最熱門的互動資料視覺化技術

http://x.co/d3course

Kirby Wu

plotdb.com


V4

02.html

加入顏色樣式

引入 d3 module

```
<script src="d3-selection-multi.min.js">
</script>
```

設定樣式

```
d3.select("body")
  .text("Hello World")
  .styles({
    background: "red"
  });
```

http://x.co/d3course


HTML
CSS
JS

Chart.js

ean and engaging charts for designers and dev

Download

Documentation

CHARTIST.JS

SIMPLE RESPONSIVE CHARTS

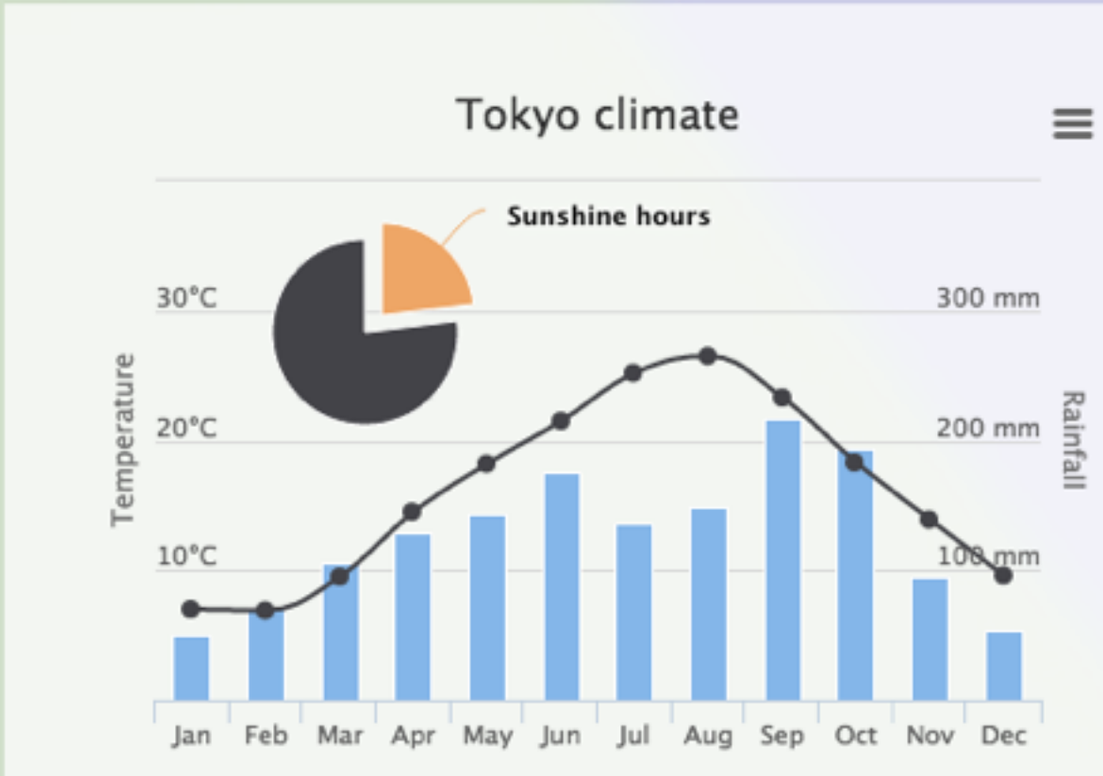
HIGHCHARTS

Create interactive charts easily for your web projects.

Used by tens of thousands of developers and 61 out of the world's 100 largest companies, Highcharts is the simplest yet most flexible charting API on the market.

READ MORE »

DOWNLOAD »



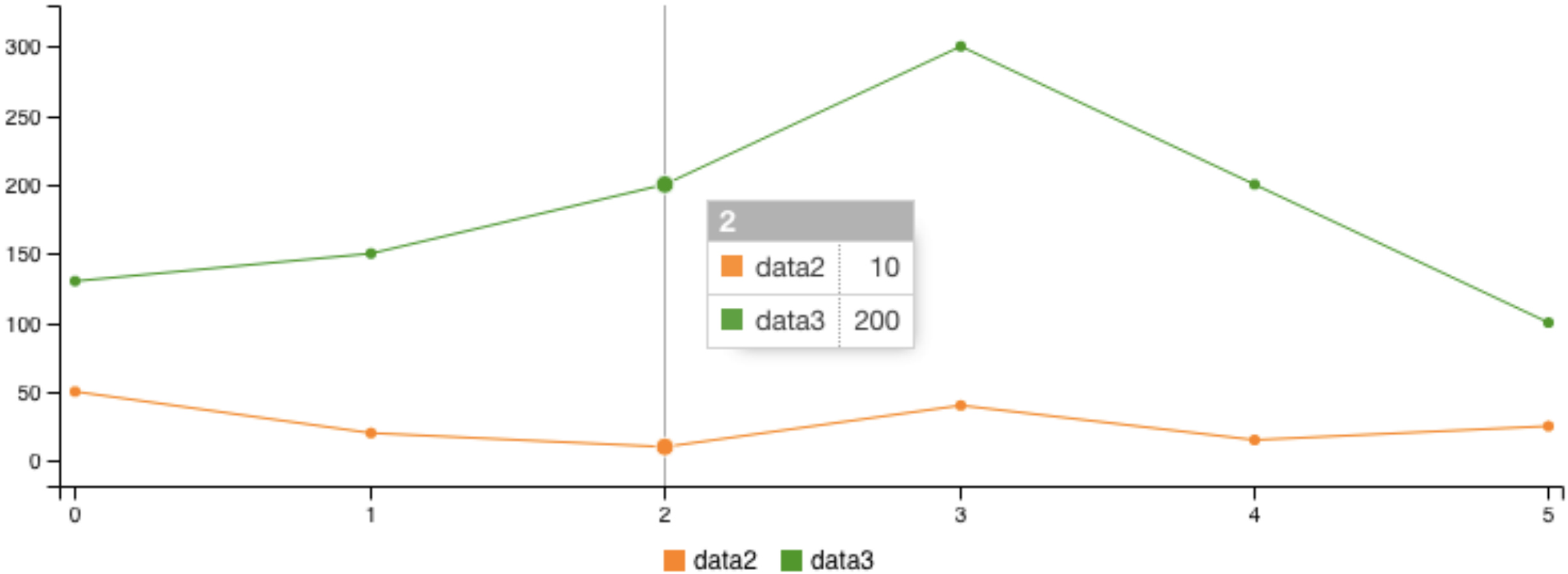
現有的圖表工具

- amchart
- chart.js
- chartist.js
- echarts (baidu)
- google chart
- highchart
- paper.js
- zingchart
- two.js
- raphaeljs
- nvd3
- c3.js
- n3 charts
- ember chart
- fusion chart
- flot
- uvCharts
- plotly.js
- sigma.js
- fusion charts
- n3 charts
- anychart
- xchart
- dc.js

Why D3.js?

Visualizing != Charting

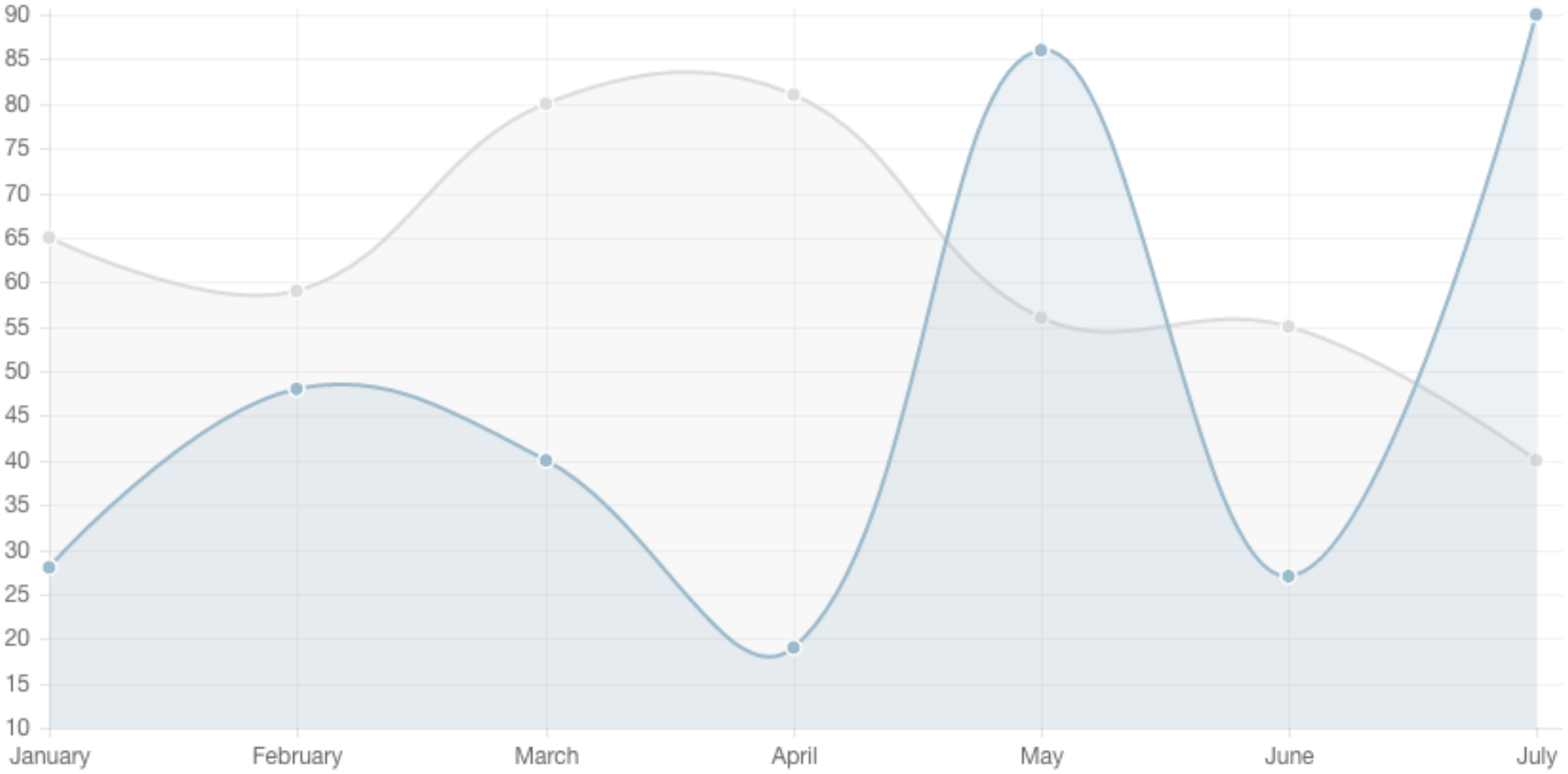
C3.js


300
250
200
150
100
50
0
0
1
2
3
4
5
2
data2 10
data3 200
data2
data3

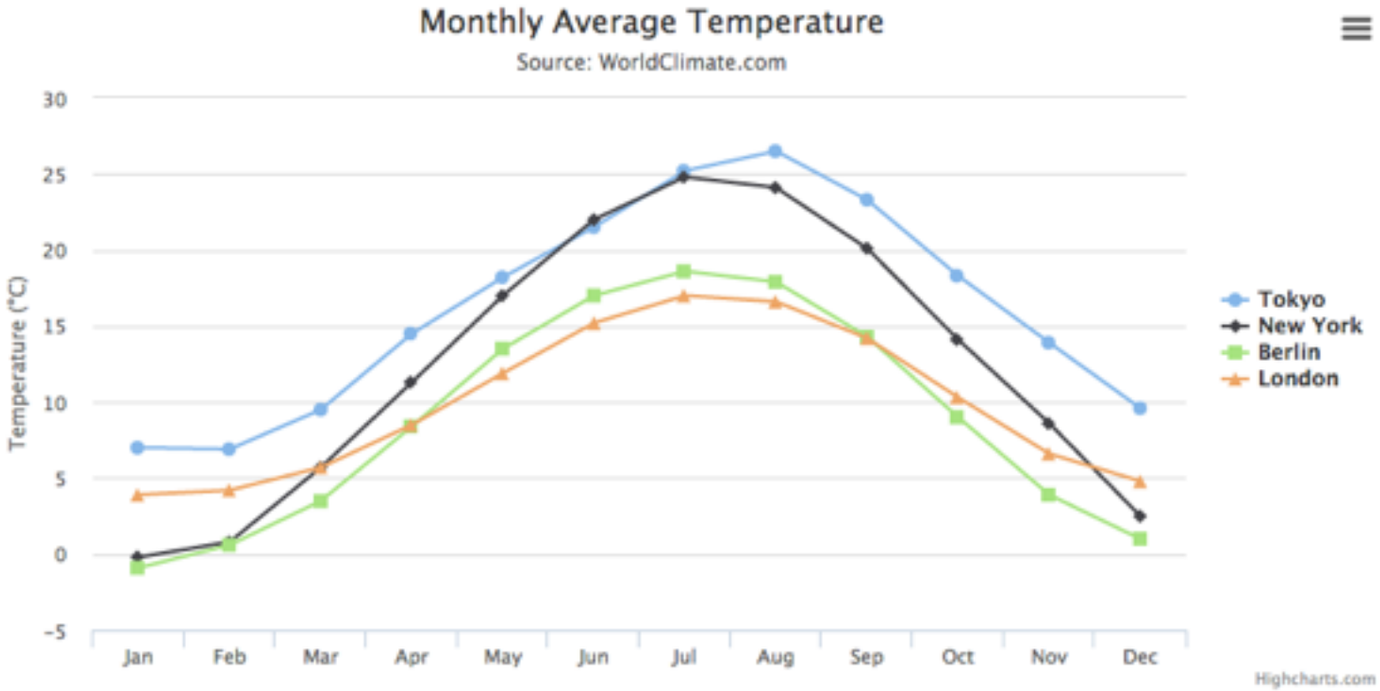
n3-charts

Awesome charts for Angular

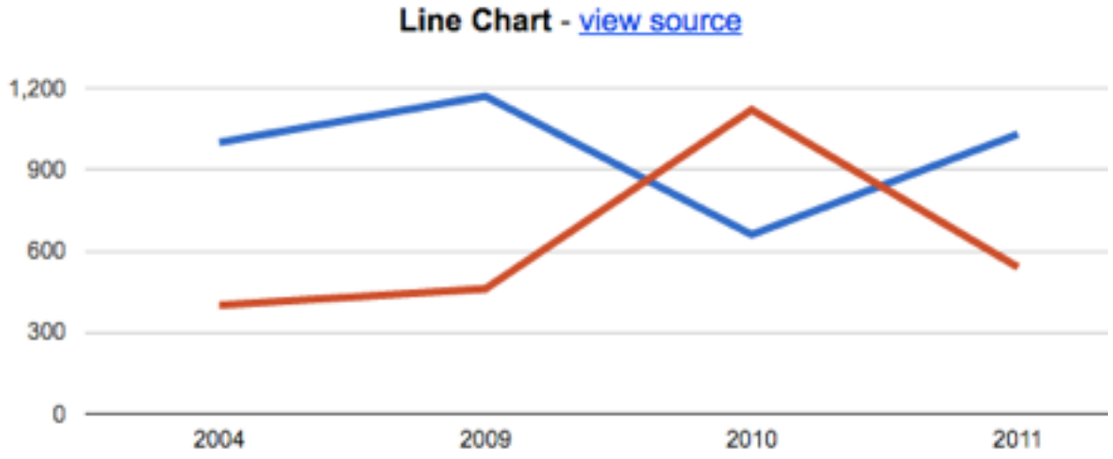
Chart.js

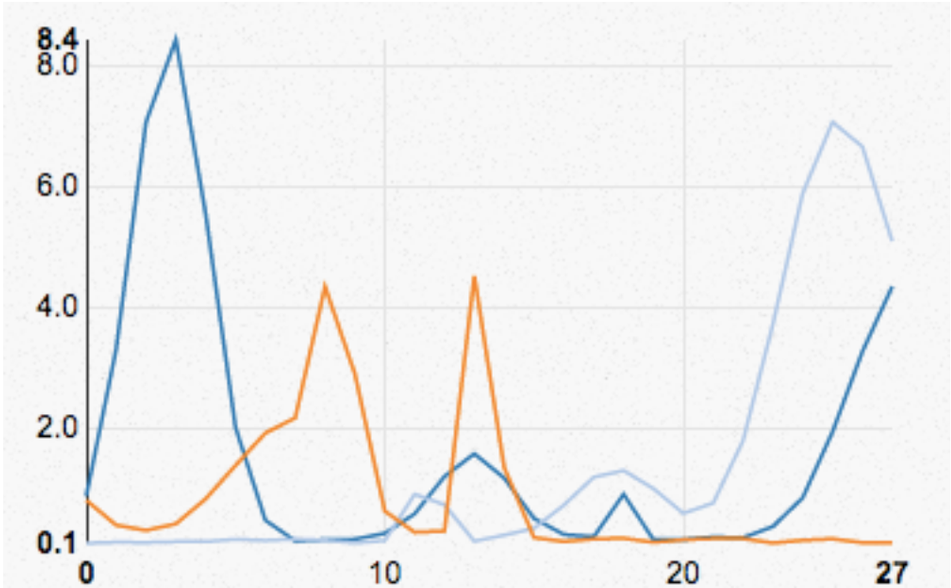

90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
January
February
March
April
May
June
July

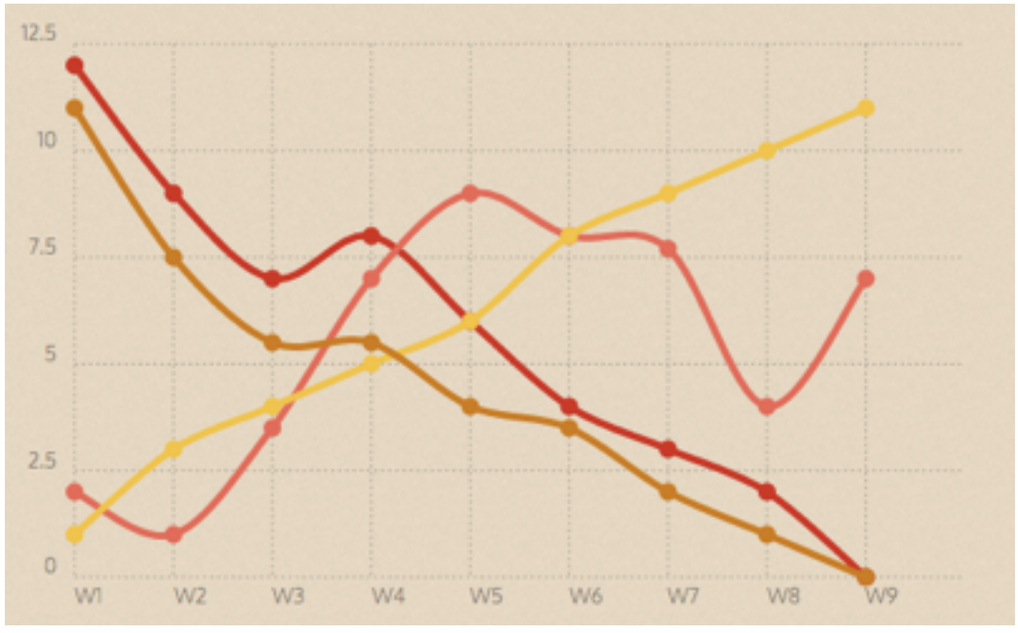
Highcharts & Amcharts

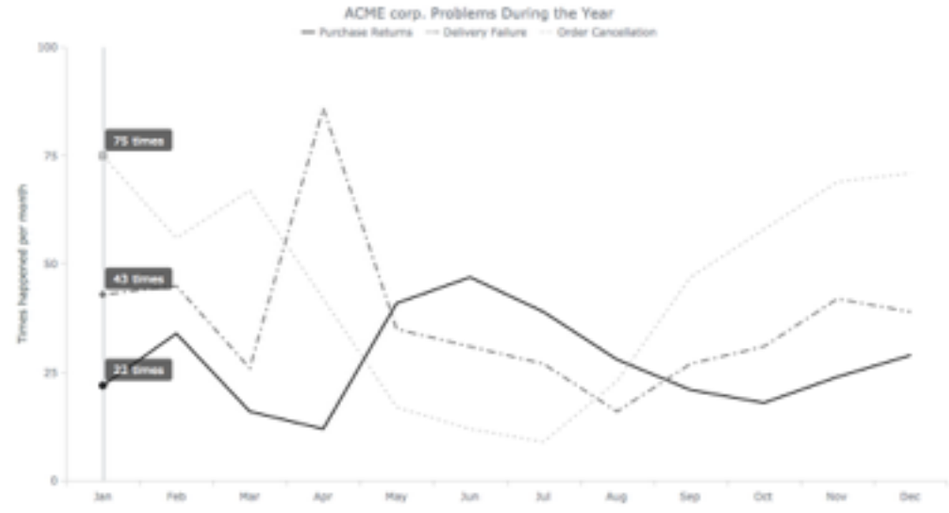


Anychart, Google Chart, Chartist, nvd3

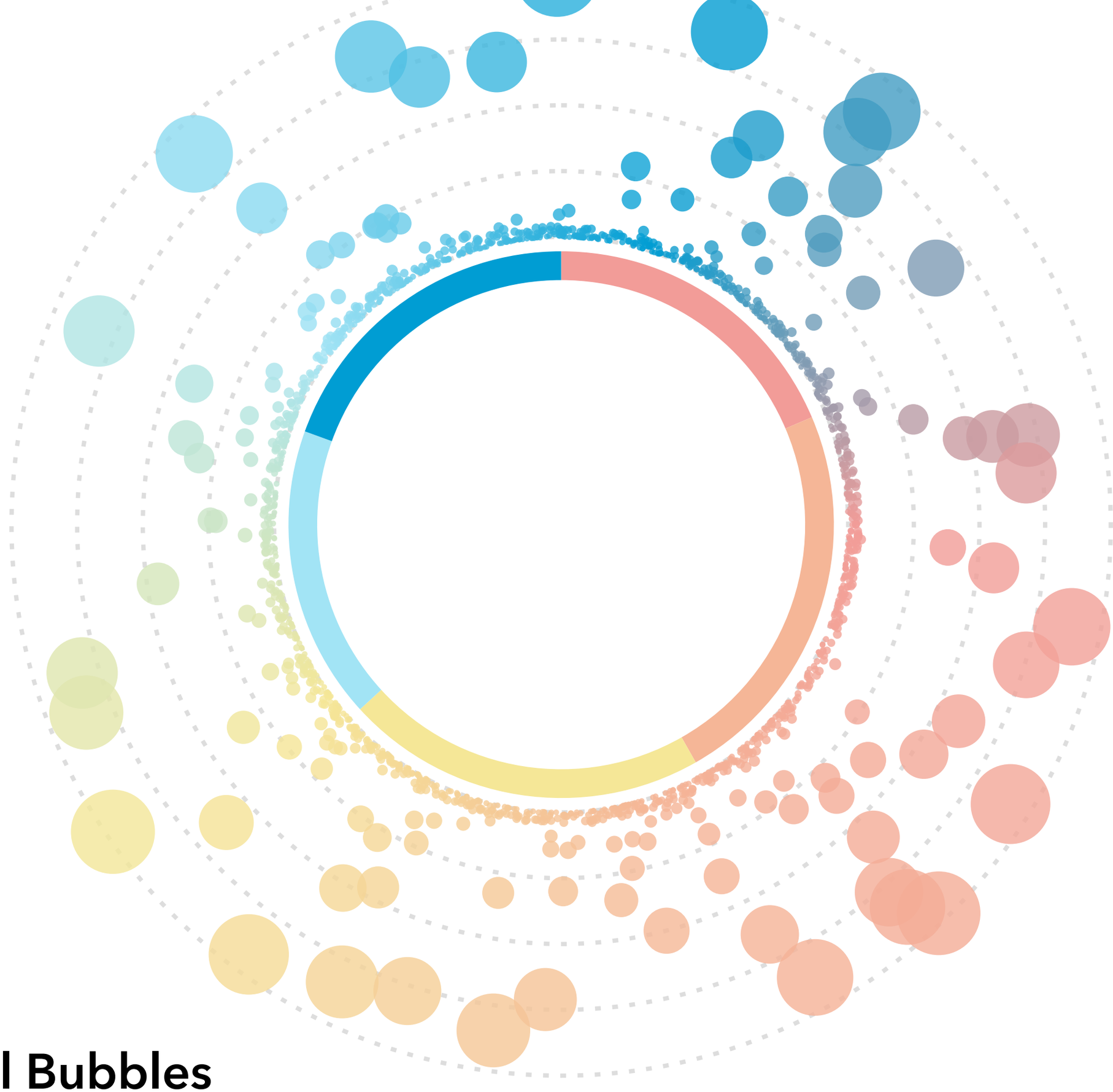








2. Flexibility

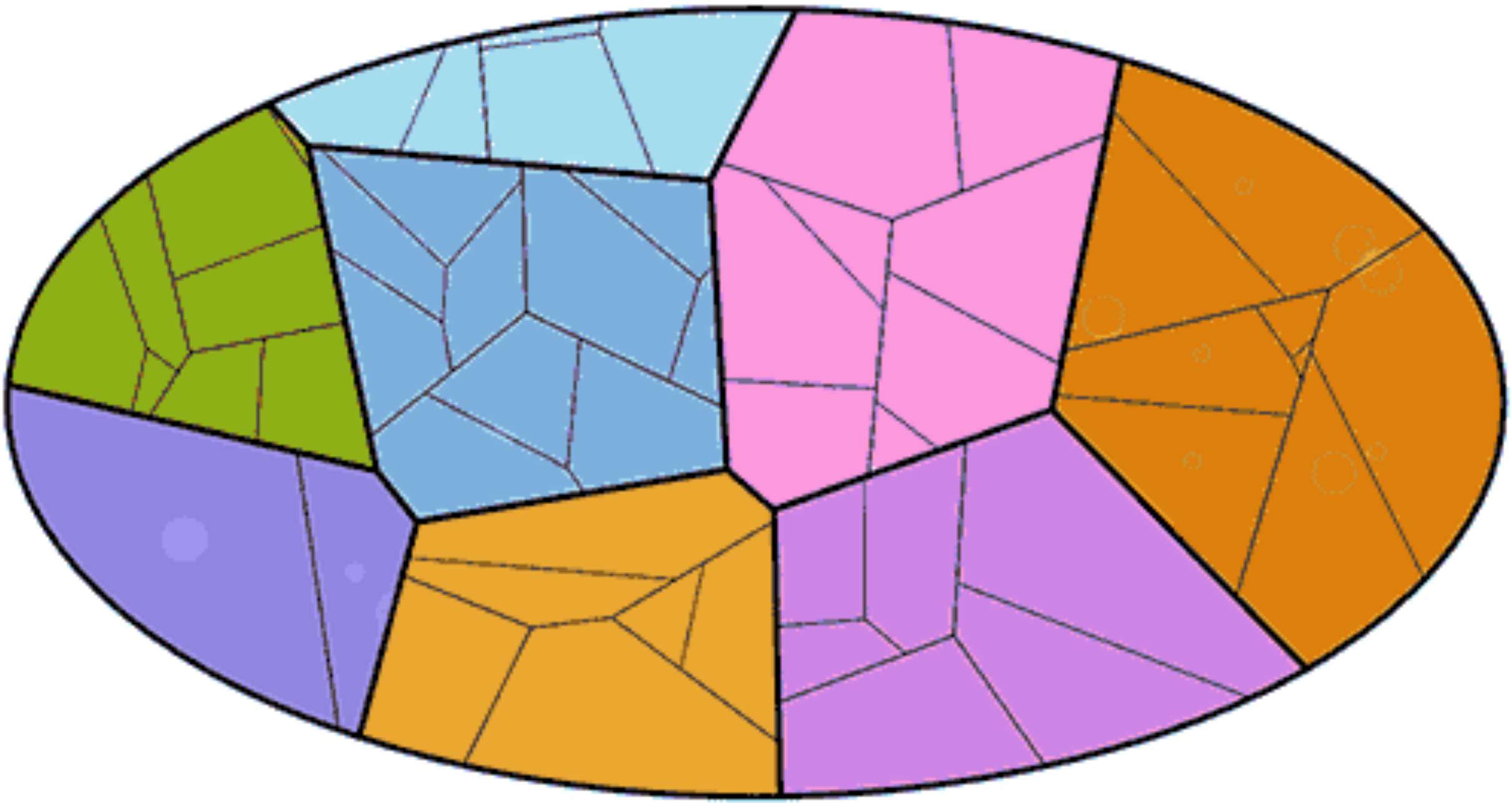


Radial Bubbles

http://plotdb.com/chart/?k=s972

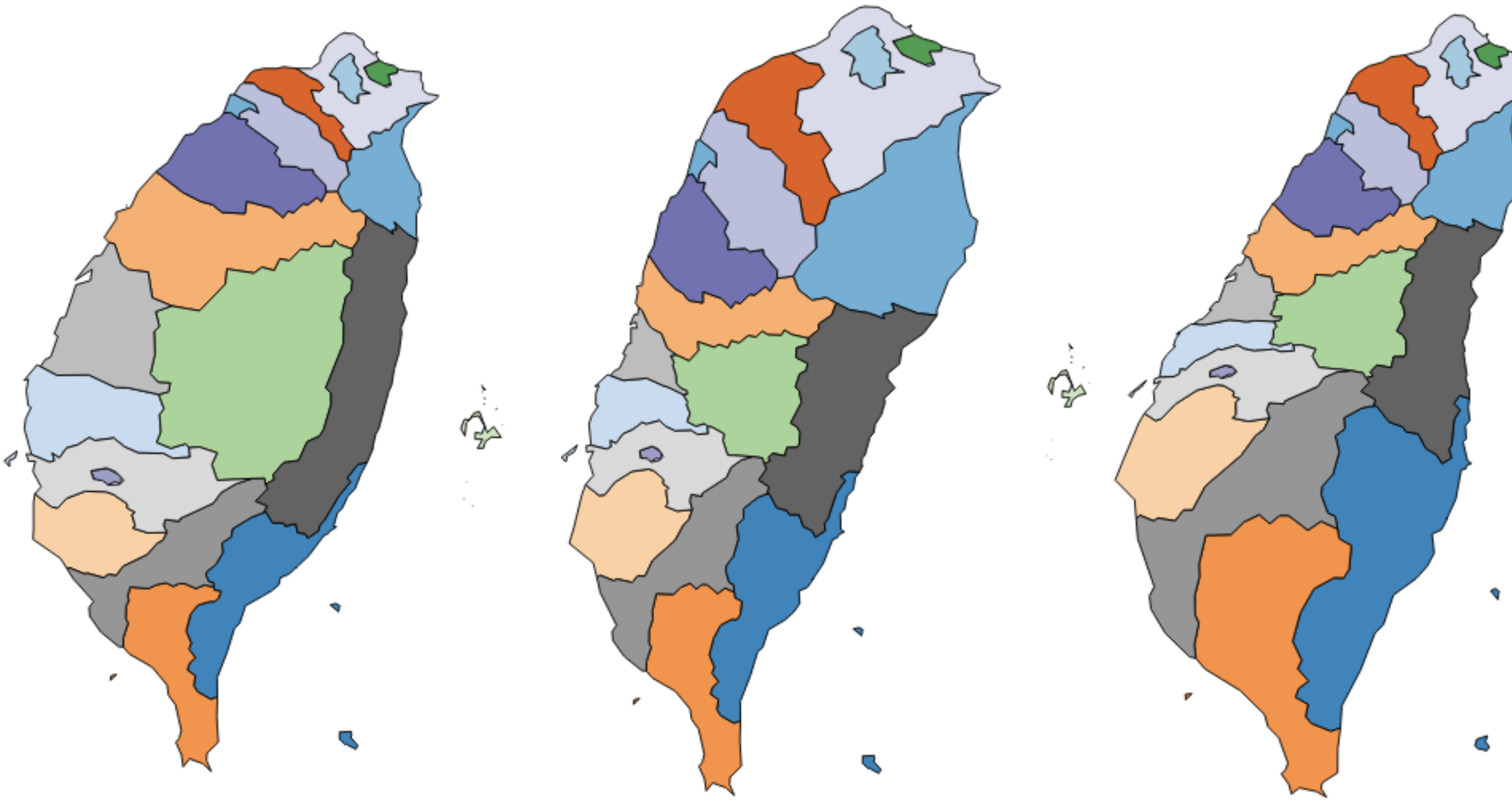
Sticky Bubbles

http://plotdb.com/chart/?k=s978



Voronoi Treemap

http://zbryikt.github.io/voronoijs/



Map Distortion

http://zbryikt.github.io/visualize/ajd3/

Continuous Transition

http://bl.ocks.org/mbostock/1256572

Environment Setup

1. 下載 D3course.zip, 解壓縮

2. 開啟 index.html

1. 下載 D3course.zip, 解壓縮

2. 開啟 index.html

```
<body></body>
<script src="d3.v4.min.js"></script>
<script>

</script>

```

在第四行的位置插入

```
d3.select("body")
    .text("Hello World");
```

Begin with D3

Select Target

```
d3.select("body")
  .text("Hello World");
```

Cascade Function Call

Set Attributes

d3.select
d3.interpolate
d3.transition
d3.scaleLinear
d3.forceSimulation
d3.pack
d3.hierarchy

……

Modularised Code

- **before: d3.v3.js**
- **after:**
  - **d3.v4.js**
  - **d3-selection-multi.v1.js**
  - **d3-array.v0.8.js**
  - **……**

still provide a common bundle
(~30 modules)

~70 modules
available

V4

Modularised Code

use D3 in NodeJS

```
fs = require("fs");
dsv = require("d3-dsv");
csv = fs.readFileSync("data.csv")
        .toString();
result = dsv.csvParse(csv);
```

