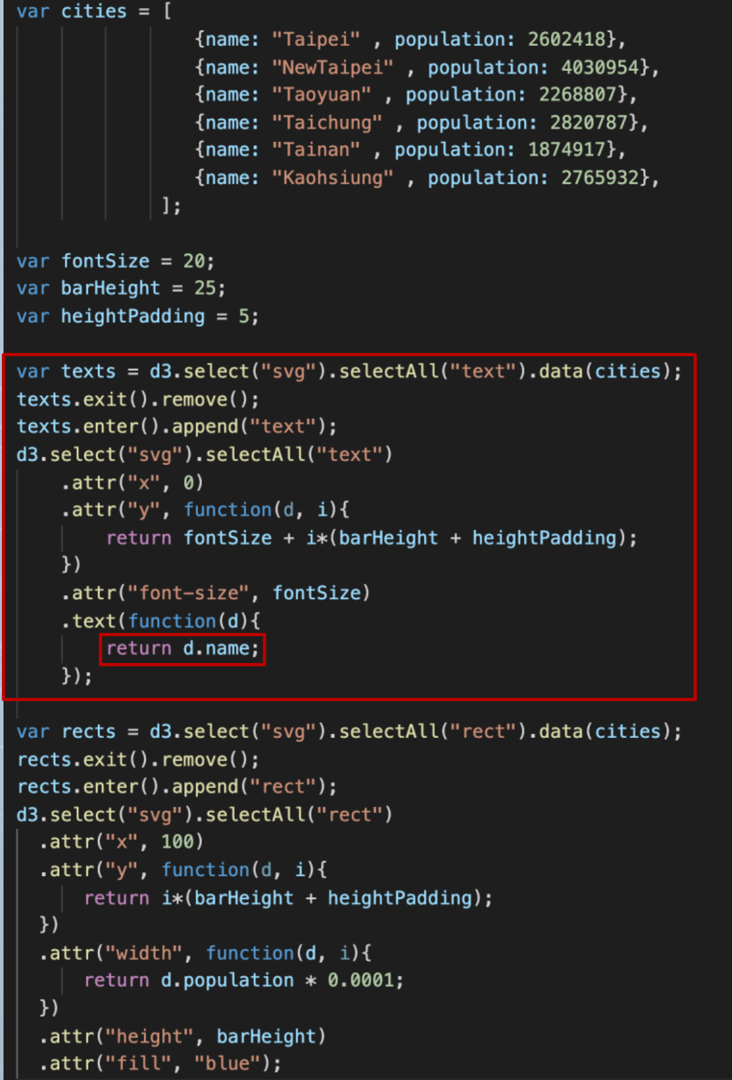


```
var cities = [
                {name: "Taipei" , population: 2602418},
                {name: "NewTaipei" , population: 4030954},
                {name: "Taoyuan" , population: 2268807},
                {name: "Taichung" , population: 2820787},
                {name: "Tainan" , population: 1874917},
                {name: "Kaohsiung" , population: 2765932},
             ];

var fontSize = 20;
var barHeight = 25;
var heightPadding = 5;

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return fontSize + i*(barHeight + heightPadding);
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities);
rects.exit().remove();
rects.enter().append("rect");
d3.select("svg").selectAll("rect")
  .attr("x", 100)
  .attr("y", function(d, i){
      return i*(barHeight + heightPadding);
  })
  .attr("width", function(d, i){
      return d.population * 0.0001;
  })
  .attr("height", barHeight)
  .attr("fill", "blue");
```

Ex04-1

- main.js

- Add text to the webpage



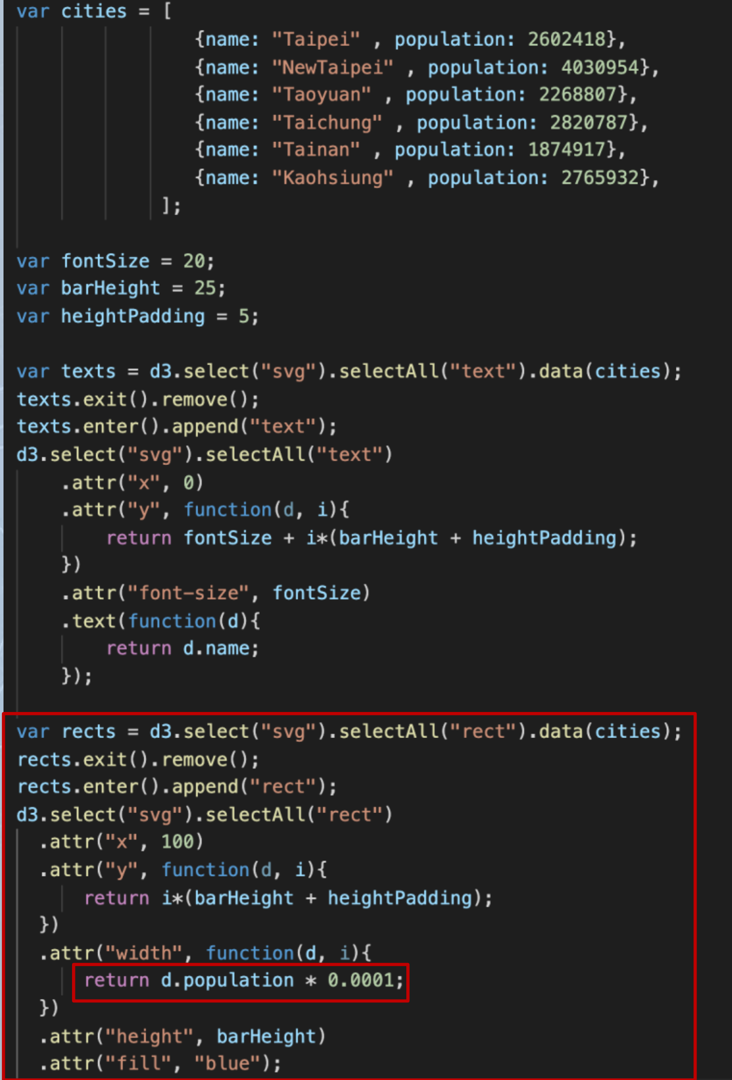
```
var cities = [
                {name: "Taipei" , population: 2602418},
                {name: "NewTaipei" , population: 4030954},
                {name: "Taoyuan" , population: 2268807},
                {name: "Taichung" , population: 2820787},
                {name: "Tainan" , population: 1874917},
                {name: "Kaohsiung" , population: 2765932},
            ];

var fontSize = 20;
var barHeight = 25;
var heightPadding = 5;

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return fontSize + i*(barHeight + heightPadding);
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities);
rects.exit().remove();
rects.enter().append("rect");
d3.select("svg").selectAll("rect")
  .attr("x", 100)
  .attr("y", function(d, i){
      return i*(barHeight + heightPadding);
  })
  .attr("width", function(d, i){
      return d.population * 0.0001;
  })
  .attr("height", barHeight)
  .attr("fill", "blue");
```

Ex04-1

- main.js

- Add bars to the webpage
- Note: the populations are too large. We have to multiply them with a factor (0.0001) before setting the width of the bars

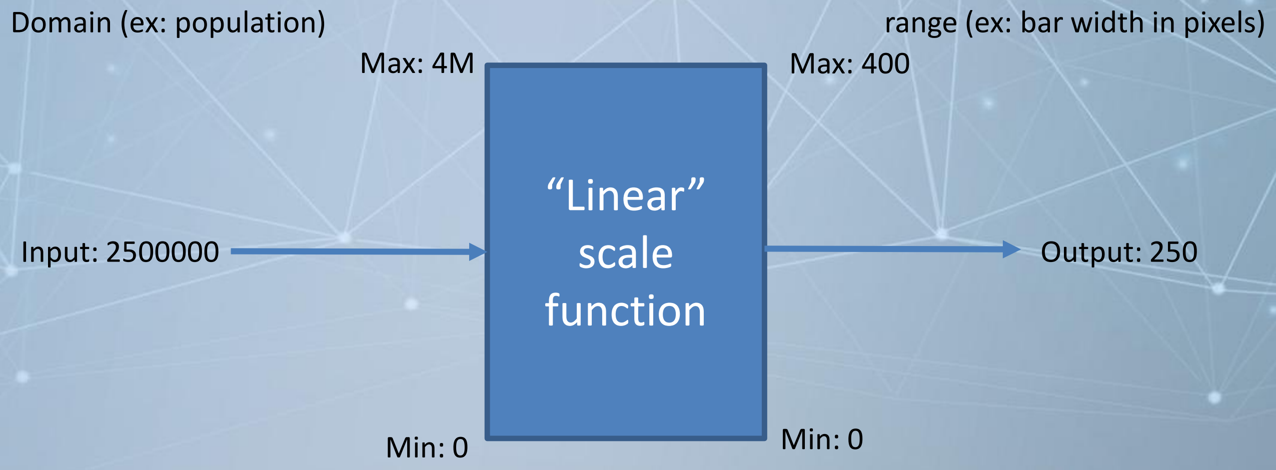
- Better way to do this?



```
var cities = [
                {name: "Taipei" , population: 2602418},
                {name: "NewTaipei" , population: 4030954},
                {name: "Taoyuan" , population: 2268807},
                {name: "Taichung" , population: 2820787},
                {name: "Tainan" , population: 1874917},
                {name: "Kaohsiung" , population: 2765932},
            ];

var fontSize = 20;
var barHeight = 25;
var heightPadding = 5;

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return fontSize + i*(barHeight + heightPadding);
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities);
rects.exit().remove();
rects.enter().append("rect");
d3.select("svg").selectAll("rect")
  .attr("x", 100)
  .attr("y", function(d, i){
      return i*(barHeight + heightPadding);
  })
  .attr("width", function(d, i){
      return d.population * 0.0001;
  })
  .attr("height", barHeight)
  .attr("fill", "blue");
```

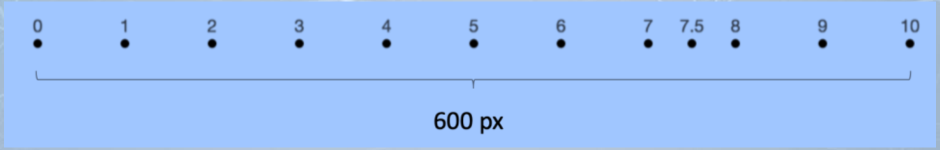
Scales in D3

- Scales are the function that maps an **input domain** to an **output range**



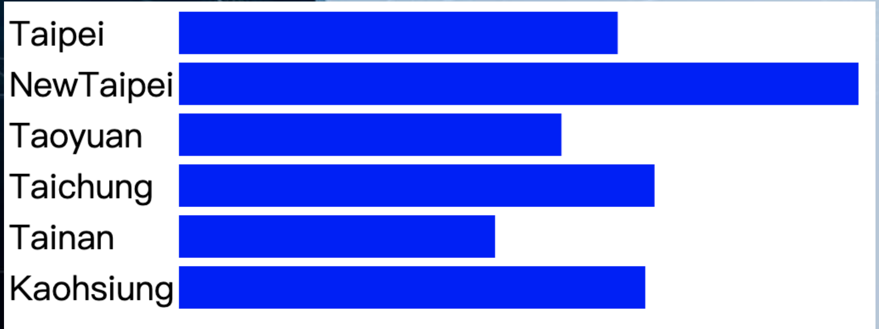
d3.scaleLinear

- d3.scaleLinear().domain([$I_{\text{min}}, I_{\text{max}}$]).range([$O_{\text{min}}, O_{\text{max}}$])
- y = m*x + b
- x: input, y: output, m=$\frac{O_{\text{max}}-O_{min}}{I_{\text{max}}-I_{min}}$, b= O_{min}
- Example:
  - d3.scaleLinear().domain([0, 10]).range([0, 600])

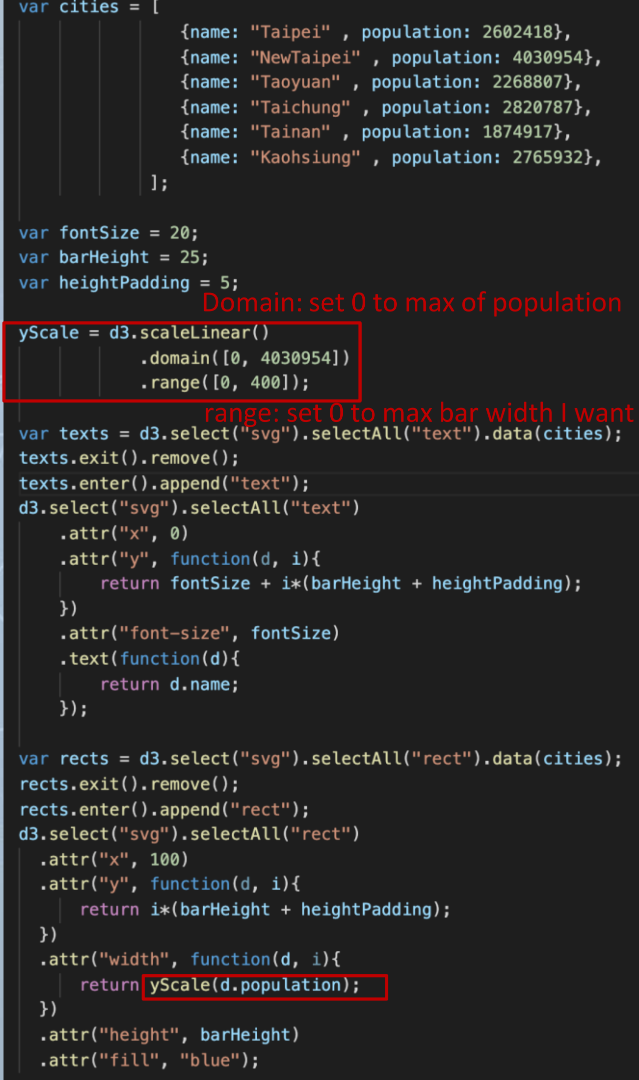


Ex04-2

- d3.scaleLinear()
- Files
  - index.html
  - main.js



Use "yScale" function to transform "population" to bar width in pixel



```
var cities = [
                {name: "Taipei" , population: 2602418},
                {name: "NewTaipei" , population: 4030954},
                {name: "Taoyuan" , population: 2268807},
                {name: "Taichung" , population: 2820787},
                {name: "Tainan" , population: 1874917},
                {name: "Kaohsiung" , population: 2765932},
              ];

var fontSize = 20;
var barHeight = 25;
var heightPadding = 5;
```

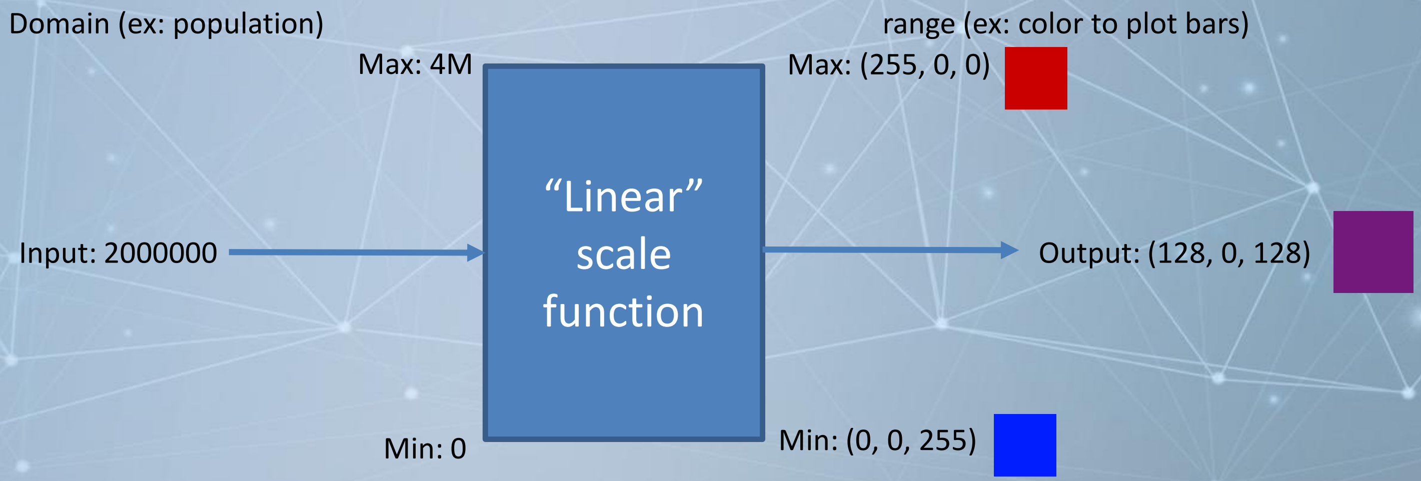
Domain: set 0 to max of population

```
yScale = d3.scaleLinear()
            .domain([0, 4030954])
            .range([0, 400]);
```

range: set 0 to max bar width I want

```
var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return fontSize + i*(barHeight + heightPadding);
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities);
rects.exit().remove();
rects.enter().append("rect");
d3.select("svg").selectAll("rect")
  .attr("x", 100)
  .attr("y", function(d, i){
    return i*(barHeight + heightPadding);
  })
  .attr("width", function(d, i){
    return yScale(d.population);
  })
  .attr("height", barHeight)
  .attr("fill", "blue");
```