V4

Modularised Code

make my bundle

```
cat d3-array.js d3-selection.js > d3.bundle.js
```

https://plotdb.github.io/d3-module-picker/

加入顏色樣式

引入適當 module

```
<script src="d3-selection-multi.v1.min.js">
</script>
```

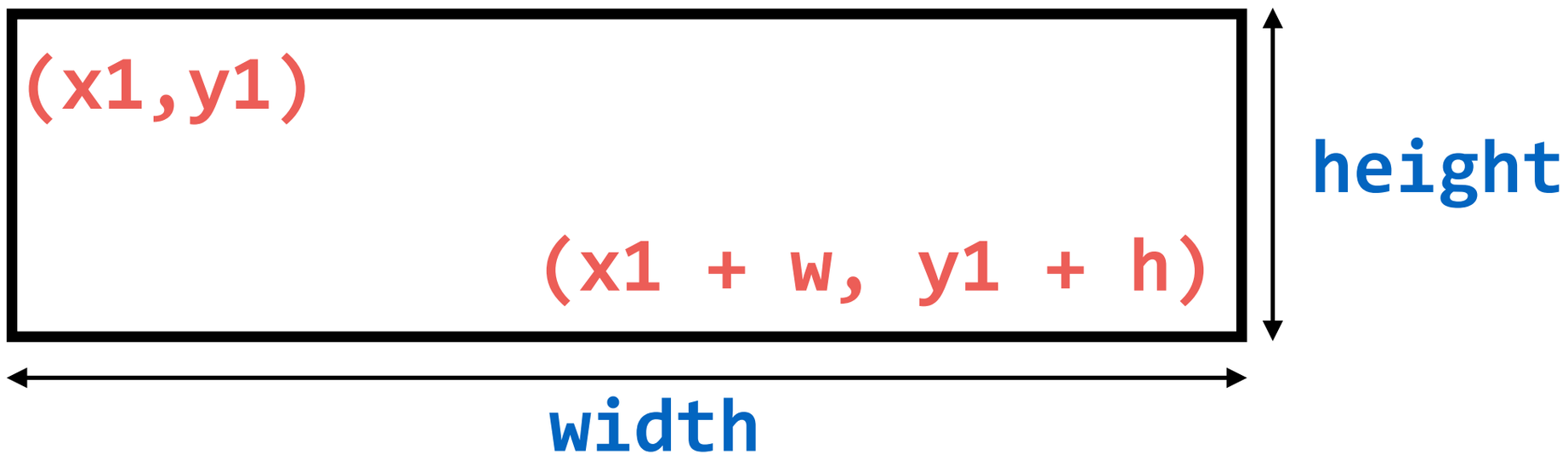
設定樣式

```
d3.select("body")
   .text("Hello World")
   .styles({
     background: "red"
   });
```


SVG

```
<body>
<svg width="<width>"
     height="<height>"
     viewBox="<x y w h>">
</svg>
</body>
```

```
<body>
<svg width="<width>"
     height="<height>"
     viewBox="<x y w h>">
</svg>
</body>
```



常用的 SVG 元素

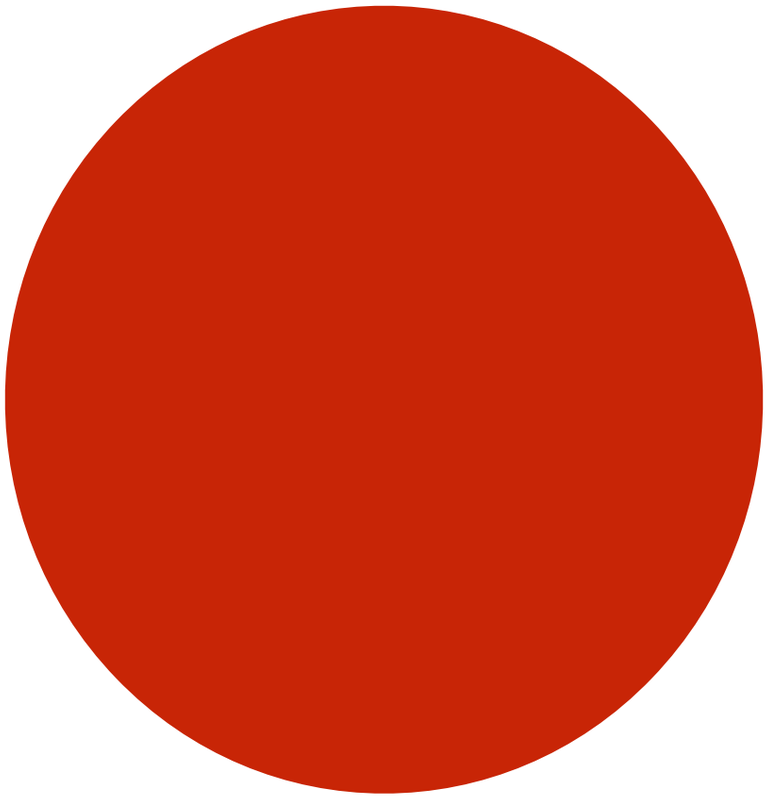
`<circle/>`	`cx, cy, r`
`<rect/>`	`x, y, width, height`
`<line/>`	`x1, y1, x2, y2`
`<path/>`	`d`
`<text></text>`	`x, y, dx, dy`

共通屬性	fill, stroke, stroke-width, transform

```
<body>
<svg width="800px"
     height="600px"
     viewBox="0 0 800 600">
  <rect x="10" y="10"
        width="100"
        height="20"
        fill="red"/>
</svg>
</body>
```

利用 SVG 畫一個紅色的圓

03.html

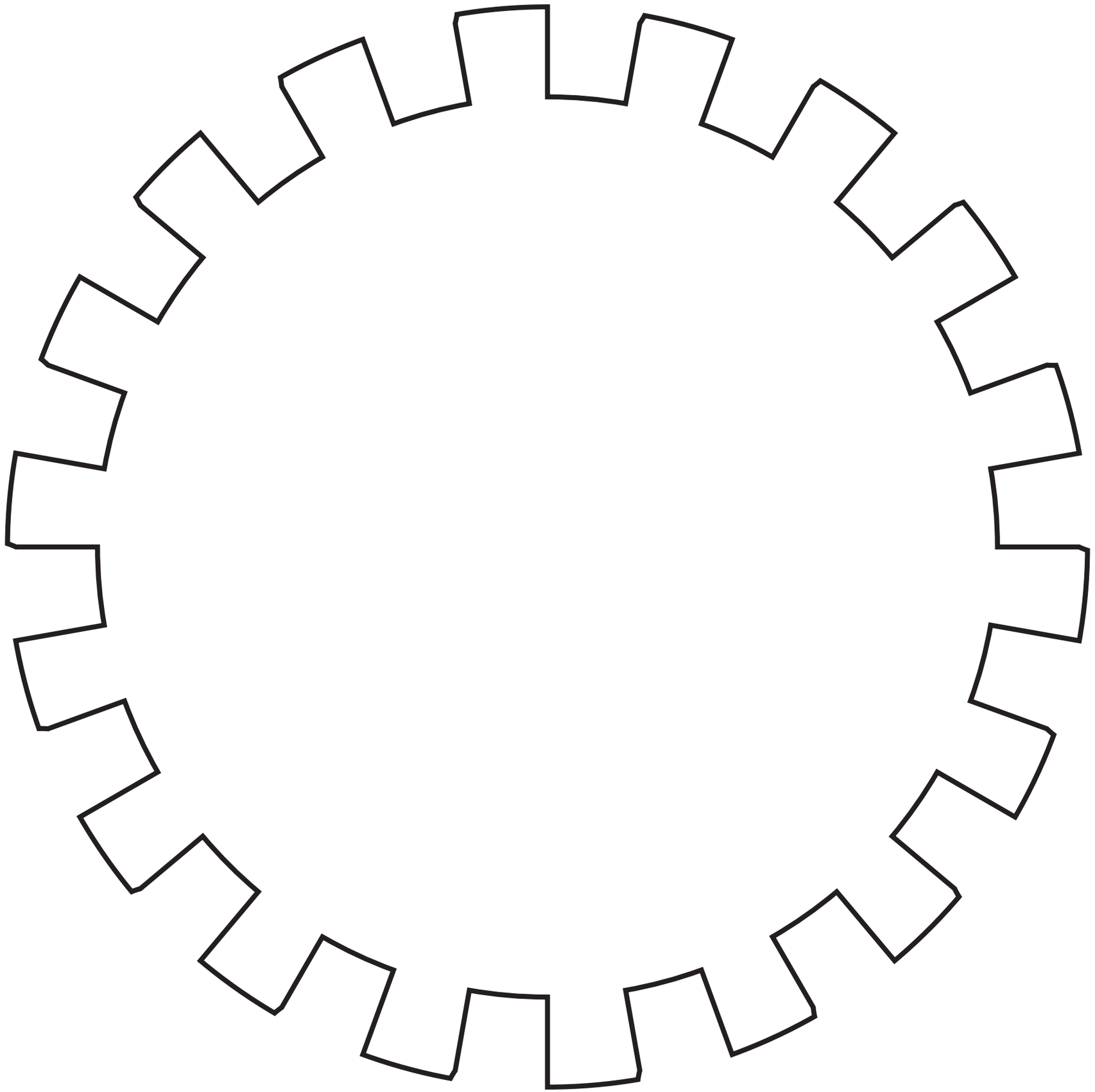


利用 SVG 畫一個臉

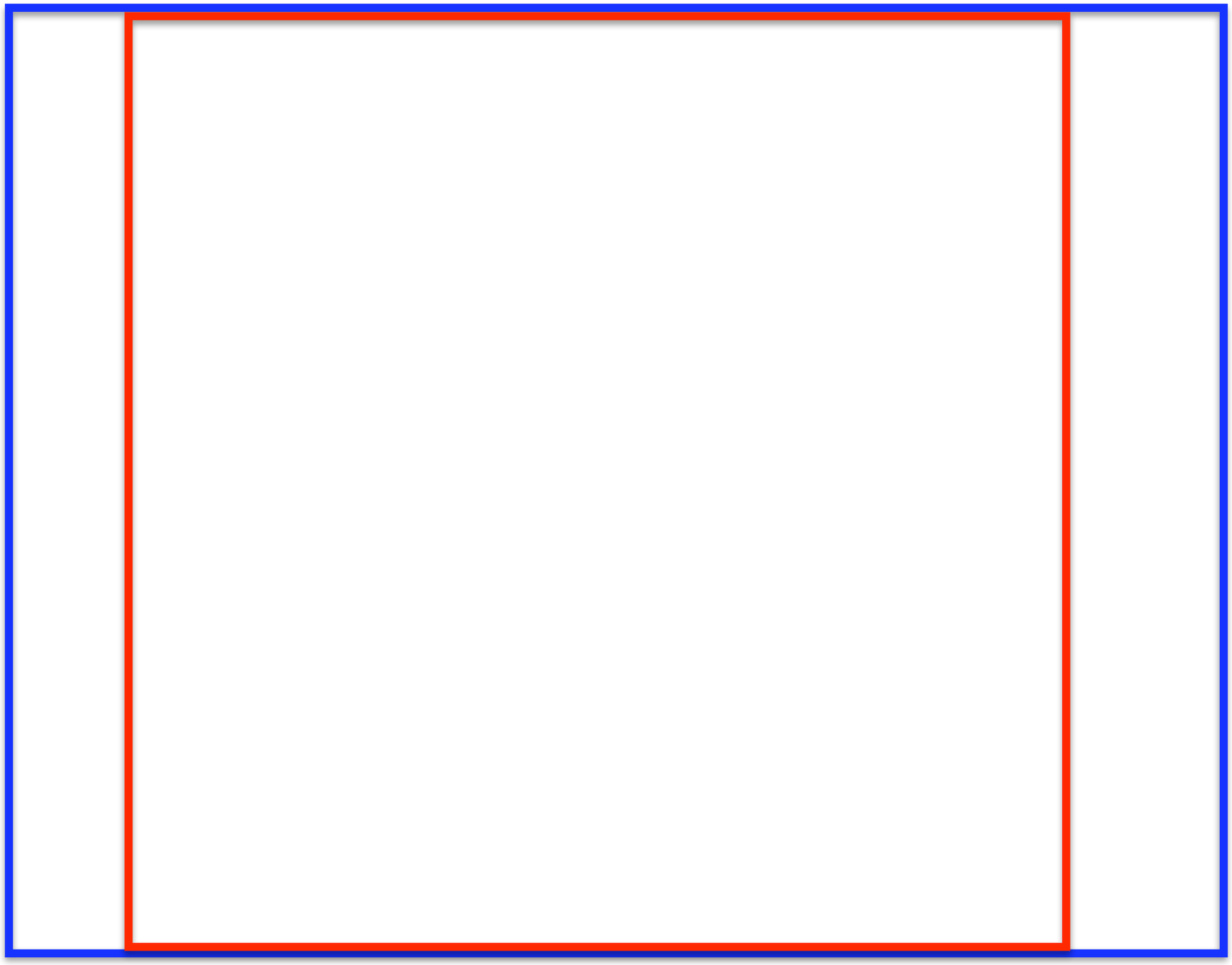
04.html

基本款

加分款

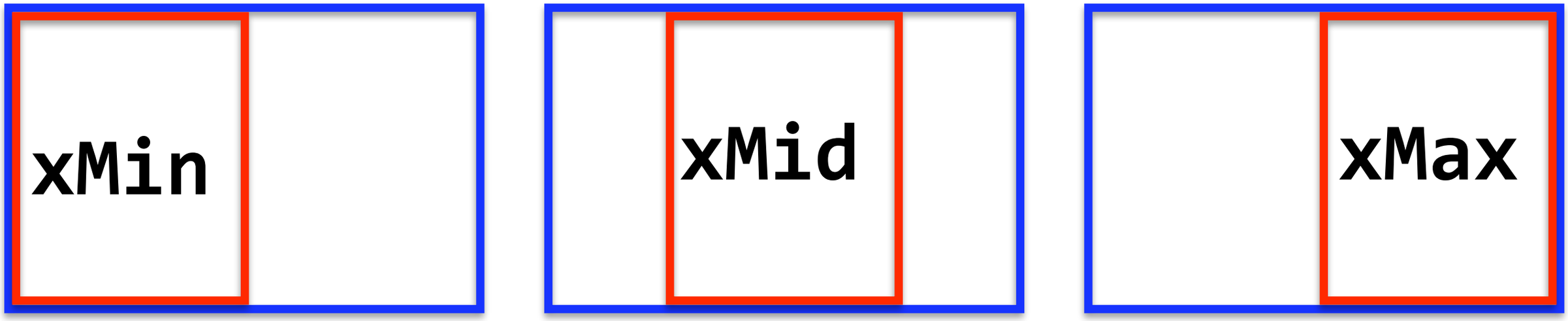


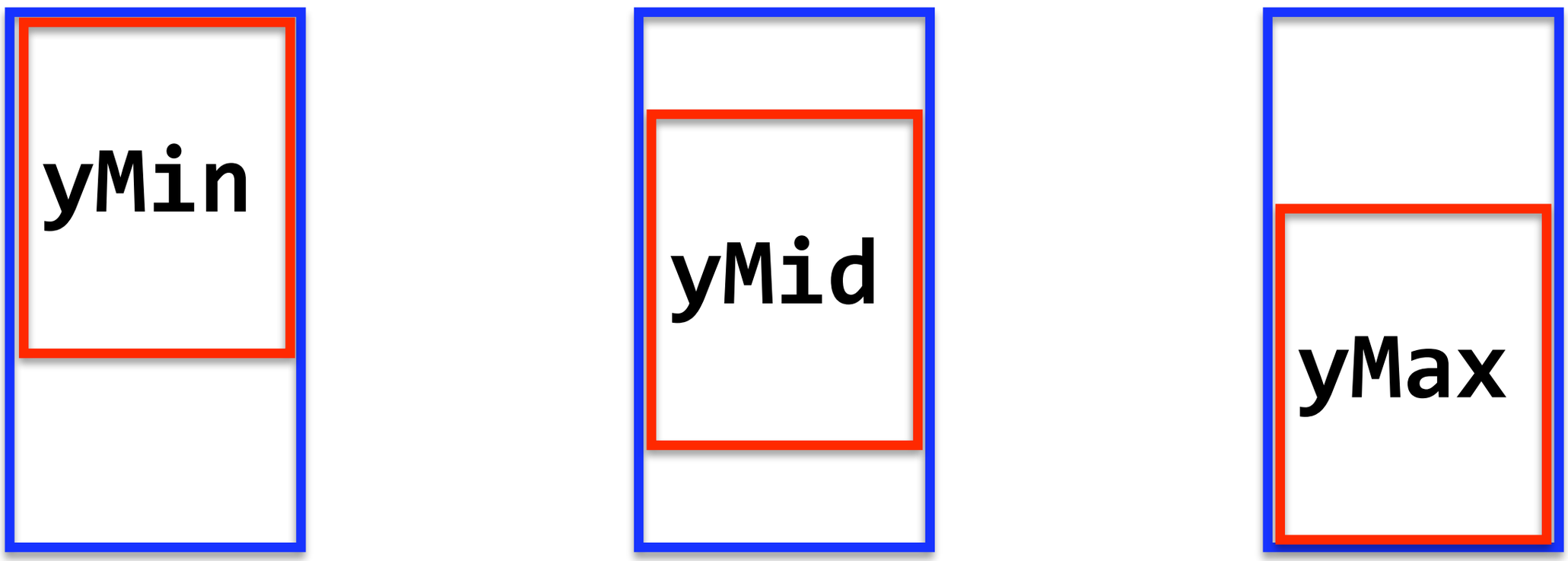
<?xml version="1.0" encoding="utf-8"?> <!-- Generator: Adobe Illustrator 19.0.0, SVG Export Plug-In . SVG Version: 6.00 Build 0) --> <svg version="1.1" baseProfile="tiny" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" viewBox="0 0 1024 576" xml:space="preserve"> <path id="XMLID_39_" fill="#FFFFFF" stroke="#231F20" stroke-miterlimit="10" d="M574.4,270.8c1.1-6.1,1.6-12.1,1.6-18.1 c-0.6-0.2-1.1-0.4-1.7-0.7H558c0-5.3-0.5-10.6-1.4-15.6l17.7-3.1c-1.1-6.1-2.6-11.9-4.6-17.6c-0.6,0-1.2,0-1.8,0l-15.3,5.6 c-1.8-5-4-9.7-6.6-14.2l15.6-9c-3.1-5.3-6.6-10.3-10.4-14.9c-0.6,0.2-1.1,0.4-1.7,0.6l-12.5,10.5c-3.4-4-7.1-7.7-11.1-11.1 l11.6-13.8c-4.7-4-9.7-7.5-14.8-10.5c-0.4,0.4-0.9,0.8-1.4,1.1L513,174c-4.5-2.6-9.3-4.8-14.2-6.6l6.1-16.9 c-5.8-2.1-11.7-3.7-17.5-4.8c-0.3,0.5-0.6,1.1-0.9,1.6l-2.8,16.1c-5.1-0.9-10.3-1.4-15.6-1.4v-18c-6.2,0-12.2,0.5-18.1,1.5 c-0.1,0.6-0.2,1.2-0.4,1.8l2.8,16.1c-5.2,0.9-10.3,2.3-15.2,4.1l-6.1-16.9c-5.8,2.1-11.3,4.7-16.5,7.6c0.1,0.6,0.2,1.2,0.3,1.8 L423,174c-4.5,2.6-8.8,5.7-12.8,9l-11.6-13.8c-4.7,4-9,8.3-12.9,12.8c0.3,0.5,0.6,1,0.9,1.6l12.5,10.5c-3.4,4-6.4,8.3-9,12.8 l-15.6-9c-3.1,5.3-5.7,10.8-7.7,16.4c0.5,0.4,0.9,0.8,1.4,1.2l15.3,5.6c-1.8,4.9-3.1,9.9-4.1,15.2l-17.7-3.1 c-1.1,6.1-1.6,12.1-1.6,18.1c0.6,0.2,1.1,0.4,1.7,0.7H378v0c0,5.3,0.5,10.6,1.4,15.6l-17.7,3.1c1.1,6.1,2.6,11.9,4.6,17.6 c0.6,0,1.2,0,1.8,0l15.3-5.6c1.8,5,4,9.7,6.6,14.2l-15.6,9c3.1,5.3,6.6,10.3,10.4,14.9c0.6-0.2,1.1-0.4,1.7-0.6l12.5-10.5 c3.4,4,7.1,7.7,11.1,11.1l-11.6,13.8c4.7,4,9.7,7.5,14.8,10.5c0.4-0.4,0.9-0.8,1.4-1.1L423,330c4.5,2.6,9.3,4.8,14.2,6.6l-6.1,16.9 c5.8,2.1,11.7,3.7,17.5,4.8c0.3-0.5,0.6-1.1,0.9-1.6l2.8-16.1c5.1,0.9,10.3,1.4,15.6,1.4v18c6.2,0,12.2-0.5,18.1-1.5 c0.1,0.6,0.2,1.2,0.4,1.8l-2.8-16.1c5.2-0.9,10.3-2.3,15.2-4.1l6.1,16.9c5.8-2.1,11.3-4.7,16.5-7.6c-0.1-0.6-0.2-1.2-0.3-1.8 L513,330c4.5-2.6,8.8-5.7,12.8-9l11.6,13.8c4.7-4,9-8.3,12.9-12.8c-0.3-0.5-0.6-1-0.9-1.6l-12.5-10.5c3.4-4,6.4-8.3,9-12.8l15.6,9 c3.1-5.3,5.7-10.8,7.7-16.4c-0.5-0.4-0.9-0.8-1.4-1.2l-15.3-5.6c1.8-4.9,3.1-9.9,4.1-15.2L574.4,270.8z"/></svg>



```
<svg
  width="800px"
  height="600px"
  viewBox="0 0 400 300"
  preserveAspectRatio="xMidYMid"
>
```

preserveAspectRatio



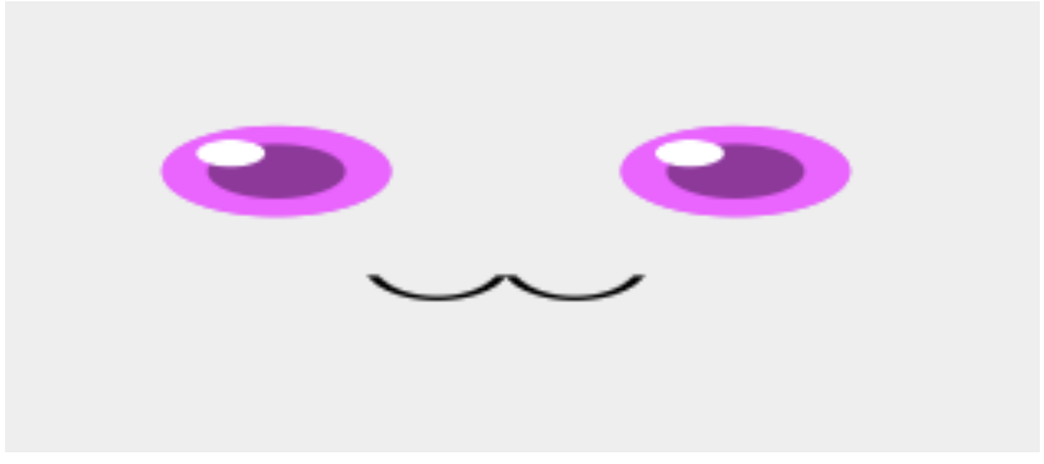


none

Play With viewBox

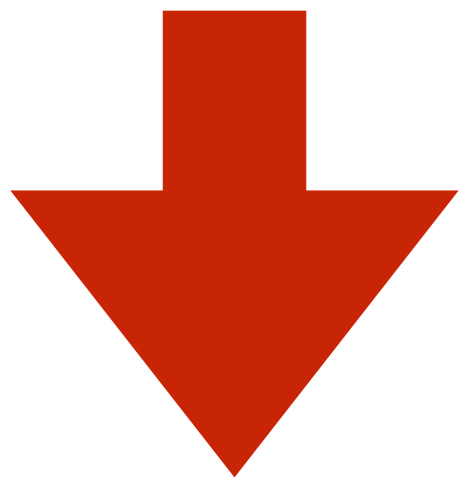
preserveAspectRatio="none"

調整 viewBox 裡的值



V4

```
d3.select("body")
  .text("Hello World");
```



```
d3.select("circle")
   .attrs({
     fill: "black"
   });
```

V4

```
d3.selectAll("circle")
    .attrs({
      fill: "black"
});
```

利用 D3.js 把所有圓圈變成紅色

05.html

加分題：你能只把眼睛的外圈變紅嗎？


```
d3.select("circle")
  .attr("fill", "red");
```

```
d3.selectAll("circle")
  .attrs({
    fill: "red"
  });
```

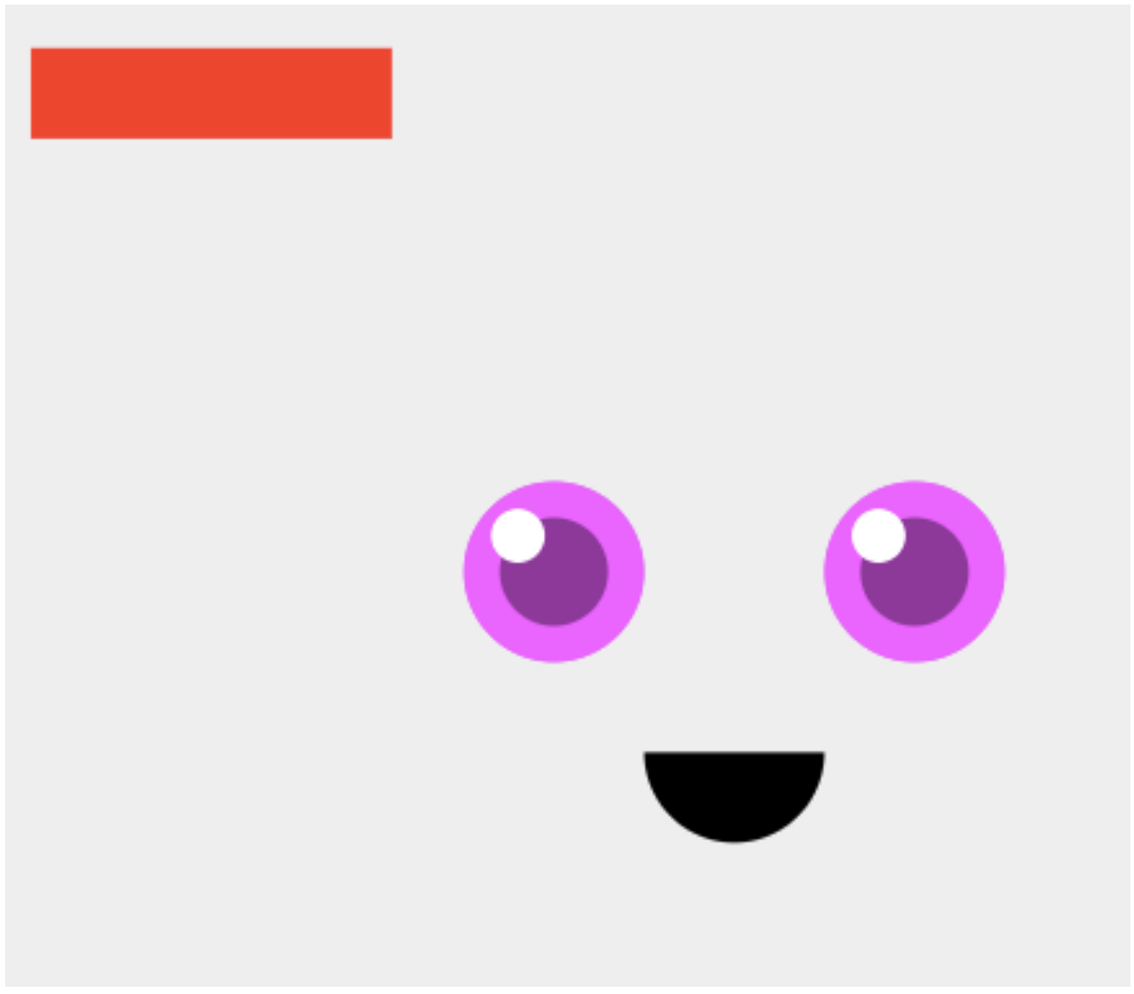
```
d3.select("svg")
    .append("rect")
    .attr("fill", "red");
```

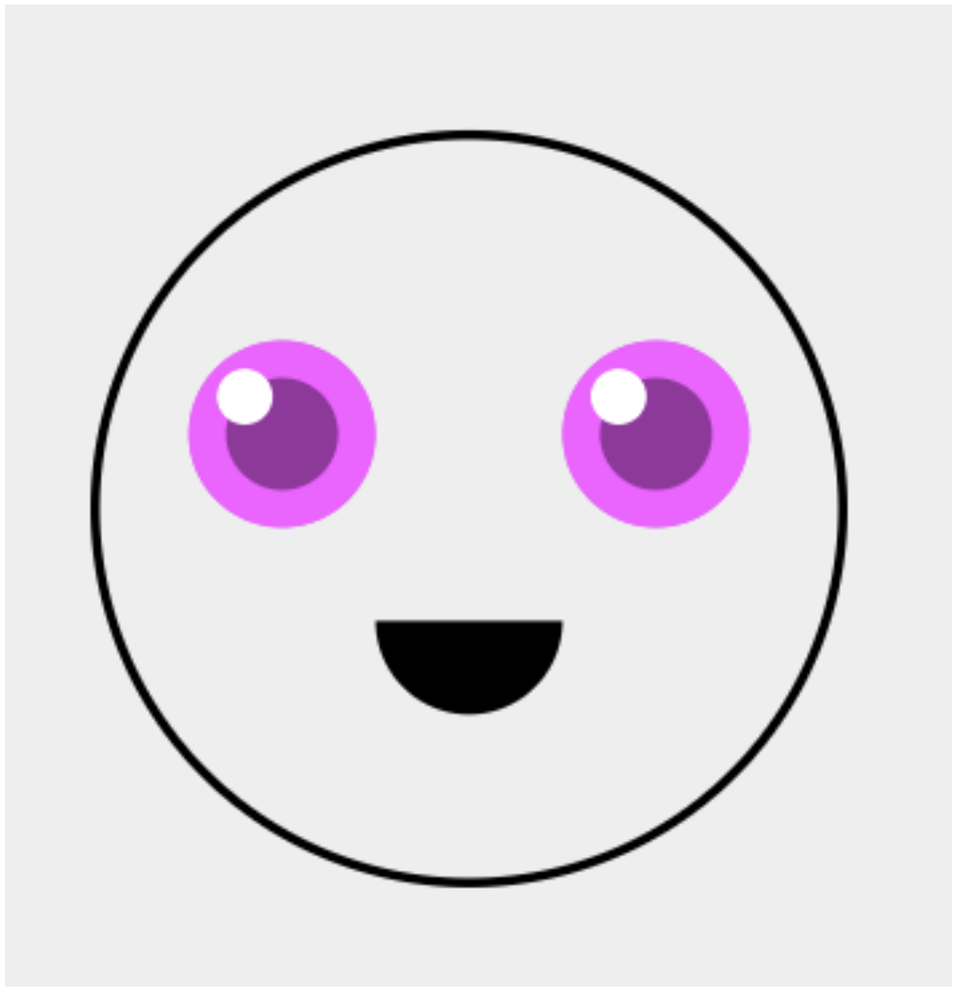
利用 D3.js 在笑臉旁插入一個紅色的矩形

06.html

加分題

用 D3.js加入臉的輪廓





```
d3.select("svg")
    .append("rect")
    .attrs({
      width: 10,
      height: 20
});
```

functional style

注意傳回值

以物件賦值

```
var node = d3.select("body");

node.attrs({"data-toggle": "#me"})

node.styles({"width": "100%"})

node.text("Hello World!")

node.html("<b>Hello World!</b>")
```

```
node.text("Hello World!")
```

```
<body>Hello World!</body>
```

```
node.html("<b>Hello World!</b>")
```

```
<body><b>Hello World!</b></body>
```

```
node.attrs({"data-toggle": "#me"})
```

```
<body data-toggle="#me"></body>
```

```
node.styles({"width": "100%"})
```

```
<body style="width:100%"></body>
```

```
var node = d3.select("body");

newnode = node.append("div");
```

```
<body><div></div></body>
```

return "div"

```
newnode.remove();
```

```
<body></body>
```

我要打十個

暴力法

```
= d3.select(“svg”).append(“rect”).attrs({…})
```

x 10

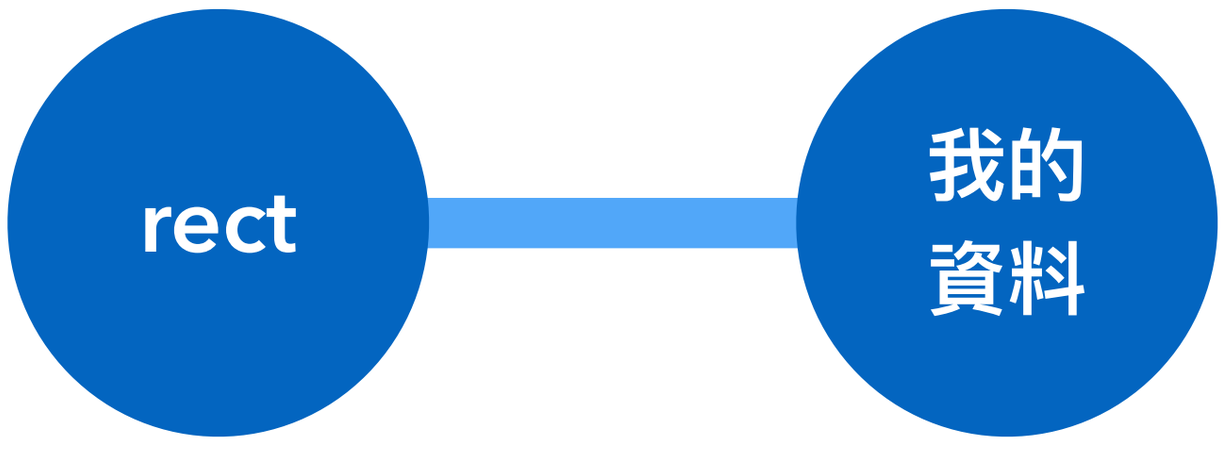
```
= d3.select(“svg”).append(“rect”).attrs({…})
  d3.select(“svg”).append(“rect”).attrs({…})
  d3.select(“svg”).append(“rect”).attrs({…})
  d3.select(“svg”).append(“rect”).attrs({…})
  d3.select(“svg”).append(“rect”).attrs({…})
  d3.select(“svg”).append(“rect”).attrs({…})
  d3.select(“svg”).append(“rect”).attrs({…})
  d3.select(“svg”).append(“rect”).attrs({…})
  d3.select(“svg”).append(“rect”).attrs({…})
  d3.select(“svg”).append(“rect”).attrs({…})
```