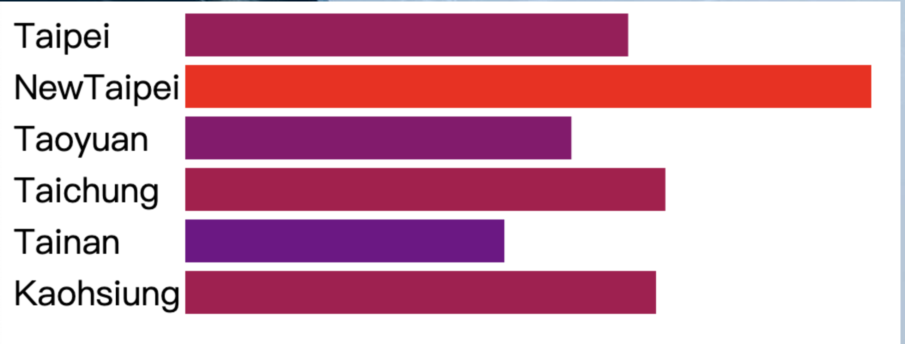
d3.scaleLinear for Color

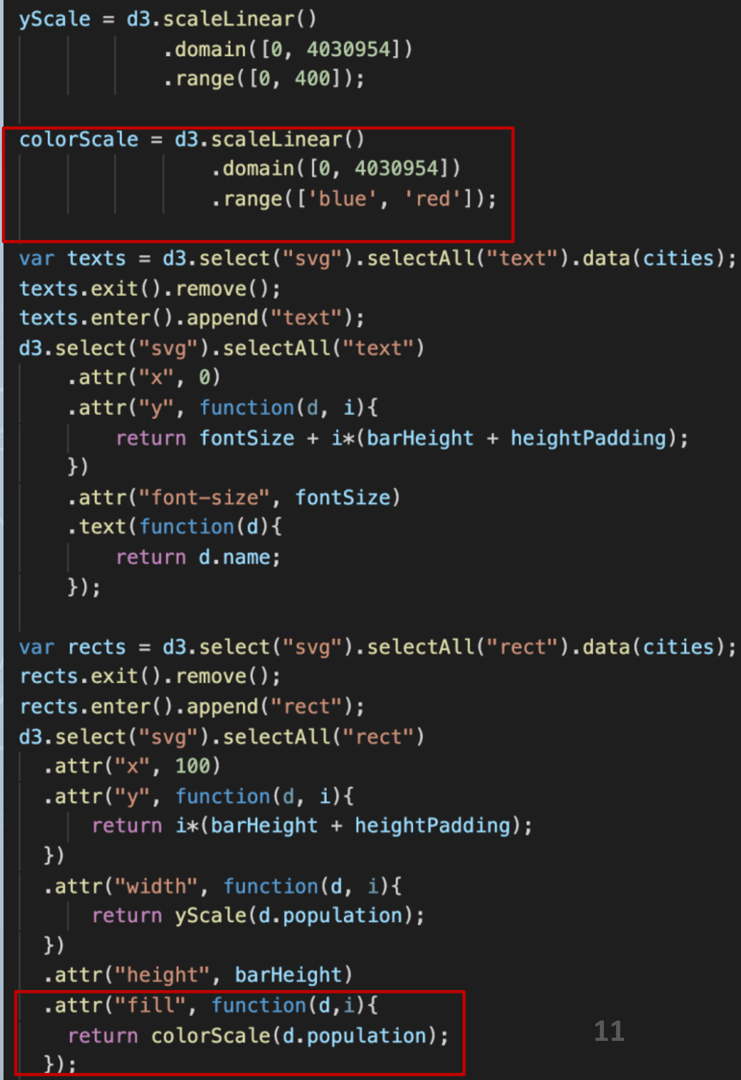


main.js

Ex04-3

- d3.scaleLinear() for color
- Files
  - index.html
  - main.js

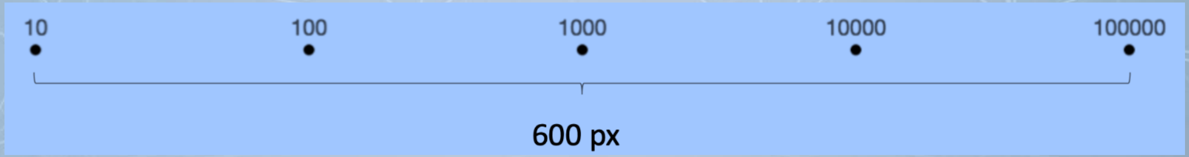




```
yScale = d3.scaleLinear()
            .domain([0, 4030954])
            .range([0, 400]);

colorScale = d3.scaleLinear()
                .domain([0, 4030954])
                .range(['blue', 'red']);

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return fontSize + i*(barHeight + heightPadding);
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities);
rects.exit().remove();
rects.enter().append("rect");
d3.select("svg").selectAll("rect")
  .attr("x", 100)
  .attr("y", function(d, i){
      return i*(barHeight + heightPadding);
  })
  .attr("width", function(d, i){
      return yScale(d.population);
  })
  .attr("height", barHeight)
  .attr("fill", function(d,i){
    return colorScale(d.population);
  });
```

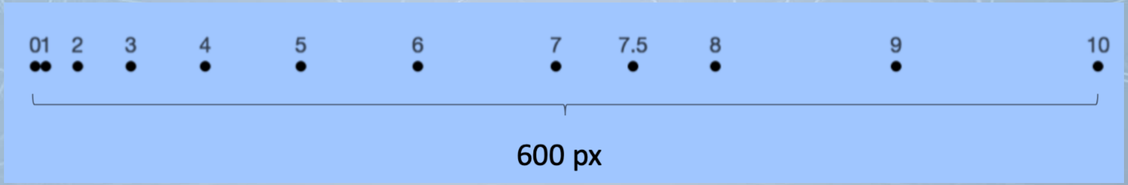
d3.scaleLog()

- d3.scaleLog().domain([$I_{\min}, I_{\max}$]).range([$O_{\min}, O_{\max}$]).base(n)
- $y = m * \log_n(x) + b$
- x: input, y: output, m=$\frac{O_{\max} - O_{\min}}{\log_n(I_{\max}) - \log_n(I_{min})}$, b= $O_{\min}$
- Example:
  - d3.scaleLog().domain([10, 100000]).range([0, 600]).base(10)



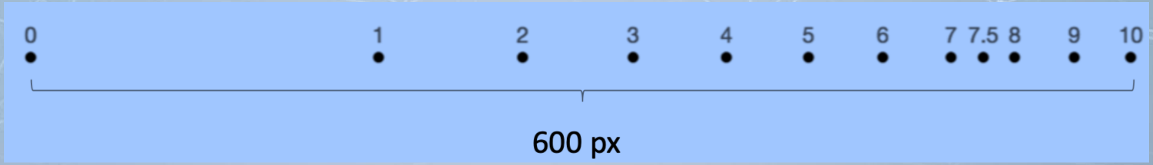
d3.scalePow()

- d3.scalePow().domain([$I_{\min}, I_{\max}$]).range([$O_{\min}, O_{\max}$]).exponent(n)
- $y = m * x^n + b$
- x: input, y: output, m=$\frac{O_{\max}-O_{\min}}{I_{max}^{n}-I_{min}^{n}}$, b= $O_{\min}$
- Example:
  - d3.scalePow().domain([0, 10]).range ([0, 600]).exponent(2)



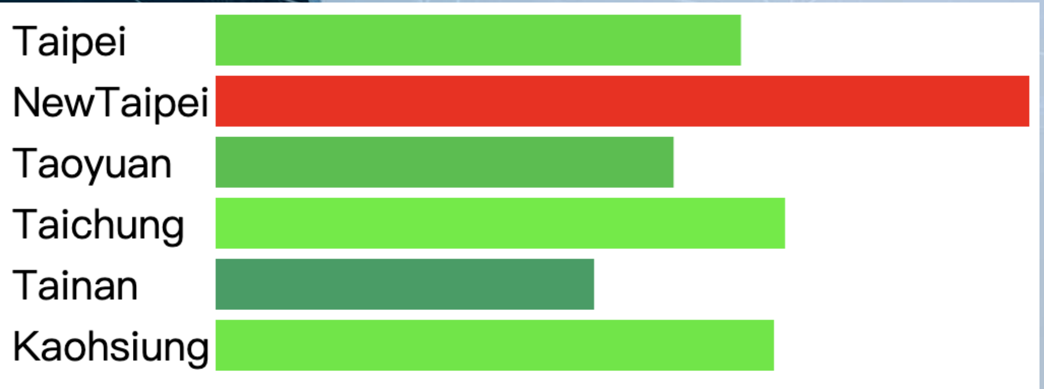
d3.scaleSqrt()

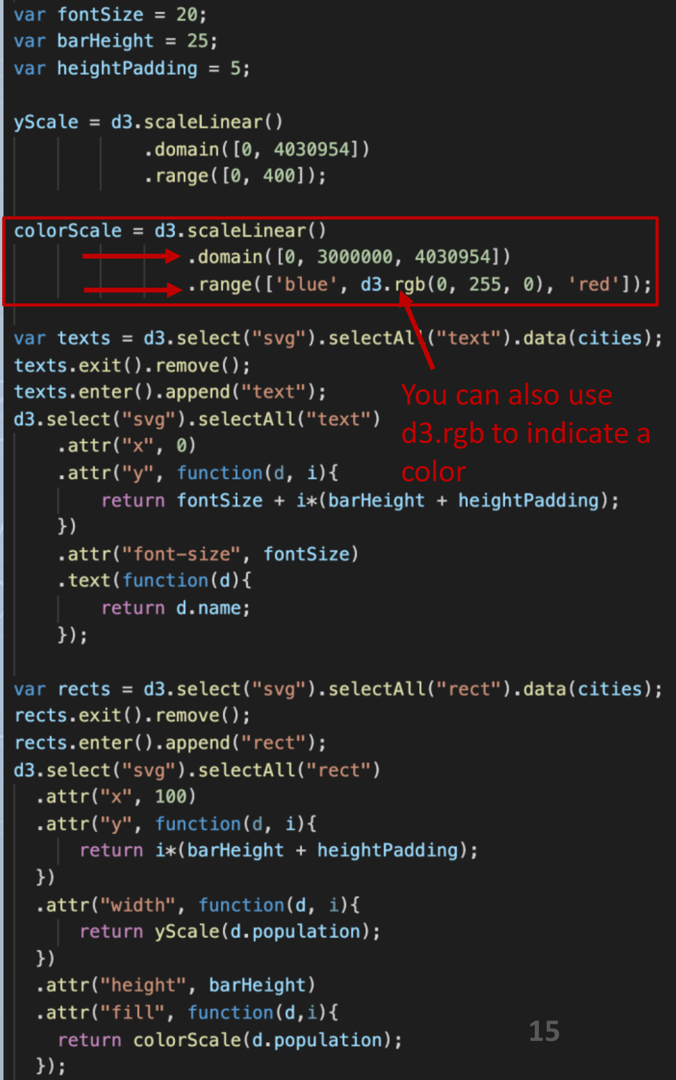
- d3.scalePow().domain([$I_{\min}, I_{\max}$]).range([$O_{\min}, O_{\max}$])
- $y = m * \sqrt{x} + b$
- x: input, y: output, m=$\frac{O_{\max} - O_{min}}{\sqrt{I_{max}} - \sqrt{I_{min}}}$, b= $O_{\min}$
- Example:
  - d3.scalePow().domain([0, 10]).range ([0, 600])



.domain() and .range()

- Actually, you can pass an array with multiple elements to .domain() and .range() of all scale functions
  - It performs piecewise interpolation
- Modify Ex04-3





```
var fontSize = 20;
var barHeight = 25;
var heightPadding = 5;

yScale = d3.scaleLinear()
            .domain([0, 4030954])
            .range([0, 400]);

colorScale = d3.scaleLinear()
                .domain([0, 3000000, 4030954])
                .range(['blue', d3.rgb(0, 255, 0), 'red']);

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return fontSize + i*(barHeight + heightPadding);
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities);
rects.exit().remove();
rects.enter().append("rect");
d3.select("svg").selectAll("rect")
  .attr("x", 100)
  .attr("y", function(d, i){
      return i*(barHeight + heightPadding);
  })
  .attr("width", function(d, i){
      return yScale(d.population);
  })
  .attr("height", barHeight)
  .attr("fill", function(d,i){
    return colorScale(d.population);
  });
```

You can also use d3.rgb to indicate a color

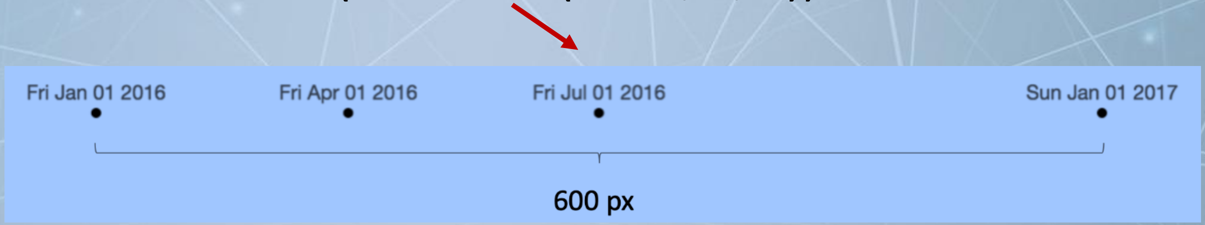
Try it

- Try to modify Ex04-2 by using different scale function

- More reading for d3 scale functions
  - https://observablehq.com/@d3/d3-scalelinear
  - This link is for scaleLinear. But it probably can give you more sense about how to utilize other d3 scale functions

d3.scaleTime()

- Similar to scaleLinear, but
  - The domain is expressed as an array of dates
- dateToWidth = d3.scaleTime().domain([new Date(2016, 0, 1), new Date(2017, 0, 1)]).range([0, 600])
- Ex: dateToWidth(new Date(2016, 6, 1))



- Try Ex04-4

d3.scaleSequential()

- Mapping continuous values to an output range determined by a present or custom interpolator
  - Useful to map to a **continuous** colormap
  - Colormap?

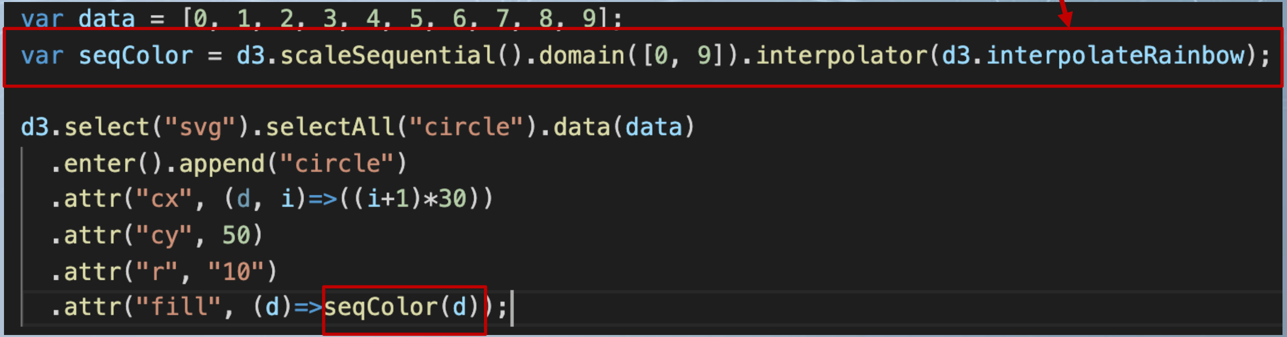
- d3.scaleSequential().domain(**DOMAIN**).interpolator(**INTERPOLATOR**)
  - INTERPOLATOR usually is a color map.
  - Ex: d3. interpolateBrBG, d3.interpolateRainbow …..
  - Check the d3 predefined color map here
    - https://github.com/d3/d3-scale-chromatic/blob/master/README.md

Ex04-5

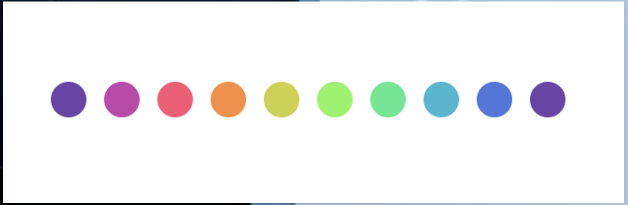
- Create 10 circles and color them by d3 interpolator

- main.js

Change here to use different continuous color map



```
var data = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
var seqColor = d3.scaleSequential().domain([0, 9]).interpolator(d3.interpolateRainbow);

d3.select("svg").selectAll("circle").data(data)
  .enter().append("circle")
  .attr("cx", (d, i)=>((i+1)*30))
  .attr("cy", 50)
  .attr("r", "10")
  .attr("fill", (d)=>seqColor(d));
```

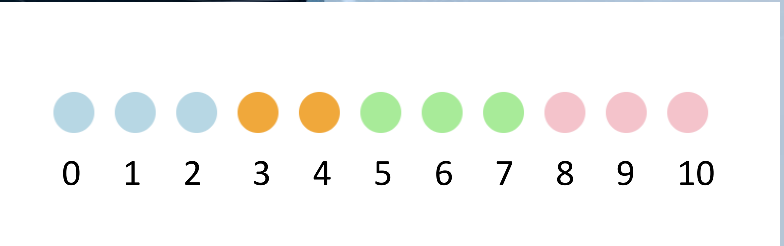


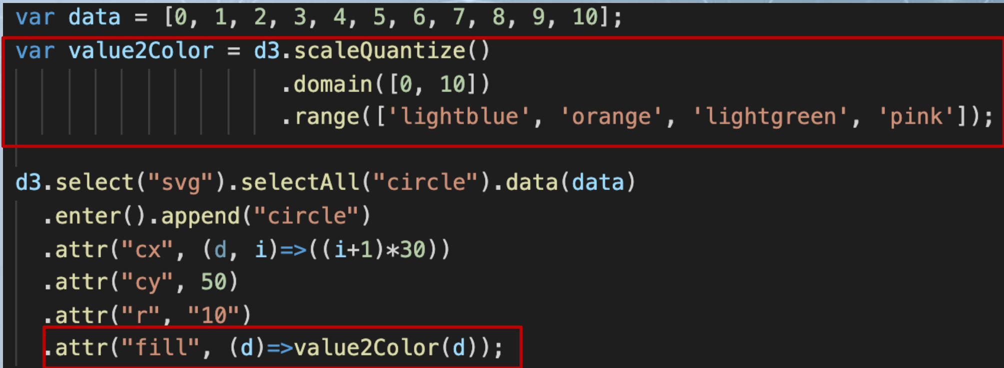
d3.scaleQuantize()

- Map continuous input to **discrete** output
- Ex04-6
  - domain value is [0, 10]
  - Range is 4 discrete value(color)
  - It will divide the domain into 4 intervals

In this example:

domain value < 2.5 is mapped to 'lightblue'
2.5<= domain value < 5.0 is mapped to 'orange'
5.0<= domain value < 7.5 is mapped to 'lightgreen'
7.5<= domain value is mapped to 'pink'

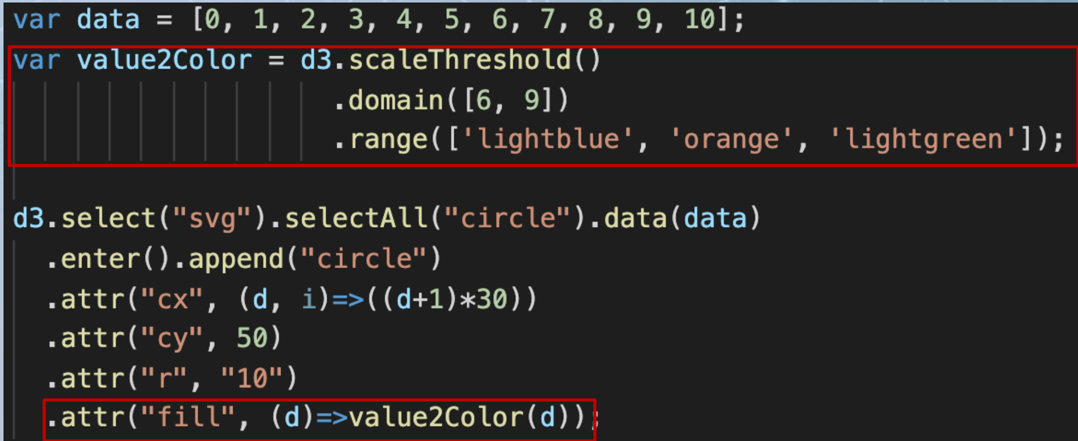




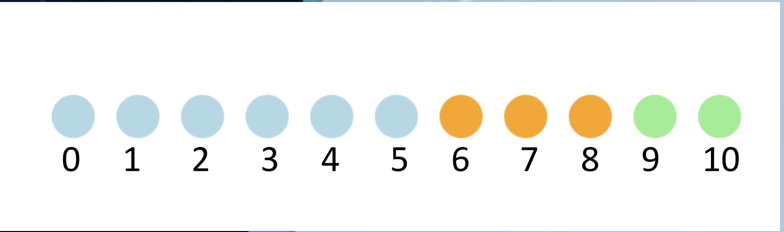
```
var data = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
var value2Color = d3.scaleQuantize()
                    .domain([0, 10])
                    .range(['lightblue', 'orange', 'lightgreen', 'pink']);

d3.select("svg").selectAll("circle").data(data)
  .enter().append("circle")
  .attr("cx", (d, i)=>((i+1)*30))
  .attr("cy", 50)
  .attr("r", "10")
  .attr("fill", (d)=>value2Color(d));
```

d3.scaleThreshold()

- Map arbitrary subsets of the domain to discrete values in the range
- Ex04-7
  - 6 is the boundary between ‘lightblue’ and ‘orange’
  - 9 is the boundary between ‘orange’ and ‘lightgreen’
  - So, it maps
    - domain value < 6 to ‘lightblue’
    - 6<=domain value < 9 to ‘orange’
    - 9<=domain value to ‘lightgreen’

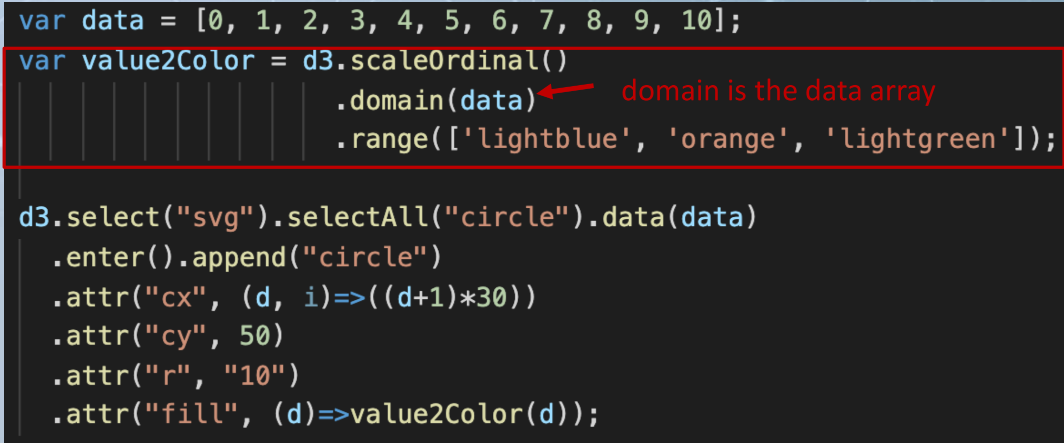


```
var data = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
var value2Color = d3.scaleThreshold()
                    .domain([6, 9])
                    .range(['lightblue', 'orange', 'lightgreen']);

d3.select("svg").selectAll("circle").data(data)
  .enter().append("circle")
  .attr("cx", (d, i)=>((d+1)*30))
  .attr("cy", 50)
  .attr("r", "10")
  .attr("fill", (d)=>value2Color(d));
```

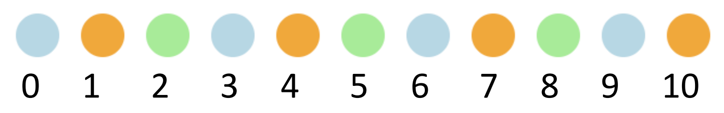


d3.scaleOrdinal()

- **Discrete** input to **discrete** output
- d3.scaleOrdinal maps the 1st, 2nd ,3rd … values in domain to the 1st, 2nd ,3rd … value in range, respectively
  - The range array will repeat if it is shorter than input array
- Ex04-8

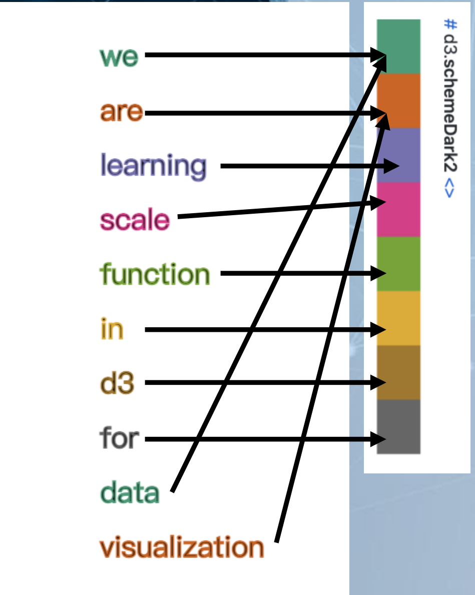


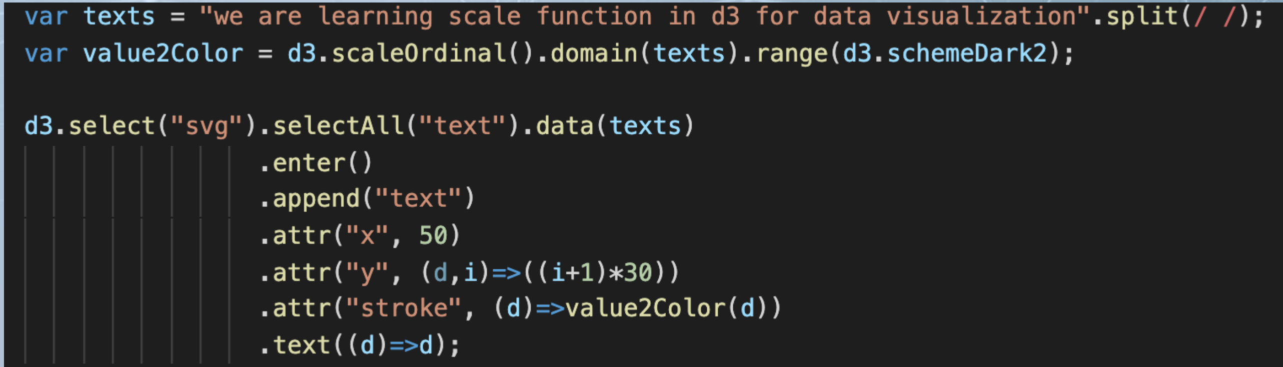
```
var data = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
var value2Color = d3.scaleOrdinal()
                        .domain(data)
                        .range(['lightblue', 'orange', 'lightgreen']);

d3.select("svg").selectAll("circle").data(data)
  .enter().append("circle")
  .attr("cx", (d, i)=>((d+1)*30))
  .attr("cy", 50)
  .attr("r", "10")
  .attr("fill", (d)=>value2Color(d));
```



Ex04-9

- One more example for d3.scaleOrdinal()
- Use D3 built-in color map
  - https://github.com/d3/d3-scale-chromatic/blob/master/README.md#categorical (categorical section)
- d3.scaleOrdinal maps the 1st, 2nd ,3rd … values in domain to the 1st, 2nd ,3rd … values in range, respectively

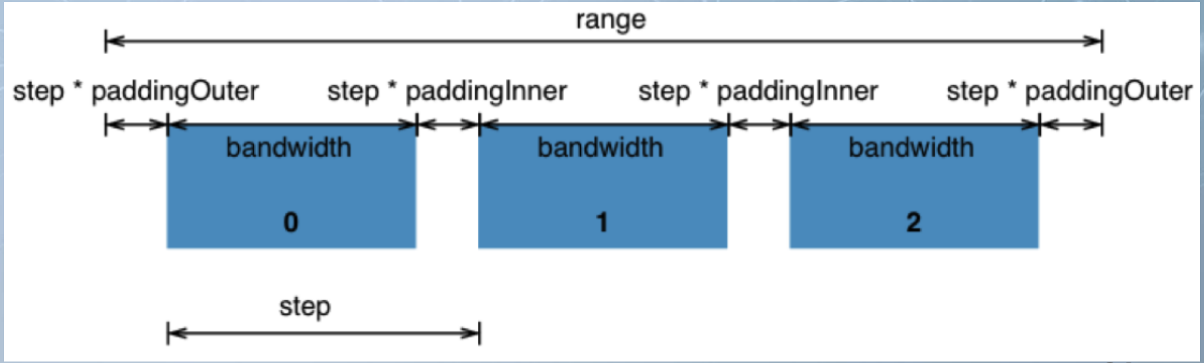



```
var texts = "we are learning scale function in d3 for data visualization".split(/ /);
var value2Color = d3.scaleOrdinal().domain(texts).range(d3.schemeDark2);

d3.select("svg").selectAll("text").data(texts)
                .enter()
                .append("text")
                .attr("x", 50)
                .attr("y", (d,i)=>((i+1)*30))
                .attr("stroke", (d)=>value2Color(d))
                .text((d)=>d);
```

d3.scaleBand()

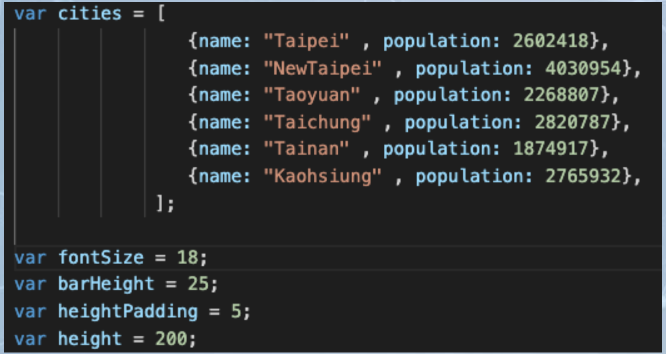
- Discrete output values are automatically computed by the scale by dividing the continuous range into uniform bands
  - Band scales are typically used for bar charts with an ordinal or categorical dimension
- d3.scaleBand.domain().range().paddingOuter().paddingInner();

paddingOuter and paddingInner are ratio of "step"
So, they are greater than 0 and smaller than 1

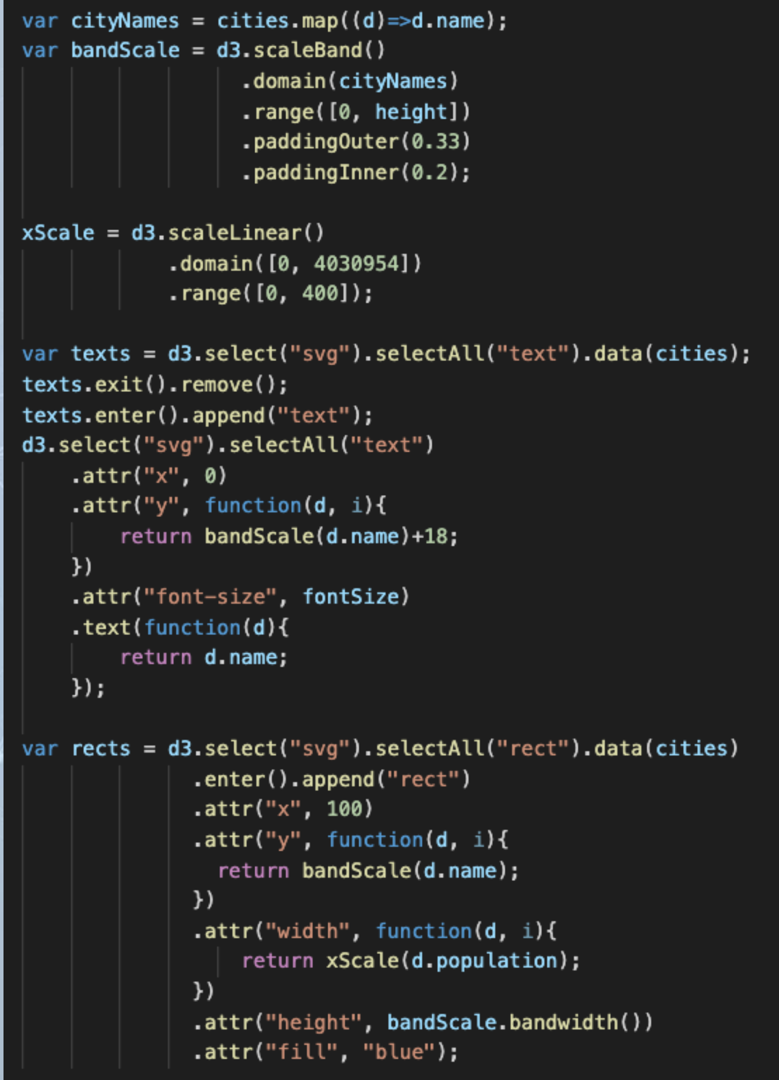


Ex04-10

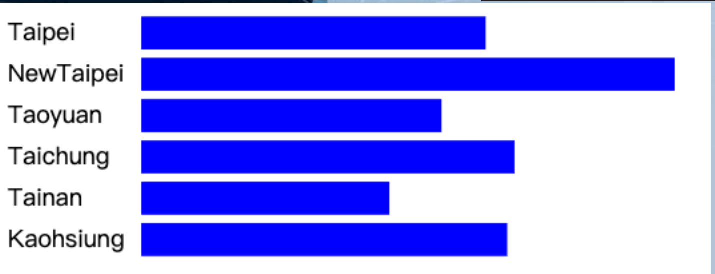
- A classic use of d3.scaleBand() – bar chart
  - Modify from Ex04-10



```
var cities = [
                {name: "Taipei" , population: 2602418},
                {name: "NewTaipei" , population: 4030954},
                {name: "Taoyuan" , population: 2268807},
                {name: "Taichung" , population: 2820787},
                {name: "Tainan" , population: 1874917},
                {name: "Kaohsiung" , population: 2765932},
            ];

var fontSize = 18;
var barHeight = 25;
var heightPadding = 5;
var height = 200;
```

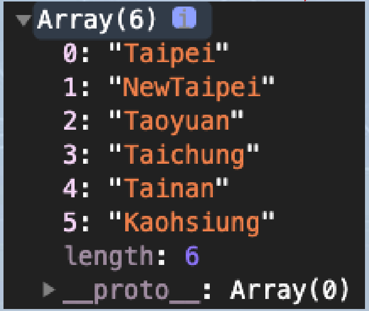


```
var cityNames = cities.map((d)=>d.name);
var bandScale = d3.scaleBand()
                    .domain(cityNames)
                    .range([0, height])
                    .paddingOuter(0.33)
                    .paddingInner(0.2);

xScale = d3.scaleLinear()
            .domain([0, 4030954])
            .range([0, 400]);

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return bandScale(d.name)+18;
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities)
                .enter().append("rect")
                .attr("x", 100)
                .attr("y", function(d, i){
                  return bandScale(d.name);
                })
                .attr("width", function(d, i){
                    return xScale(d.population);
                })
                .attr("height", bandScale.bandwidth())
                .attr("fill", "blue");
```

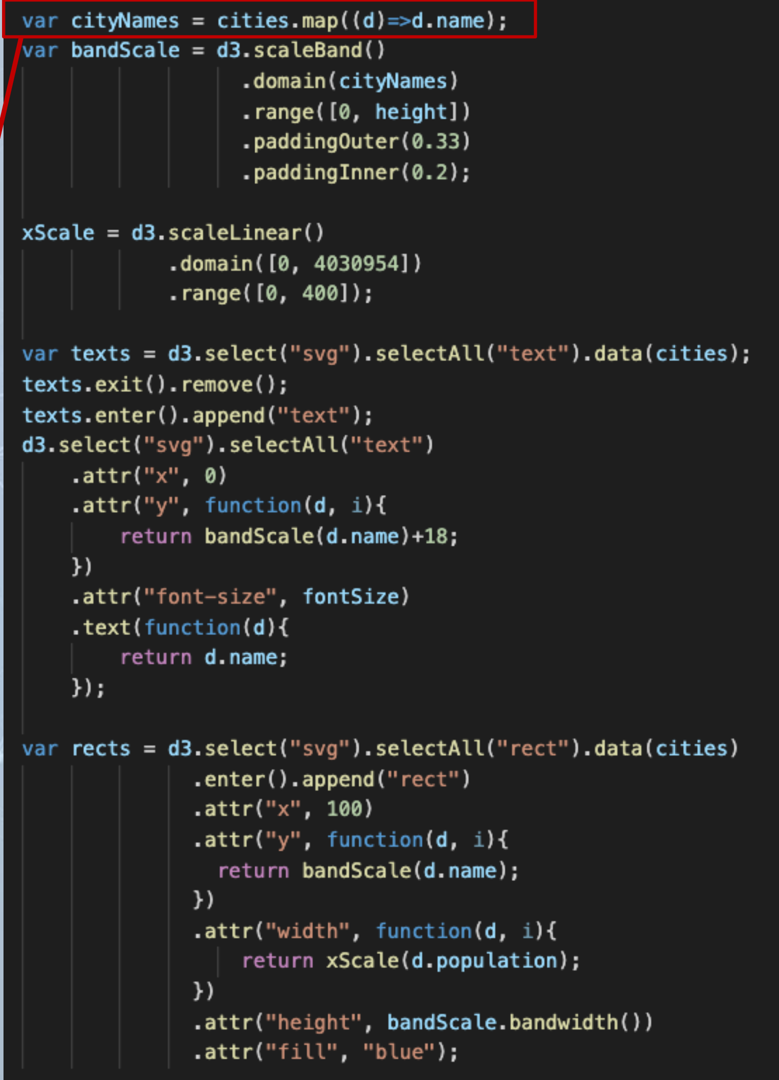


Ex04-10

- A classic use of d3.scaleBand() – bar chart
  - Modify from Ex04-10



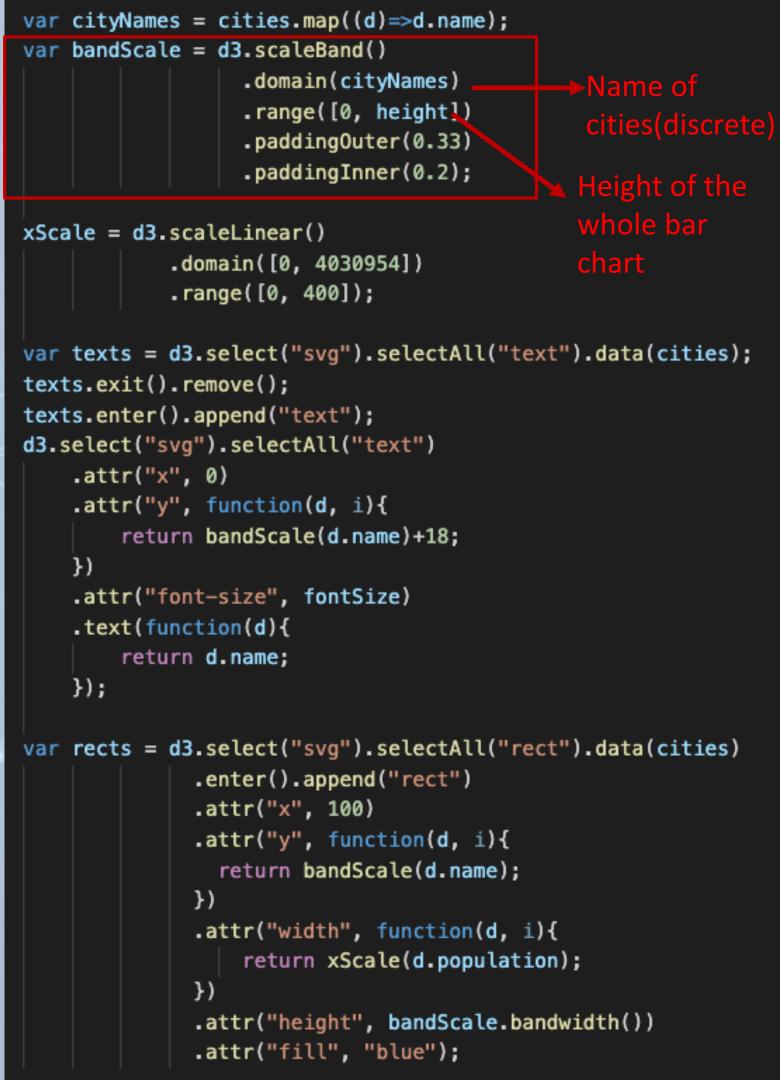
```
▼Array(6)
  0: "Taipei"
  1: "NewTaipei"
  2: "Taoyuan"
  3: "Taichung"
  4: "Tainan"
  5: "Kaohsiung"
  length: 6
 ▶__proto__: Array(0)
```



```
var cityNames = cities.map((d)=>d.name);
var bandScale = d3.scaleBand()
                    .domain(cityNames)
                    .range([0, height])
                    .paddingOuter(0.33)
                    .paddingInner(0.2);

xScale = d3.scaleLinear()
                .domain([0, 4030954])
                .range([0, 400]);

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return bandScale(d.name)+18;
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities)
                .enter().append("rect")
                .attr("x", 100)
                .attr("y", function(d, i){
                  return bandScale(d.name);
                })
                .attr("width", function(d, i){
                    return xScale(d.population);
                })
                .attr("height", bandScale.bandwidth())
                .attr("fill", "blue");
```