使用迴圈

```
for(var i =1; i<=10;i++) {
   d3.select("svg")
      .append("rect")
      .attrs({
        width: data[i]
      });
}
```

Data Binding

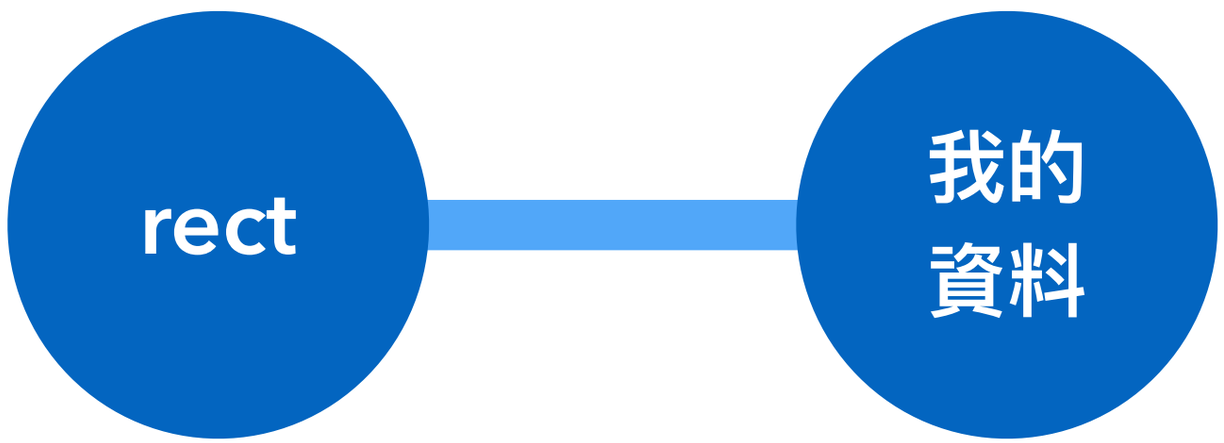
```
d3.select("rect")
    .datum( 我的資料 )
```



Data Binding

d3.select("rect").datum(我的資料)

```
d3.select("rect").datum();
```



```
d3.select("rect:nth-of-type(5)")
```

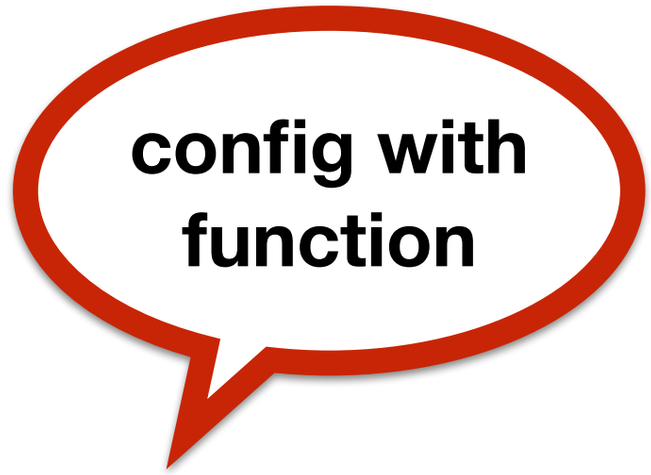
靠

```
.datum()
```

就對了

```
d3.select("rect:nth-of-type(5)")
  .attrs({
    width: 50
  })
```

```
d3.select("rect")
   .datum( 我的資料 )
   .attrs({
      "width":function(d,i) {
         return this;
      }
});
```



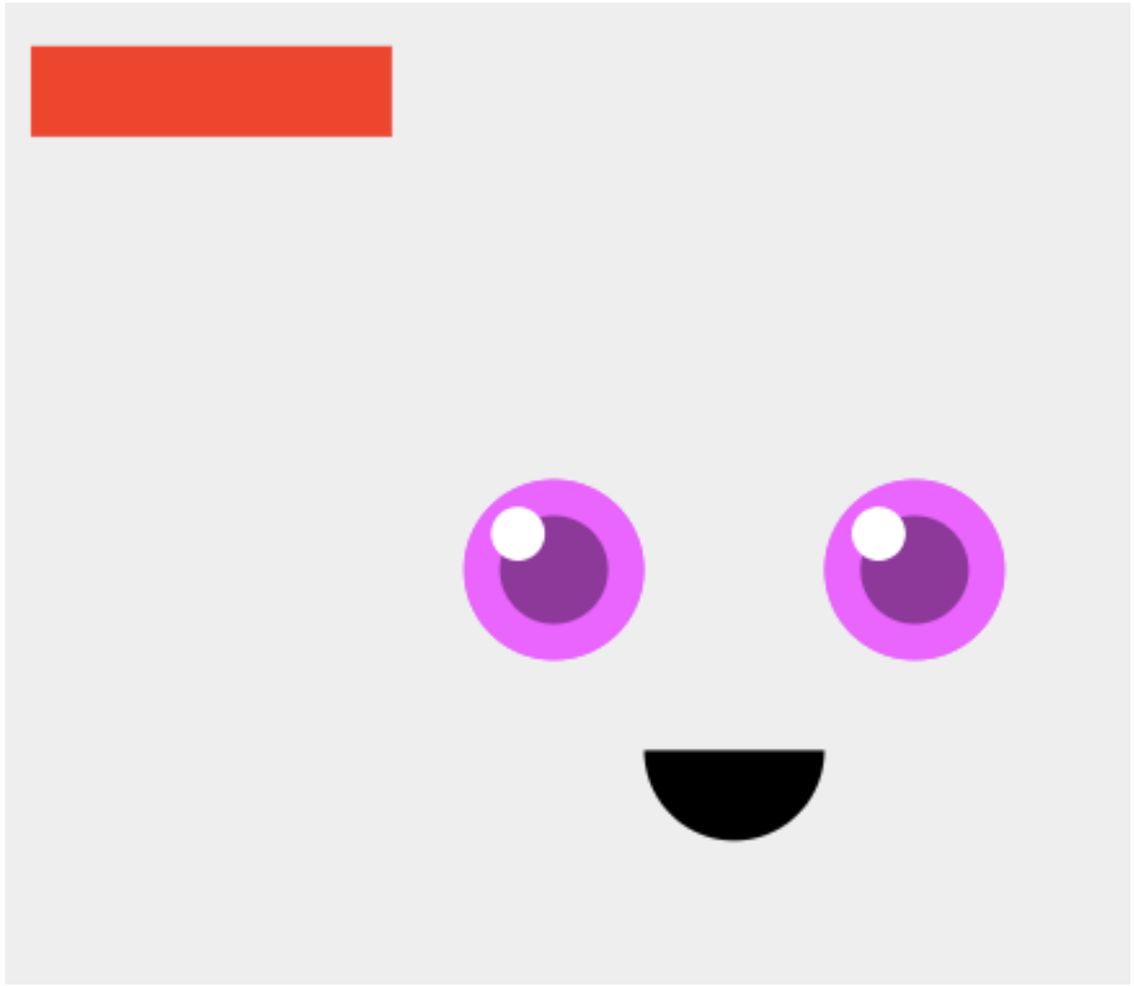

```
   d    = 我的資料
this    = <rect></rect>
   i    = 0
width = 回傳值
```

用datum與函式的方式設定方塊的顏色

07.html

加分題

用函式設定臉的背景色



```
for(var i =1; i<=10;i++) {
   d3.select("svg")
      .append("rect")
      .datum(data[i])
      .attrs({
         width: function(d,i) {
            return d * 10;
         }
      });
}
```

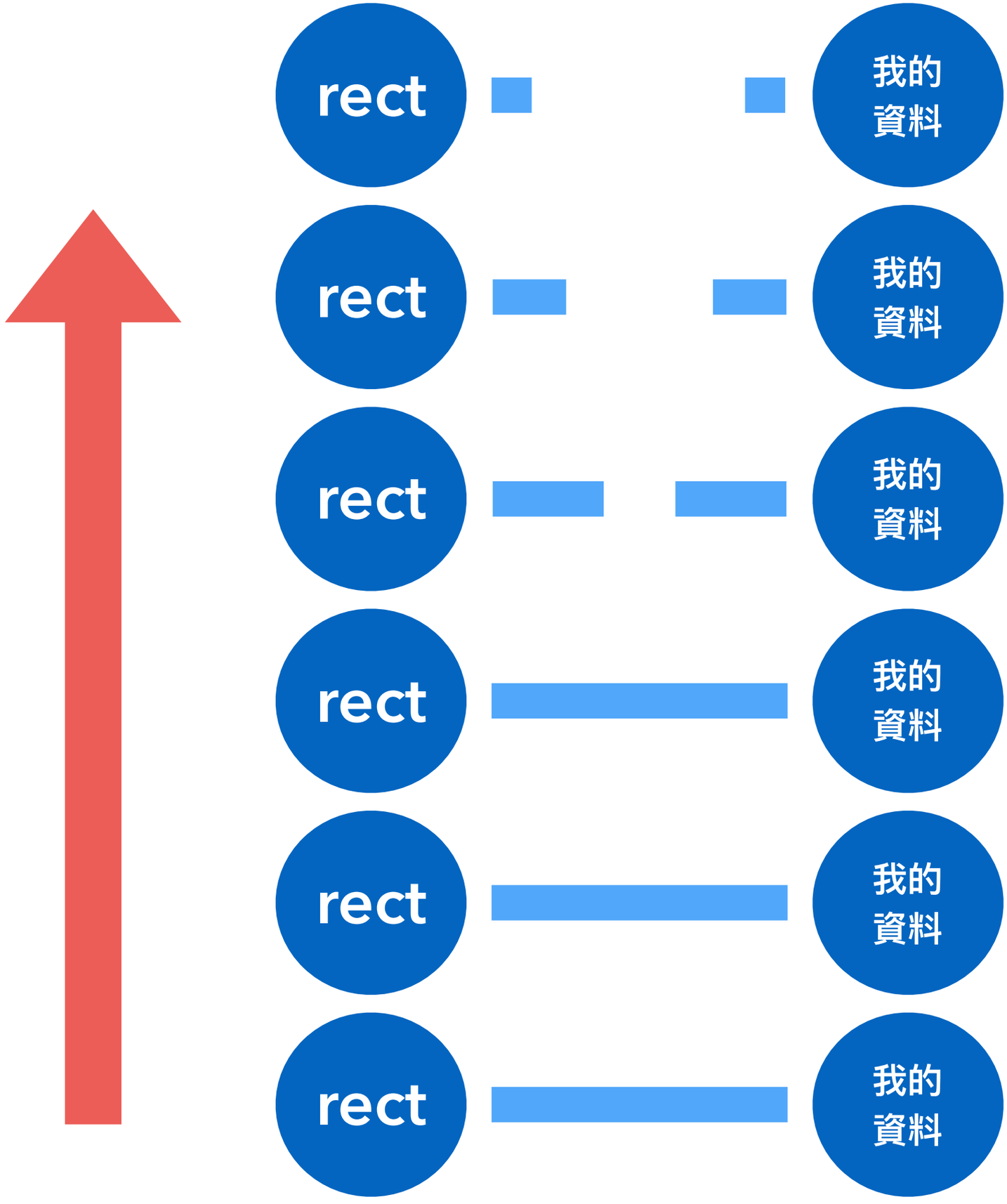
```
for(var i =1; i<=10;i++) {
   d3.select("svg")
      .append("rect")
      .datum(data[i])
      .attrs({
         width: function(d,i) {
            return d * 10;
         }
      });
   }
```

到底要多少個？

多出來的
怎麼辦？

幫我自動

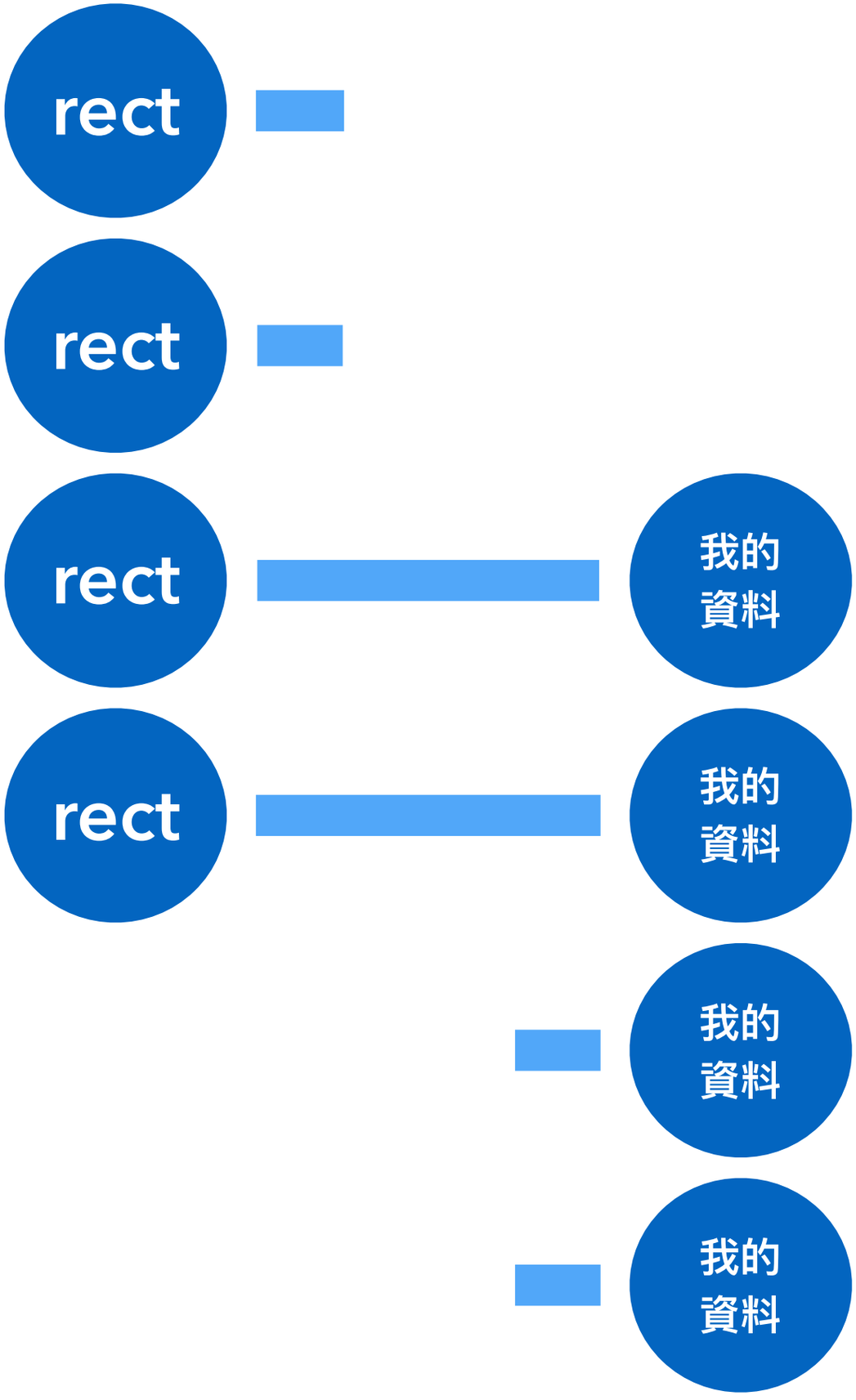
綁定資料



資料已經
不見的

之前已經
綁好的

還沒建立
元素的



我們需要

元素陣列

資料陣列

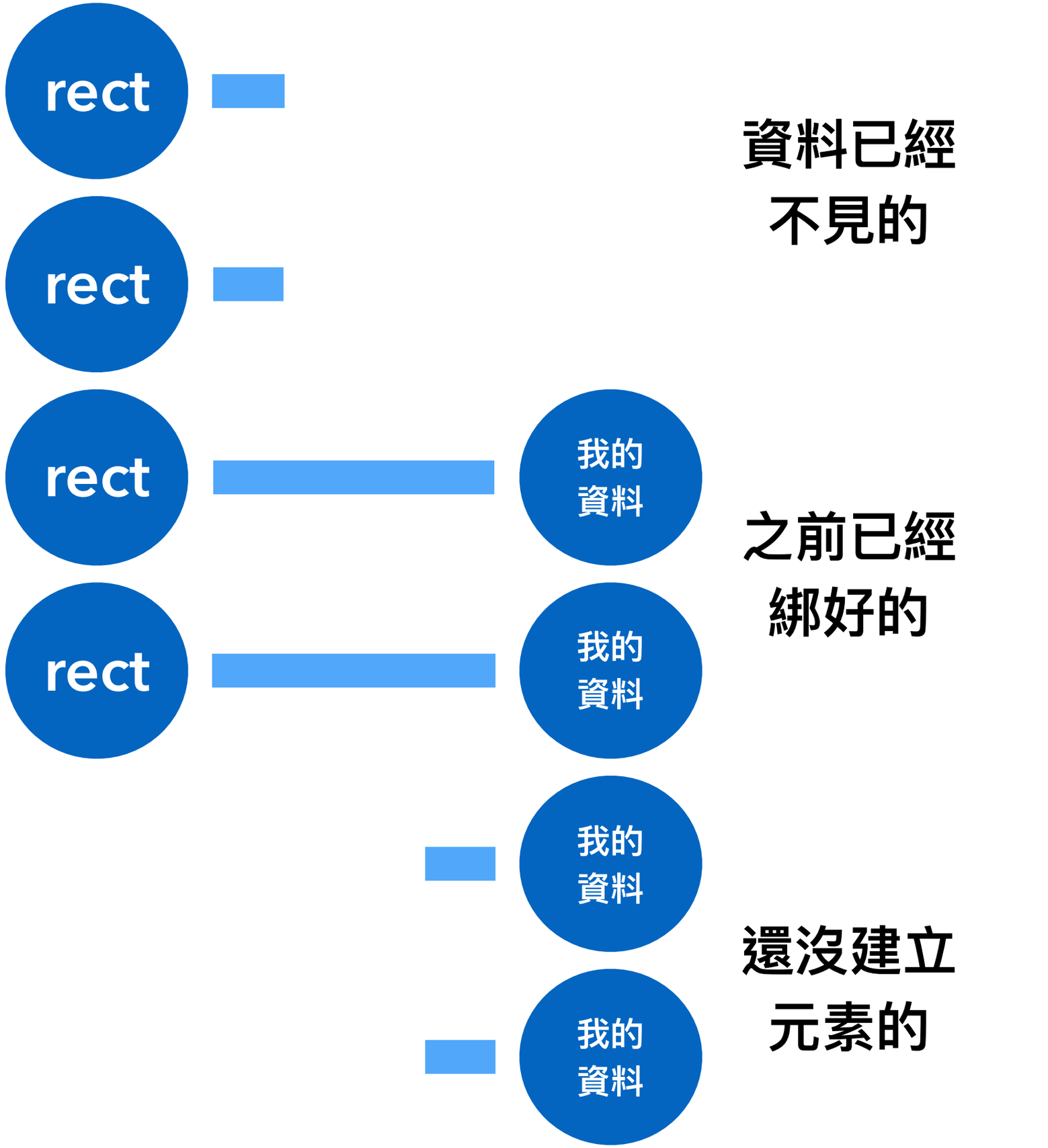
綁定動作

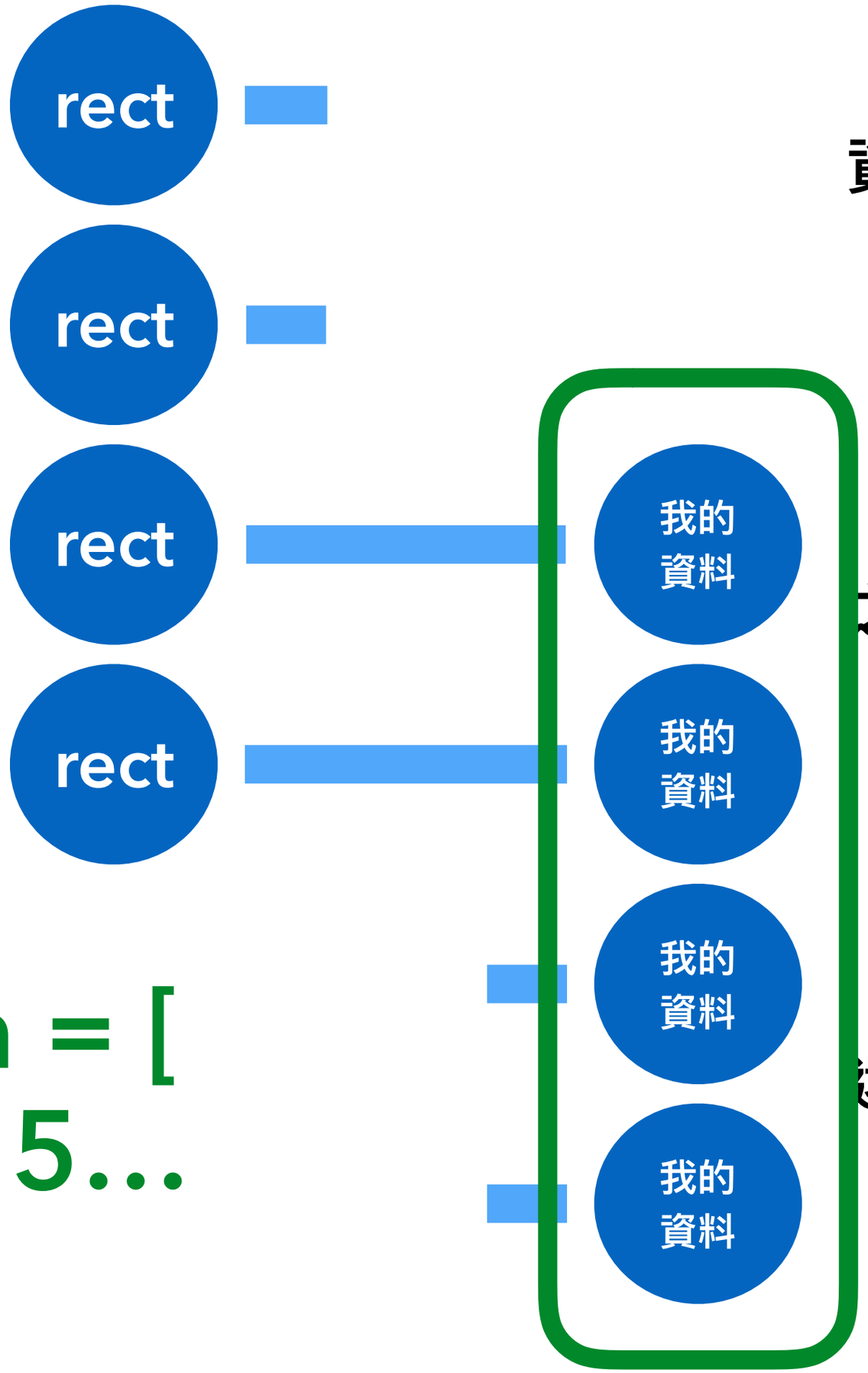
我們想要

A 集合

B 集合

C 集合





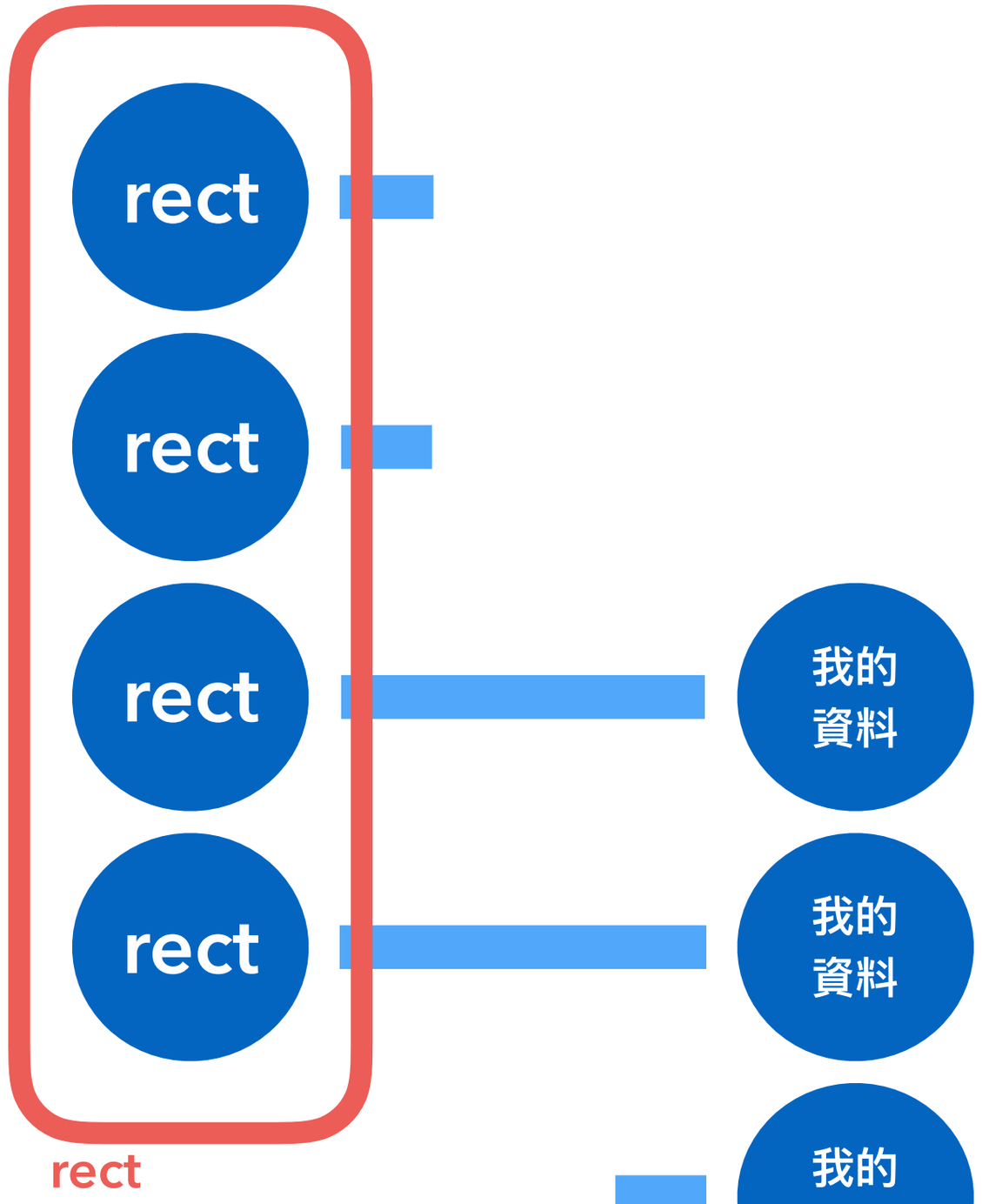
資料已經
不見的

之前已經
綁好的

還沒建立
元素的

```
var mydata = [
   3, 1, 4, 1, 5...
];
```

```
var mydata = [
   3, 1, 4, 1, 5...
];
```



資料已經
不見的

之前已經
綁好的

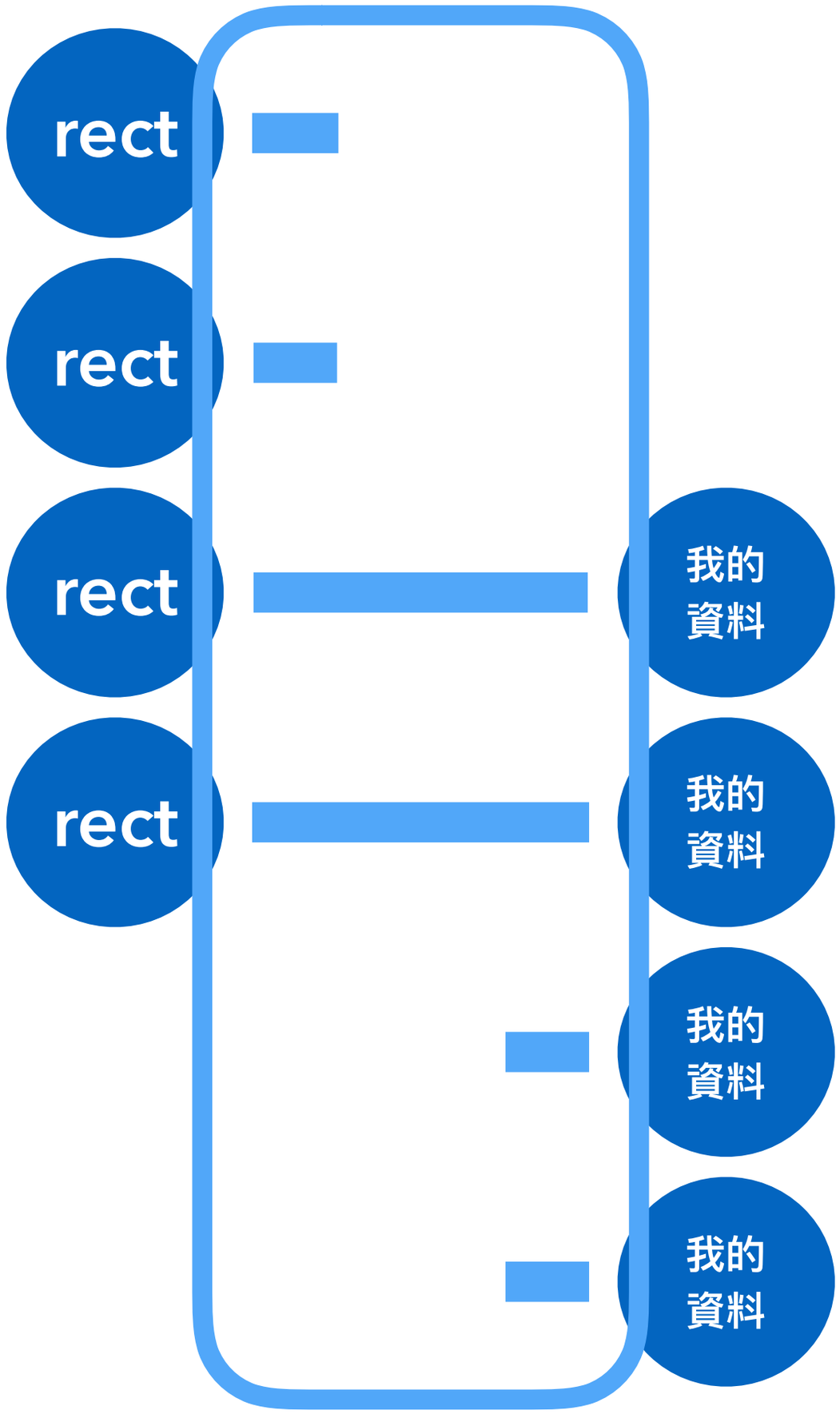
還沒建立
元素的

d3.selectAll("rect")

```
var mydata = [
   3, 1, 4, 1, 5...
];
```

d3.selectAll("rect")

.data(mydata)

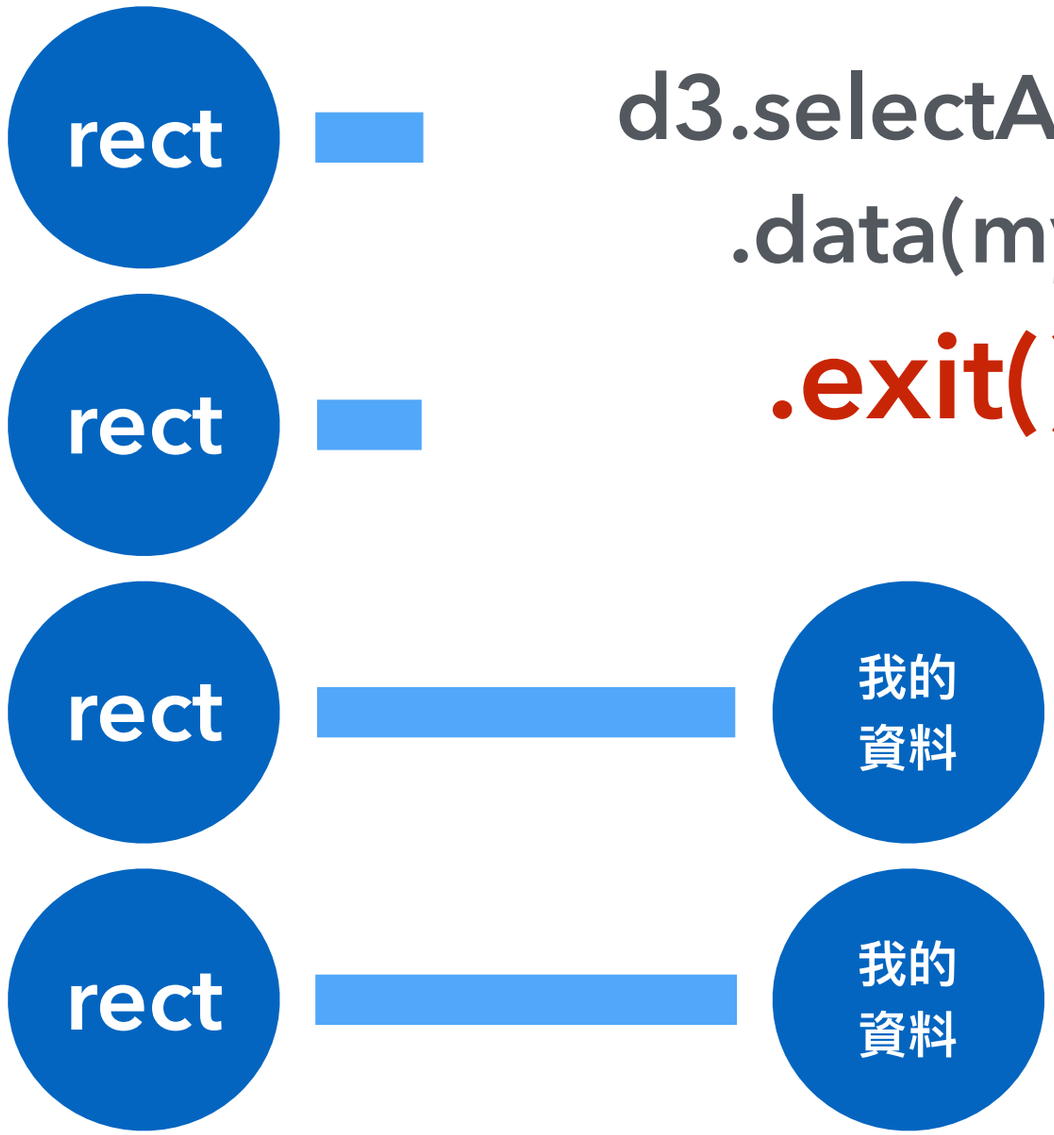


資料已經
不見的

之前已經
綁好的

還沒建立
元素的

d3.selectAll("rect")
.data(mydata)