Ex04-10

- A classic use of d3.scaleBand() – bar chart
  - Modify from Ex04-10

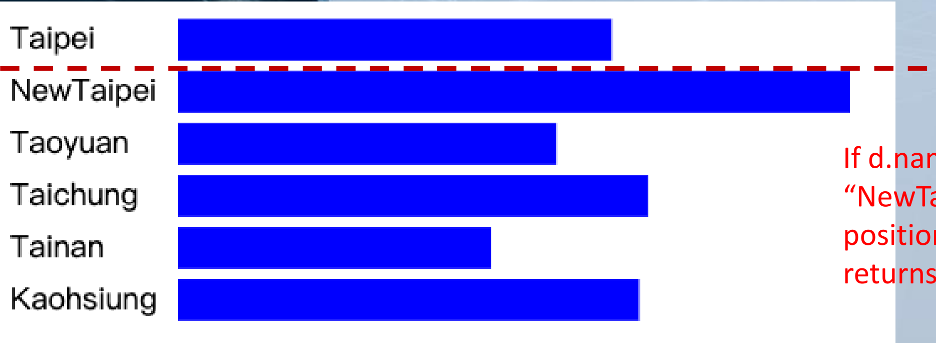


```
var cityNames = cities.map((d)=>d.name);
var bandScale = d3.scaleBand()
                    .domain(cityNames)
                    .range([0, height])
                    .paddingOuter(0.33)
                    .paddingInner(0.2);

xScale = d3.scaleLinear()
              .domain([0, 4030954])
              .range([0, 400]);

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return bandScale(d.name)+18;
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities)
                .enter().append("rect")
                .attr("x", 100)
                .attr("y", function(d, i){
                  return bandScale(d.name);
                })
                .attr("width", function(d, i){
                    return xScale(d.population);
                })
                .attr("height", bandScale.bandwidth())
                .attr("fill", "blue");
```

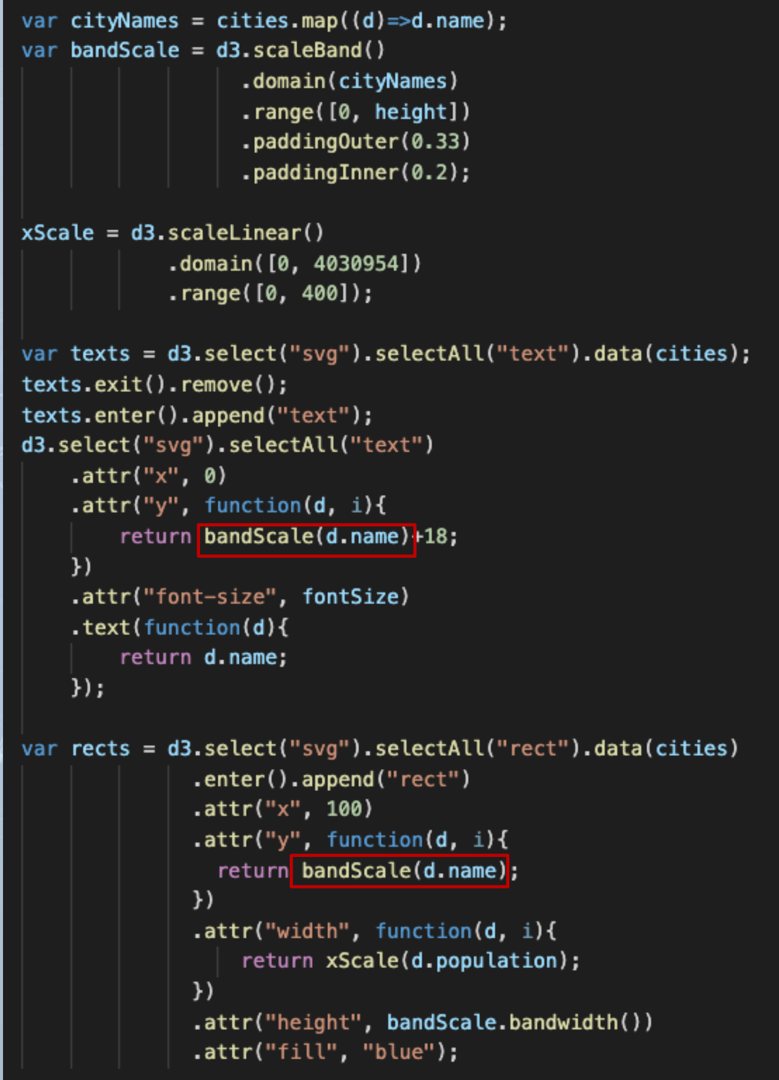
Ex04-10

- A classic use of d3.scaleBand() – bar chart
  - Modify from Ex04-10

Give bandScale() a city name, it returns you the starting height of the bar



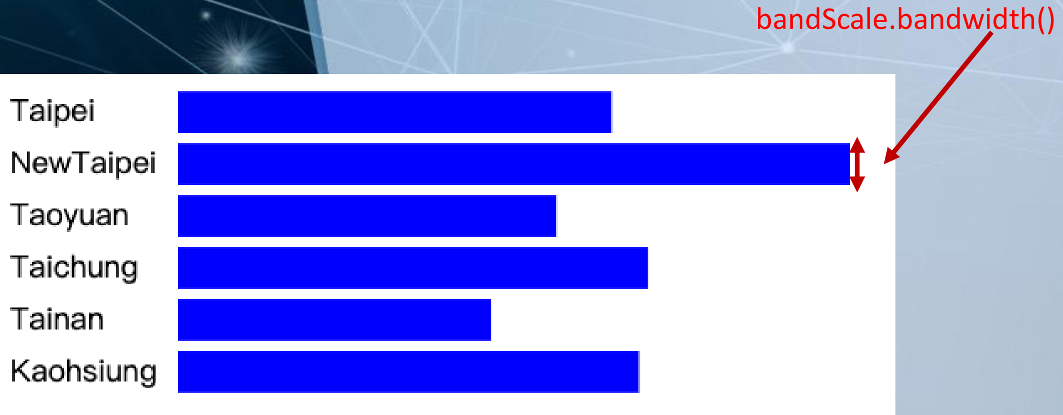
If d.name is "NewTaipei", this y position is what it returns

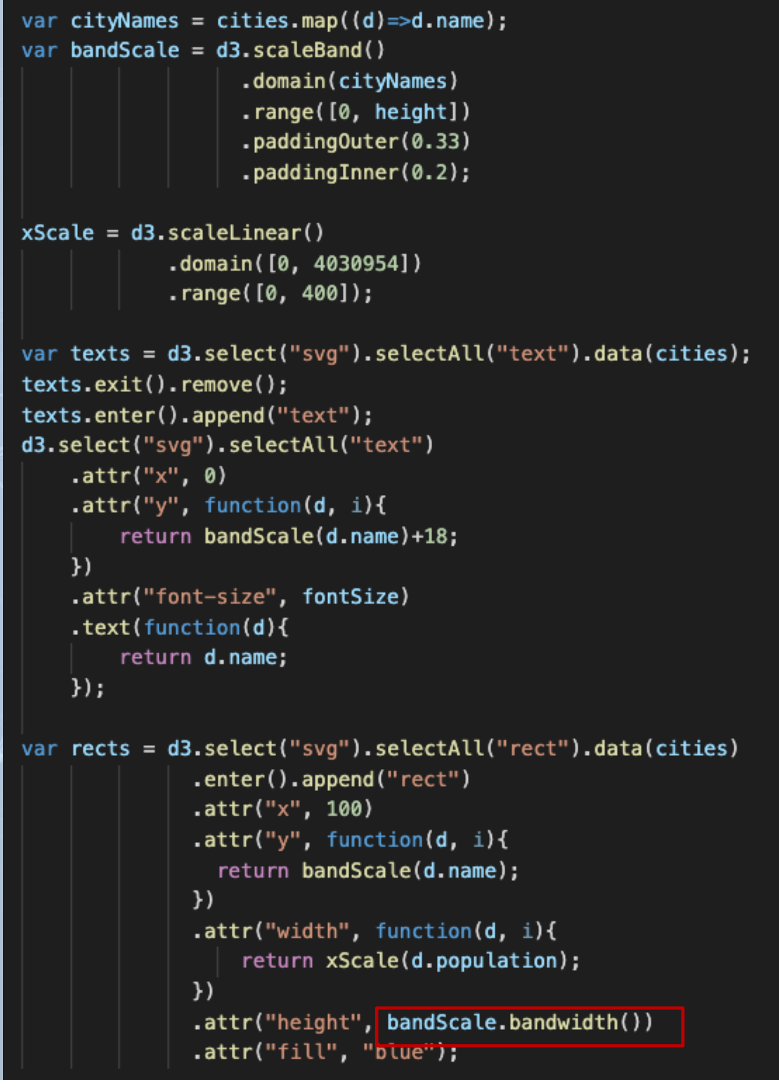


```
var cityNames = cities.map((d)=>d.name);
var bandScale = d3.scaleBand()
                  .domain(cityNames)
                  .range([0, height])
                  .paddingOuter(0.33)
                  .paddingInner(0.2);

xScale = d3.scaleLinear()
            .domain([0, 4030954])
            .range([0, 400]);

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return bandScale(d.name)+18;
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities)
                .enter().append("rect")
                .attr("x", 100)
                .attr("y", function(d, i){
                  return bandScale(d.name);
                })
                .attr("width", function(d, i){
                    return xScale(d.population);
                })
                .attr("height", bandScale.bandwidth())
                .attr("fill", "blue");
```

Ex04-10

- A classic use of d3.scaleBand() – bar chart
  - Modify from Ex04-10



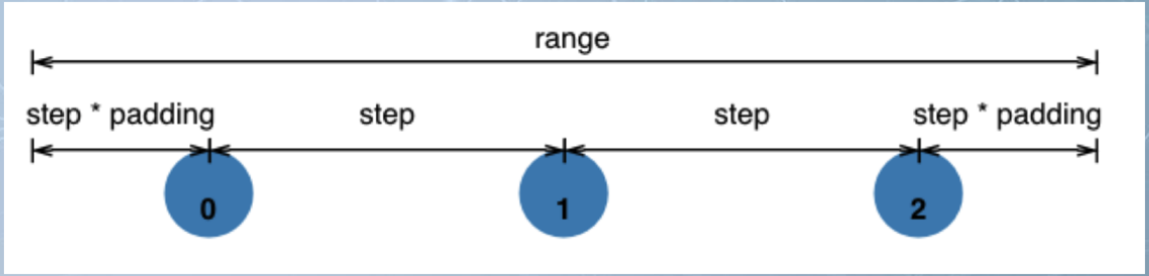


```
var cityNames = cities.map((d)=>d.name);
var bandScale = d3.scaleBand()
                    .domain(cityNames)
                    .range([0, height])
                    .paddingOuter(0.33)
                    .paddingInner(0.2);

xScale = d3.scaleLinear()
              .domain([0, 4030954])
              .range([0, 400]);

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return bandScale(d.name)+18;
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("rect").data(cities)
                .enter().append("rect")
                .attr("x", 100)
                .attr("y", function(d, i){
                  return bandScale(d.name);
                })
                .attr("width", function(d, i){
                    return xScale(d.population);
                })
                .attr("height", bandScale.bandwidth())
                .attr("fill", "blue");
```

d3.scalePoint()

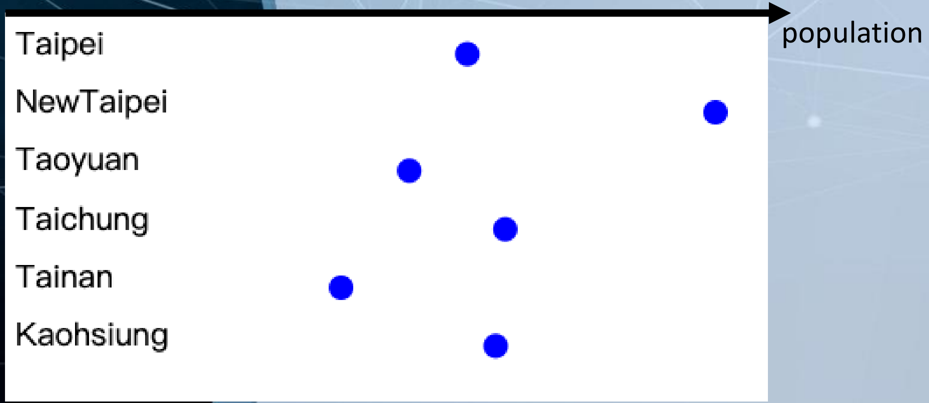
- Point scales are a variant of band scales with the bandwidth fixed to 0
- d3.scalePoint().domain().range().padding();

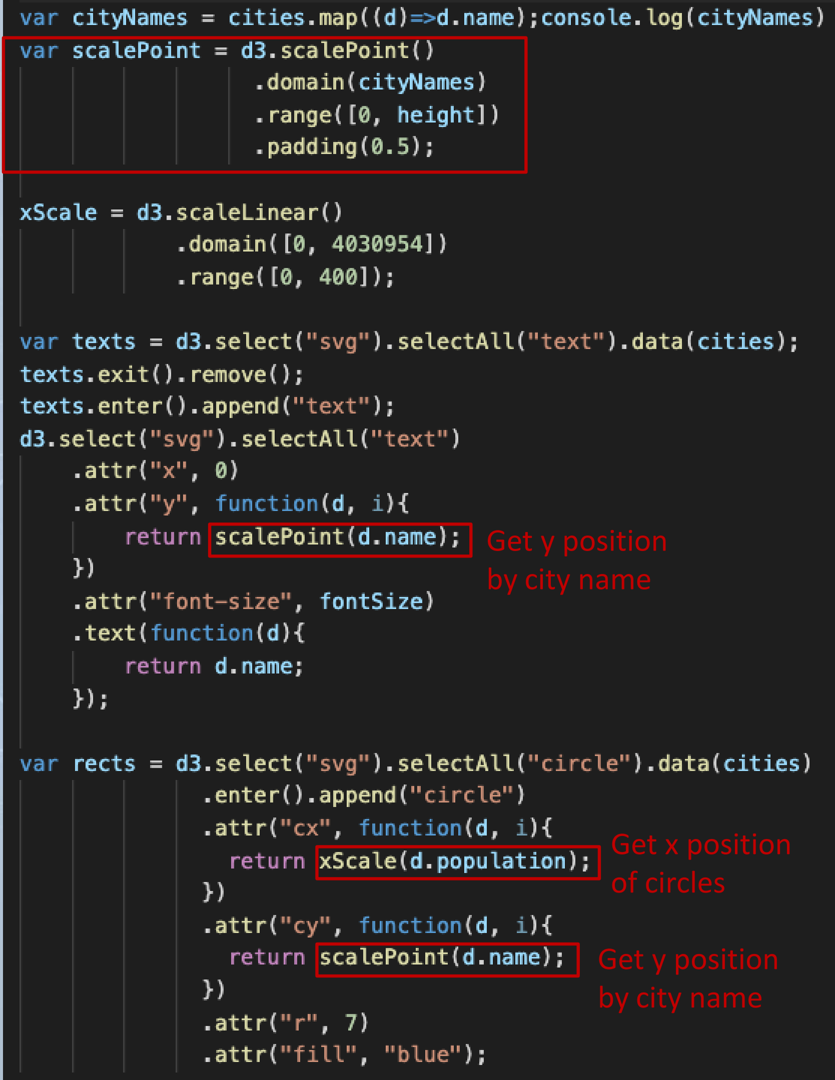
padding is also a ratio



Ex04-11

- A classic use of d3.scaleBand() – scatter plot





```
var cityNames = cities.map((d)=>d.name);console.log(cityNames)
var scalePoint = d3.scalePoint()
                    .domain(cityNames)
                    .range([0, height])
                    .padding(0.5);

xScale = d3.scaleLinear()
                .domain([0, 4030954])
                .range([0, 400]);

var texts = d3.select("svg").selectAll("text").data(cities);
texts.exit().remove();
texts.enter().append("text");
d3.select("svg").selectAll("text")
    .attr("x", 0)
    .attr("y", function(d, i){
        return scalePoint(d.name);
    })
    .attr("font-size", fontSize)
    .text(function(d){
        return d.name;
    });

var rects = d3.select("svg").selectAll("circle").data(cities)
                .enter().append("circle")
                .attr("cx", function(d, i){
                  return xScale(d.population);
                })
                .attr("cy", function(d, i){
                  return scalePoint(d.name);
                })
                .attr("r", 7)
                .attr("fill", "blue");
```