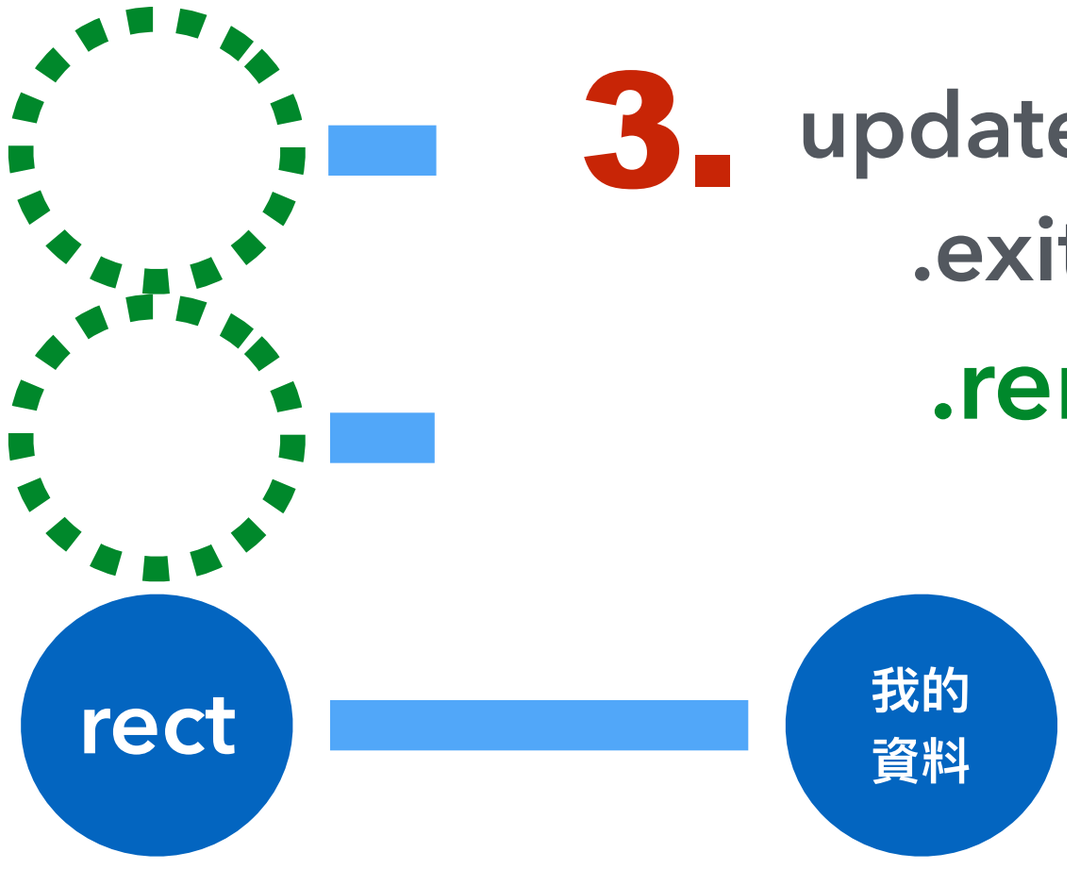


d3.selectAll("rect")
.data(mydata)
.exit()

d3.selectAll("rect")
.data(mydata)
.enter()

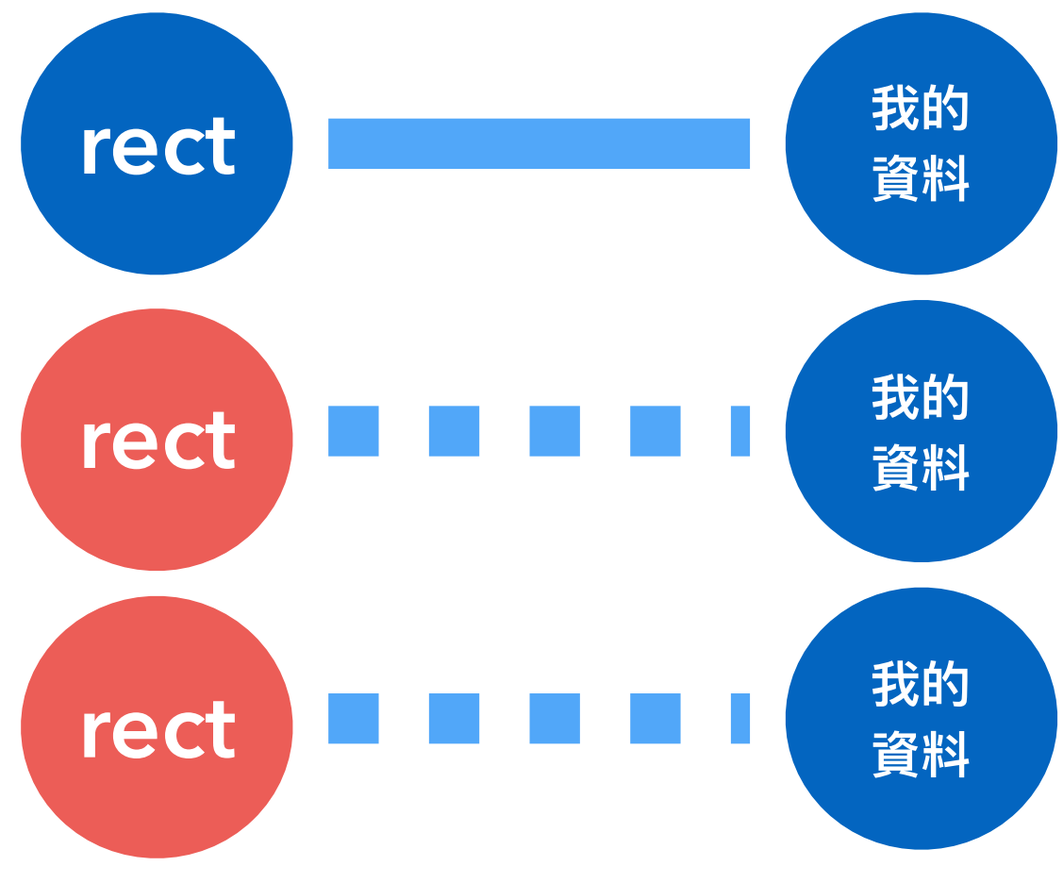
1.

```
var update = d3
    .selectAll("rect")
    .data(mydata);
```



2.

```
update
    .enter()
    .append("rect")
```



3.

```
update
    .exit()
    .remove()
```

```
var update = d3
    .select("svg")
    .selectAll("rect")
    .data([3,1,4,1,5,9]);

update.enter()
    .append("rect");

update.exit()
    .remove();
```

```
var update = d3
   .select("svg")
   .selectAll("rect")
   .data([3,1,4,1,5,9]);
update.enter().append("rect");
update.exit().remove();
```

```
d3.selectAll("rect")
   .attrs({
      width: function(d,i) {
         ....
      },
      ....
   });
```

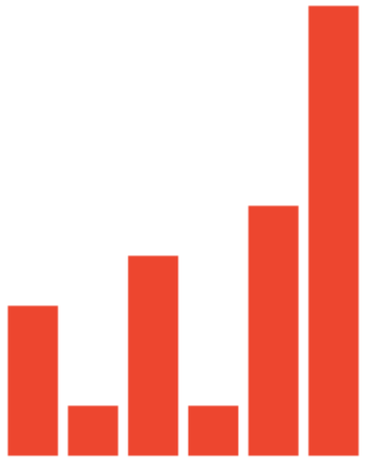
```
var update = d3.select("svg")
  .selectAll("rect").data([3,1,4,1,5,9]);
update.enter().append("rect");
update.exit().remove();

d3.select("svg").selectAll("rect").attrs({
  x: 10,
  y: function(d,i) { return i * 12; },
  width: function(d,i) { return d * 10; },
  height: 10,
  fill: "red"
});
```

畫一個垂直的長條圖

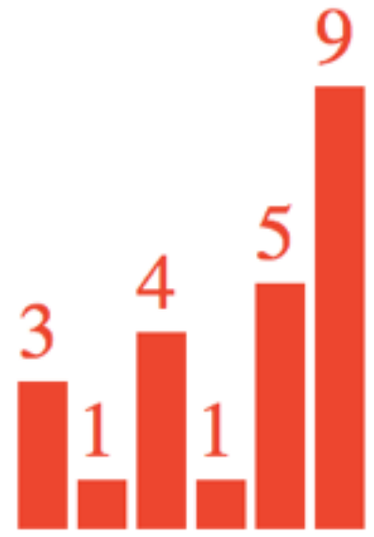
範例資料：[3,1,4,1,5,9,2,6,5]

範例如下:

加分題1



加分題2

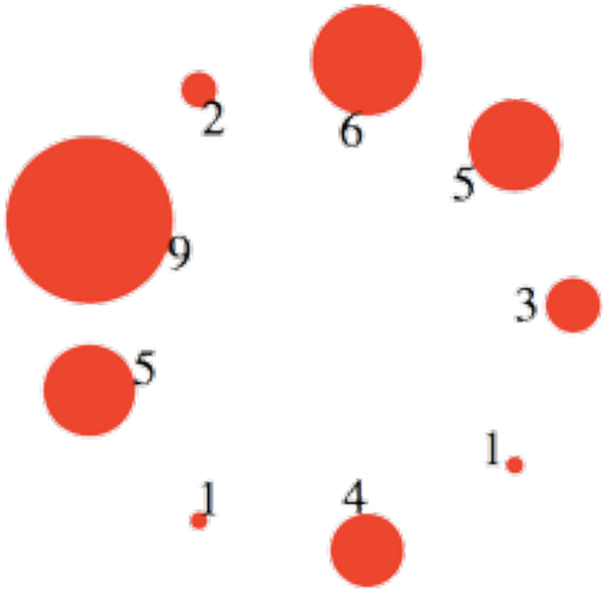


(使用 <text> 與 .text)

加分題3

(使用 circle)

加分題4



Recap

利用函數設定參數

```
d3.select("body")
  .styles({
    "width": function() {…},
}).text(function() { … });
```

```
d3.select("body")
    .datum( 我的資料 )
    .styles(
        "width":function(d,i) {
            return this;
    );
```

綁定的資料

元素的順序

設定的值

元素本身

```
d3.select("svg")
  .select(function(d,i) {
    return this;
  })
```

```
d3.select("svg")
  .selectAll("rect")
  .data(function(d,i) {
    return this;
  })
```

串連式的函式呼叫

```
d3.select("body")
  .styles({"width": "100%"})
  .text("hello world!")
  .attrs( … )
```

注意傳回值

```
d3.select(“svg”)       // 傳回 [svg]
  .selectAll(“rect”) // 傳回 [rect,…]
  .attrs(…)                    // 傳回 [rect,…]
  .data(…) // 傳回 [rect,](update selection版)
```

注意傳回值

```
(d3.area()                          // 傳回函式
    .x()                            // 傳回函式
    .y0()                           // 傳回函式
    .y1()                           // 傳回函式
)({ … });                           // 傳回字串
```

D3 Selection

選取 svg (只取第一個)

```
d3.select("svg")
```

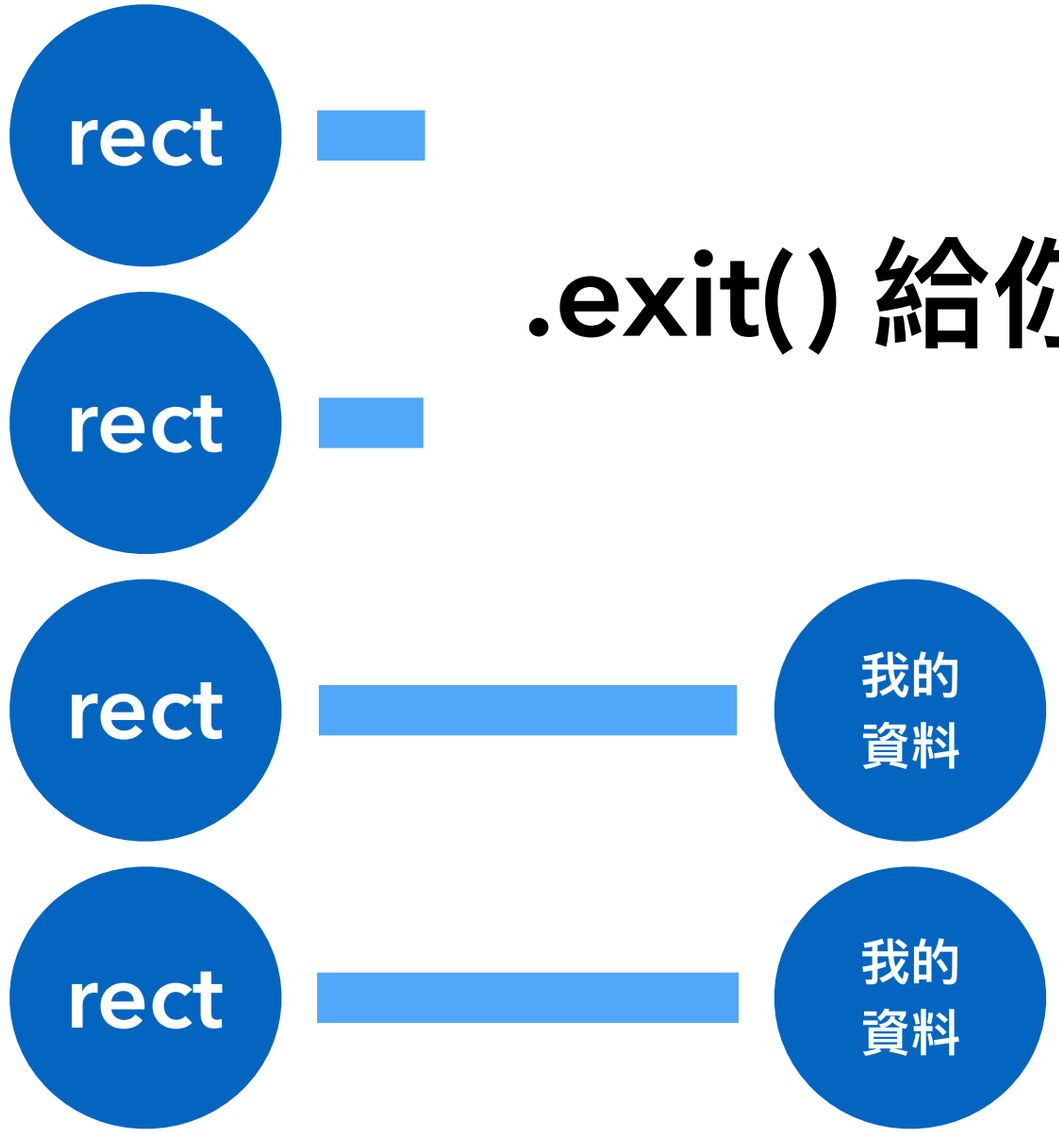
選取 rect (全部)

```
d3.selectAll("rect")
```

選取 svg 下的所有 rect

```
d3.select("svg").selectAll("rect")
```

資料已經
不見的



.exit() 給你元素

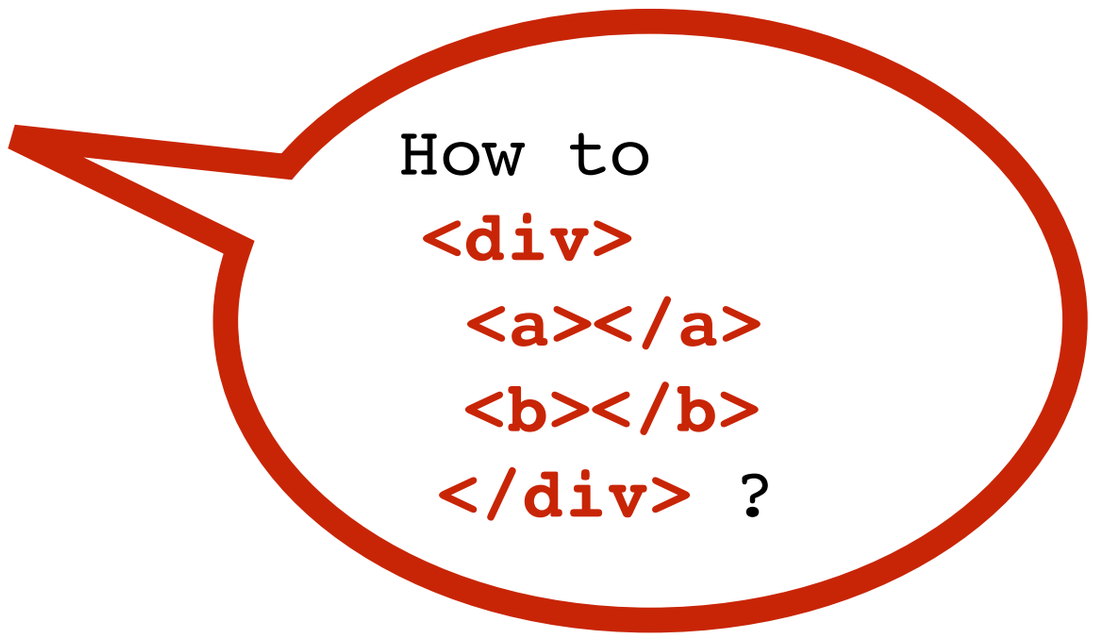
之前已經
綁好的

.enter() 給你棒子

還沒建立
元素的

DOM 結構調整

```
var update = d3.select(“body”)
  .selectAll(“div”)
  .data(data);

update.exit().remove();
update.enter().append(“div”)
  .append(“a”)
  .append(“b”);
```



子元素的資料

```
update.exit().remove();
update.enter().append(“div”)
    .append(“a”).attrs({
        “width”:
        function(D,i) {
            return D + “px”;
        }
});
```

D 的值是多少？

文件結構 – 使用 “each” 深入

```
update.enter().append(“div”)
.each(function(d,i) {
    var node = d3.select(this);
    node.append(“a”);
    node.append(“b”);
});
```

Nested Data Binding

```
update.enter().append(“div”)
.each(function(d,i) {
      d3.select(this).selectAll(“a”)
            .data([“a”,”b”,”c”])
            .enter().append(“a”);
});
```

```
update.enter().append(“div”)
.each(function(d,i) {
    d3.select(this).selectAll(“a”)
        .data([“a”,”b”,”c”])
        .enter().append(“a”);
});
```

```
d3.selectAll(“div > a”)
.attrs({
  width:function(D,i) { …… }
});
```

D 的值又是多少？

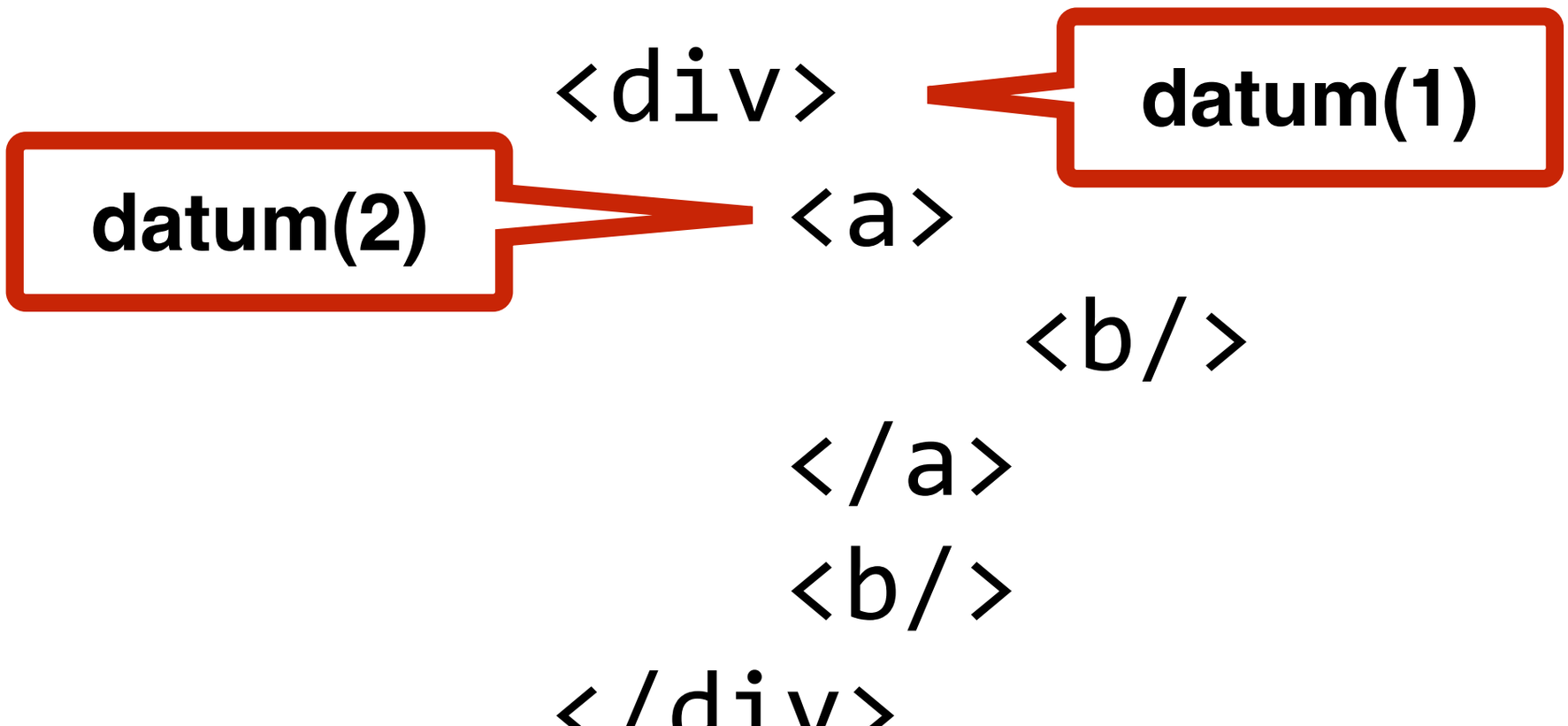
```
update.exit().remove();
update.enter().append(“div”)
  .append(“a”).attrs({
    “width”:
    function(D,i) {
        return D + “px”;
    }
  });
```

```
update.enter().append(“div”)
.each(function(d,i) {
    d3.select(this).selectAll(“a”)
        .data([“a”,”b”,”c”])
        .enter().append(“a”);
});

d3.selectAll(“div > a”)
.attrs({
  width:function(D,i) { …… }
});
```

資料的儲存方式

1. 資料存在元素的 __data__ 屬性中
2. D3js 會往父元素不斷尋找 __data__
3. 只會在 append 時探尋一次資料

QUIZ:



1. **d3.select(“b”)**
2. **d3.select(“div > b”)**

Recap

d3.

select
selectAll

make selection

data binding

data
datum

update selection

enter
exit

attrs
styles
text / html

alter attributes

append
remove

manipulate node

來點動畫吧！

```
d3.selectAll("rect")
  .attrs({width: 0})
  .attrs({width: 100})
```

來點動畫吧！

```
d3.selectAll("rect")
  .attrs({width: 0})
  .transition()
  .attrs({width: 100})
```

來點動畫吧！

利用
Duration
設定長度

利用 Delay
延遲發動

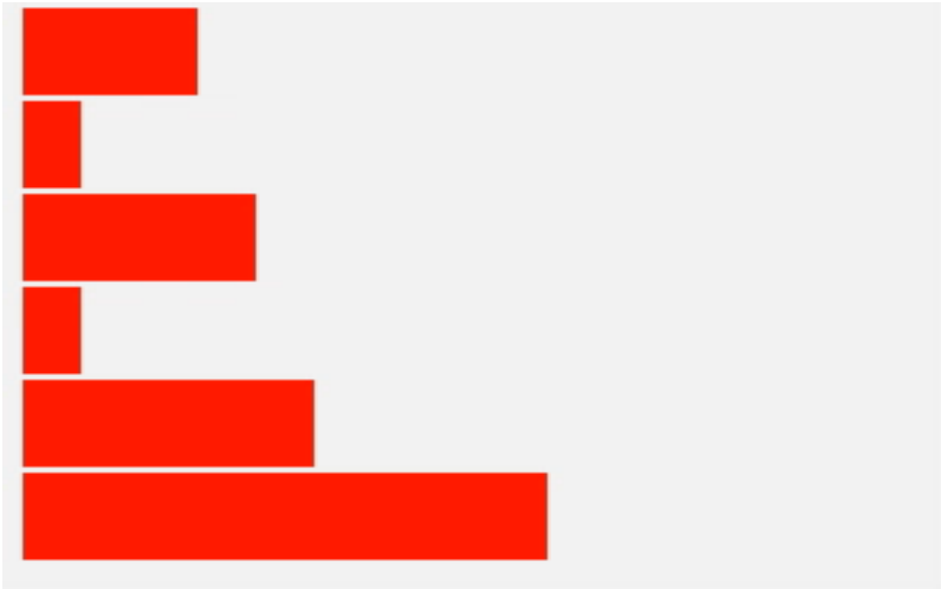
```
d3.selectAll("rect")
   .attrs({width: 0})
   .transition()
   .duration(1000)
   .delay(1000)
   .attrs({width: 100})
```

單位: 千分之一秒

畫一個長出來的長條圖

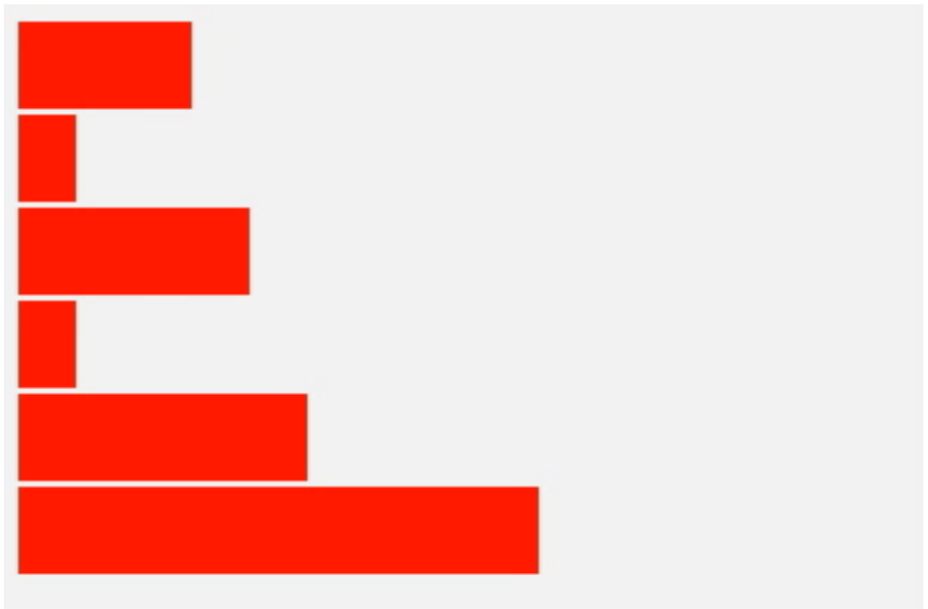
09.html

基本題

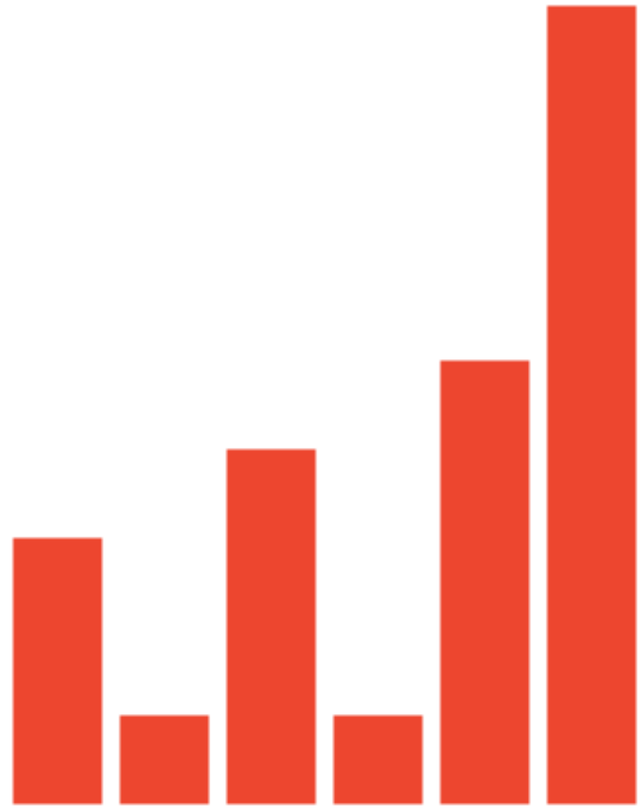


加分題1

讓每個 Bar 長出來的速度不一樣



加分題2: 從下面往上長



```
d3.selectAll("rect")
  .attrs({width: 0})
  .transition()
  .attrs({width: 100})
```


```
d3.selectAll("rect")
  .attrs({width: 0})

d3.selectAll("rect")
  .transition()
  .attrs({width: 100})
```

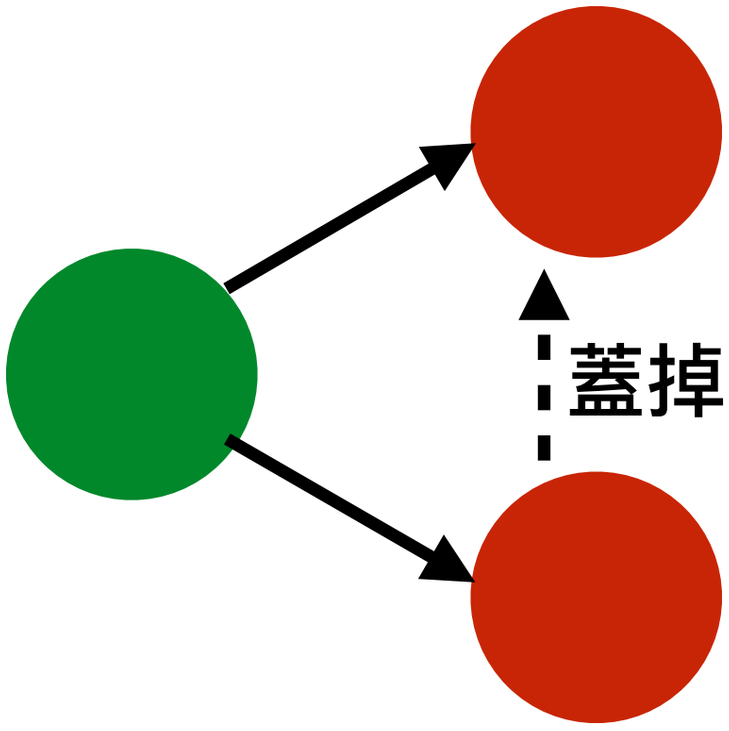
Quiz

多個 transition 會怎樣？

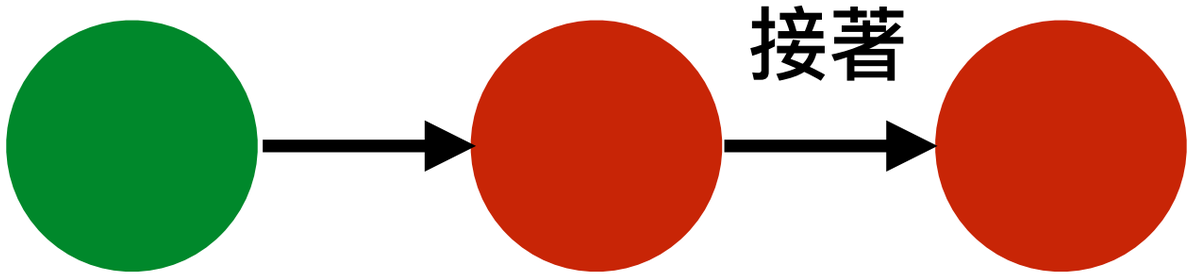
```
d3.selectAll(“rect”)
    .transition().duration(1000)
        .attrs({fill: “#f00”})

d3.selectAll(“rect”)
  .transition().duration(1000)
      .attrs({fill: “#0f0”});
```

Chaining Transition

```
d3.selectAll(“rect”)
    .transition().duration(1000)
        .attrs({fill: “#f00”})
    .transition().duration(1000)
        .attrs({fill: “#0f0”})
```



多個 Transition



Chained Transition

Transition Naming

```
d3.selectAll(“rect”)
    .transition("opacity")
        .attrs({opacity: 1})
```

```
d3.selectAll("rect")
  .transition("fill")
      .attrs({fill: “#0f0”})
```

Good Pattern

Binding 與 Styling

```
d3.selectAll(“rect”)
   .data(data)
   .enter()
   .append(“rect”)
   .attrs({
      class: “…”
   });
```

只做初始設定

```
d3.selectAll(“rect”)
   .attrs({
      …
   }).styles({
      …
   })
```

要分開

Good Pattern

Binding

```
function bind(data) {
  d3.selectAll(“rect”)
    .data(data)
    .enter()
    .append(“rect”)
    .attrs({
      class: “…”
    });
}
```

Styling

```
function render() {
    d3.selectAll(“rect”)
      .attrs({
          …
      }).styles({
        …
      })
}
```

包裝起來，可重複使用

Good Pattern

Binding

```
function bind(data) {
  d3.selectAll(“rect”)
    .data(data)
    .enter()
    .append(“rect”)
    .attrs({
      class: “…”
    });
}
```

Styling

```
function render(delay) {
    d3.selectAll(“rect”)
      .transition()
      .delay(delay)
      .attrs({
          …
      }).styles({
        …
      })
}
```