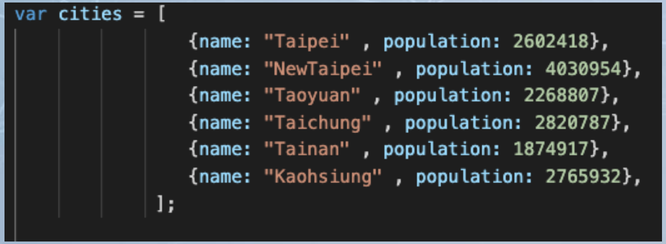
Get y position by city name

Get x position of circles

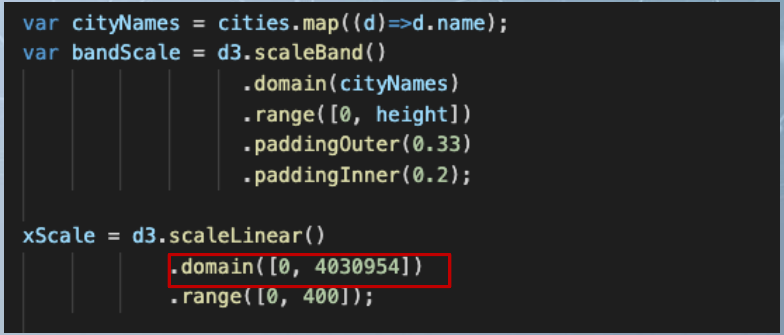
Get y position by city name

d3.min(), d3.max() and d3.extent()

- In previous examples, I always manually calculate the min, max and extent of my data.
  - If so, I always change my code when the data is updated



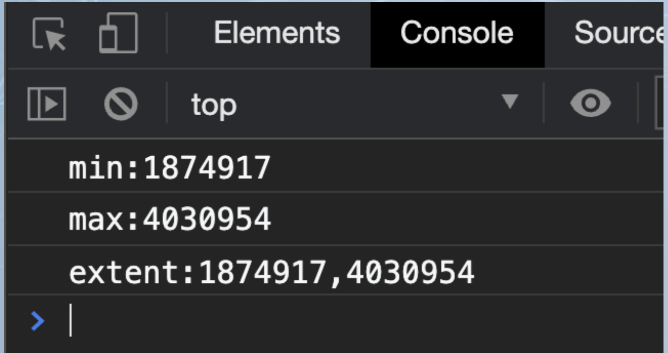
```
var cities = [
            {name: "Taipei" , population: 2602418},
            {name: "NewTaipei" , population: 4030954},
            {name: "Taoyuan" , population: 2268807},
            {name: "Taichung" , population: 2820787},
            {name: "Tainan" , population: 1874917},
            {name: "Kaohsiung" , population: 2765932},
        ];
```



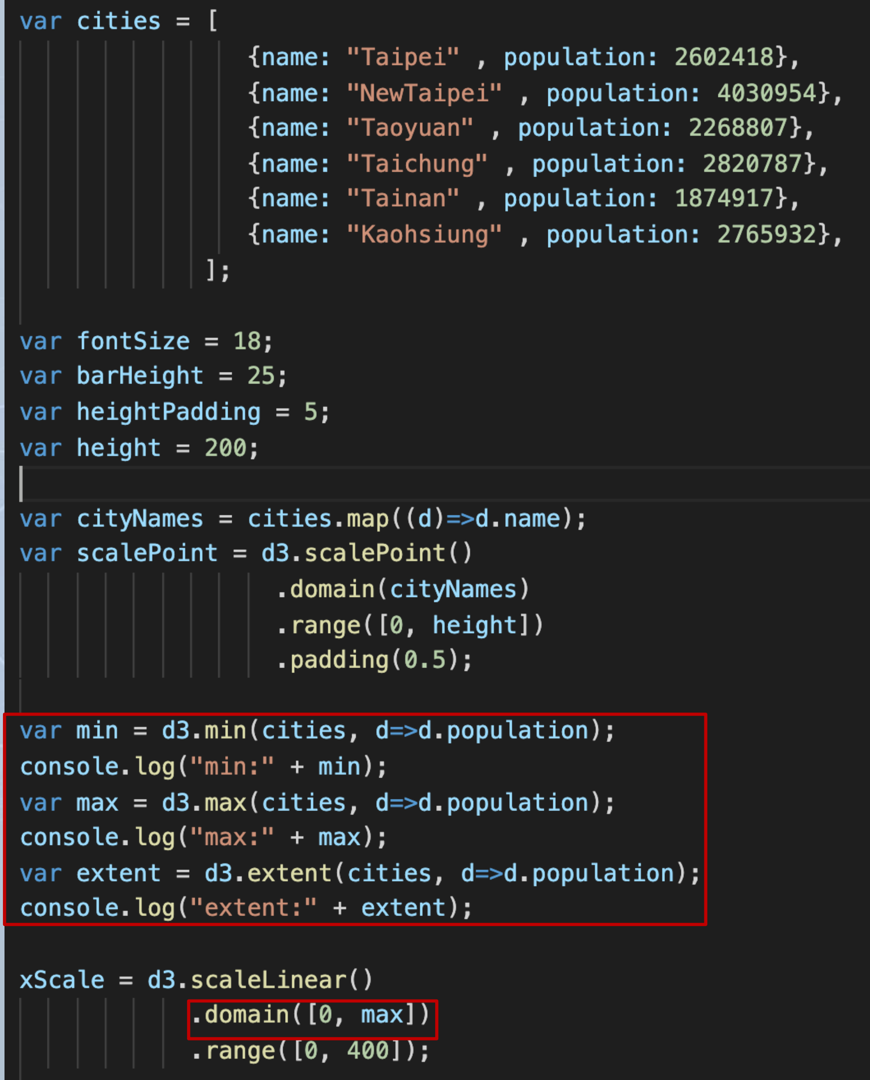
```
var cityNames = cities.map((d)=>d.name);
var bandScale = d3.scaleBand()
                    .domain(cityNames)
                    .range([0, height])
                    .paddingOuter(0.33)
                    .paddingInner(0.2);

xScale = d3.scaleLinear()
                .domain([0, 4030954])
                .range([0, 400]);
```

Ex04-12

- main.js



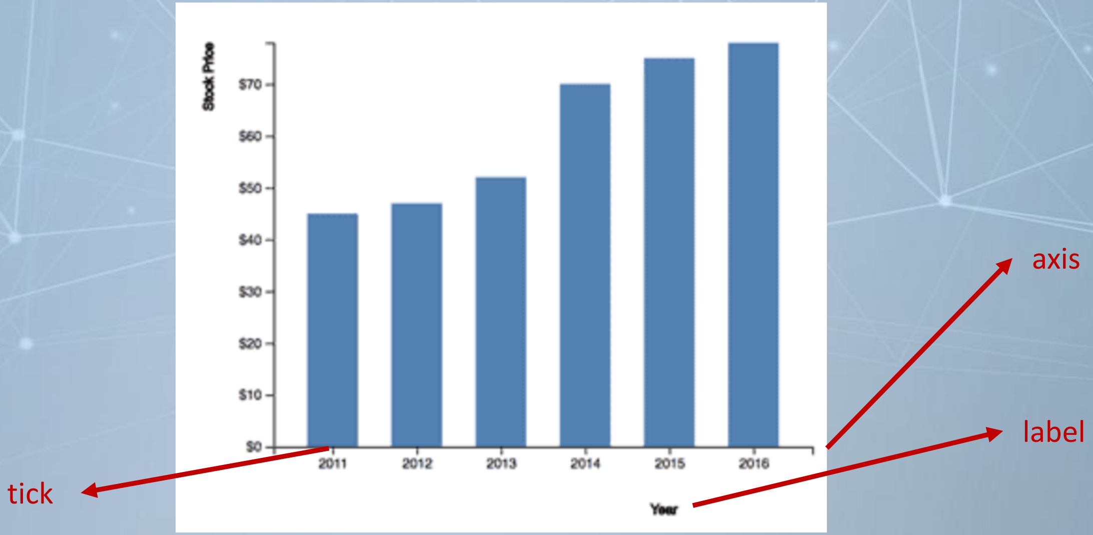
```
min:1874917
max:4030954
extent:1874917,4030954
```



```
var cities = [
                {name: "Taipei" , population: 2602418},
                {name: "NewTaipei" , population: 4030954},
                {name: "Taoyuan" , population: 2268807},
                {name: "Taichung" , population: 2820787},
                {name: "Tainan" , population: 1874917},
                {name: "Kaohsiung" , population: 2765932},
             ];

var fontSize = 18;
var barHeight = 25;
var heightPadding = 5;
var height = 200;

var cityNames = cities.map((d)=>d.name);
var scalePoint = d3.scalePoint()
                  .domain(cityNames)
                  .range([0, height])
                  .padding(0.5);

var min = d3.min(cities, d=>d.population);
console.log("min:" + min);
var max = d3.max(cities, d=>d.population);
console.log("max:" + max);
var extent = d3.extent(cities, d=>d.population);
console.log("extent:" + extent);

xScale = d3.scaleLinear()
            .domain([0, max])
            .range([0, 400]);
```

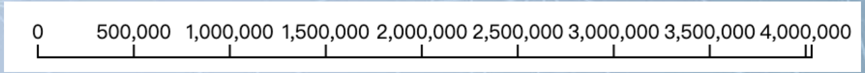
Axis, Tick and Label

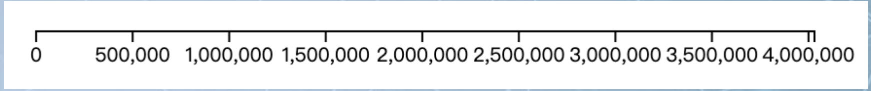
- The axis, tick and label are important for users to interpret the visualization

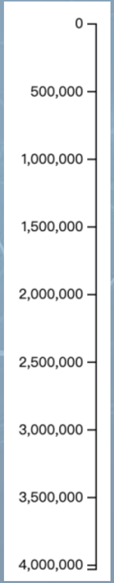


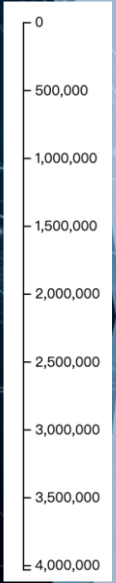
Axis

- Another important reason to use d3 scale function is that we can easily add the **axis** to the visualization
- Four functions to create different axes
  - d3.axisTop()
  - d3.axisButton()
  - d3.axisLeft()
  - d3.axisRight()
- Argument to these function? Scale function





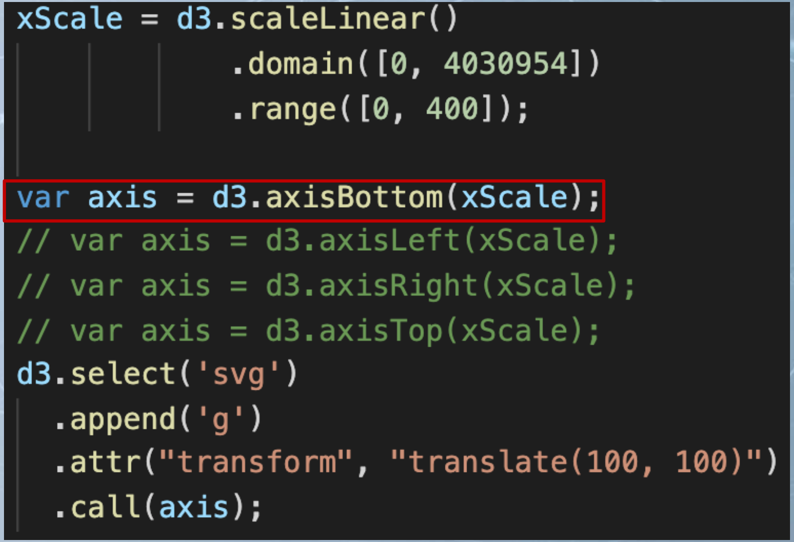




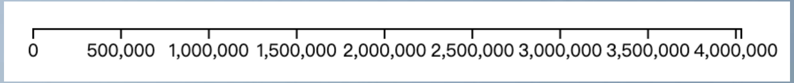
Ex04-13

- main.js

- **In this example, we create a button axis**
- It takes a scale function as the input argument.
  - domain in the scale function will be the data range to draw the ticks
  - The range in the scale function will be the length of the axis in pixel
- We usually add the axis to a ‘g’ in svg and use transform to move it to where we want
  - How to show the axis? “.call(axis)”



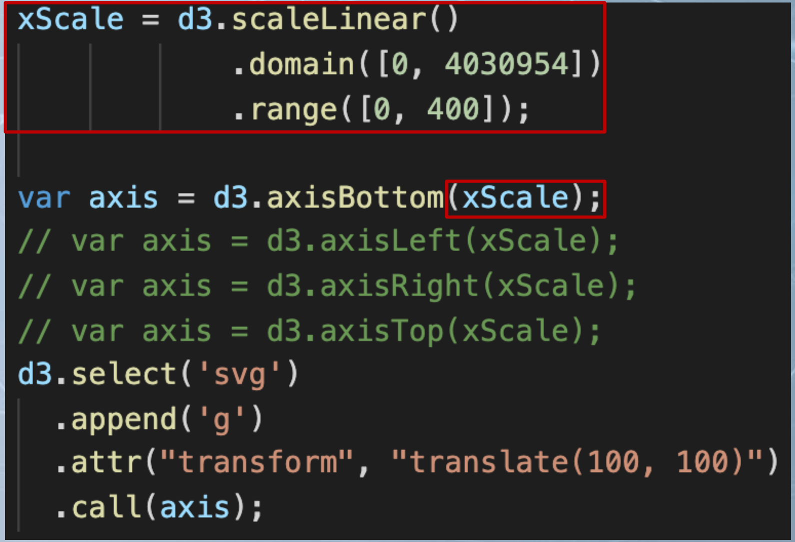
```
xScale = d3.scaleLinear()
                .domain([0, 4030954])
                .range([0, 400]);

var axis = d3.axisBottom(xScale);
// var axis = d3.axisLeft(xScale);
// var axis = d3.axisRight(xScale);
// var axis = d3.axisTop(xScale);
d3.select('svg')
  .append('g')
  .attr("transform", "translate(100, 100)")
  .call(axis);
```



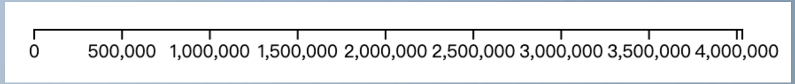
Ex04-13

- main.js

- In this example, we create a button axis
- **It takes a scale function as the input argument.**
  - domain in the scale function will be the data range to draw the ticks
  - The range in the scale function will be the length of the axis in pixel
- We usually add the axis to a 'g' in svg and use transform to move it to where we want
  - How to show the axis? ".call(axis)"



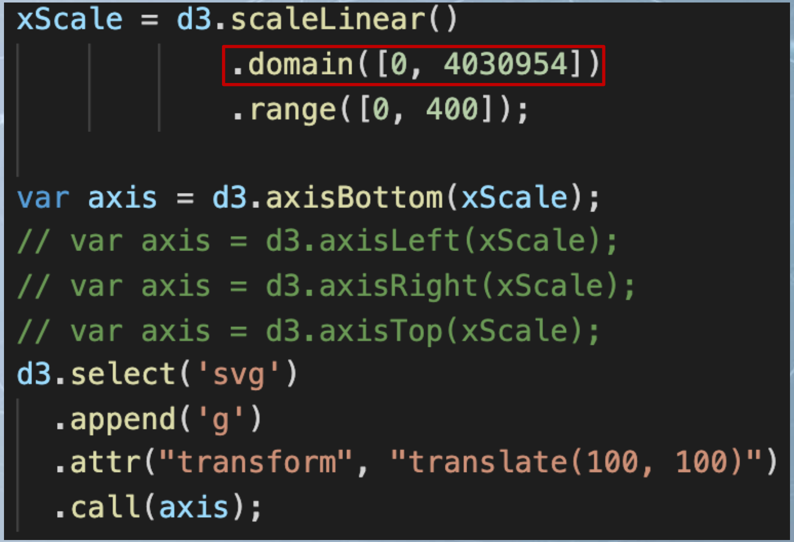
```
xScale = d3.scaleLinear()
            .domain([0, 4030954])
            .range([0, 400]);

var axis = d3.axisBottom(xScale);
// var axis = d3.axisLeft(xScale);
// var axis = d3.axisRight(xScale);
// var axis = d3.axisTop(xScale);
d3.select('svg')
  .append('g')
  .attr("transform", "translate(100, 100)")
  .call(axis);
```



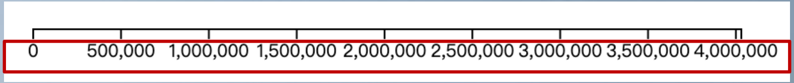
Ex04-13

- main.js

- In this example, we create a button axis
- It takes a scale function as the input argument.
  - **domain in the scale function will be the data range to draw the ticks**
  - The range in the scale function will be the length of the axis in pixel
- We usually add the axis to a ‘g’ in svg and use transform to move it to where we want
  - How to show the axis? “.call(axis)”



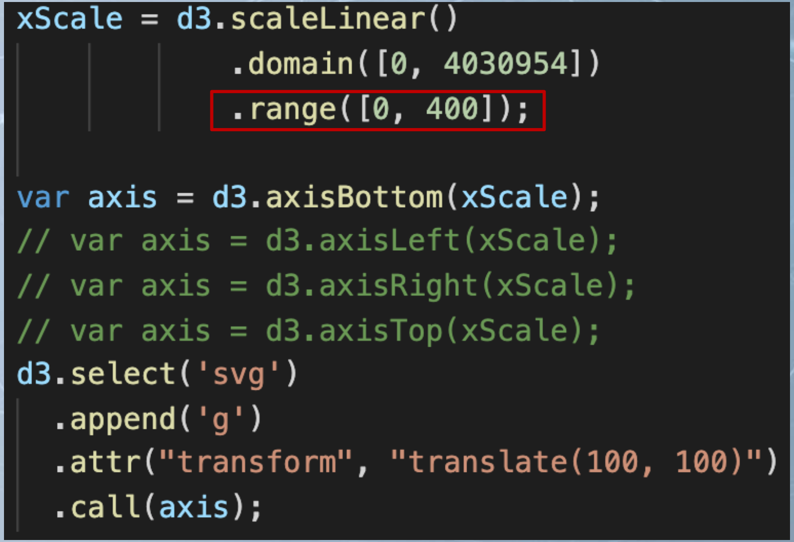
```
xScale = d3.scaleLinear()
                .domain([0, 4030954])
                .range([0, 400]);

var axis = d3.axisBottom(xScale);
// var axis = d3.axisLeft(xScale);
// var axis = d3.axisRight(xScale);
// var axis = d3.axisTop(xScale);
d3.select('svg')
  .append('g')
  .attr("transform", "translate(100, 100)")
  .call(axis);
```



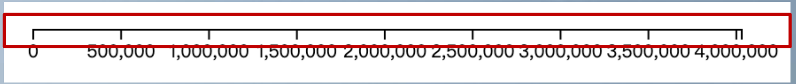
Ex04-13

- main.js

- In this example, we create a button axis
- It takes a scale function as the input argument.
  - domain in the scale function will be the data range to draw the ticks
  - **The range in the scale function will be the length of the axis in pixel**
- We usually add the axis to a 'g' in svg and use transform to move it to where we want
  - How to show the axis? ".call(axis)"



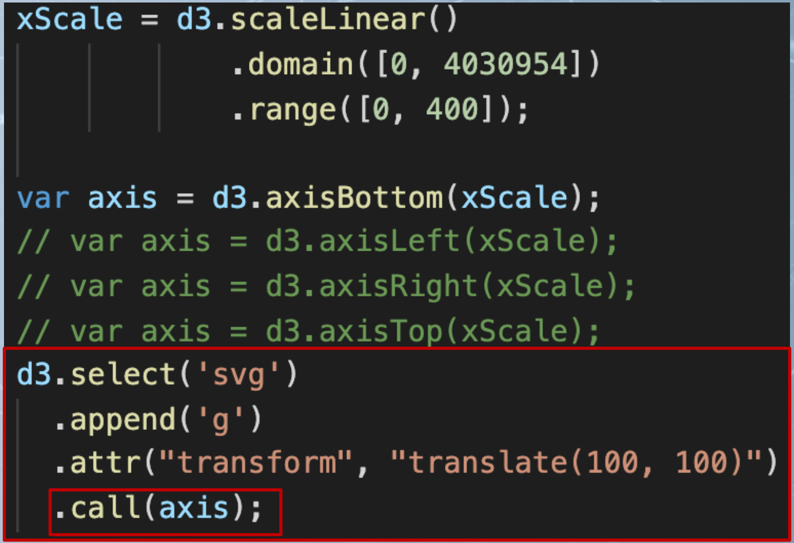
```
xScale = d3.scaleLinear()
            .domain([0, 4030954])
            .range([0, 400]);

var axis = d3.axisBottom(xScale);
// var axis = d3.axisLeft(xScale);
// var axis = d3.axisRight(xScale);
// var axis = d3.axisTop(xScale);
d3.select('svg')
  .append('g')
  .attr("transform", "translate(100, 100)")
  .call(axis);
```



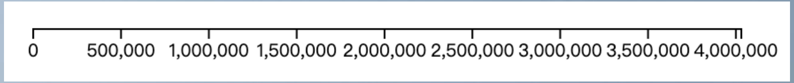
Ex04-13

- main.js

- **In this example, we create a button axis**
- It takes a scale function as the input argument.
  - domain in the scale function will be the data range to draw the ticks
  - The range in the scale function will be the length of the axis in pixel
- **We usually add the axis to a 'g' in svg and use transform to move it to where we want**
  - **How to show the axis? ".call(axis)"**



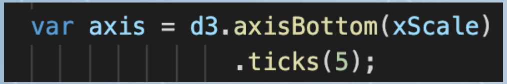
```
xScale = d3.scaleLinear()
            .domain([0, 4030954])
            .range([0, 400]);

var axis = d3.axisBottom(xScale);
// var axis = d3.axisLeft(xScale);
// var axis = d3.axisRight(xScale);
// var axis = d3.axisTop(xScale);
d3.select('svg')
  .append('g')
  .attr("transform", "translate(100, 100)")
  .call(axis);
```



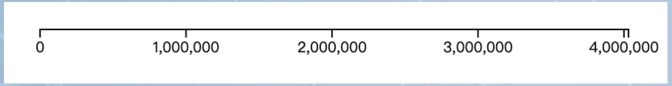
Ticks

- Check Ex04-14

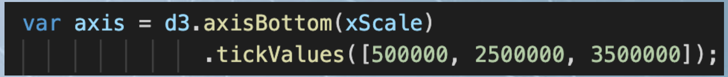
Set number of ticks



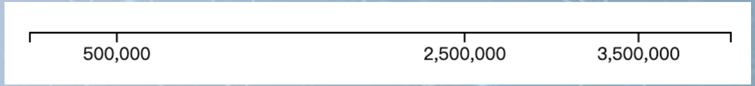
```
var axis = d3.axisBottom(xScale)
             .ticks(5);
```



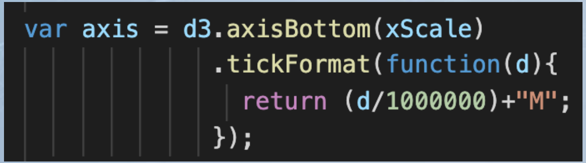
Explicit Tick Values



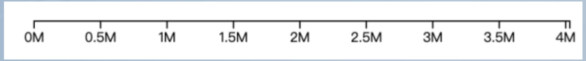
```
var axis = d3.axisBottom(xScale)
             .tickValues([500000, 2500000, 3500000]);
```



Text Format



```
var axis = d3.axisBottom(xScale)
             .tickFormat(function(d){
               return (d/1000000)+"M";
             });
```

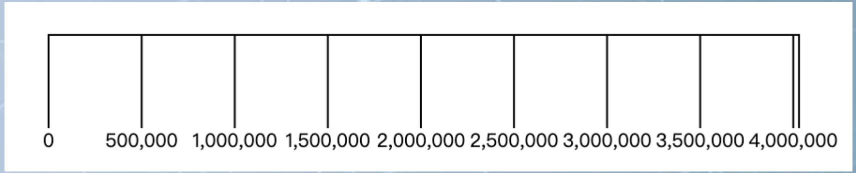


Tick Size

- Check Ex05-15

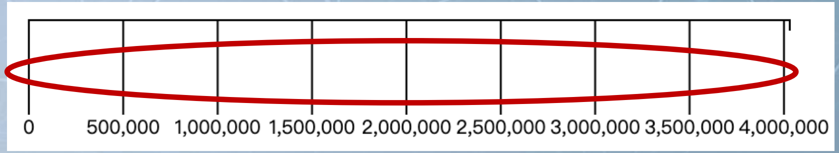
Set all tick size

```
var axis = d3.axisBottom(xScale)
             .tickSize(50);
```



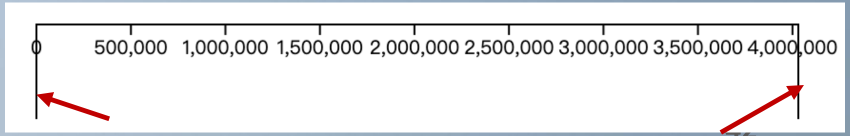
Set inner tick size

```
var axis = d3.axisBottom(xScale)
             .tickSizeInner(50);
```



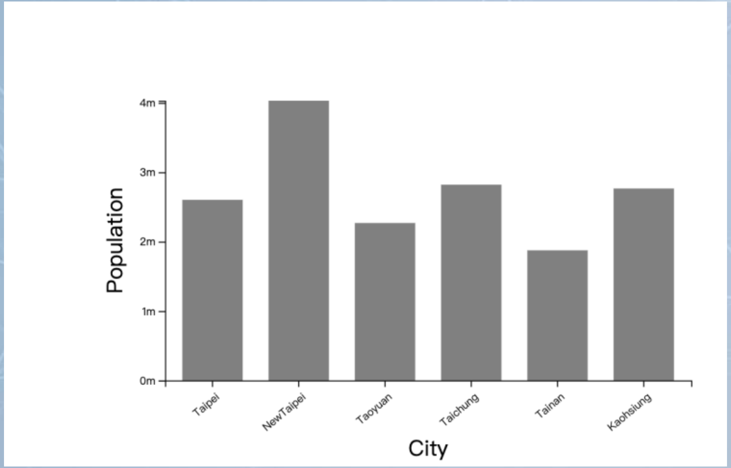
Set outer tick size

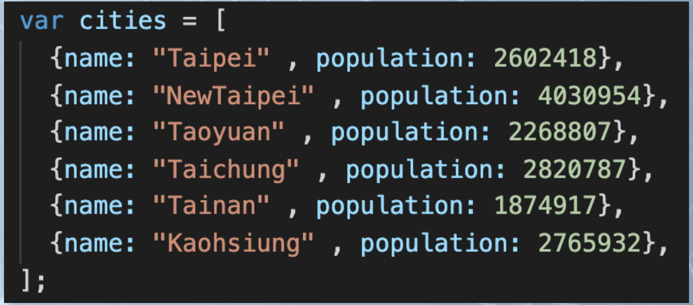
```
var axis = d3.axisBottom(xScale)
             .tickSizeOuter(50);
```



Ex04-16

- Make a bar chart



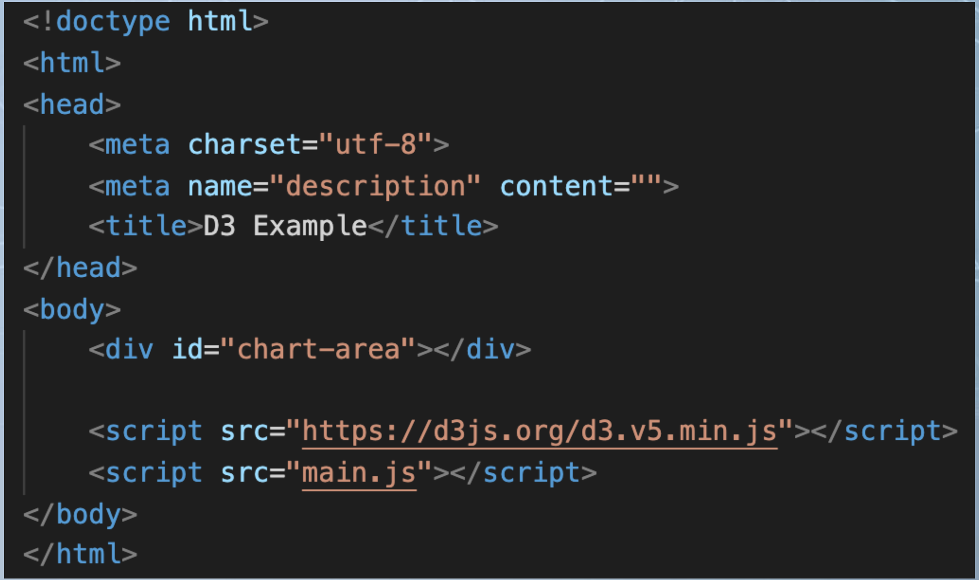


```
var cities = [
  {name: "Taipei" , population: 2602418},
  {name: "NewTaipei" , population: 4030954},
  {name: "Taoyuan" , population: 2268807},
  {name: "Taichung" , population: 2820787},
  {name: "Tainan" , population: 1874917},
  {name: "Kaohsiung" , population: 2765932},
];
```

Our data in main.js

Ex04-16

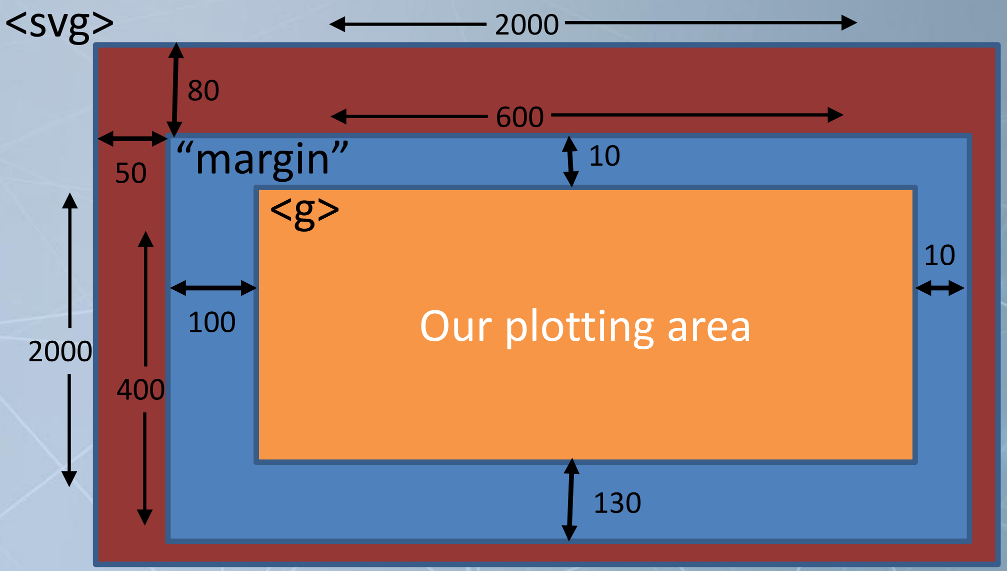
- index.js
  - Now, our index.html only has a <div>. We will append svg to it in main.js

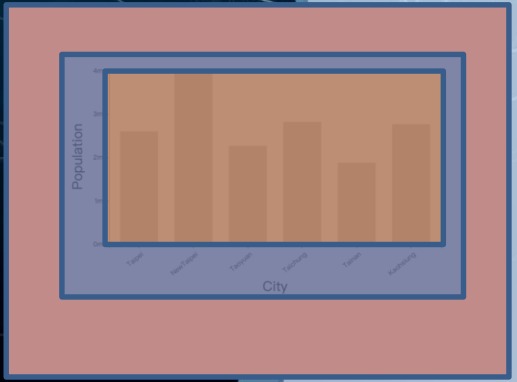


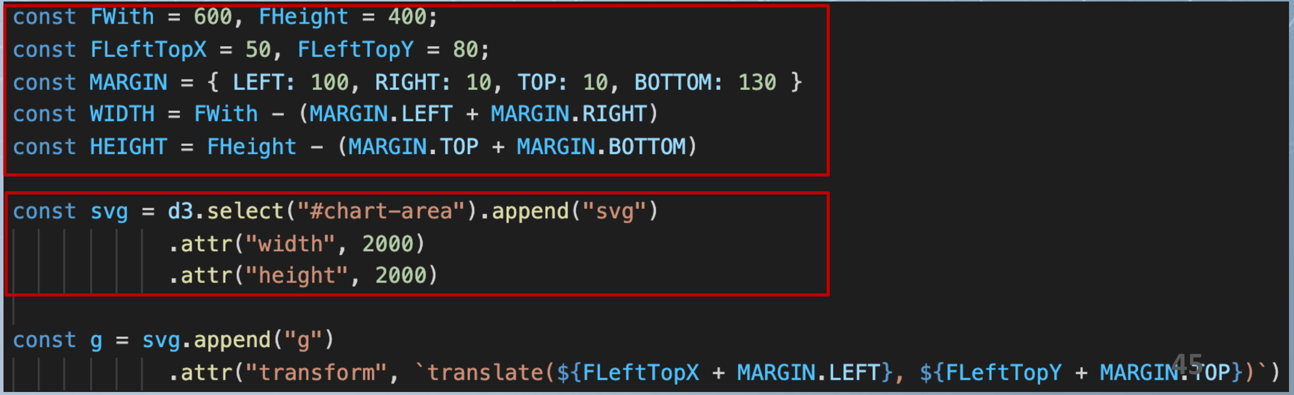
```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="description" content="">
    <title>D3 Example</title>
</head>
<body>
    <div id="chart-area"></div>

    <script src="https://d3js.org/d3.v5.min.js"></script>
    <script src="main.js"></script>
</body>
</html>
```

Ex04-16

- main.js
  - We usually use variables to set the width and height of the svg, and the margin of our plot
- Why margin?
  - You may have multiple plots on the webpage and you may want to make/adjust the distance between them later



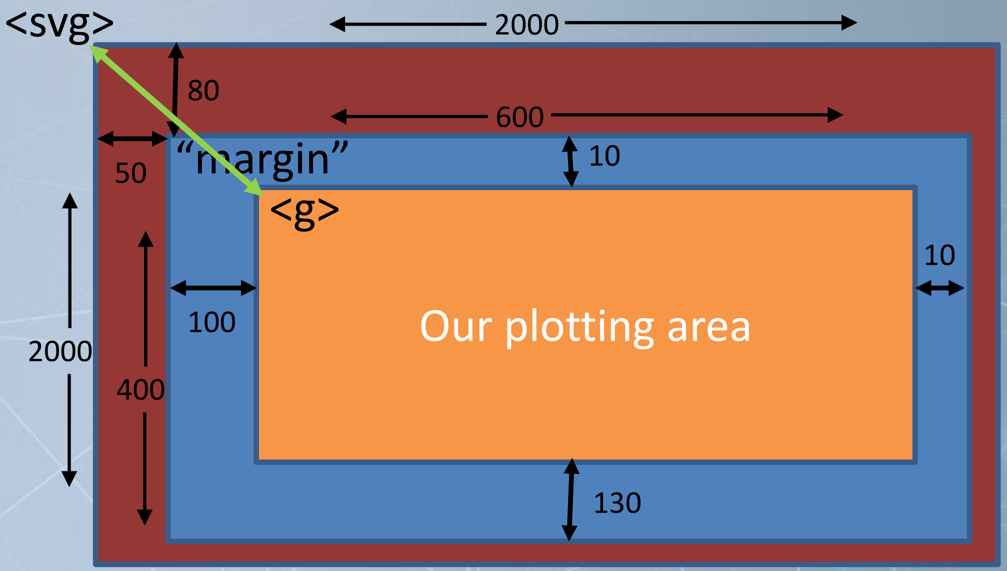


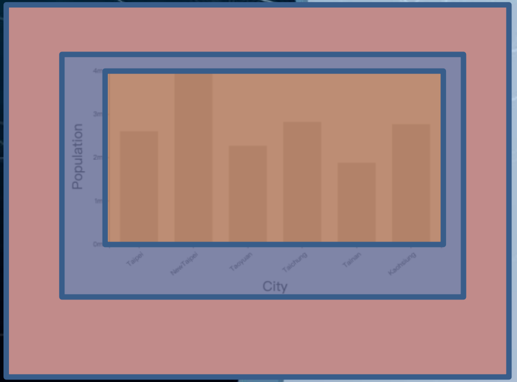


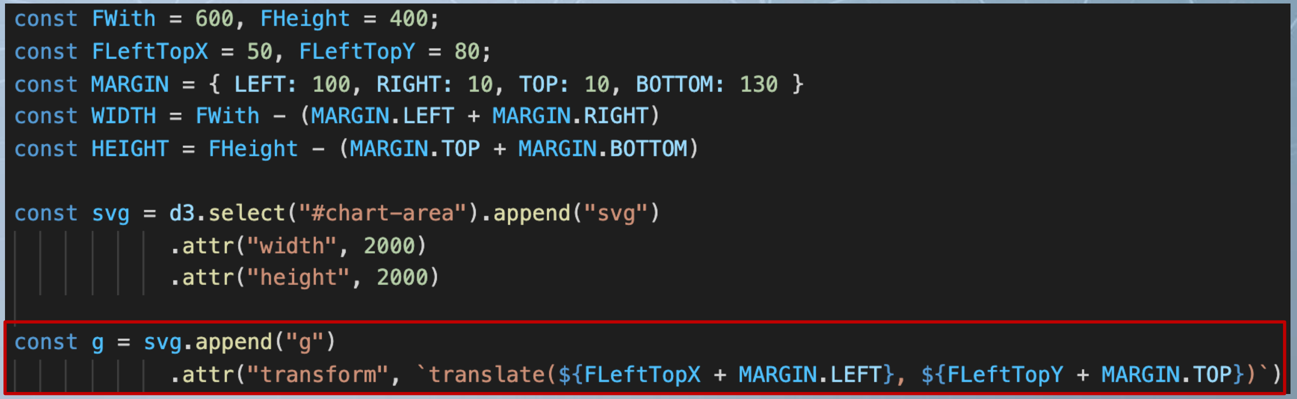
```
const FWith = 600, FHeight = 400;
const FLeftTopX = 50, FLeftTopY = 80;
const MARGIN = { LEFT: 100, RIGHT: 10, TOP: 10, BOTTOM: 130 }
const WIDTH = FWith - (MARGIN.LEFT + MARGIN.RIGHT)
const HEIGHT = FHeight - (MARGIN.TOP + MARGIN.BOTTOM)

const svg = d3.select("#chart-area").append("svg")
      .attr("width", 2000)
      .attr("height", 2000)

const g = svg.append("g")
      .attr("transform", `translate(${FLeftTopX + MARGIN.LEFT}, ${FLeftTopY + MARGIN.TOP})`)
```