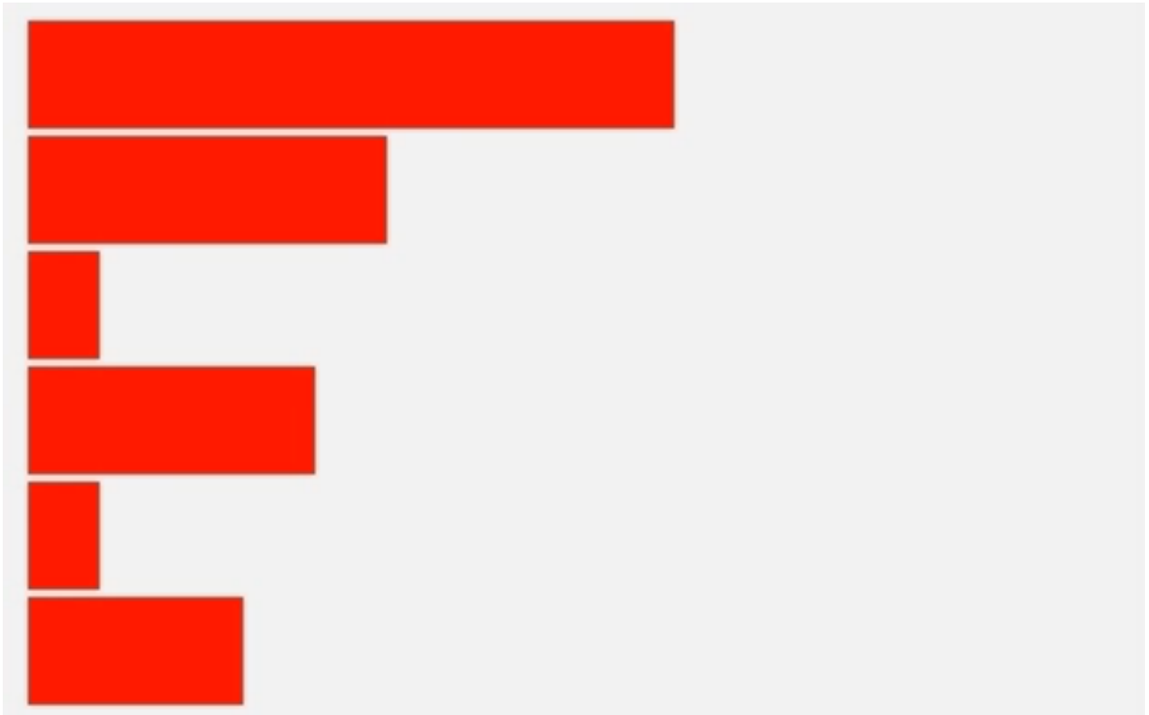
為了動畫，帶個延遲參數

Rendering

```
bind([3,1,4,1,5,9]);
render(0);
bind([9,5,1,4,1,3]);
render(1000);
```

10.html

畫一個長條圖，從 [3,1,4,1,5,9] 變形到 [9,5,1,4,1,3]



加分題

讓圖表在兩組資料間不斷來回變形

```
setInterval(
  function() { … },
  millisecond
);
```

Chaining Transition


```
d3.selectAll(“rect”)
    .transition()
        .delay(1000)
        .duration(1000)
            .attrs({fill: “#f00”})
    .transition()
        .delay(2000)
        .duration(1000)
            .attrs({fill: “#0f0”})
```

Quiz

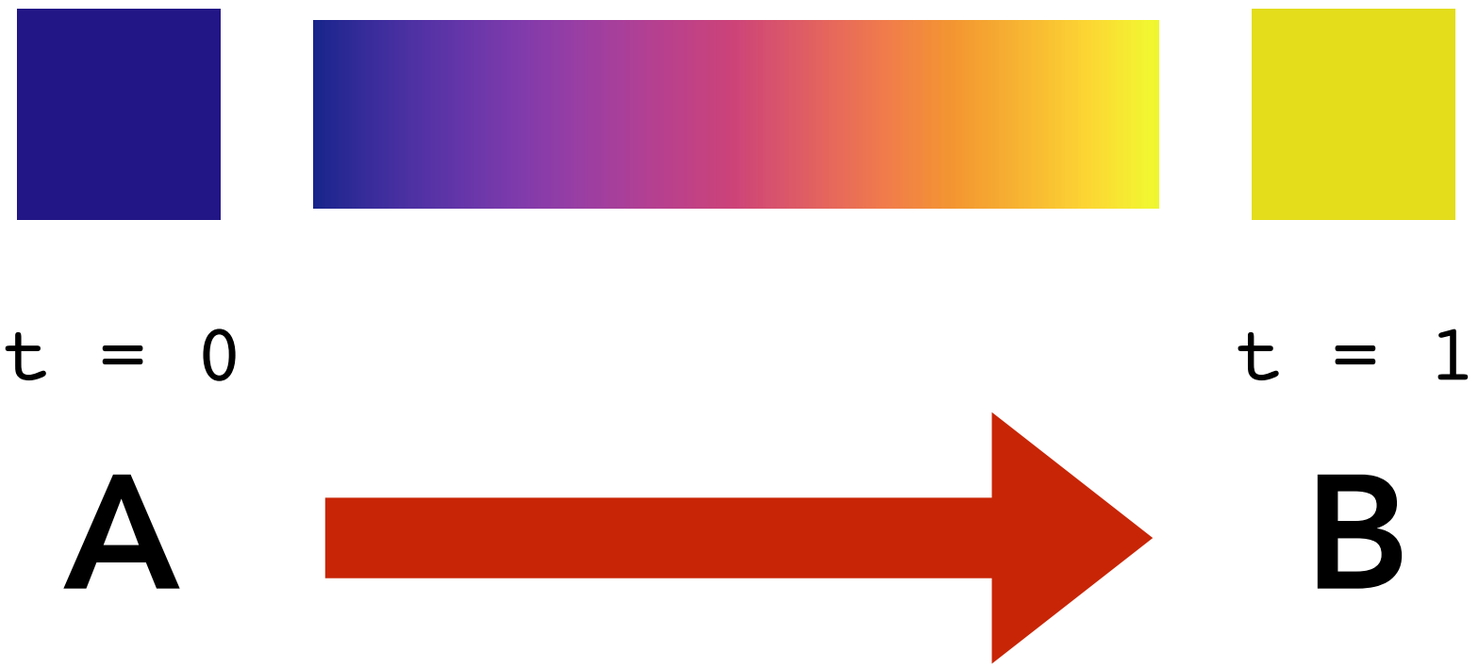
字串參數可以做動畫嗎？

```
d3.selectAll("rect")
  .attrs({fill: "blue"})
  .transition().duration(100)
  .attrs({fill: "yellow"});
```

start

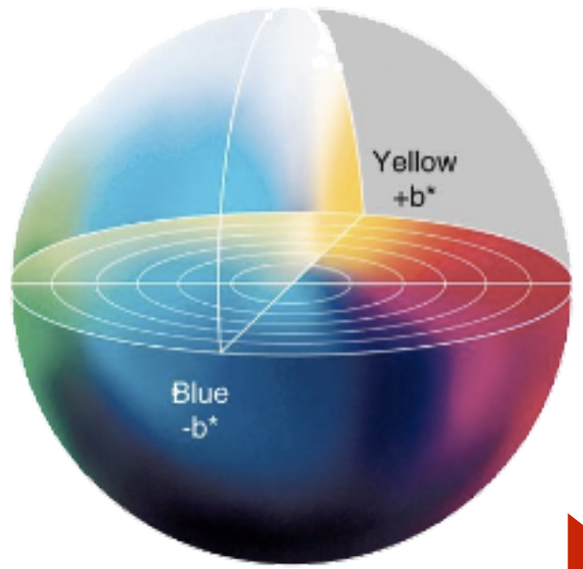
end

動畫的原理



動畫的原理

t = 0



t = 1

blue → yellow

初始值　　中間值　　目標值

i("blue","yellow",t)

動畫的原理

線性內插

t = 0　**i = 5*(1- t) + 9 * t**　t = 1

5　9

初始值　中間值　目標值

i(5,9,t)

"blue" → "yellow"

d3.interpolate("blue","yellow")

↓

d3.interpolateRgb

d3.interpolateString

d3.interpolateArray

d3.interpolateObject

Simple Number Interpolator

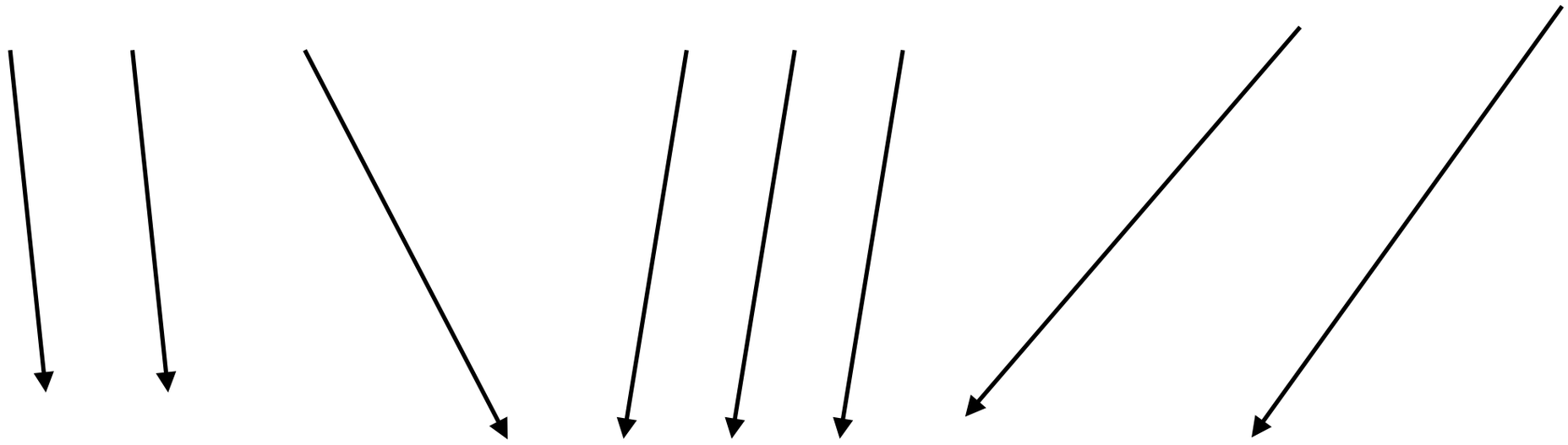
```
function(a, b) {
  a = +a;
  return function(t) {
    return a + (b - a) * t;
  };
});
```

Quiz

字串參數可以做動畫嗎？

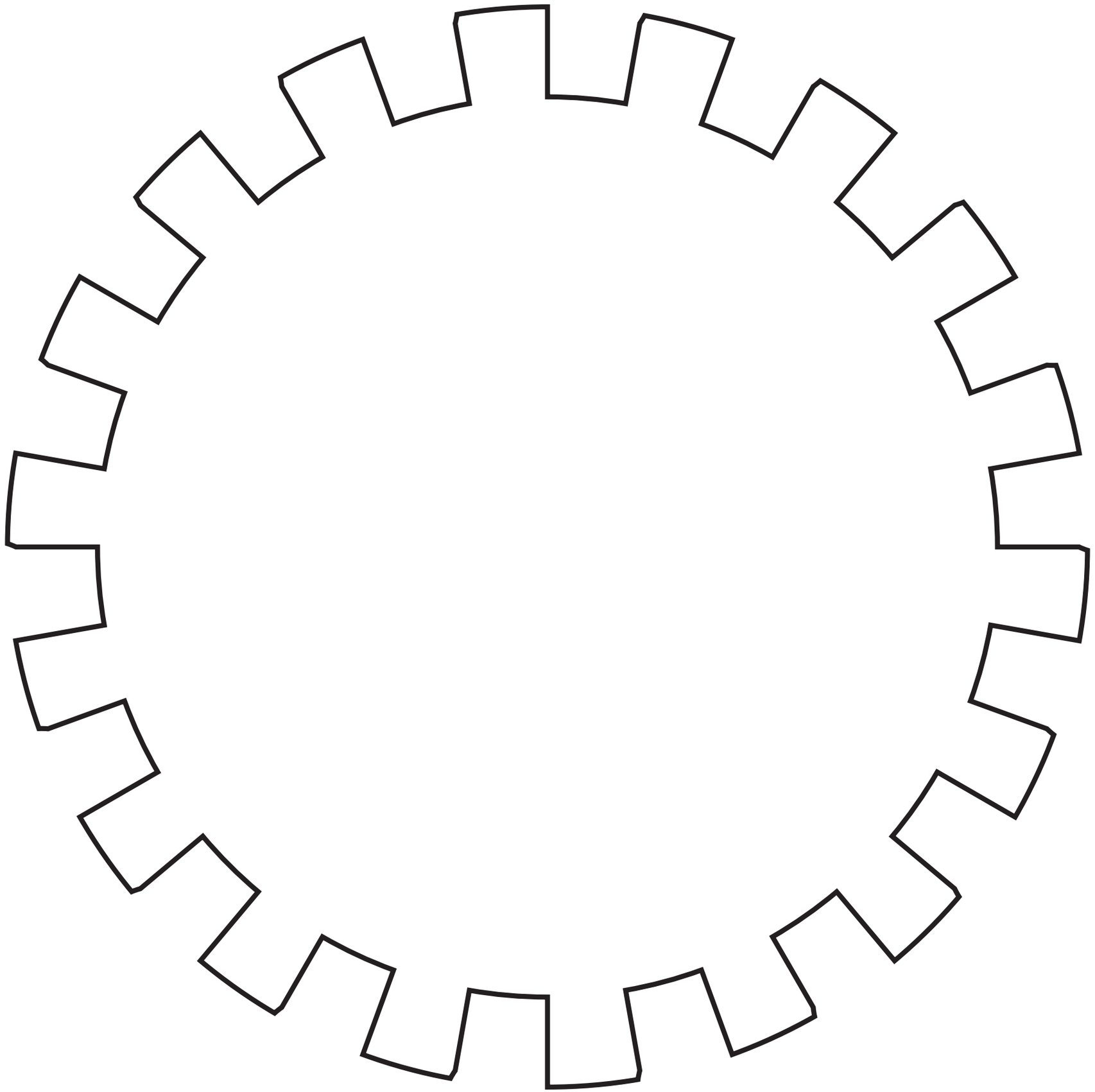
```
d3.selectAll("rect")
  .attrs({fill: "Sharkspear"})
  .transition().duration(100)
  .attrs({fill: "Oedipus"});
```

1鄉2里共3夫子，不識4書5經6義，竟敢教789子，10分大膽



10室9貧，湊得8兩7錢6分5毫4厘，尚且3心2意，1等下流

1.94790193749999997室2.7372570625貧，湊得3.526612187499999 6兩4.3159673125錢5.1053224375分5.894677562499999毫706.3218865625厘，尚且9.2627429374999 99心2意，1等下流



<?xml version="1.0" encoding="utf-8"?> <!-- Generator: Adobe Illustrator 19.0.0, SVG Export Plug-In . SVG Version: 6.00 Build 0) --> <svg version="1.1" baseProfile="tiny" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" viewBox="0 0 1024 576" xml:space="preserve"> <path id="XMLID_39_" fill="#FFFFFF" stroke="#231F20" stroke-miterlimit="10" d="M574.4,270.8c1.1-6.1,1.6-12.1,1.6-18.1 c-0.6-0.2-1.1-0.4-1.7-0.7H558c0-5.3-0.5-10.6-1.4-15.6l17.7-3.1c-1.1-6.1-2.6-11.9-4.6-17.6c-0.6,0-1.2,0-1.8,0l-15.3,5.6 c-1.8-5-4-9.7-6.6-14.2l15.6-9c-3.1-5.3-6.6-10.3-10.4-14.9c-0.6,0.2-1.1,0.4-1.7,0.6l-12.5,10.5c-3.4-4-7.1-7.7-11.1-11.1 l11.6-13.8c-4.7-4-9.7-7.5-14.8-10.5c-0.4,0.4-0.9,0.8-1.4,1.1L513,174c-4.5-2.6-9.3-4.8-14.2-6.6l6.1-16.9 c-5.8-2.1-11.7-3.7-17.5-4.8c-0.3,0.5-0.6,1.1-0.9,1.6l-2.8,16.1c-5.1-0.9-10.3-1.4-15.6-1.4v-18c-6.2,0-12.2,0.5-18.1,1.5 c-0.1,0.6-0.2,1.2-0.4,1.8l2.8,16.1c-5.2,0.9-10.3,2.3-15.2,4.1l-6.1-16.9c-5.8,2.1-11.3,4.7-16.5,7.6c0.1,0.6,0.2,1.2,0.3,1.8 L423,174c-4.5,2.6-8.8,5.7-12.8,9l-11.6-13.8c-4.7,4-9,8.3-12.9,12.8c0.3,0.5,0.6,1,0.9,1.6l12.5,10.5c-3.4,4-6.4,8.3-9,12.8 l-15.6-9c-3.1,5.3-5.7,10.8-7.7,16.4c0.5,0.4,0.9,0.8,1.4,1.2l15.3,5.6c-1.8,4.9-3.1,9.9-4.1,15.2l-17.7-3.1 c-1.1,6.1-1.6,12.1-1.6,18.1c0.6,0.2,1.1,0.4,1.7,0.7H378v0c0,5.3,0.5,10.6,1.4,15.6l-17.7,3.1c1.1,6.1,2.6,11.9,4.6,17.6 c0.6,0,1.2,0,1.8,0l15.3-5.6c1.8,5,4,9.7,6.6,14.2l-15.6,9c3.1,5.3,6.6,10.3,10.4,14.9c0.6-0.2,1.1-0.4,1.7-0.6l12.5-10.5 c3.4,4,7.1,7.7,11.1,11.1l-11.6,13.8c4.7,4,9.7,7.5,14.8,10.5c0.4-0.4,0.9-0.8,1.4-1.1L423,330c4.5,2.6,9.3,4.8,14.2,6.6l-6.1,16.9 c5.8,2.1,11.7,3.7,17.5,4.8c0.3-0.5,0.6-1.1,0.9-1.6l2.8-16.1c5.1,0.9,10.3,1.4,15.6,1.4v18c6.2,0,12.2-0.5,18.1-1.5 c0.1,0.6,0.2,1.2,0.4,1.8l-2.8-16.1c5.2-0.9,10.3-2.3,15.2-4.1l6.1,16.9c5.8-2.1,11.3-4.7,16.5-7.6c-0.1-0.6-0.2-1.2-0.3-1.8 L513,330c4.5-2.6,8.8-5.7,12.8-9l11.6,13.8c4.7-4,9-8.3,12.9-12.8c-0.3-0.5-0.6-1-0.9-1.6l-12.5-10.5c3.4-4,6.4-8.3,9-12.8l15.6,9 c3.1-5.3,5.7-10.8,7.7-16.4c-0.5-0.4-0.9-0.8-1.4-1.2l-15.3-5.6c1.8-4.9,3.1-9.9,4.1-15.2L574.4,270.8z"/></svg>

SVG Path

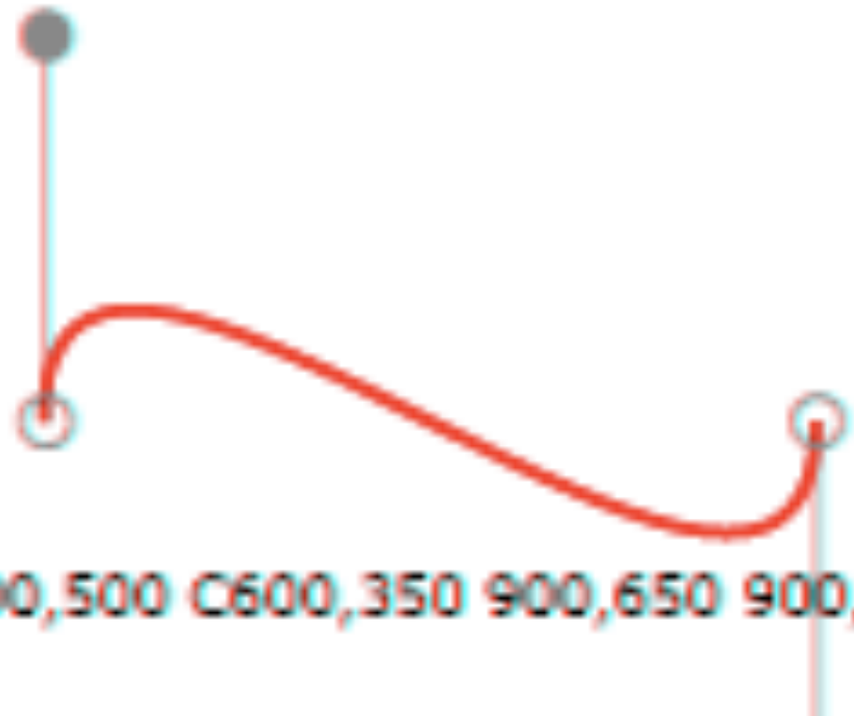
```
<path
  fill="red"
  stroke="black"
  stroke-width="2"
  d="M10 10L20 20"
/>
```

SVG Path

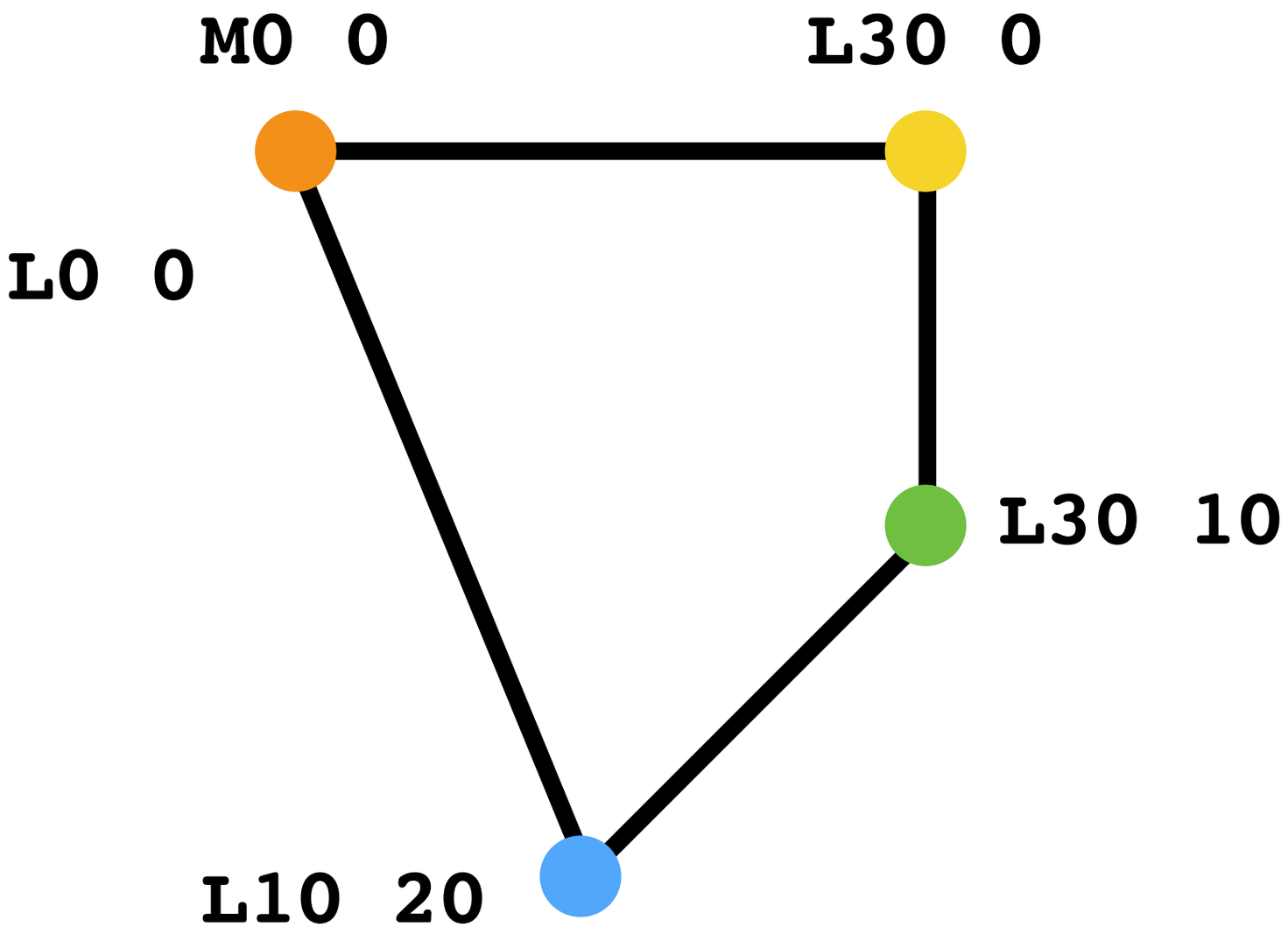
M100,200 C100,100 400,100 400,200

M600,200 C675,100 975,100 900,200

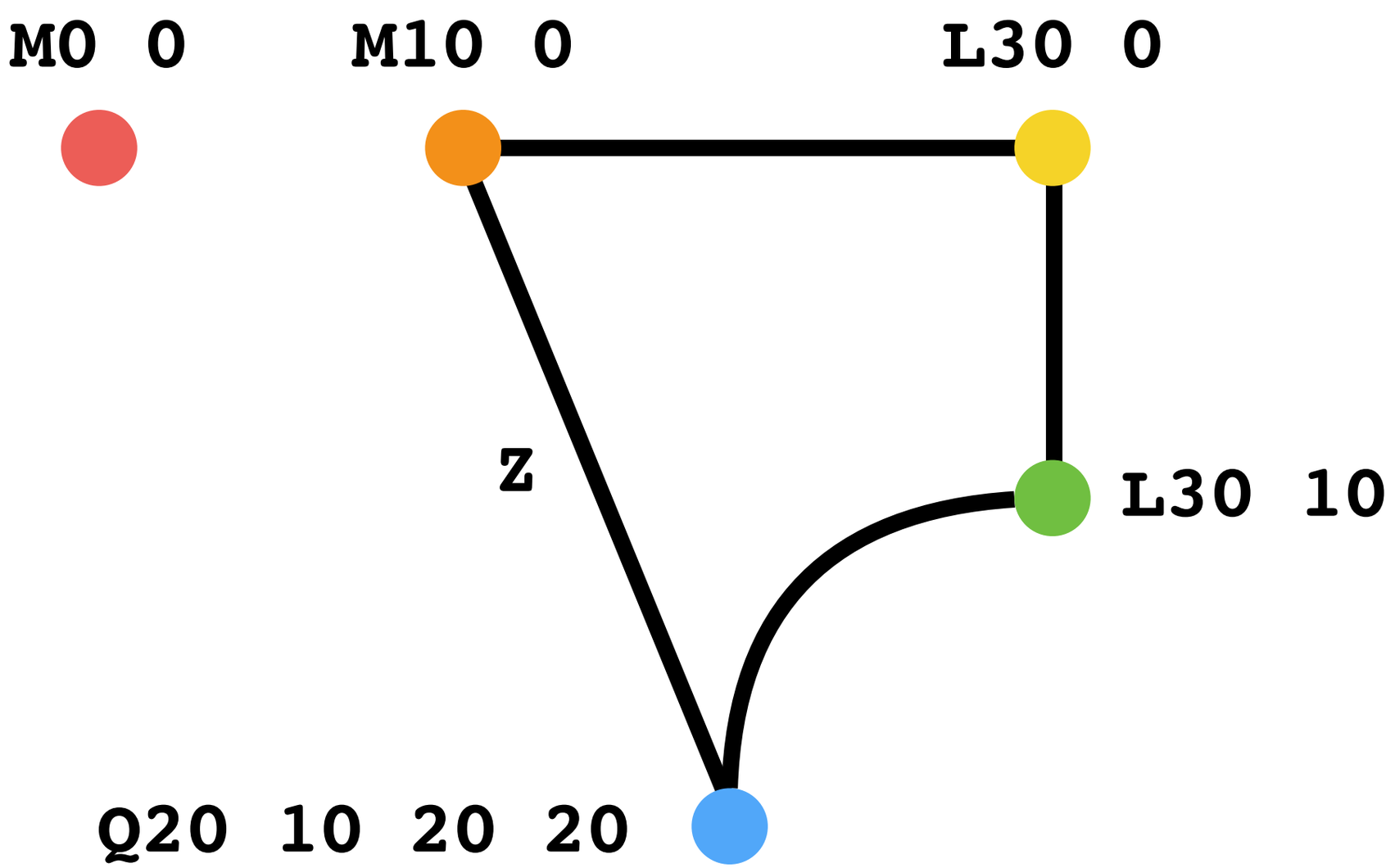
M100,500 C25,400 475,400 400,500



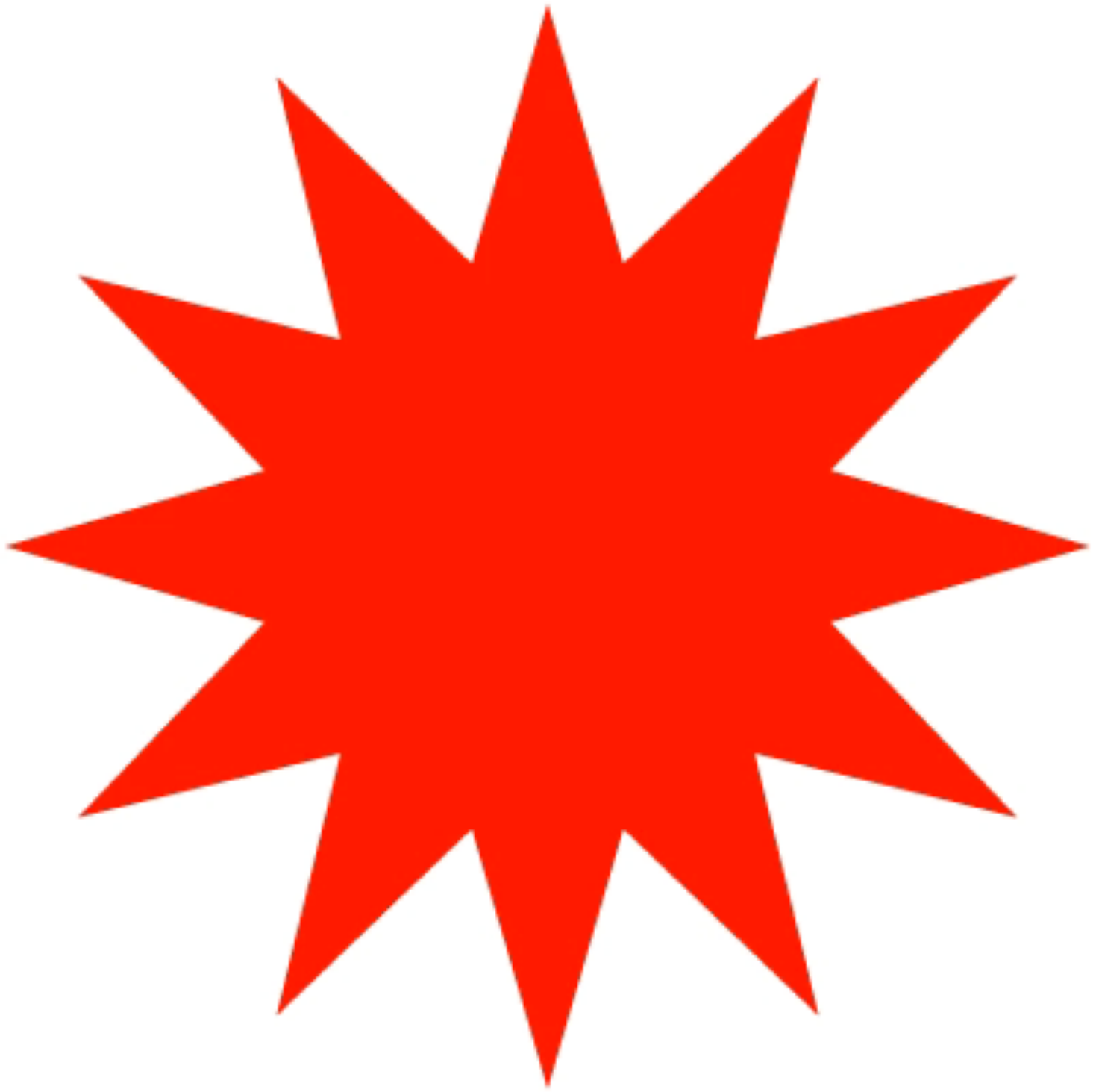
M600,500 C600,350 900,650 900,500




```
<path d="M0 0 L30 0 L 30 10 L10 20 L0 0"/>
```



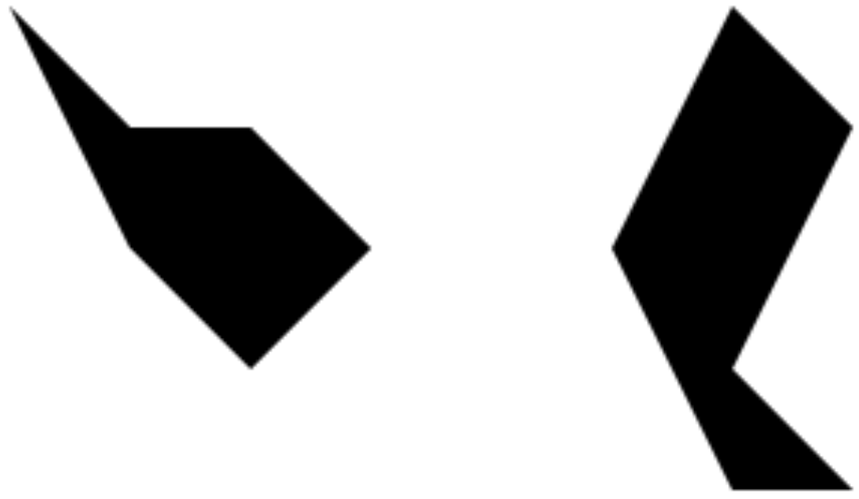
M0 0 M10 0 L30 0 L 30 10 Q20 10 20 20 Z



利用 SVG Path 隨便畫兩個不一樣的六邊形

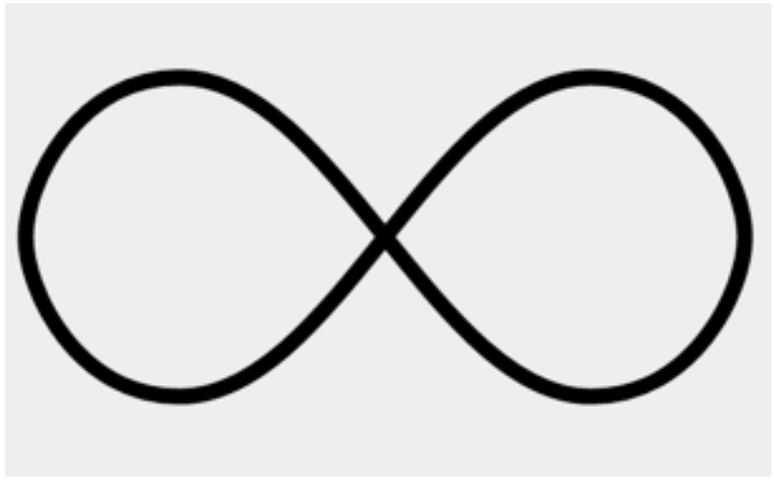
11.html
11-2.html

基本題



加分題

製作下面這個圖形



```
<svg>
  <path d="…"/>
</svg>
<script>
d3.select("path")
  .transition()
  .attrs({
    d: "……"
  });
</script>
```

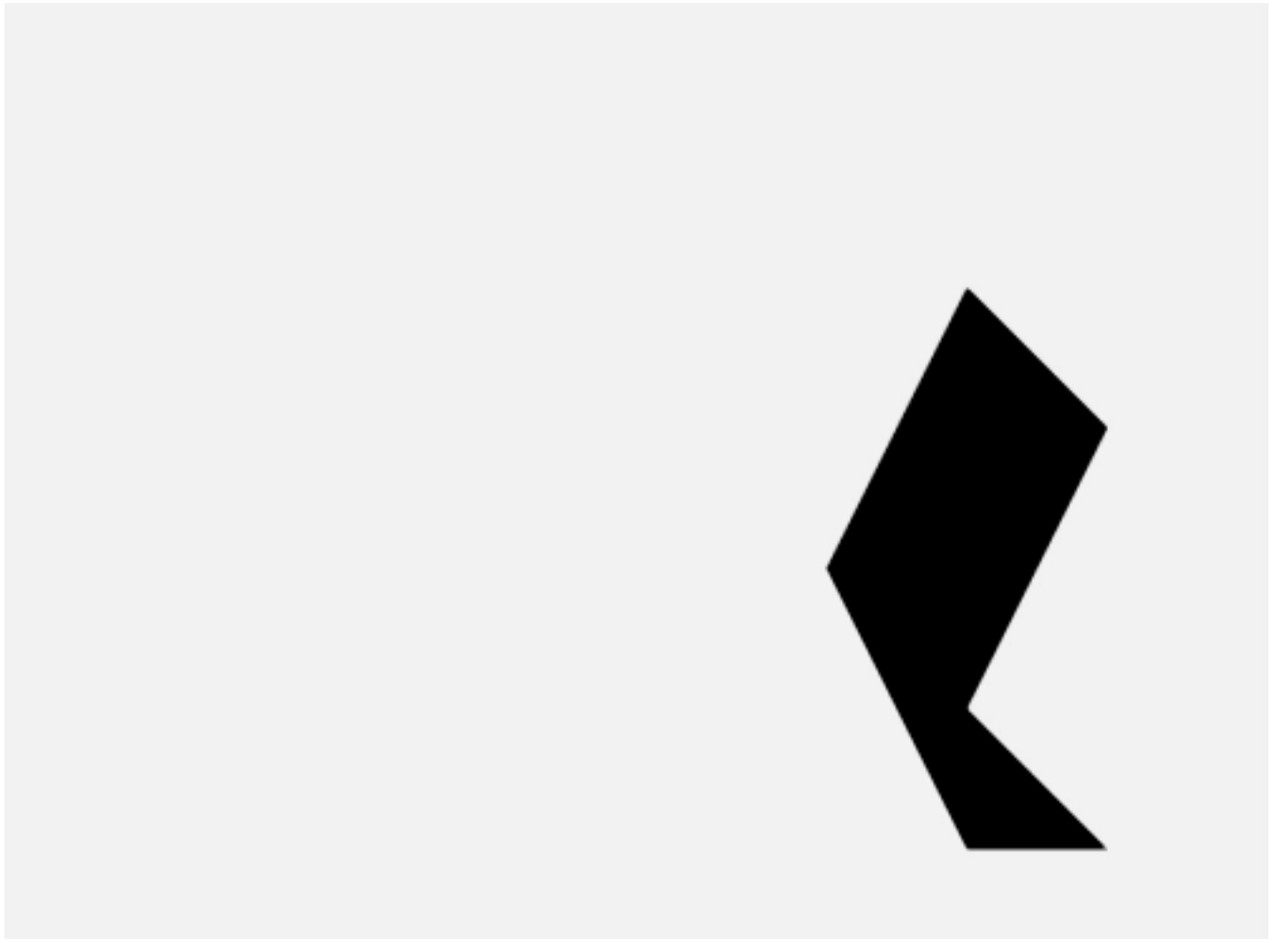
第一個形狀

變形到...

第二個形狀

12.html

將上一個練習中的第一個六邊形
利用 D3.js 變形到第二個六邊形

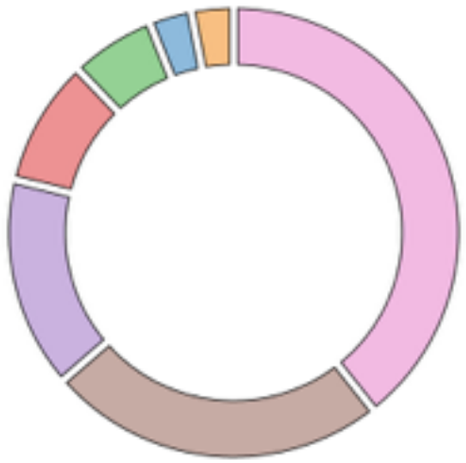


加分題
星星變太陽無限循環

複雜形狀不用自己畫

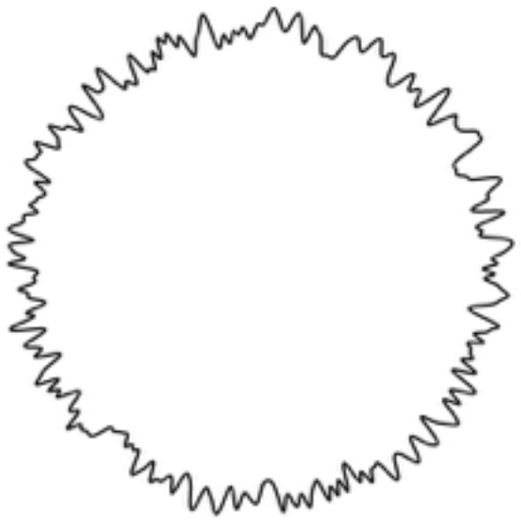
就使用 d3-shape 吧

d3.arc

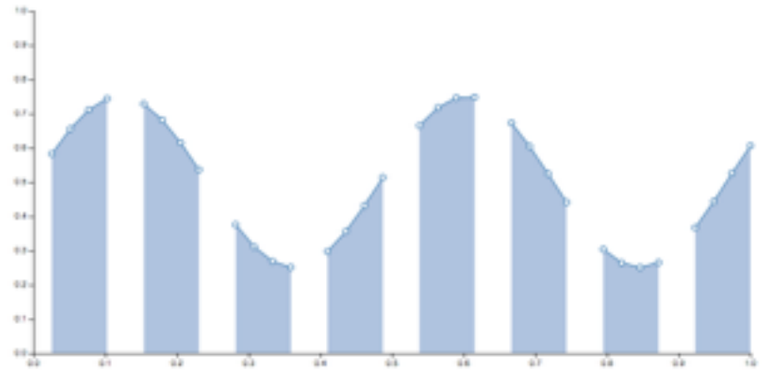


d3.line

d3.radialLine



d3.area



以 d3.area 為例

設定區域圖
的座標

先建立
Area Builder

```
builder = d3.area()
    .x(……)
    .y0(……)
    .y1(……);
```

當成函式
呼叫 Builder

```
builder([3,1,4,1,5,9]);
```

```
builder = d3.area()
   .x(function(d,i) { return i; })
   .y0(function(d,i) { return 0;})
   .y1(function(d,i) { return d; });

builder([3,1,4,1,5,9]);
```

idx	value
0	3
1	1
2	4

x	y0	y1
0	0	3
1	0	1
2	0	4

```
builder = d3.area()
    .x(……)
    .y0(……)
    .y1(……);

d3.select("svg").append("path")
    .attrs({
      d: builder([3,1,4,1,5,9])
    });
```

```
builder = d3.area()
    .x(……)
    .y0(……)
    .y1(……);

d3.select("svg").append("path")
    .datum([3,1,4,1,5,9])
    .attrs({
      d: builder
    });
```

```
d3.select("svg")
    .append("path")
    .datum([3,1,4,1,5,9])
    .attrs({ d: builder })
    .datum([9,5,1,4,1,3])
    .transition()
    .attrs({ d: builder });
```

```
builder = d3.line()
    .x(……)
    .y(……);
```

```
builder = d3.radialLine()
    .angle(……)
    .radius(……);
```

製作動態區域圖，從 [3,1,4,1,5,9] 變到 [9,5,1,4,1,3]

13.html
13-2.html

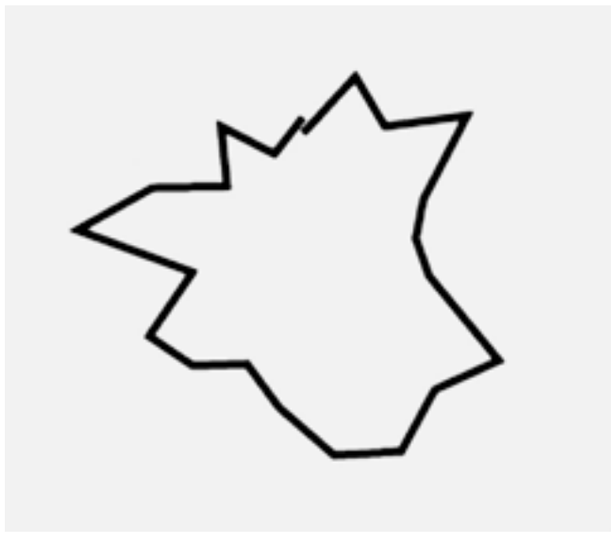
基本題

加分題1

讓動畫不斷重覆

加分題2

利用圓周率製作圓弧曲線



無限重覆的動畫？

```
d3.selectAll(“rect”)
    .transition().duration(1000)
        .attrs({fill: “#f00”})
   .transition().duration(1000)
        .attrs({fill: “#0f0”})
   .transition().duration(1000)
        .attrs({fill: “#0f0”})

    ……

.transition().duration(1000)
        .attrs({fill: “#0f0”})
.transition().duration(1000)
        .attrs({fill: “#0f0”})
```

使用迴圈？

```
var node = d3.select("svg")
                .selectAll("rect");

for(var i = 1; i < 100; i++) {
  node.transition()
      .delay(i * 1000)
      .attrs(…);
}
```

使用 Timer?

```
var node = d3.select("svg")
            .selectAll("rect");

setInterval(function() {
  node.transition().attrs(…);
}, 1000);
```

使用 RequestAnimationFrame ?

```
function tick(time) {
  if(!start || time - start > 1000) {
    node.transition()……;
    start = time;
  }
  requestAnimationFrame(tick);
}
requestAnimationFrame(tick);
```

利用動畫事件

start / end /interrupt


```
d3.select("svg")
  .selectAll("rect")
  .transition().duration(1000)
  .attrs(…)
  .on("end", function(d,i) {

  });
```

利用動畫事件

start / end /interrupt

```
d3.select("svg")
   .selectAll("rect")
  .transition().duration(1000)
.on("start", function repeat() {
    d3.active(this)
       .transition()
       .attrs(……)
       .on("end", repeat);
 });
```

利用動畫事件

start / end /interrupt

```
repeat = function() {
    var t = d3.transition()
    .on("end", repeat);
    selection.transition(t)……
};
```

利用動畫事件

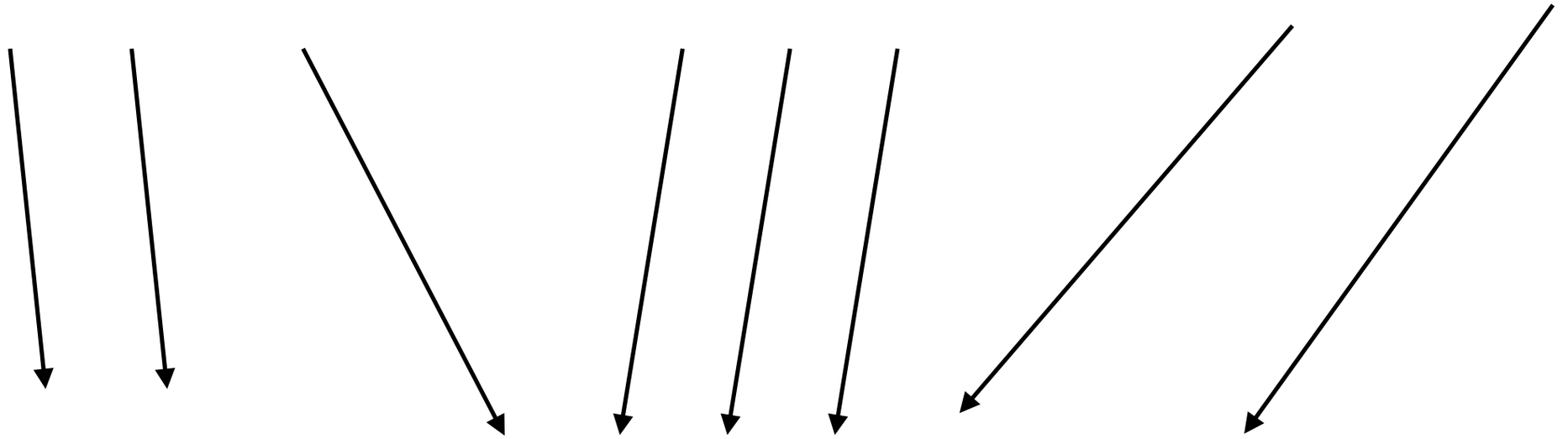
start / end /interrupt

```
repeat = function() {
  var t = d3.transition()
  .tween("tween", function() {
    ……
  })
  .on("end", repeat);
};
```

Customize Tweening

```
transition
    .attrTween(name, func)
    .styleTween(name, func)
    .tween(name , func)
```

1鄉2里共3夫子，不識4書5經6義，竟敢教789子，10分大膽



10室9貧，湊得8兩7錢6分5毫4厘，尚且3心2意，1等下流

```
d3.select("body")
  .transition().duration(2000)
  .tween("tanbohu",function() {
    return function(t) {
      d3.select(this).text(
        d3.interpolate(
          "1鄉2里共3夫子，不識4書5經6義，竟敢教789子，10分大膽",
          "10室9貧，湊得8兩7錢6分5毫4厘，尚且3心2意，1等下流"
        )(t)
      );
    };
  });
```

動畫的原理

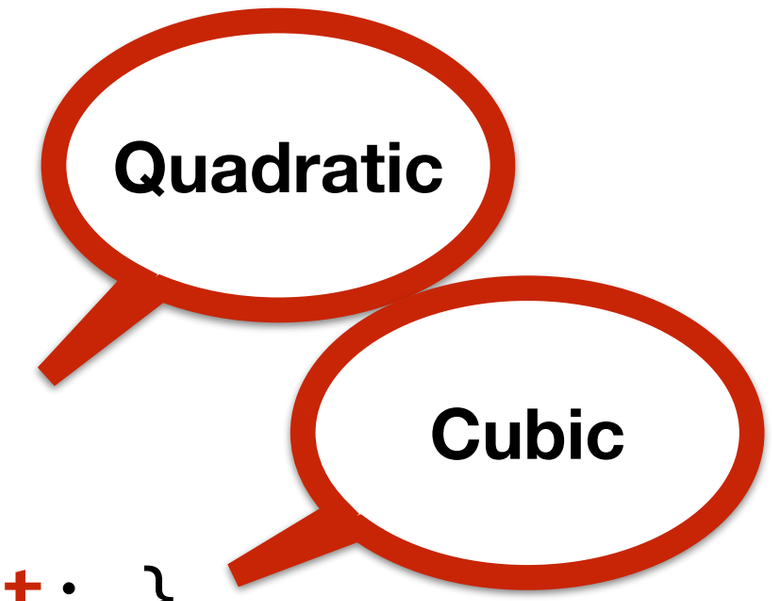
初始值 A

目標值 B

內插函式 i

時間函式 t

```
function time(t) {
    return t;
}
```




```
function time(t) { return t * t; }

function time(t) { return t * t * t; }

function time(t) { return Math.sin(t * Math.PI/2); }
```

easeInSine

easeOutSine

easeInOutSine

easeInQuad

easeOutQuad

easeInOutQuad

easeInCubic

easeOutCubic

easeInOutCubic

easeInQuart

easeOutQuart

easeInOutQuart

easeInQuint

easeOutQuint

easeInOutQuint

easeInExpo

easeOutExpo

easeInOutExpo

easeInCirc

easeOutCirc

easeInOutCirc

easeInBack

easeOutBack

easeInOutBack

easeInElastic

easeOutElastic

easeInOutElastic

easeInBounce

easeOutBounce

easeInOutBounce

```
… .transition()
    .ease(type) …
```

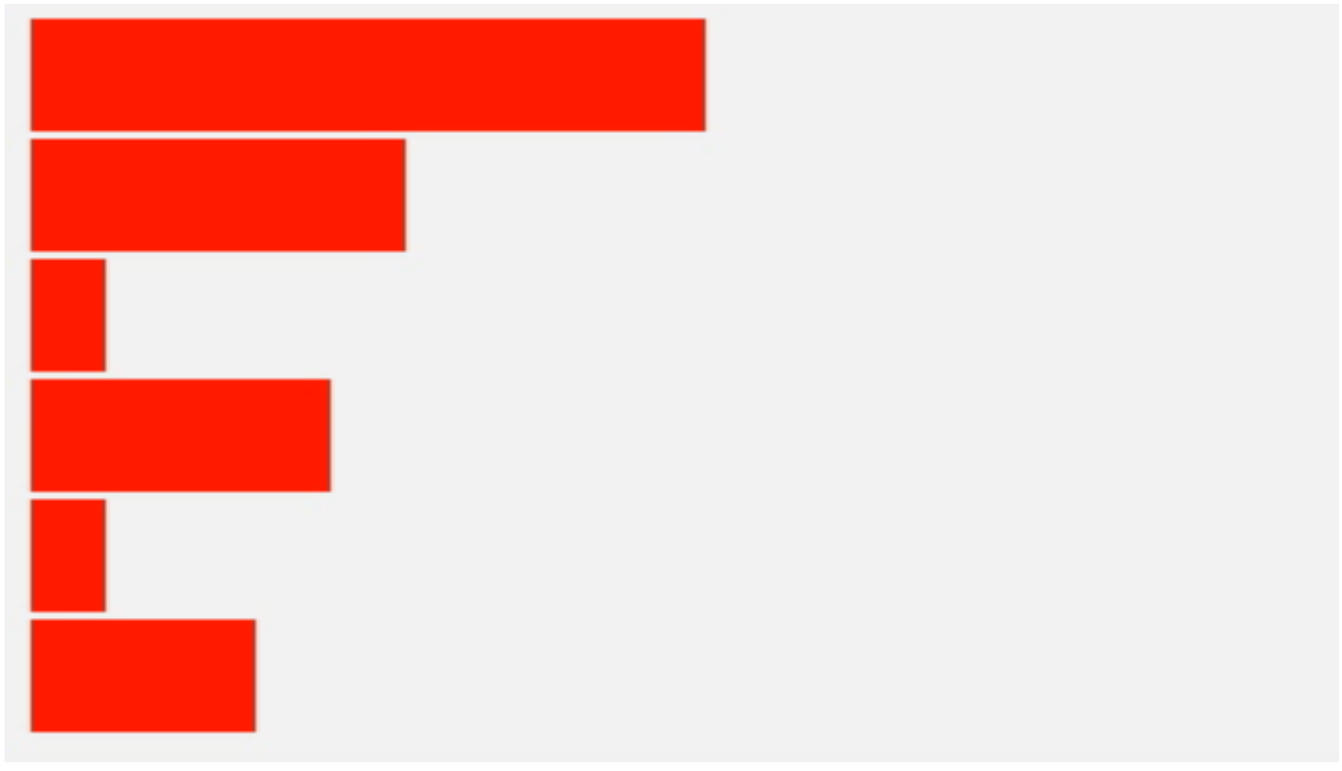
d3-ease

transition.ease(type)

d3.easeBounce	d3.easeBounceOut
d3.easeBack	d3.easeBackOut
d3.easeElastic	d3.easeElasticOut
d3.easeCircle	d3.easeCircleOut
d3.easeSin	d3.easeSinOut

14.html

利用練習 10 中的長條圖動畫，嘗試至少三種不同的 Ease 函式



加分題

你能讓每個長條用的 Ease 函式都不同嗎？

Recap

- 串連動畫： transition().transition()
  - 延遲啟動 - delay()
  - 動畫長度 - duration()
- 平行動畫： transition(“a”); transition(“b”);
- 時間函式：d3.ease
- 客製動畫：d3.tween

資料到座標需要轉換

```
d3.selectAll("rect").attrs({
    x: 10,
    y: function(d,i) {
        return 10 + d * 10;
    },
});
```

資料到座標需要轉換

```
d3.selectAll("rect").attrs({
  x: 10,
  y: function(d,i) { return 10 + d * 10; },
});
```

```
d3.selectAll("text").attrs({
  x: 10,
  y: function(d,i) { return 10 + d * 10; },
});

d3.selectAll("circle").attrs({
  cx: 10,
  cy: function(d,i) { return 10 + d * 10; },
});
```

資料到座標需要轉換

```
yscale = function(d) {
   return 10 + d * 10;
}
```

```
d3.selectAll("rect").attrs({
   x: 10,
   y: function(d,i) {
      return yscale(d);
   },
});
```

資料到座標需要轉換

[3,1,4,1,5,9]

輸入範圍 1 ~ 9

對應範圍 10 ~ 800

轉換方式：

10 + (input - 1) * (800 - 10) / (9 - 1)

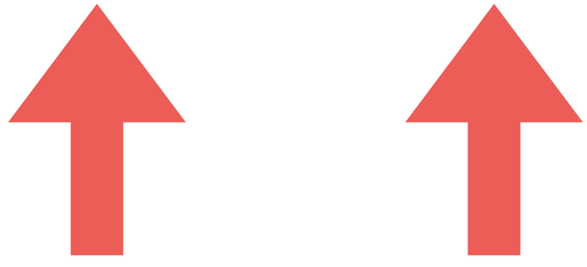
D3 Scales


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

```
scale(0.5);
```

D3 Scales

```
var scale = d3.scaleLinear();
scale
   .domain( [  1,    9] )
   .range(  [ 10, 800] );
```



start end

15.html

利用 Linear Scale 將 1 ~ 9 這個資料範圍

對應到 10 ~ 800 這個螢幕範圍

求資料 4 對應的座標值

(可以用 d3.select("body").text(答案) 的方式取值)

加分題1： 利用 scale.invert() 求螢幕座標 "768" 對應的資料值

加分題2： 若對應到螢幕範圍 800 ~ 100, 答案又是多少呢？

17.html

利用 Linear Scale 製作 Radial Bar Chart

1. 弧狀長條的長度對應到數值大小
2. 資料為 [3,1,4,1,5,9,2,6]
3. 由內往外排列

基本題

加分題

利用 LinearScale
將數值對應到顏色並為長條上色

```
arc = d3.arc();
arc({
  startAngle: 0,   // 角度起點
  endAngle: ?,     // 角度終點
  innerRadius: ?,  // 弧內半徑
  outerRadius: ?   // 弧外半徑
});
```

```
arc = d3.arc();
d3.select("path").attrs({
  d: arc({
      startAngle: 0,    // 角度起點
      endAngle: ?,      // 角度終點
      innerRadius: ?,   // 弧內半徑
      outerRadius: ?    // 弧外半徑
  });
});
```

```
arc = d3.arc();
d3.select("path").attrs({
  d: function(d,i) {
    arc({
      startAngle: 0,
      endAngle: d * Math.PI * 2 / 9,
      innerRadius: i * 10,
      outerRadius: (i + 1) * 10
    });
  }
});
```

17.html

利用 Linear Scale 製作 Radial Bar Chart

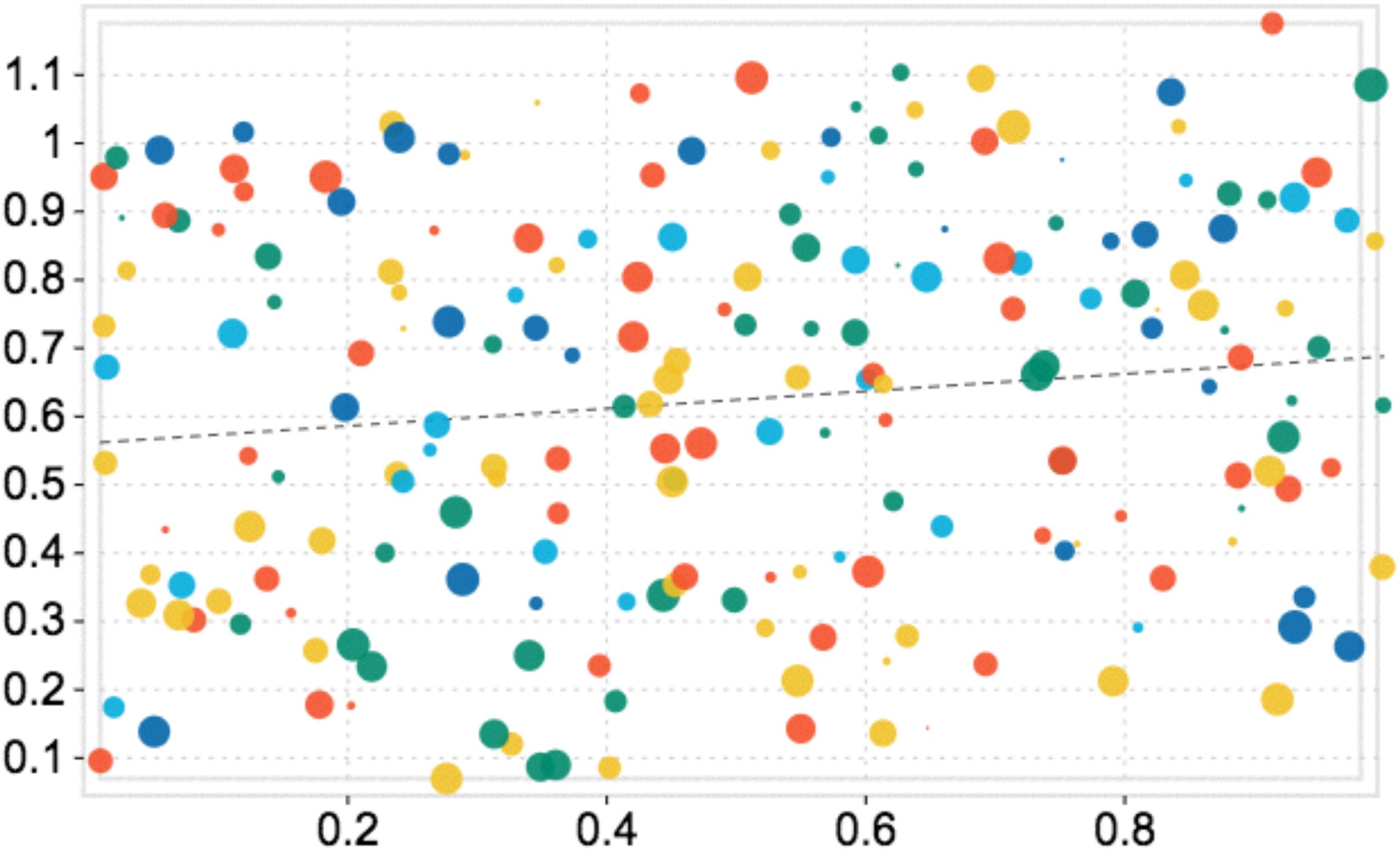
1. 弧狀長條的長度對應到數值大小
2. 資料為 [3,1,4,1,5,9,2,6]
3. 由內往外排列

基本題

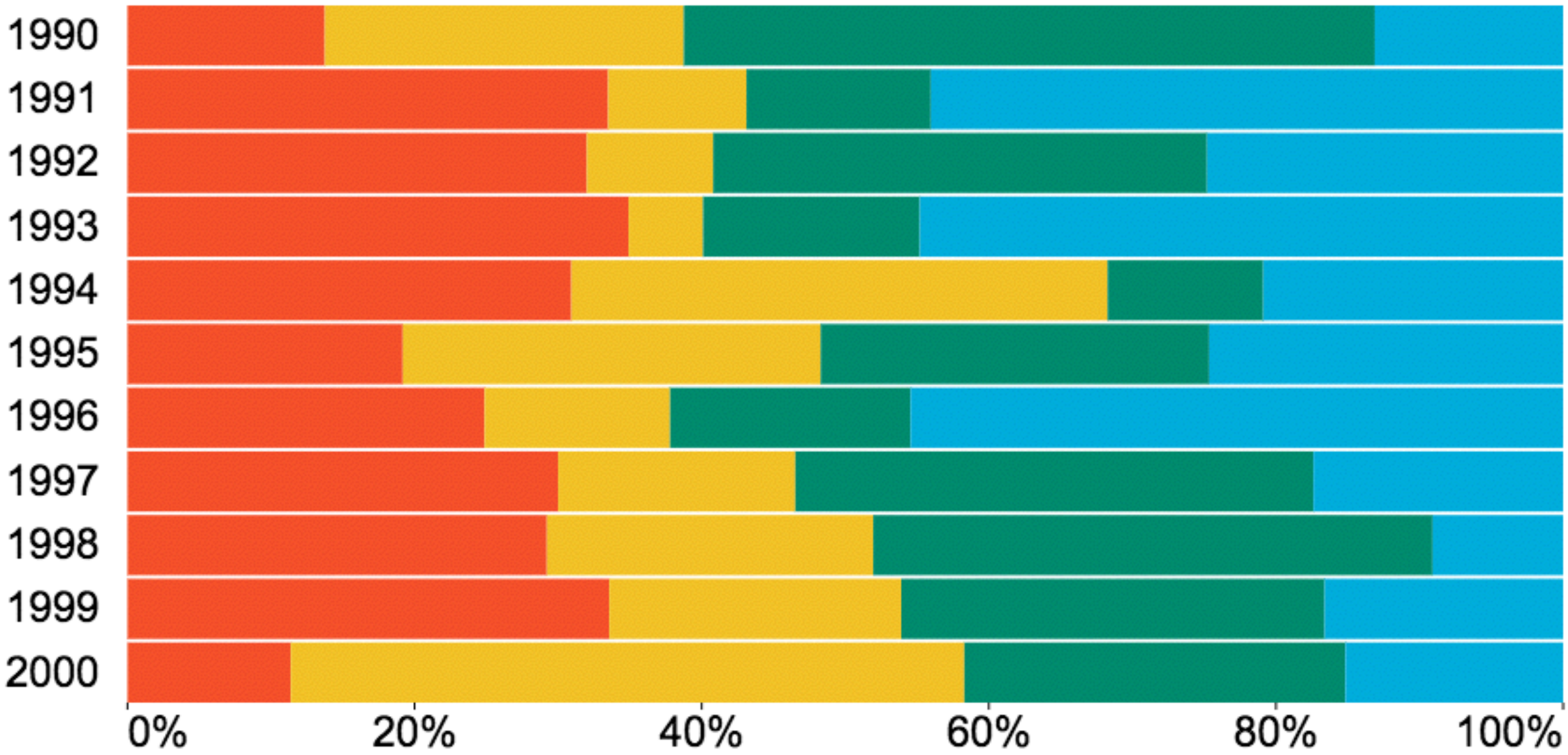
加分題

利用 LinearScale
將數值對應到顏色並為長條上色


輸入
scale
輸出

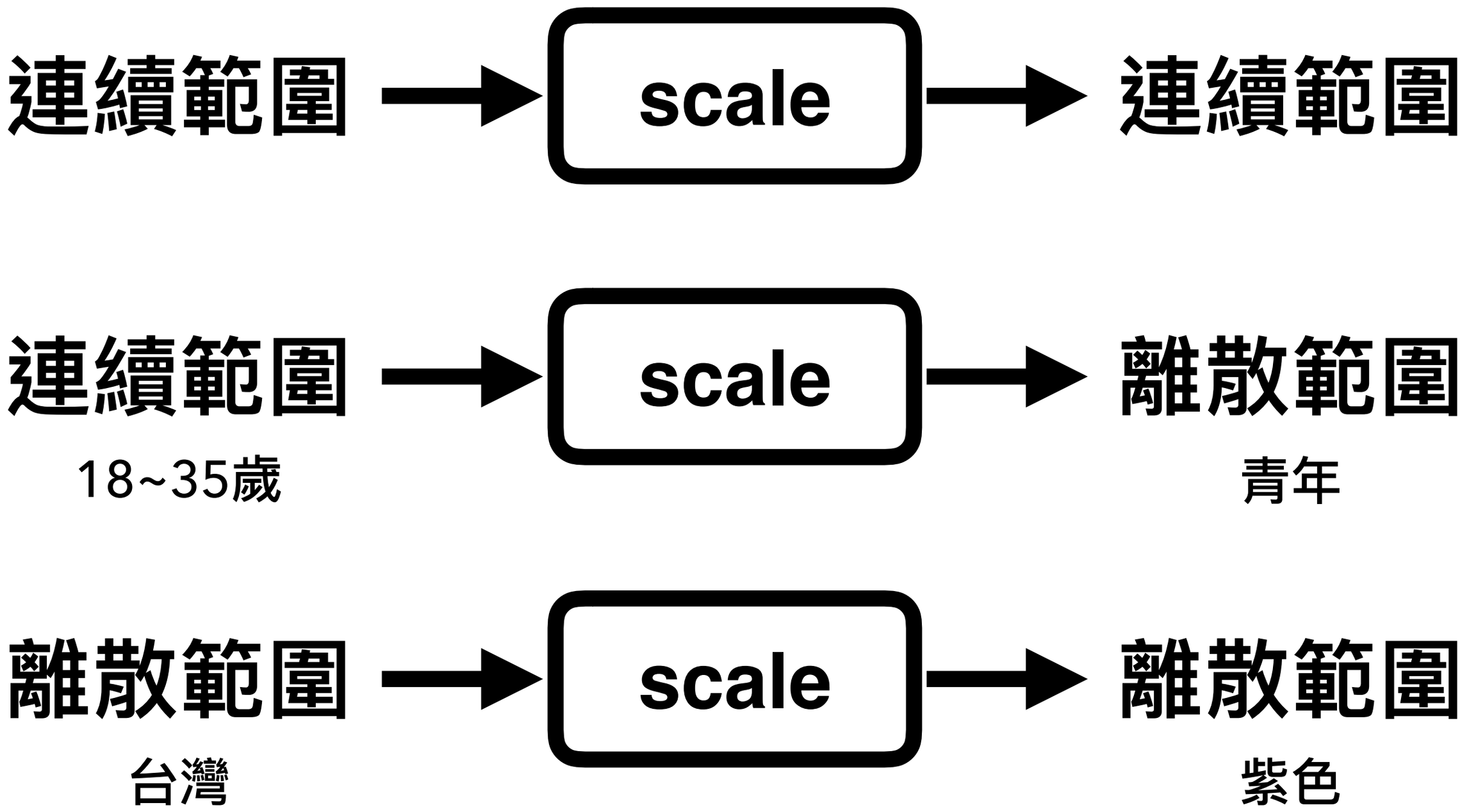


(0 ~ 1) → scale → (30~768)



年度 → scale → Y 軸座標

國家? → scale → 顏色?