Ex04-16

- main.js
  - We usually use variables to set the width and height of the svg, and the margin of our plot
- Why margin?
  - You may have multiple plots on the webpage and you may want to make/adjust the distance between them later





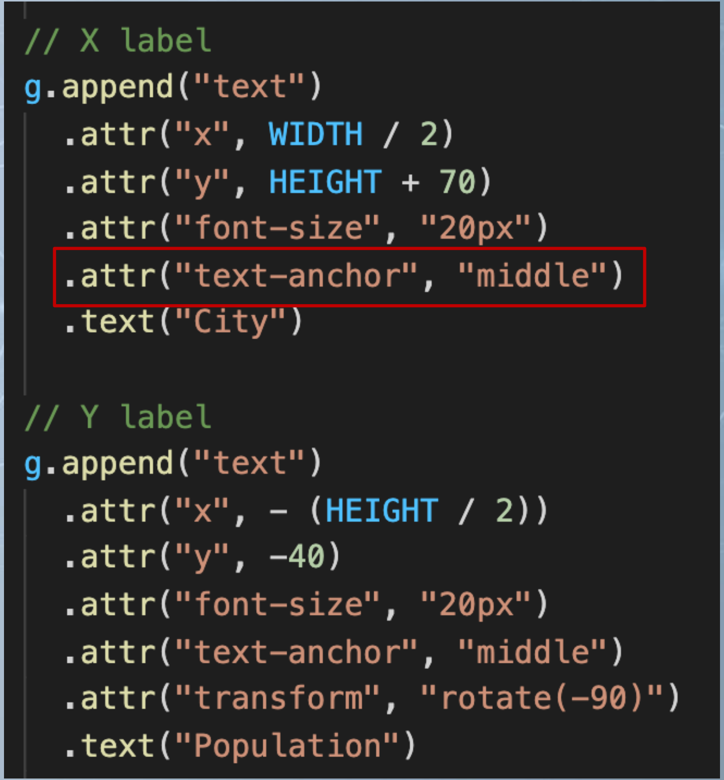


```
const FWith = 600, FHeight = 400;
const FLeftTopX = 50, FLeftTopY = 80;
const MARGIN = { LEFT: 100, RIGHT: 10, TOP: 10, BOTTOM: 130 }
const WIDTH = FWith - (MARGIN.LEFT + MARGIN.RIGHT)
const HEIGHT = FHeight - (MARGIN.TOP + MARGIN.BOTTOM)

const svg = d3.select("#chart-area").append("svg")
            .attr("width", 2000)
            .attr("height", 2000)

const g = svg.append("g")
            .attr("transform", `translate(${FLeftTopX + MARGIN.LEFT}, ${FLeftTopY + MARGIN.TOP})`)
```

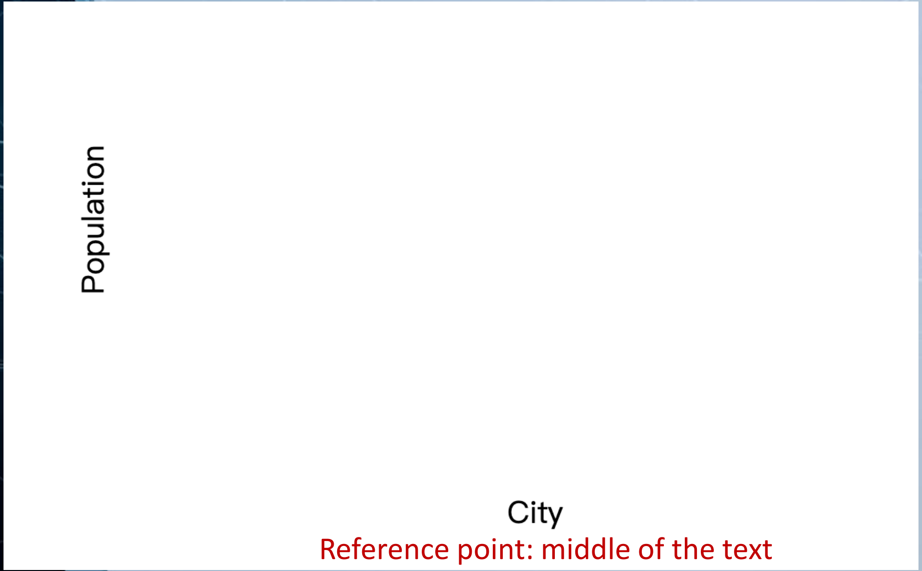
Ex04-16

- main.js
  - X label and y label

https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/text-anchor



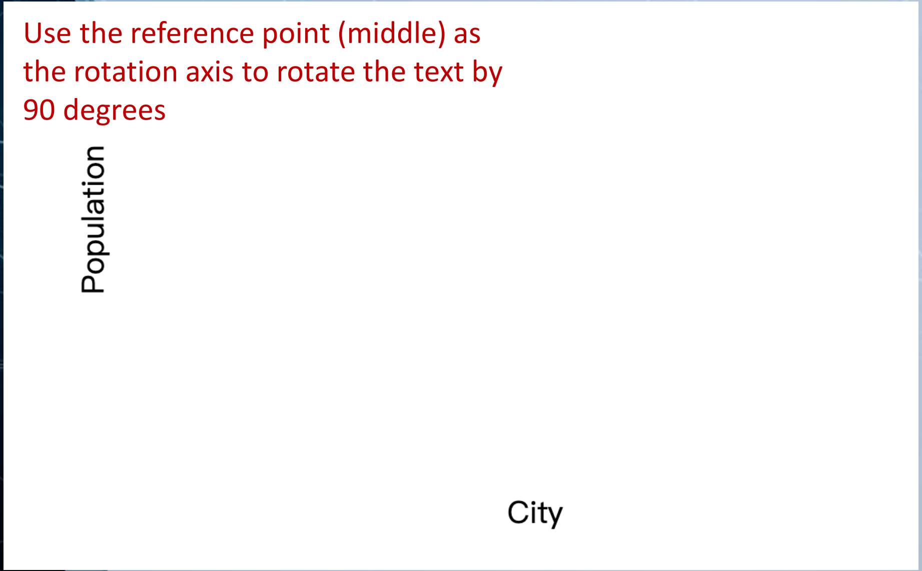
```
// X label
g.append("text")
  .attr("x", WIDTH / 2)
  .attr("y", HEIGHT + 70)
  .attr("font-size", "20px")
  .attr("text-anchor", "middle")
  .text("City")

// Y label
g.append("text")
  .attr("x", - (HEIGHT / 2))
  .attr("y", -40)
  .attr("font-size", "20px")
  .attr("text-anchor", "middle")
  .attr("transform", "rotate(-90)")
  .text("Population")
```



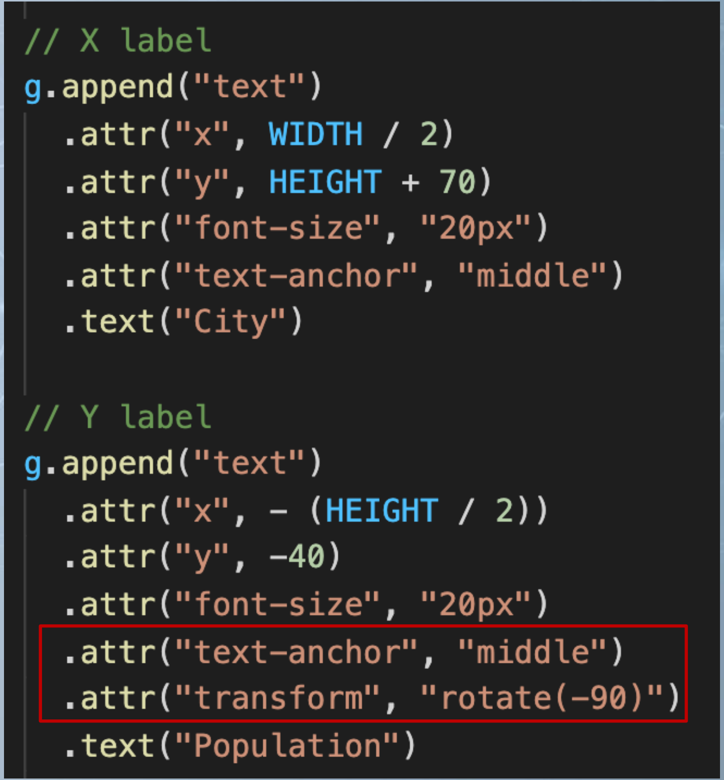
Reference point: middle of the text

Ex04-16

- main.js
  - x label and y label



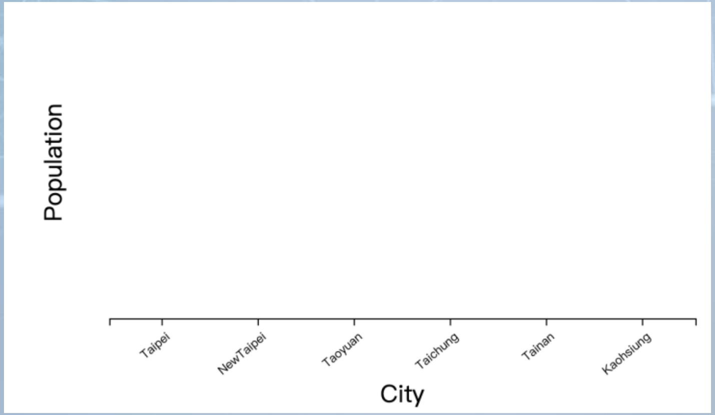
Use the reference point (middle) as the rotation axis to rotate the text by 90 degrees

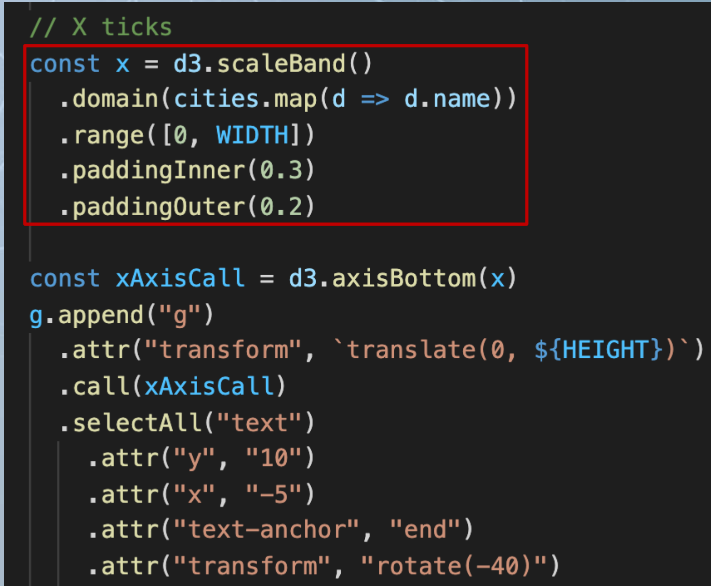


```
// X label
g.append("text")
  .attr("x", WIDTH / 2)
  .attr("y", HEIGHT + 70)
  .attr("font-size", "20px")
  .attr("text-anchor", "middle")
  .text("City")

// Y label
g.append("text")
  .attr("x", - (HEIGHT / 2))
  .attr("y", -40)
  .attr("font-size", "20px")
  .attr("text-anchor", "middle")
  .attr("transform", "rotate(-90)")
  .text("Population")
```

Ex04-16

- main.js
  - X-ticks
  - Our x domain is discrete
  - Use d3.scaleBand to map city names to x location

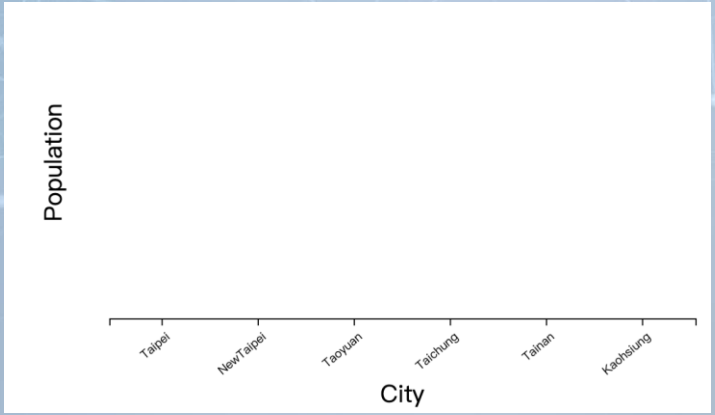


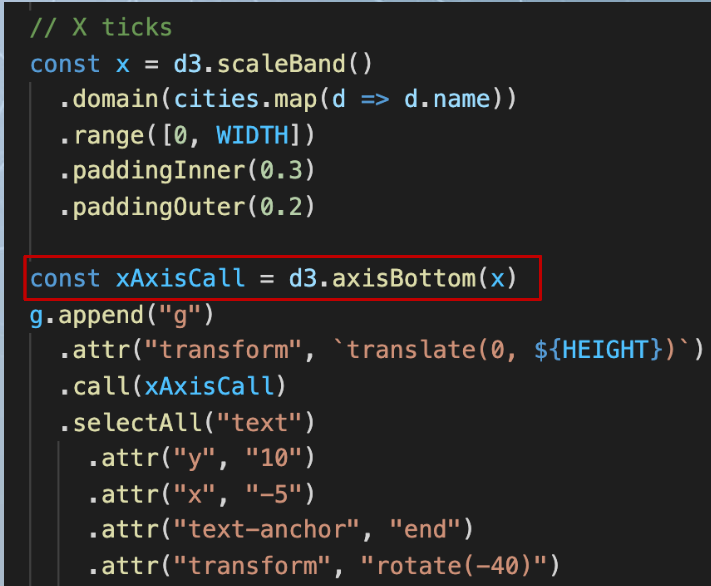


```
// X ticks
const x = d3.scaleBand()
  .domain(cities.map(d => d.name))
  .range([0, WIDTH])
  .paddingInner(0.3)
  .paddingOuter(0.2)

const xAxisCall = d3.axisBottom(x)
g.append("g")
  .attr("transform", `translate(0, ${HEIGHT})`)
  .call(xAxisCall)
  .selectAll("text")
    .attr("y", "10")
    .attr("x", "-5")
    .attr("text-anchor", "end")
    .attr("transform", "rotate(-40)")
```

Ex04-16

- main.js
  - X-ticks
  - create the axis function

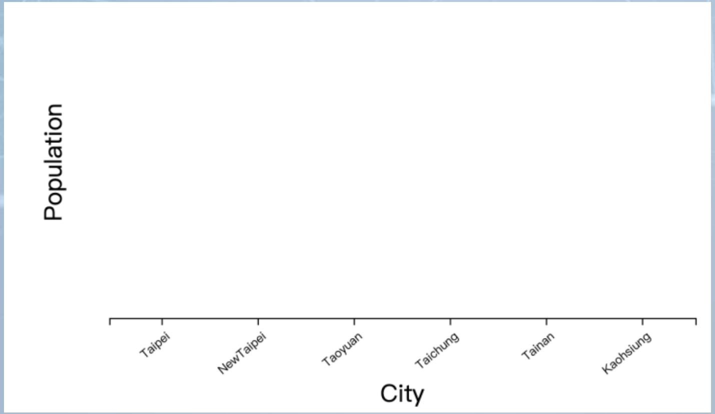


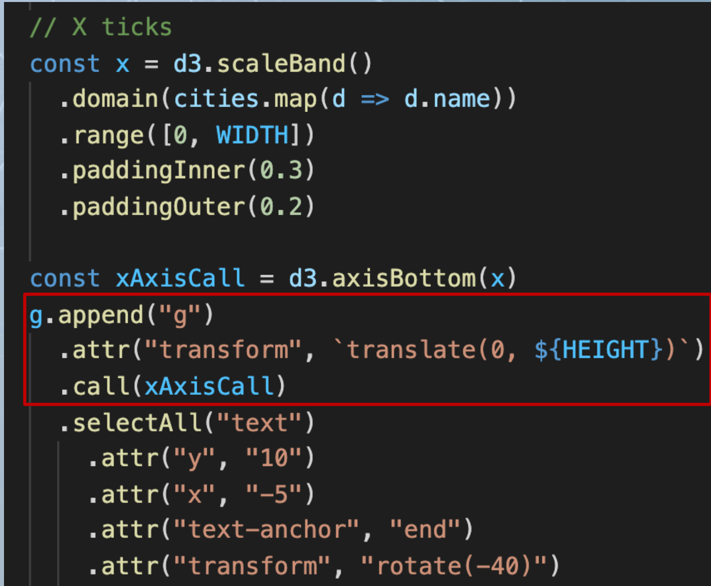


```
// X ticks
const x = d3.scaleBand()
  .domain(cities.map(d => d.name))
  .range([0, WIDTH])
  .paddingInner(0.3)
  .paddingOuter(0.2)

const xAxisCall = d3.axisBottom(x)
g.append("g")
  .attr("transform", `translate(0, ${HEIGHT})`)
  .call(xAxisCall)
  .selectAll("text")
    .attr("y", "10")
    .attr("x", "-5")
    .attr("text-anchor", "end")
    .attr("transform", "rotate(-40)")
```

Ex04-16

- main.js
  - X-ticks
  - Put the axis in a new <g>
    - Easy to translate the axis to where we want
  - Use .call() to create the axis