連續範圍
scale
連續範圍
連續範圍
18~35歲
scale
青年
離散範圍
台灣
scale
離散範圍
紫色

D3 Scales

V4

輸入連續, 輸出連續

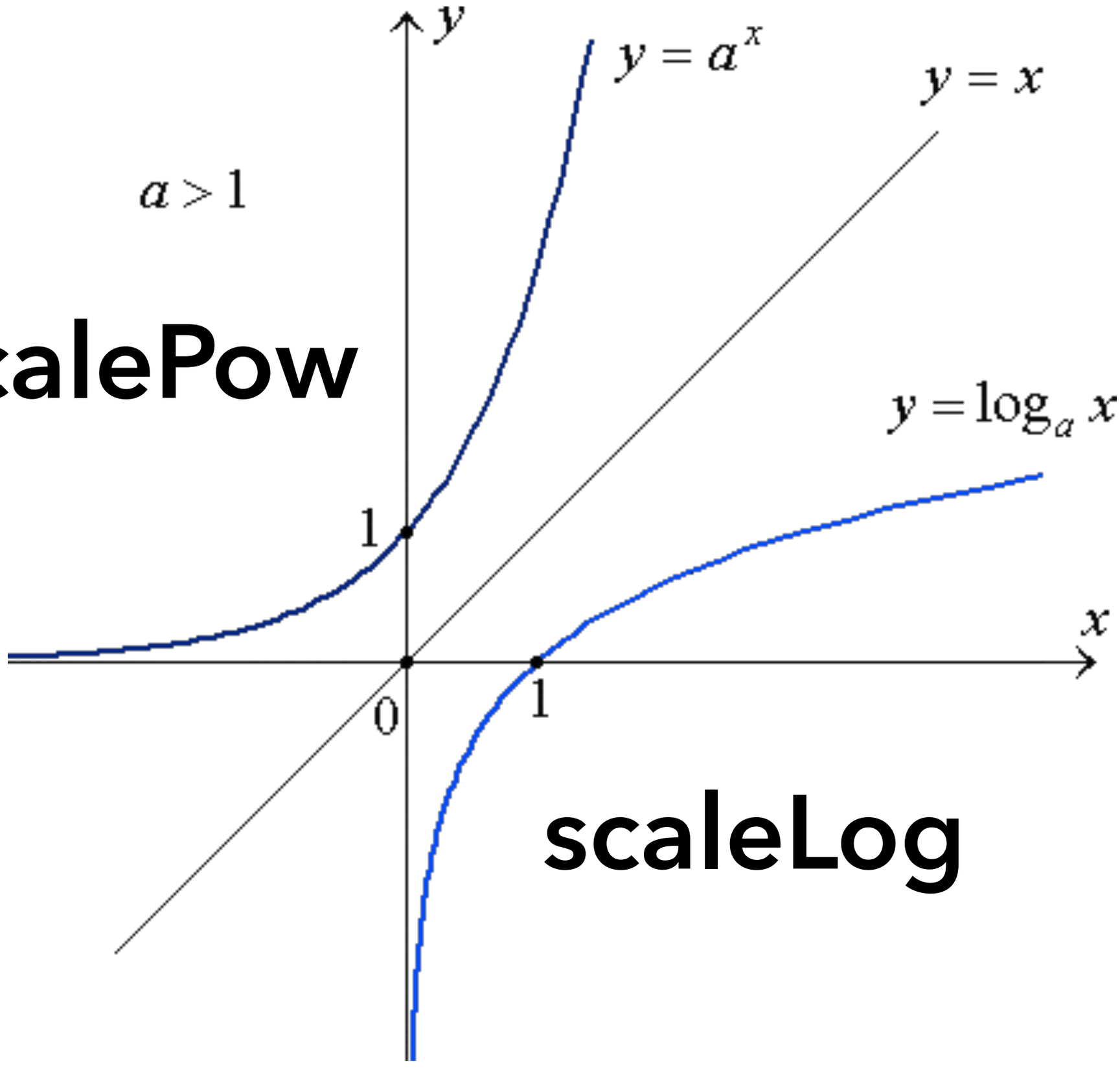
Continuous **Sequential**

輸入連續, 輸出離散

Quantize **Quantile** **Threshold**

輸入離散, 輸出離散

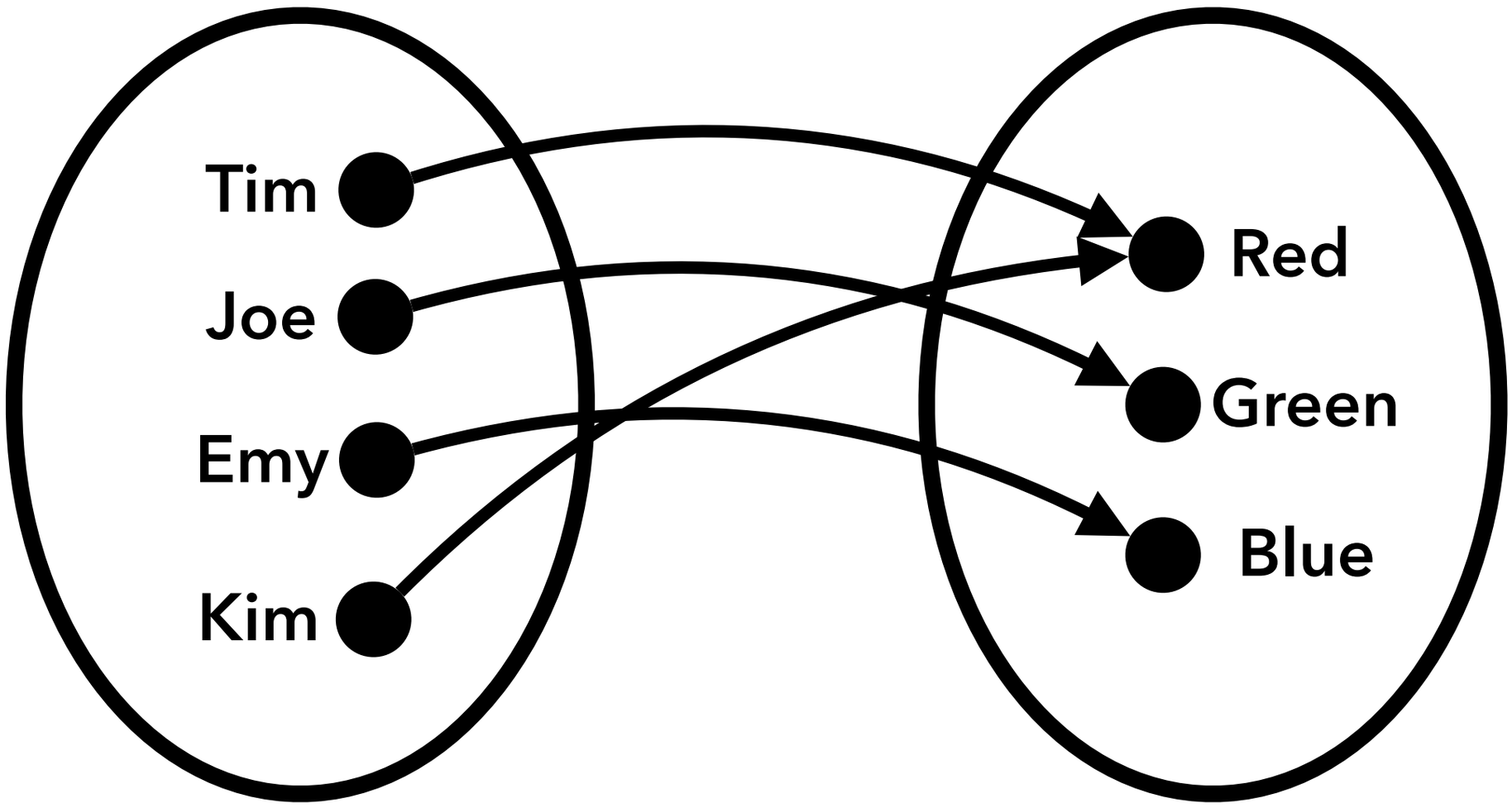
Ordinal



scalePow

scaleLog

d3.scaleOrdinal



顏色轉換： 離散 → 離散

```
var scale = d3.scaleOrdinal()
   .range([
      "#8334ad","#2b488c","#2b82a1",
      "#3fab4b","#7fc054","#edc95e",
      "#e48e11","#df3316","#8d2725"
   ]);

scale("Joe");
```

若未定義 Domain
D3.js 將自動分配

d3.scaleOrdinal(

  d3.schemeCategory20

)

d3.schemeCategory10 <>

d3.schemeCategory20 <>

d3.schemeCategory20b <>

d3.schemeCategory20c <>

顏色轉換： 連續 → 連續

利用預先定義好的顏色內插函式

```
d3.scaleSequential(
    d3.interpolateInferno,
).domain([10,800]);
```

---

d3.interpolateViridis

d3.interpolateInferno

d3.interpolateWarm

d3.interpolateCool

d3.interpolateRainbow

d3.interpolatePlasma

顏色轉換： 連續 → 離散


```
var colors = d3.quantize(
   d3.interpolateInferno,
   10
);

d3.scaleQuantize()
   .domain([0,1])
   .range(colors);
```

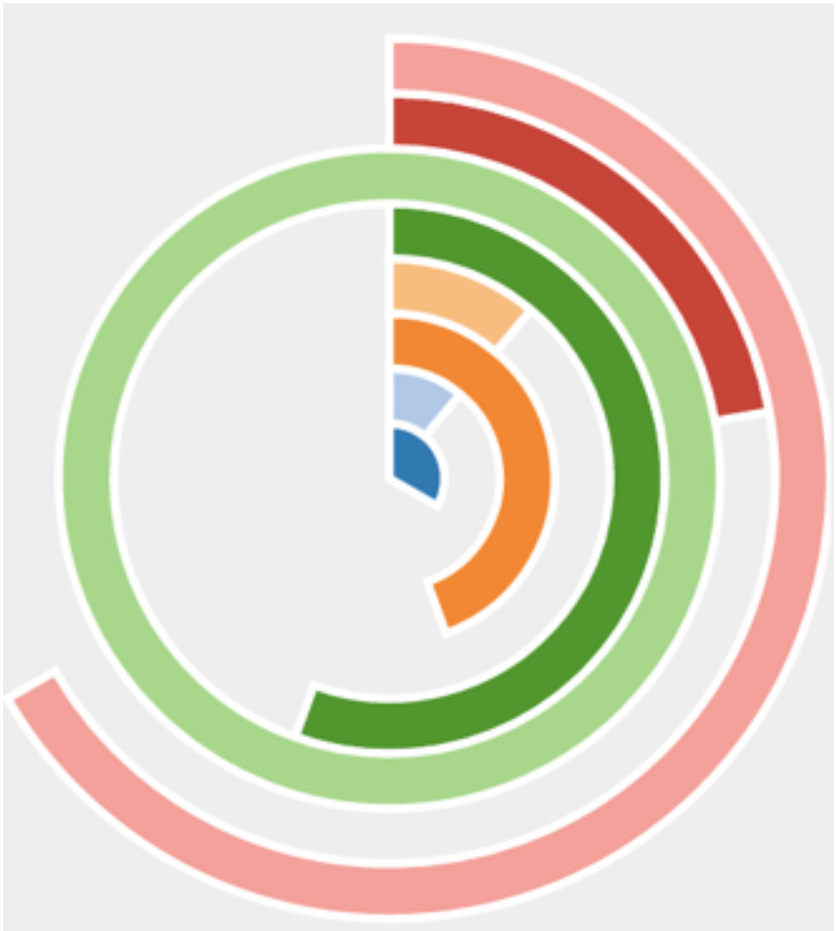
把 interpolateInferno
切成 10 塊

然後把 0 ~ 1
間的數字平均分入這
10 個顏色

將練習 17 中的長條利用 Category20 上色

18.html

基本題



加分題1

利用 interpolateInferno
將數值對應到連續的色階

加分題2

利用 interpolateInferno
將數值對應成四種顏色

有用的知識

d3.scale 是靠 d3.interpolate 算的

d3.scale.linear().domain([0,1])

.range([“#f00”, “#0f0”])

d3.scale.linear().domain([0,1])

.range([[0,0], [1,1]])

有用的知識 #2

domain 與 range 接受兩個以上的數值

但兩者的數量須一致

```
d3.scale.linear()
   .domain([-1,0,1])
  .range([“red”, “white”, “green”])
```

有用的知識 #3

利用 scale.invert(value) 進行反函式運算

- 處理滑鼠事件時特別好用
- ordinal scale 無法使用

[3,1,4,1,5,9,2,6,5,3,5,8,9,7,9 …]

```
xscale = d3.scaleLinear()
    .domain([1, 9])
    .range([10, 800]);
```

placeholder?

```
[1,45,132,200, 4943, … ]
```

```
xscale = d3.scaleLinear()
   .domain([???])
   .range([10, 790]);
```

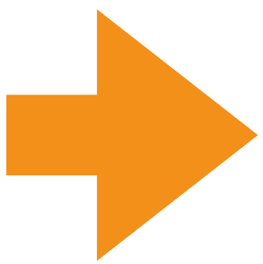
d3.extent(Array)

```
array = [3,1,4,1,5,9];
```

```
d3.extent(array)
```



```
[1,9]
```



```
[d3.min(array), d3.max(array)]
```

d3.extent(Array,accessor)

```
array = [
   {value: 3},
   {value: 1},
   {value: 4},
   ....
]

d3.extent(array,function(it) {
   return it.value
})
```

d3.range(長度)

d3.range(10) → [0,1,2,3,4,5,6,7,8,9]

```
d3.range(10).map(function(it) {
   return {
     value: Math.random()
   };
});
```

利用 d3.range 產生如下的隨機測試資料

```
[
    {value: 1, category: "A"},
    {value: 3, category: "A"},
    {value: 9, category: "B"},
    {value: 4, category: "B"},
    {value: 8, category: "C"},
]
```

長度不拘

1 ~ 100

分成三類

加分題: 利用 d3.extent 確認資料中 value 欄位的範圍是否是 [1, 100]

```
value range: [1,5]
```

想知道
資料的範圍

```
categories: ["A","B","C"]
```

```
each category:
"A": [1],
"B": [2,3],
"C": [4,5]
```

想知道有
哪幾類

想知道各類
有哪些資料

```
[
   {value: 1, category: "A"},
   {value: 2, category: "B"},
   {value: 3, category: "B"},
   {value: 4, category: "C"},
   {value: 5, category: "C"},
]
```

d3.map(data, accessor)

build map from array

```
d3.map([
   {cat: "A", value: 1},
   {cat: "B", value: 2},
   {cat: "C", value: 3}
], function(d) {
   return d.cat;
});
```

d3.map(data, accessor)

build map from array

```
{
   "A": {cat: "A", value: 1},
   "B": {cat: "B", value: 2},
   "C": {cat: "C", value: 3}
}
```

`map.get("C")` = {cat:"C", value: 3}

`map.keys()` = ["A","B","C"]

d3.nest()

group array into hierarchy structure

```
.key()                  - 取 key 函式
.map(Array)             - 轉成 Map
.object(Array)          - 轉成 Object
.entries(Array)         - 轉成 Array
```

```
a = [
   {value: 1, cat: "A"},
   {value: 2, cat: "B"},
   {value: 3, cat: "B"},
   {value: 4, cat: "C"},
   {value: 5, cat: "C"},
]

d3.nest()
   .key(function(it) {
      return it.cat; })
   .entries(a)
```

建立 nest 物件

從 cat 取 key

把 a 轉成 key/value 陣列

```
a = [
  {value: 1, cat: "A"},
  {value: 2, cat: "B"},
  {value: 3, cat: "B"},
  {value: 4, cat: "C"},
  {value: 5, cat: "C"},
]
```

```
converted = [
  {key: “A”, values: [Obj]},
  {key: “B”, values: [Obj,Obj]},
  {key: “C”, values: [Obj,Obj]},
]
```

```
a = [
  {key: "A", values: [
    {value: 1, cat: "A"}
  ]},
  {key: "B", values: [
    {value: 2, cat: "B"},
    {value: 3, cat: "B"},
  ]},
  {key: "C", values: [
    {value: 4, cat: "C"},
    {value: 5, cat: "C"},
  ]}
]
```

```
d3.nest()
  .key(function(it) {
    return it.cat; })
  .entries(a)
```

建立 nest 物件

從 cat 取 ke

V4

把 a 轉成 key/value 陣列

```
d3.nest()
  .key(function(it) {
    return it.cat; })
  .map(a)
```

建立 nest 物件

從 cat 取 key

把 a 轉成「MAP」

```
a = [
   {value: 1, cat: "A"},
   {value: 2, cat: "B"},
   {value: 3, cat: "B"},
   {value: 4, cat: "C"},
   {value: 5, cat: "C"},
]
```

```
converted = {
   “A”: [Obj]},
   “B”: [Obj,Obj]},
   “C”: [Obj,Obj]},
}
```

```
d3.nest()
    .key(function(it) {
      return it.cat; })
    .map(a).keys()
```

```
[ "A", "B", "C" ]
```

20.html

利用 練習19 的資料與 d3.nest, 確認 category 的種類

```
[ {value: 12, category: "A"},
  {value: 31, category: "A"},
  {value: 92, category: "B"},
  {value: 45, category: "B"},
{value: 83, category: "C"}]
```

↓

d3.nest()

↓

["A", "B", "C"]

加分題

確認 value 中、
十位數的數值有多少種

```
[ {value: 12, category: "A"},
  ……
  {value: 83, category: "C"}]

d3.nest()
  .key(function(it) {
     return Math.floor(it/10);
  }).map(A);

converted = {
  "4": [{value: 46,…}, …],
  "7": [{value: 76,…}, …],
  "8": [{value: 85,…}, …]
}
```

```
d3.nest()
 .key(function(it) {
    return parseInt(it/100);
 }).key(function(it) {
    return parseInt(it/10)%10;
 }).map([123]);


converted = {
  "1": [{"2": [123]}]
}
```

d3.map – es6 style map

d3.nest – 巢狀物件

d3.pairs – 陣列配對

d3.quantile – 分位數內插

d3.transpose – 轉置

d3.zip – 行列配對

d3.permute – 指定排列方式

陣列 [1,2,2,3,3,3]

雜湊
```
{ 1: Obj,
  2: Obj,
  3: Obj}
```

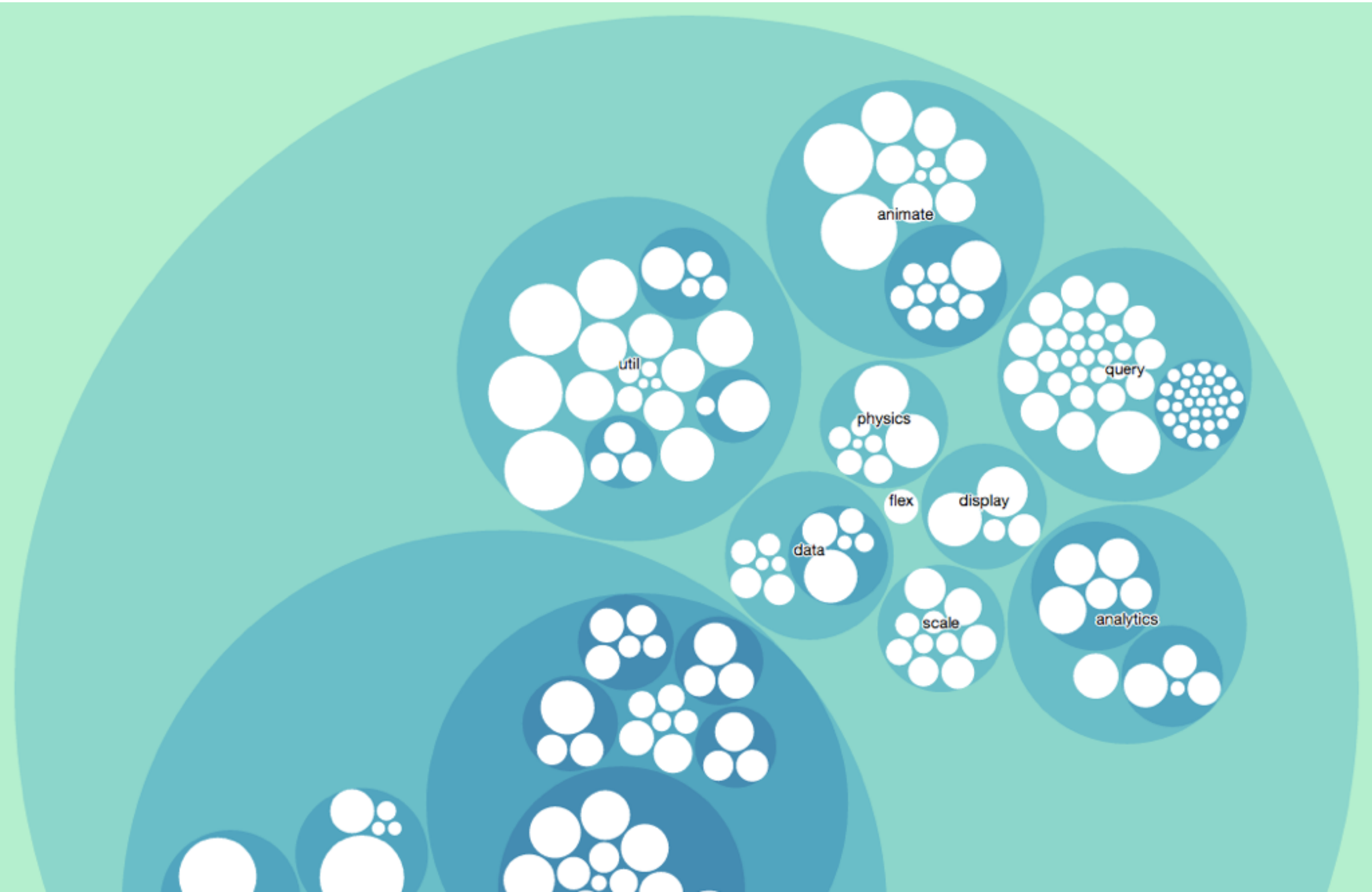
階層陣列
```
[{key:1, values: […]},
 {key:2, values: […]
 {key:3, values: […]
]
```

雜湊陣列
```
{ 1: [Obj],
  2: [Obj, Obj],
  3: [Obj, Obj, Obj]}
```

```
[[1,2],
 [2,2],
 [2,3],
 [3,3],
 [3,3]]
```

配對

Circular Treemap



d3.pack

```
var pack =
    d3.pack()
        .size([width, height]);
pack(data);
```

```
values: [
  {…},
  {
    cat: “A”,
    value: 0.447
    x: 248.867
    y: 414.396
    r: 31.373
    depth: 2
    parent: {…}
  }, {…}, …
]
```

d3.hierarchy

Data Structure for Pack Layout

V4

```
{
   children: [
      {…}, {…}, …
   ],
   value: (size)
}
```

隨機產生資料 (練習 19)

```
var data = d3.range(20).map(function(d,i) {
  return {
    value: Math.round(Math.random() * 100),
    category: ["A","B","C"][Math.floor(Math.random() * 3)]
  };
});
```

運用 d3.nest() (練習 20)

```
var ret = d3.nest()
  .key(function(d,i) { return d.category; })
  .entries(data);
```

運用 d3.nest()(練習 20)

```
var ret = d3.nest()
  .key(function(d,i) { return d.category; })
  .entries(data);
```

```
converted = [
  {key: “A”, values: [Obj]},
  {key: “B”, values: [Obj,Obj]},
  {key: “C”, values: [Obj,Obj]},
]
```

建構完整樹狀結構

```
{ values: converted }
```

```
{
   values: [
      {key: “A”, values: [Obj]},
      {key: “B”, values: [Obj,Obj]},
      {key: “C”, values: [Obj,Obj]},
   ]
}
```

隨機產生資料 (練習 19)

```
var data = d3.range(20).map(function(d,i) {
  return {
    value: Math.round(Math.random() * 100),
    category: ["A","B","C"][Math.floor(Math.random() * 3)]
  };
});
```

運用 d3.nest() (練習 20)

```
var ret = d3.nest()
  .key(function(d,i) { return d.category; })
  .entries(data);
```

建構樹狀結構

```
var root = { values: ret };
```

嘗試由隨機資料
建立一個樹狀資料結構

```
var data = d3.range(20).map(function(d,i) {
  return {
    value: Math.round(Math.random() * 100),
    category: ["A","B","C"][Math.floor(Math.random() * 3)]
  };
});

var ret = d3.nest()
  .key(function(d,i) { return d.category; })
  .entries(data);

var root = { values: ret };
```

d3.hierarchy

Data Structure for Pack Layout

```
{
  children: [
    {…}, {…}, …
  ],
  value: (size)
}
```

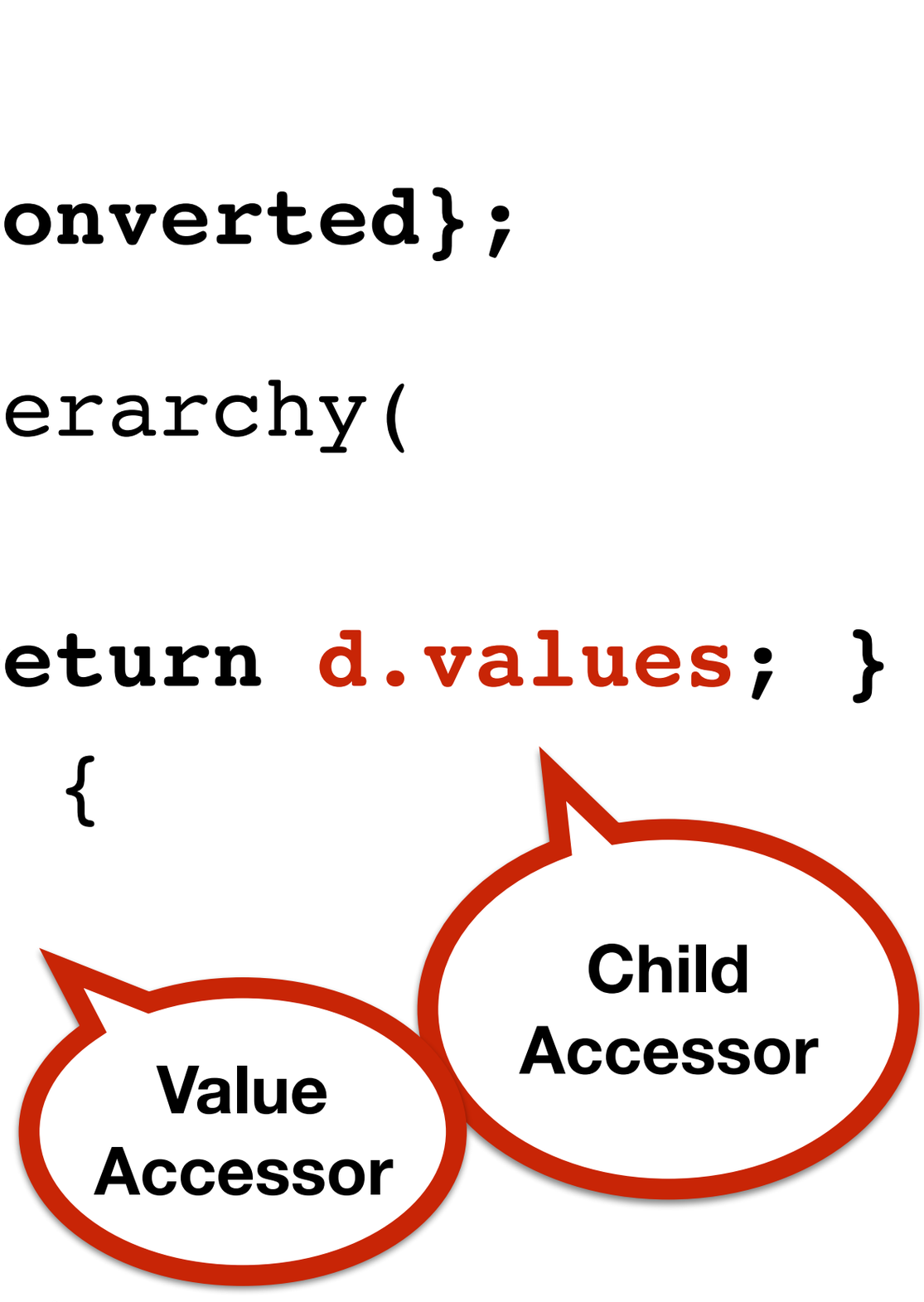
```
d3.hierarchy(
  data,
  childrenAccessor
).sum(
  valueAccessor
);
```

```
a = [
   {value: 1, cat: "A"},
   {value: 2, cat: "B"},
   {value: 3, cat: "B"},
   {value: 4, cat: "C"},
   {value: 5, cat: "C"},
]
```

```
converted =
   d3.nest()
      .key(function(it) {
        return it.cat;
      })
      .entries(a);
```

```
converted = [
  {key: "A", values: [
    {value: 1, cat: "A"}
  ]},
  {key: "B", values: [
    {value: 2, cat: "B"},
    {value: 3, cat: "B"},
  ]},
  {key: "C", values: [
    {value: 4, cat: "C"},
    {value: 5, cat: "C"},
  ]}
]
```

```
root = {values: converted};

hierarchy = d3.hierarchy(
  root,
  function(d) { return d.values; }
).sum(function(d) {
  return d.value;
});

d3.pack()
  .size([800, 400])
  (hierarchy);
```



```
values: [
  {…},
  {
    cat: “A”,
    value: 0.447
    x: 248.867
    y: 414.396
    r: 31.373
    depth: 2
    parent: {…}
  }, {…}, …
]
```

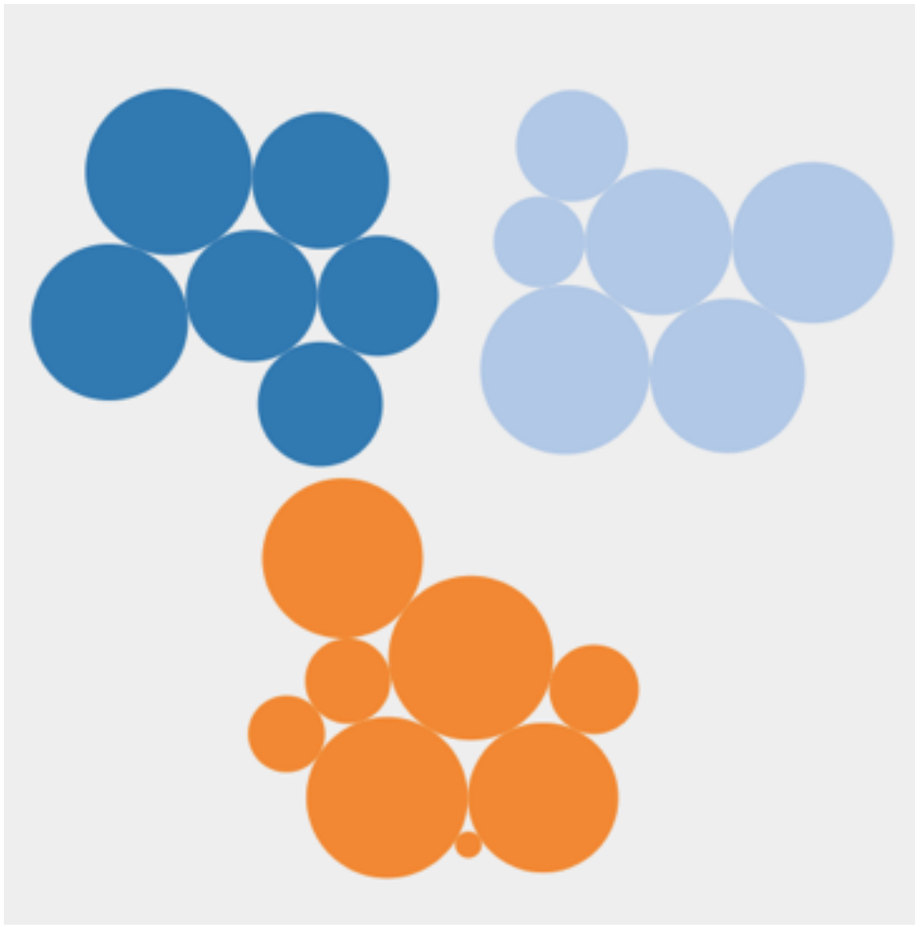
```
var leaves = hierarchy.leaves();
var pack = d3.pack().size([w,h]);
d3.select("svg")
   .selectAll("circle")
   .data(leaves)
   .enter().append("circle")
   .attrs({
     cx: function(it) { return it.x; },
     cy: function(it) { return it.y; },
     r:  function(it) { return it.r; },
     fill: "none",
     stroke: "#000"
   });
```

製作步驟

- 準備資料 (可透過 d3.nest() 協助轉換)
- 建立 Hierarchy 物件 (d3.hierarchy)
- 建立 Pack Layout 物件 (d3.pack)
- 利用 Pack Layout 更新 Hierarchy 中的資料
- 利用 Data Binding 建立元素並繪製

利用自己產生的隨機資料製作泡泡圖

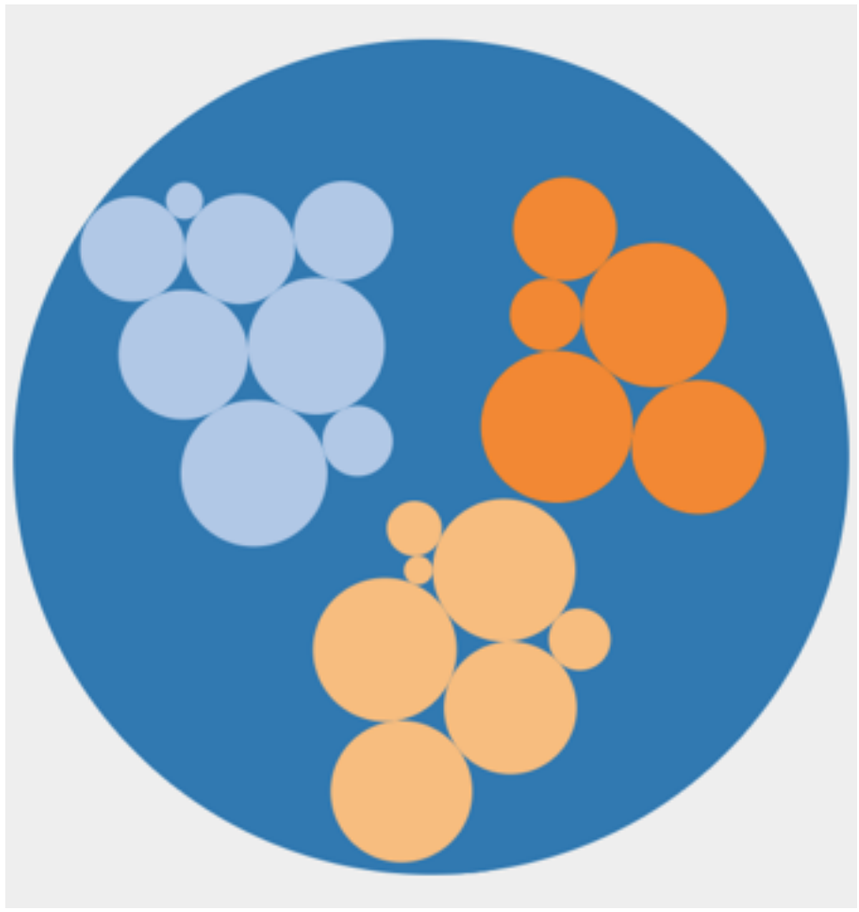
基本題



加分題

將 leaves 改為descendants

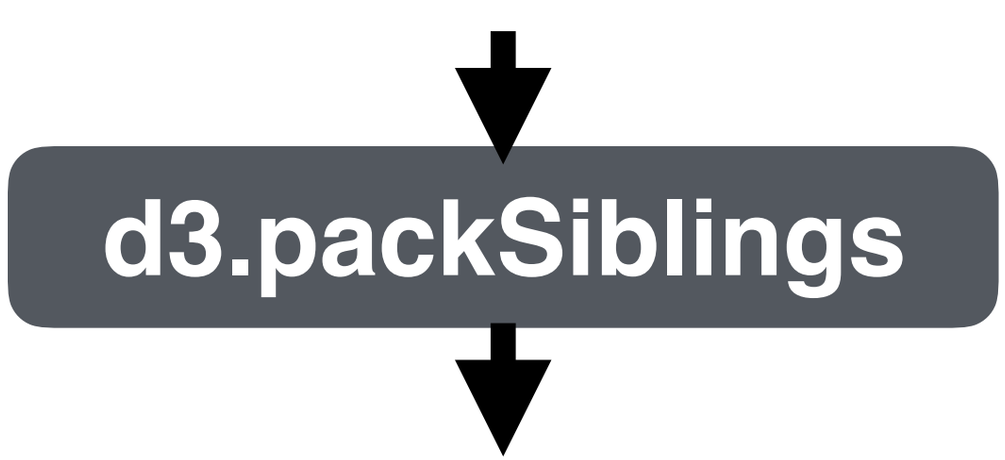
將上層泡泡一併繪出



快速製作泡泡圖

使用 d3.packSiblings

```
a = [{r: 1, cat: "A"},
     ……
     {r: 5, cat: "C"}]
```




```
a = [{r: 1, cat: "A", x: 1, y: 2},
     ……
     {r: 5, cat: "C", x: 2, y: 4}]
```

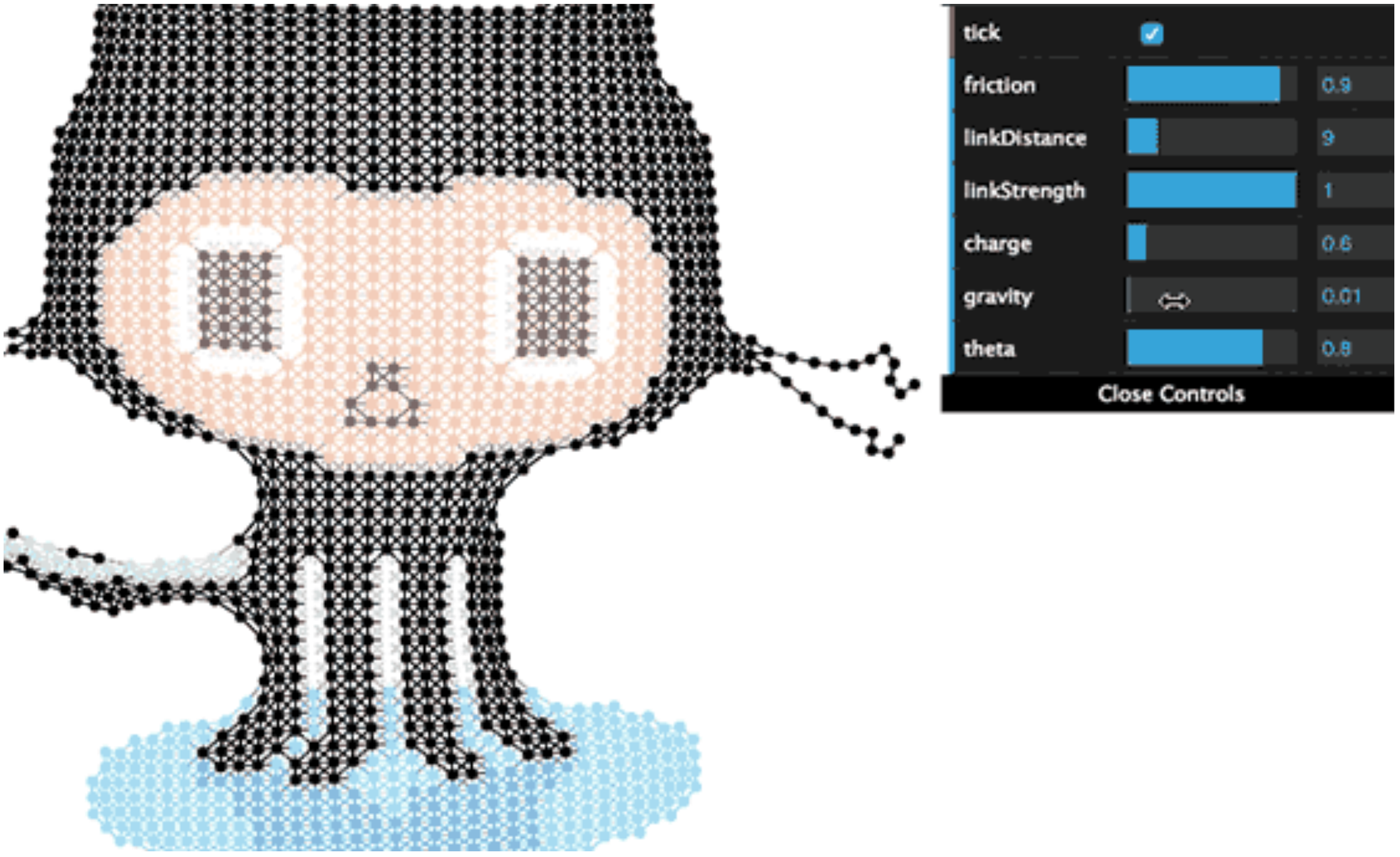
快速製作泡泡圖

使用 d3.packSiblings

```
var data = d3.range(20).map(function(d,i) {
  return {
    r: Math.round(Math.random() * 10),
    category: ……
  };
});
d3.packSiblings(data);
d3.select("svg")
  .selectAll("circle")
  .data(data)
  ……
```

使用 packSiblings 製作泡泡圖

http://bl.ocks.org/christophermanning/4527513



d3-force

```
d3.forceSimulation()  - 建立 Force 物件
  .force(name, ……)    - 套用模型
  .on("tick", ……)     - 繪製
  .nodes(data)        - 設定資料點
```


```
[
  {…},
  {
    cat: “A”,
    value: 0.447
    r: 5
    x: 248.867
    y: 414.396
    vx: 354.939
    vy: 303.569
    index: 1,
  }, {…}, …
]
```

d3-force 模型種類

- Centering – 讓質點重心保持在特定位置
- Collision – 避免質點互相重疊
- Links – 提供質點間的彈簧力
- Many-Body – 模擬多質點間的引力與斥力系統
- Positioning – 令質點移動到特定位置

```
data = [{r: 1}, {r: 2}, {r: 3}, {r: 4}];

force = d3.forceSimulation()
   .force("center",
     d3.forceCenter(600,600))

   .force("collide",
     d3.forceCollide()
       .radius(function(d,i) {return d.r;})

  .force("charge",
     d3.forceManyBody()
       .strength(10));
```

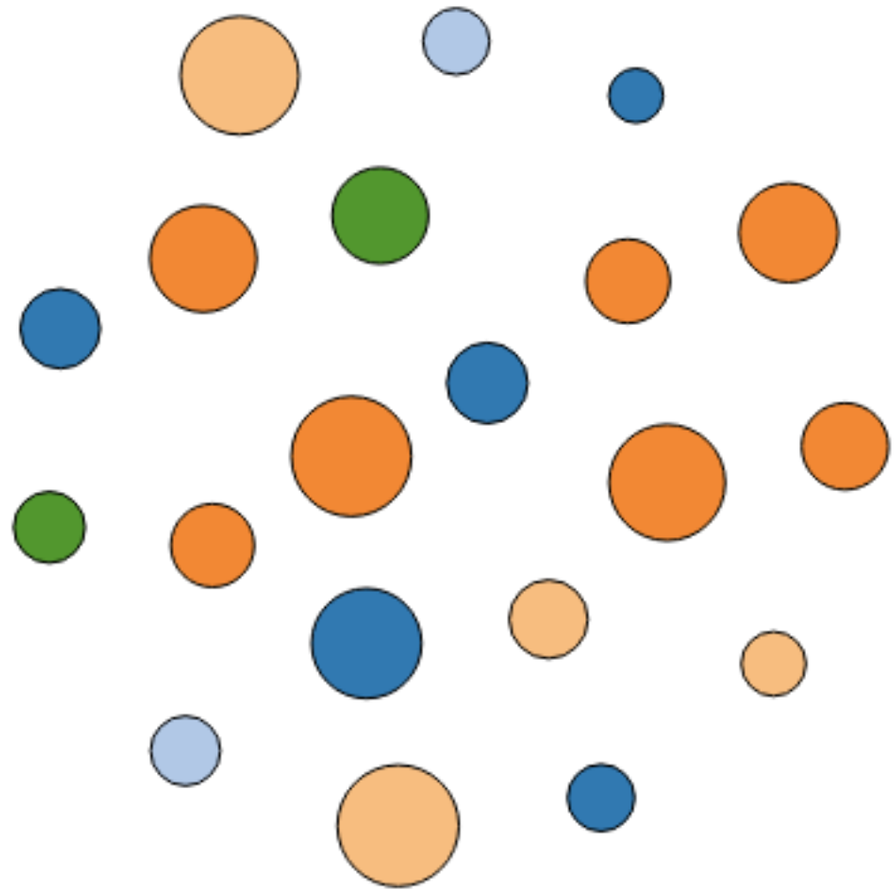
製作步驟

- 準備資料
- 利用 Data Binding 建立元素
- 建立 Force Layout 物件、設定模型
- 利用 Force Layout 更新資料
- 利用 Tick Event 繪製

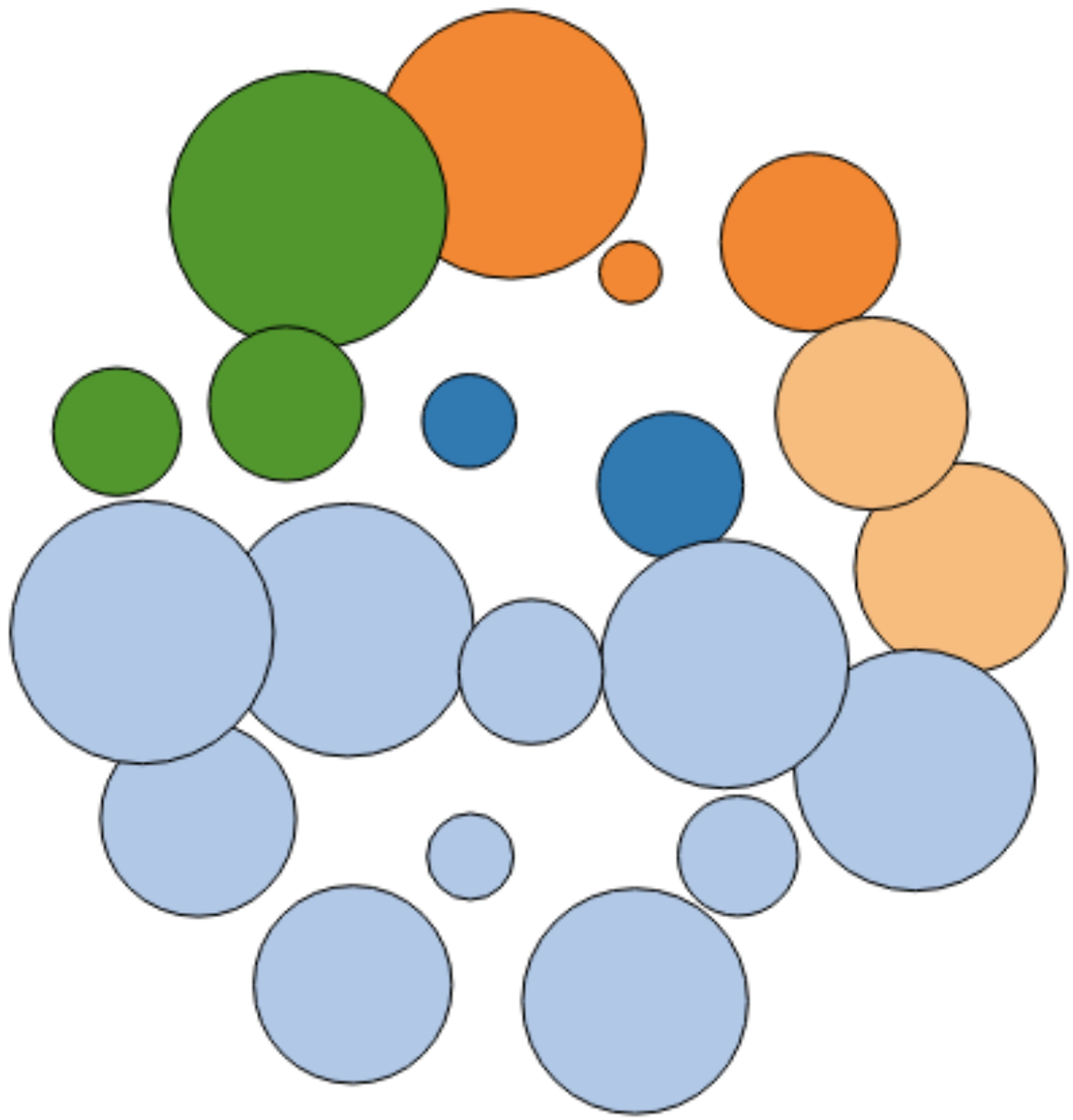
實作練習 #16

利用 #13 的結果做出彈力圖

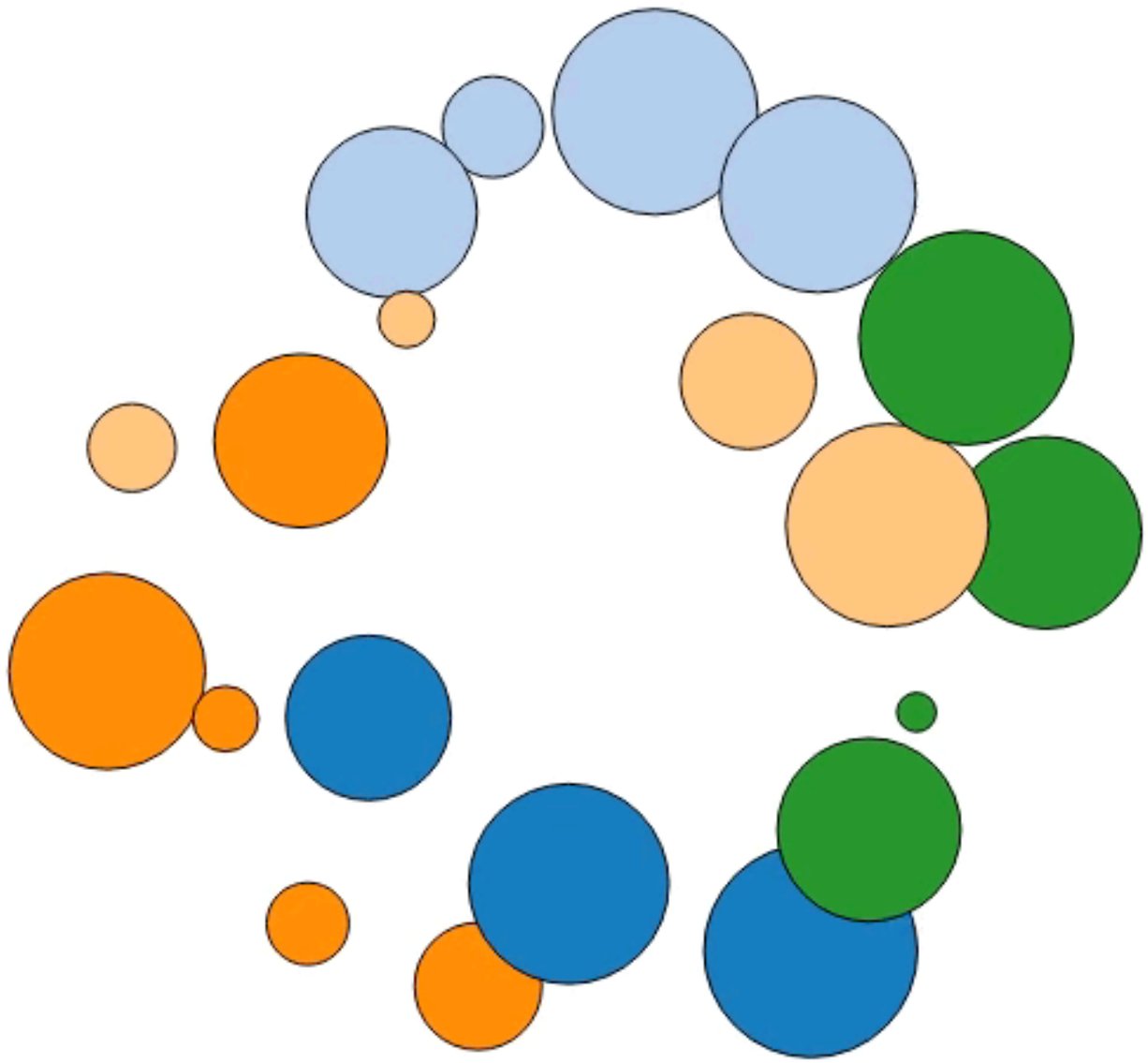
並利用 d3.schemeCategory20 著色



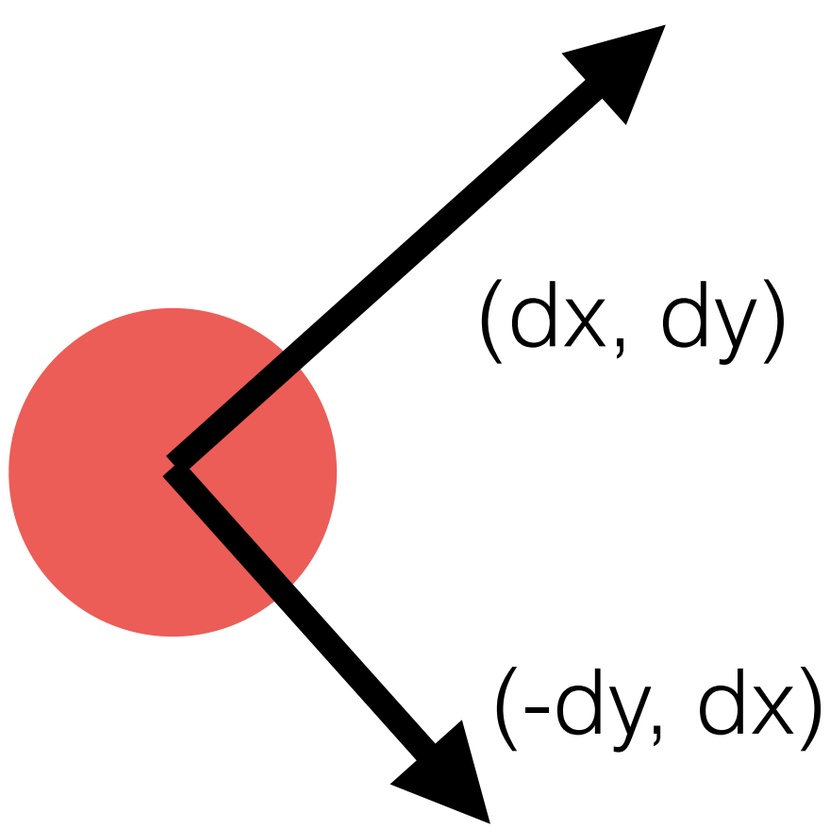
Pack Layout + Force Layout



Pack Layout + Force Layout and Rotate!

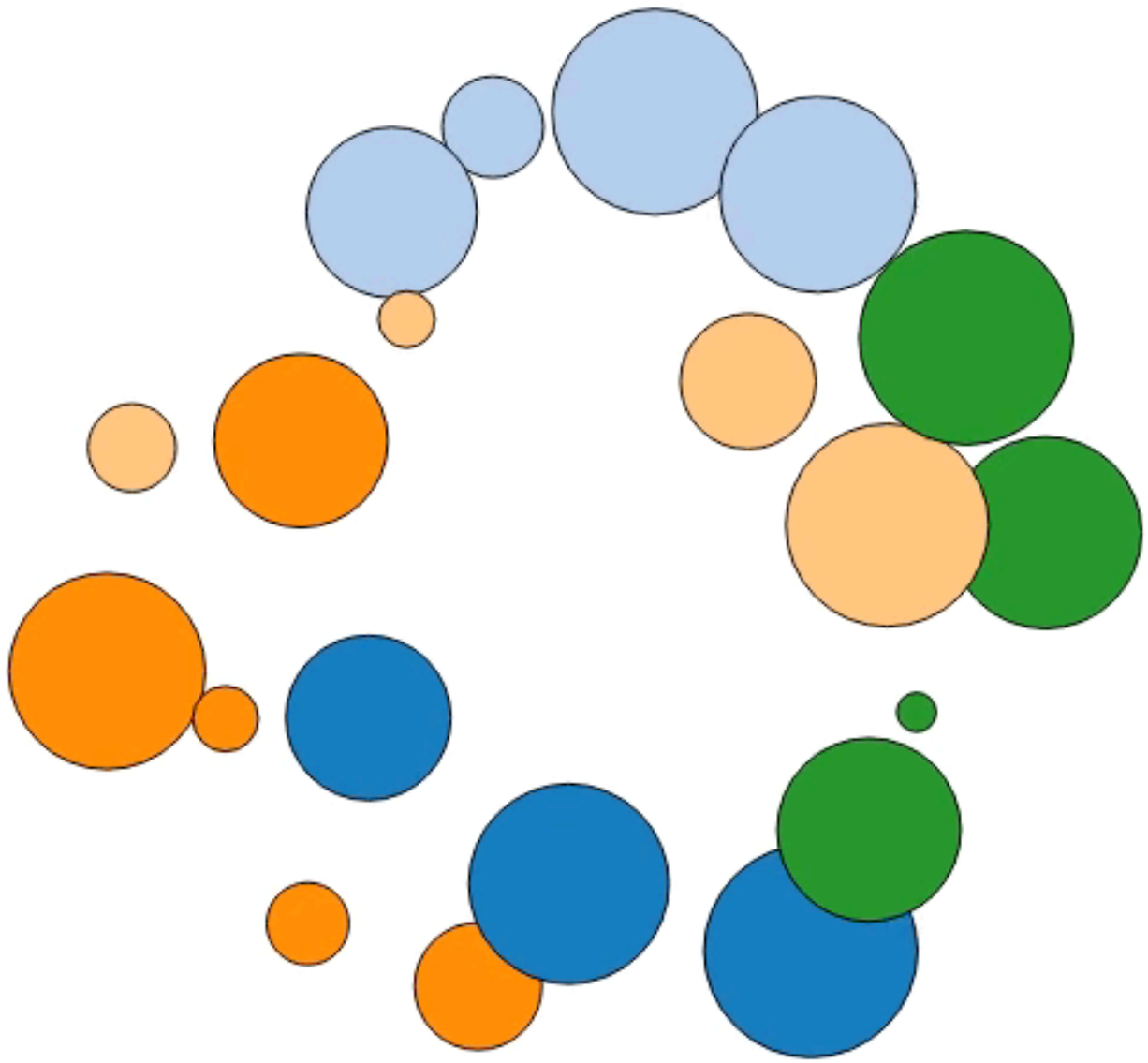


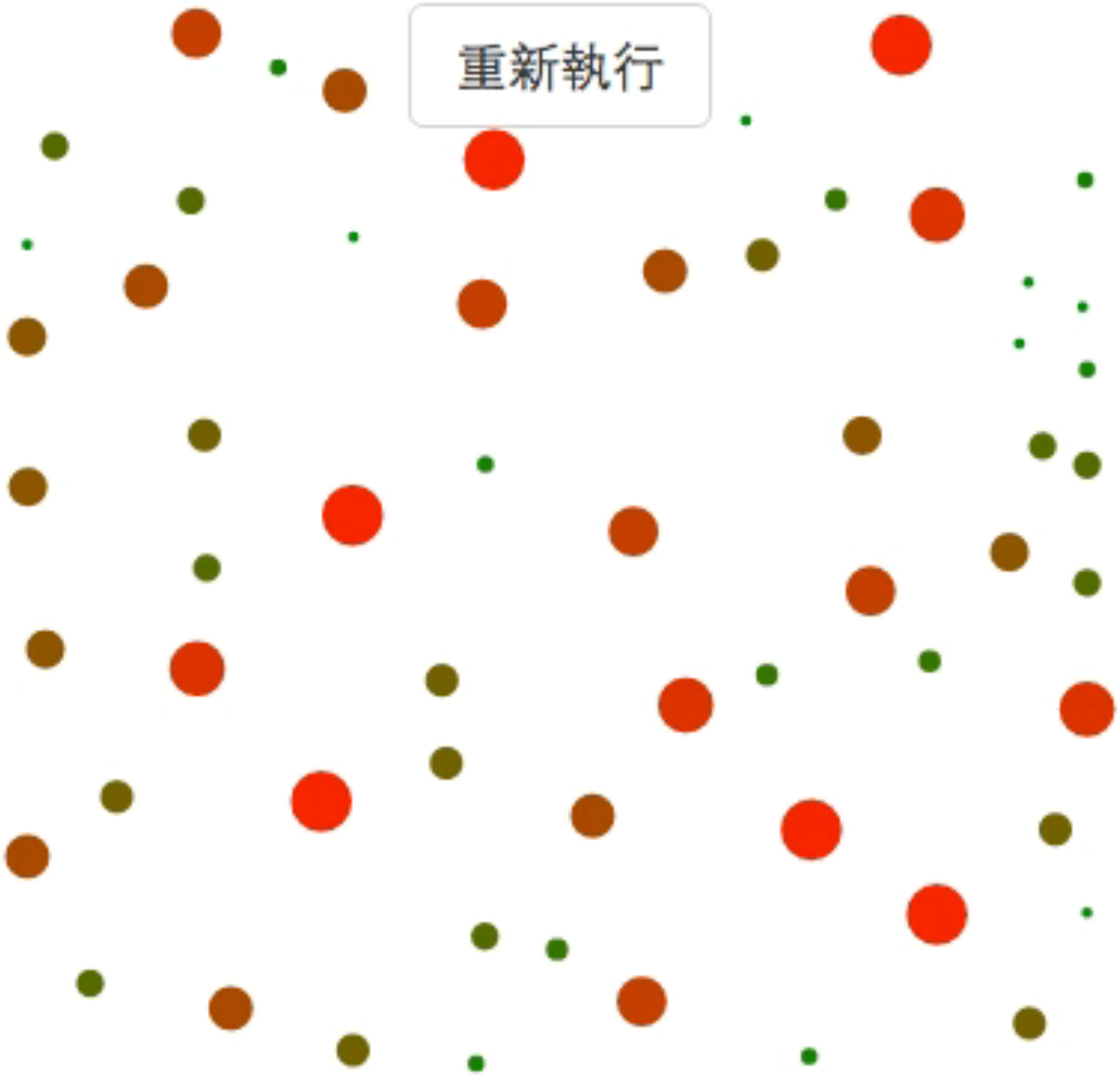
```
function tick() {
  for(var i=0;i<nodes.length;i++) {
    n = nodes[i];
    dx = 300 - n.x;
    dy = 300 - n.y;
    len = Math.sqrt(dx * dx + dy * dy );
    n.x -= dy / len;
    n.y += dx / len;
    }
}
```

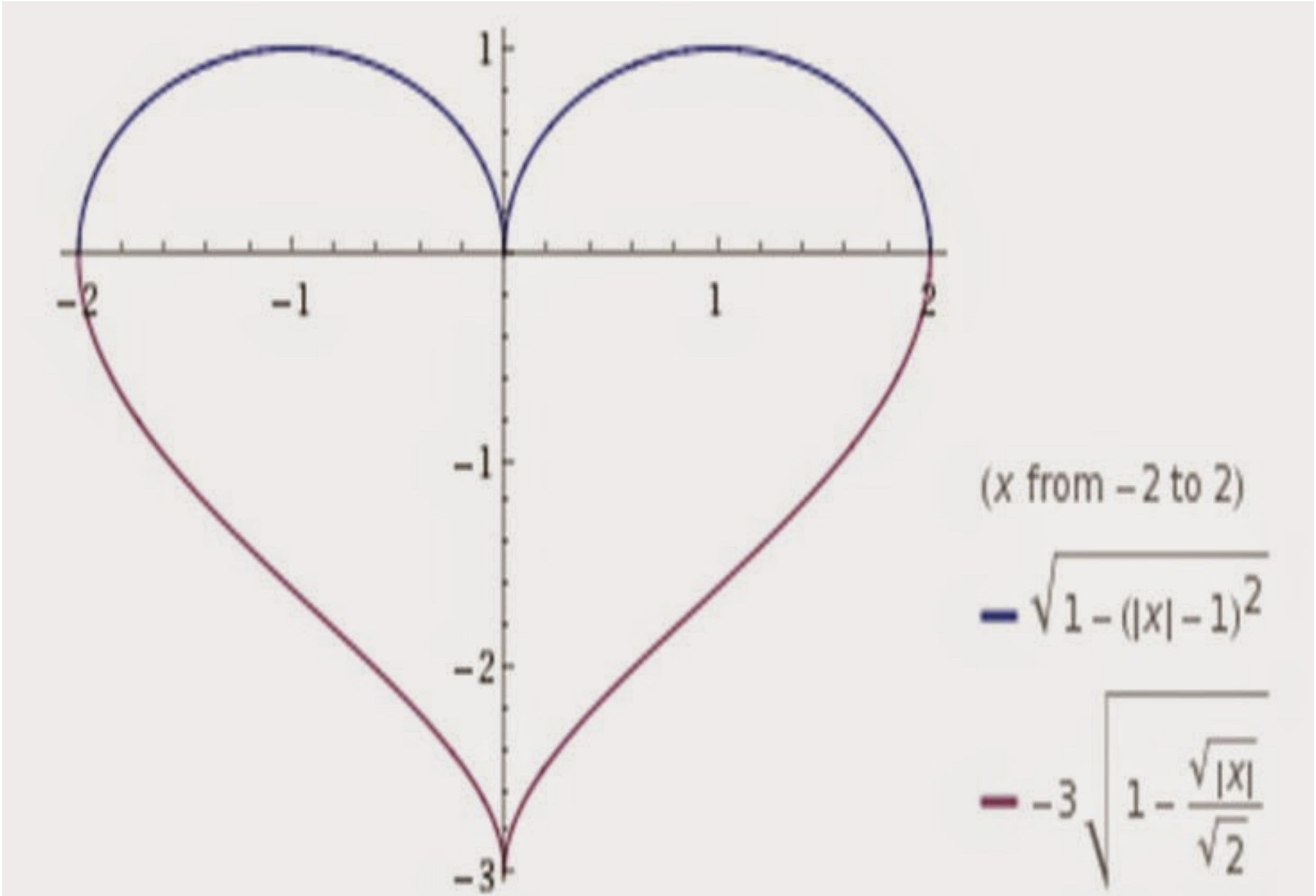


實作練習 #17

讓 #16 的結果旋轉！




重新執行


1
-2
-1
1
2
-1
-2
-3
(x from -2 to 2)
$-\sqrt{1-(|x|-1)^2}$
$-3\sqrt{1-\frac{\sqrt{|x|}}{\sqrt{2}}}$