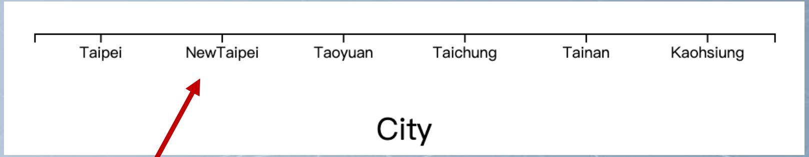


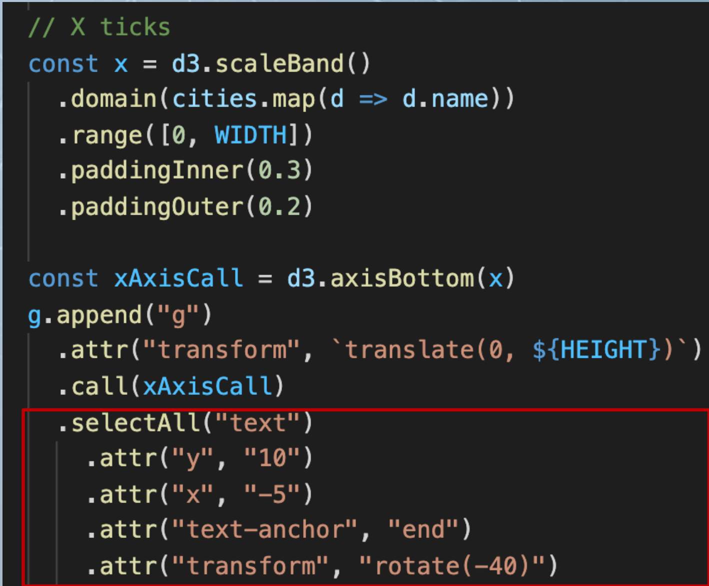


```
// X ticks
const x = d3.scaleBand()
  .domain(cities.map(d => d.name))
  .range([0, WIDTH])
  .paddingInner(0.3)
  .paddingOuter(0.2)

const xAxisCall = d3.axisBottom(x)
g.append("g")
  .attr("transform", `translate(0, ${HEIGHT})`)
  .call(xAxisCall)
  .selectAll("text")
    .attr("y", "10")
    .attr("x", "-5")
    .attr("text-anchor", "end")
    .attr("transform", "rotate(-40)")
```

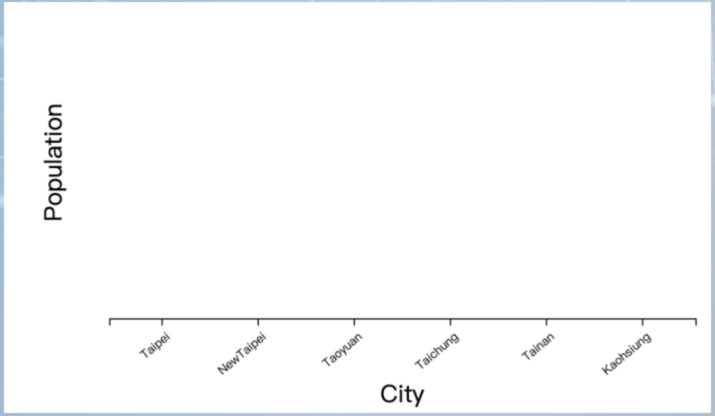
Ex04-16

- main.js
  - X-ticks
  - Without the red box code, we still have the city names on the axis, but no rotation



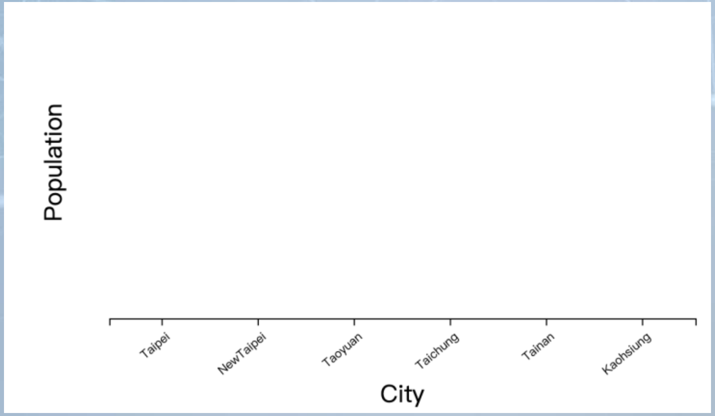


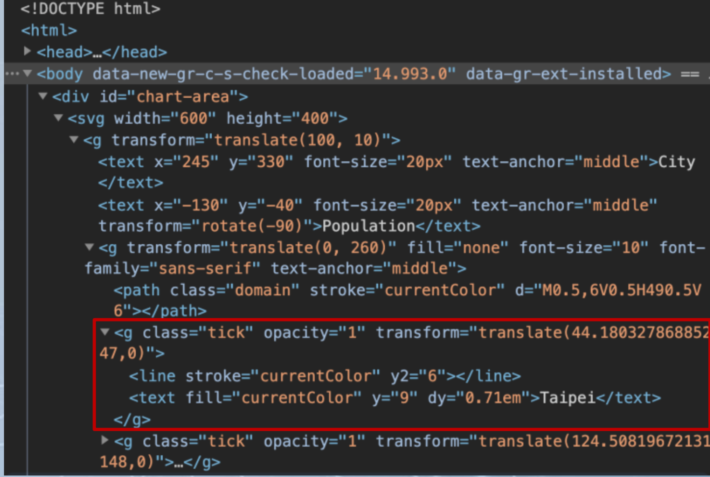
```
// X ticks
const x = d3.scaleBand()
  .domain(cities.map(d => d.name))
  .range([0, WIDTH])
  .paddingInner(0.3)
  .paddingOuter(0.2)

const xAxisCall = d3.axisBottom(x)
g.append("g")
  .attr("transform", `translate(0, ${HEIGHT})`)
  .call(xAxisCall)
  .selectAll("text")
    .attr("y", "10")
    .attr("x", "-5")
    .attr("text-anchor", "end")
    .attr("transform", "rotate(-40)")
```



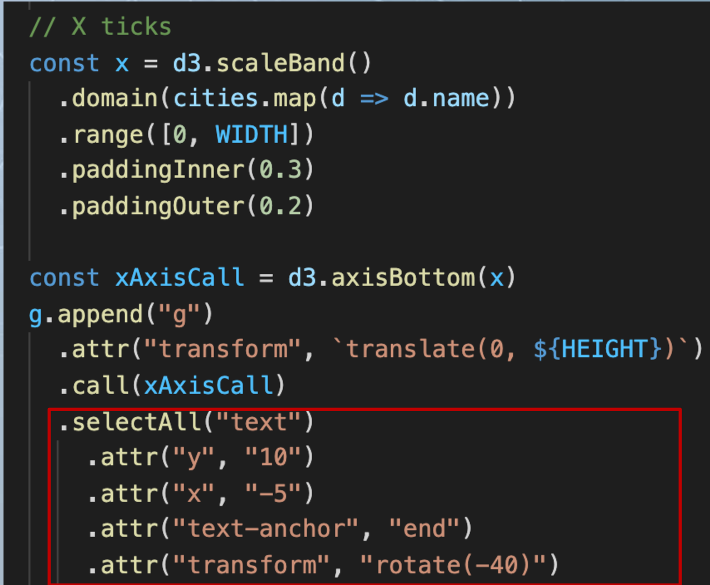
Ex04-16

- main.js
  - X-ticks
  - City names are “text”. We can select them and manipulate them.





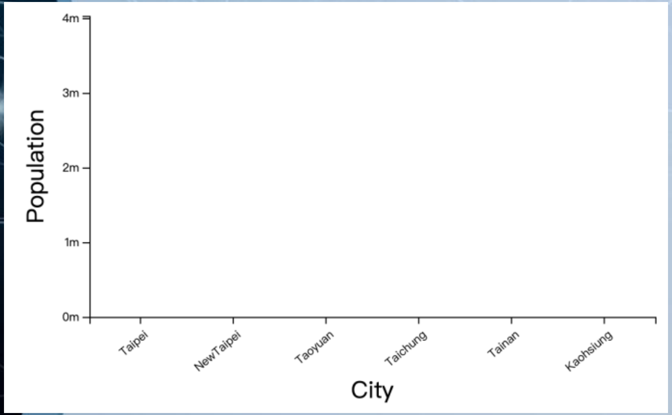
```
<!DOCTYPE html>
<html>
 ▶ <head>…</head>
…▼ <body data-new-gr-c-s-check-loaded="14.993.0" data-gr-ext-installed> ==
   ▼ <div id="chart-area">
     ▼ <svg width="600" height="400">
       ▼ <g transform="translate(100, 10)">
           <text x="245" y="330" font-size="20px" text-anchor="middle">City
           </text>
           <text x="-130" y="-40" font-size="20px" text-anchor="middle"
           transform="rotate(-90)">Population</text>
         ▼ <g transform="translate(0, 260)" fill="none" font-size="10" font-
         family="sans-serif" text-anchor="middle">
             <path class="domain" stroke="currentColor" d="M0.5,6V0.5H490.5V
             6"></path>
           ▼ <g class="tick" opacity="1" transform="translate(44.180327868852
           47,0)">
               <line stroke="currentColor" y2="6"></line>
               <text fill="currentColor" y="9" dy="0.71em">Taipei</text>
             </g>
           ▶ <g class="tick" opacity="1" transform="translate(124.50819672131
           148,0)">…</g>
```

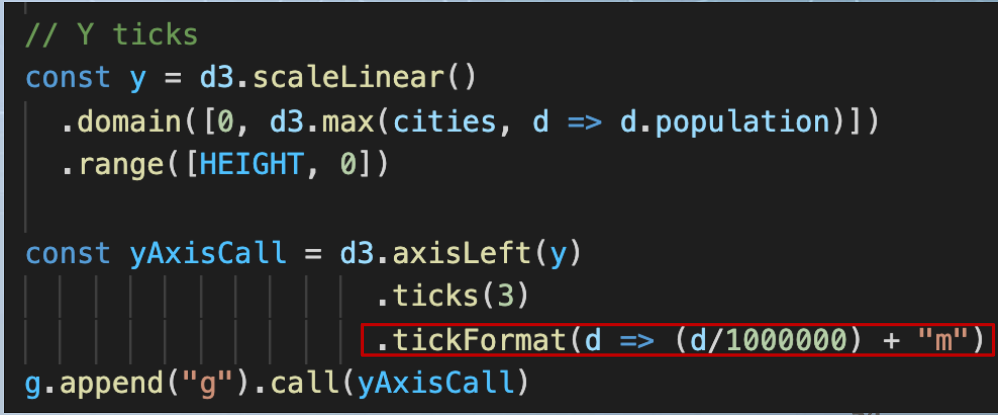


```
// X ticks
const x = d3.scaleBand()
  .domain(cities.map(d => d.name))
  .range([0, WIDTH])
  .paddingInner(0.3)
  .paddingOuter(0.2)

const xAxisCall = d3.axisBottom(x)
g.append("g")
  .attr("transform", `translate(0, ${HEIGHT})`)
  .call(xAxisCall)
  .selectAll("text")
    .attr("y", "10")
    .attr("x", "-5")
    .attr("text-anchor", "end")
    .attr("transform", "rotate(-40)")
```

Ex04-16

- main.js
  - Y ticks
  - We use ”m” to represent millions

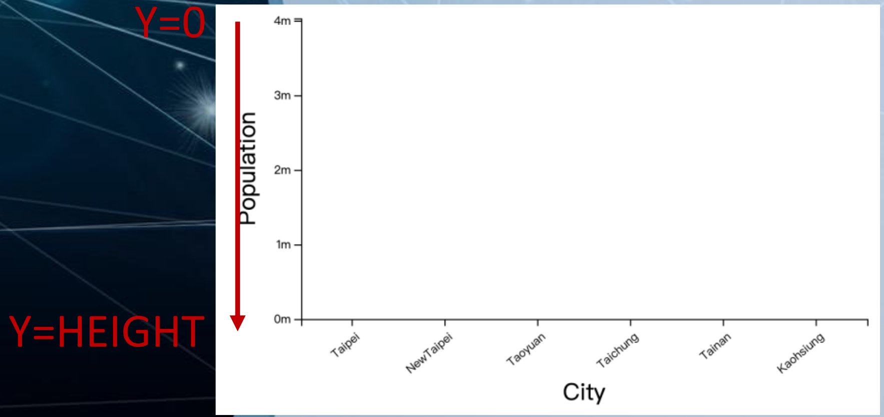


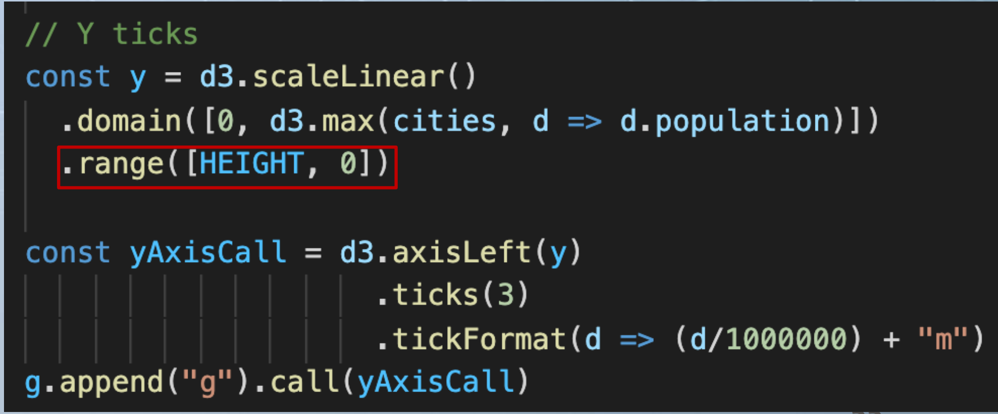


```
// Y ticks
const y = d3.scaleLinear()
  .domain([0, d3.max(cities, d => d.population)])
  .range([HEIGHT, 0])

const yAxisCall = d3.axisLeft(y)
                    .ticks(3)
                    .tickFormat(d => (d/1000000) + "m")
g.append("g").call(yAxisCall)
```

Ex04-16

- main.js
  - Y ticks
  - Why is the range [HEIGHT, 0] instead of [0, HEIGHT]?
  - We map [0, maxPopulation] to [HEIGHT, 0]
  - Y-coordinate: top is 0

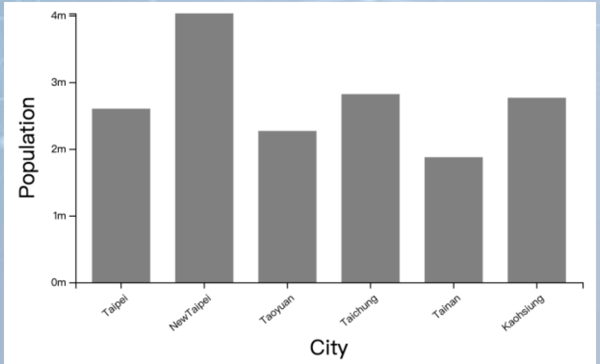


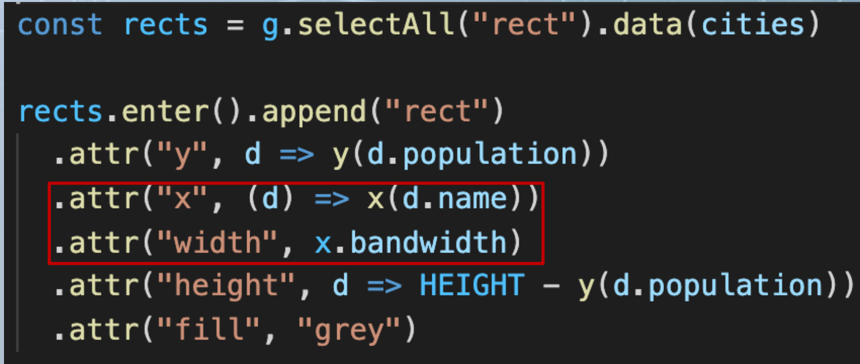


```
// Y ticks
const y = d3.scaleLinear()
  .domain([0, d3.max(cities, d => d.population)])
  .range([HEIGHT, 0])

const yAxisCall = d3.axisLeft(y)
                    .ticks(3)
                    .tickFormat(d => (d/1000000) + "m")
g.append("g").call(yAxisCall)
```

Ex04-16

- main.js
  - Draw the bars
  - x: the d3.scaleBand function
    - Get the x locations and width of the bars



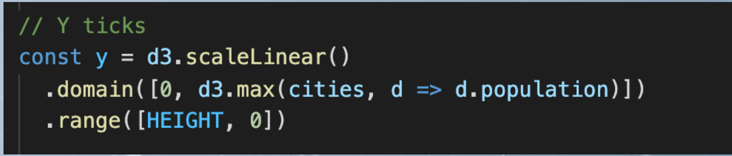


```
const rects = g.selectAll("rect").data(cities)

rects.enter().append("rect")
  .attr("y", d => y(d.population))
  .attr("x", (d) => x(d.name))
  .attr("width", x.bandwidth)
  .attr("height", d => HEIGHT - y(d.population))
  .attr("fill", "grey")
```

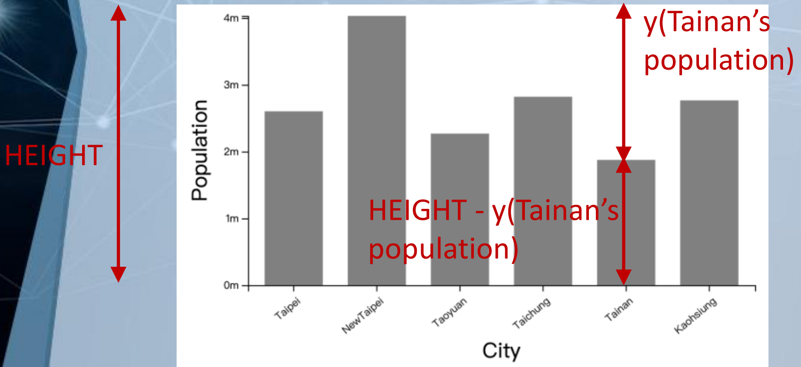
Ex04-16

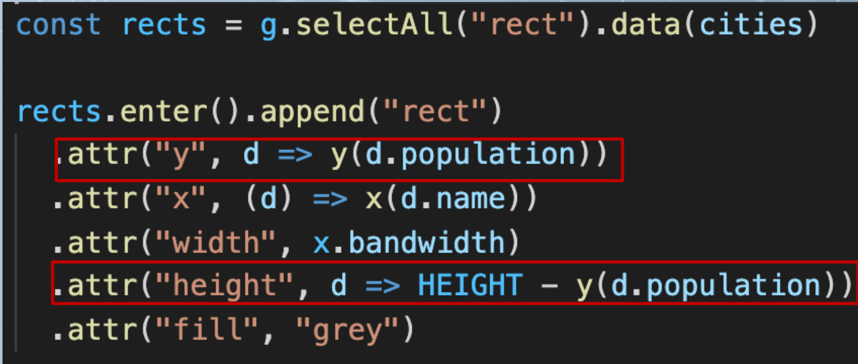
- main.js
  - Draw the bars
  - x: the d3.scaleBand function
    - Get the x locations and width of the bars
  - y: the d3.scaleLinear function
    - **Get the y location of left upper corner and height of bars**

If the input to y() is larger, the output is smaller



```
// Y ticks
const y = d3.scaleLinear()
  .domain([0, d3.max(cities, d => d.population)])
  .range([HEIGHT, 0])
```





```
const rects = g.selectAll("rect").data(cities)

rects.enter().append("rect")
  .attr("y", d => y(d.population))
  .attr("x", (d) => x(d.name))
  .attr("width", x.bandwidth)
  .attr("height", d => HEIGHT - y(d.population))
  .attr("fill", "grey")
```