
最熱門的互動資料視覺化技術

Kirby Wu

infographics.tw


HTML
CSS
JS

# Chart.js

ean and engaging charts for designers and dev

Download

Documentation

# CHARTIST.JS

SIMPLE RESPONSIVE CHARTS

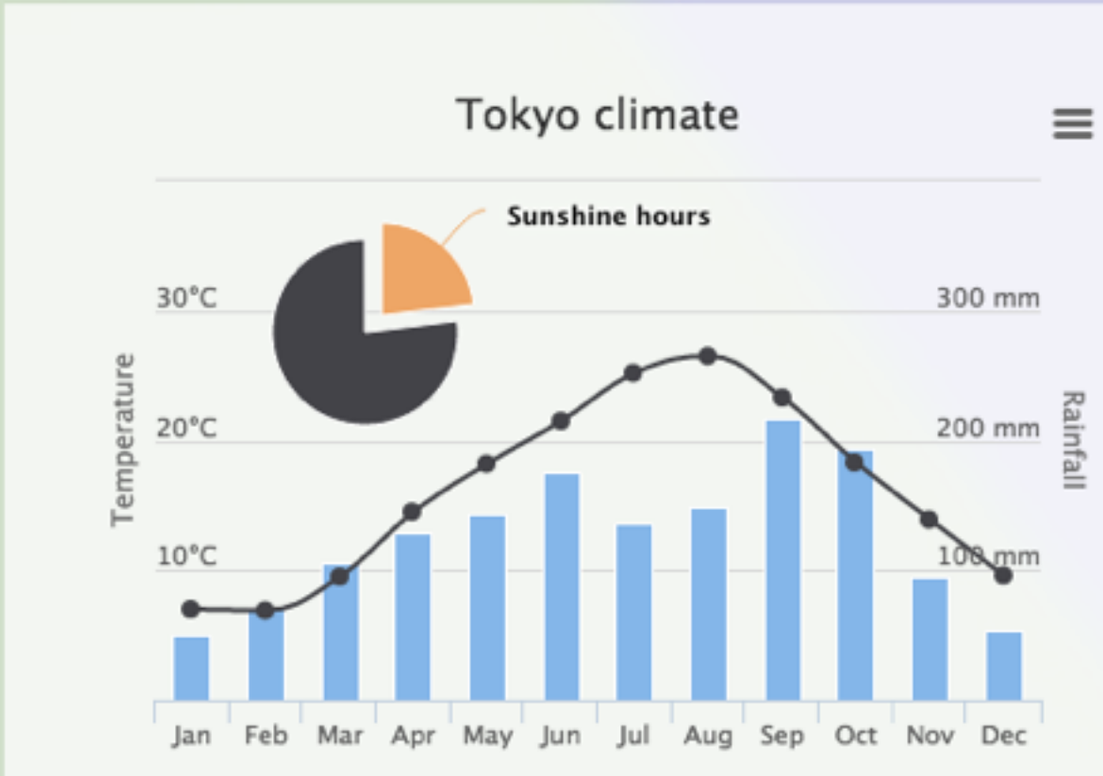

# HIGHCHARTS

**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 »



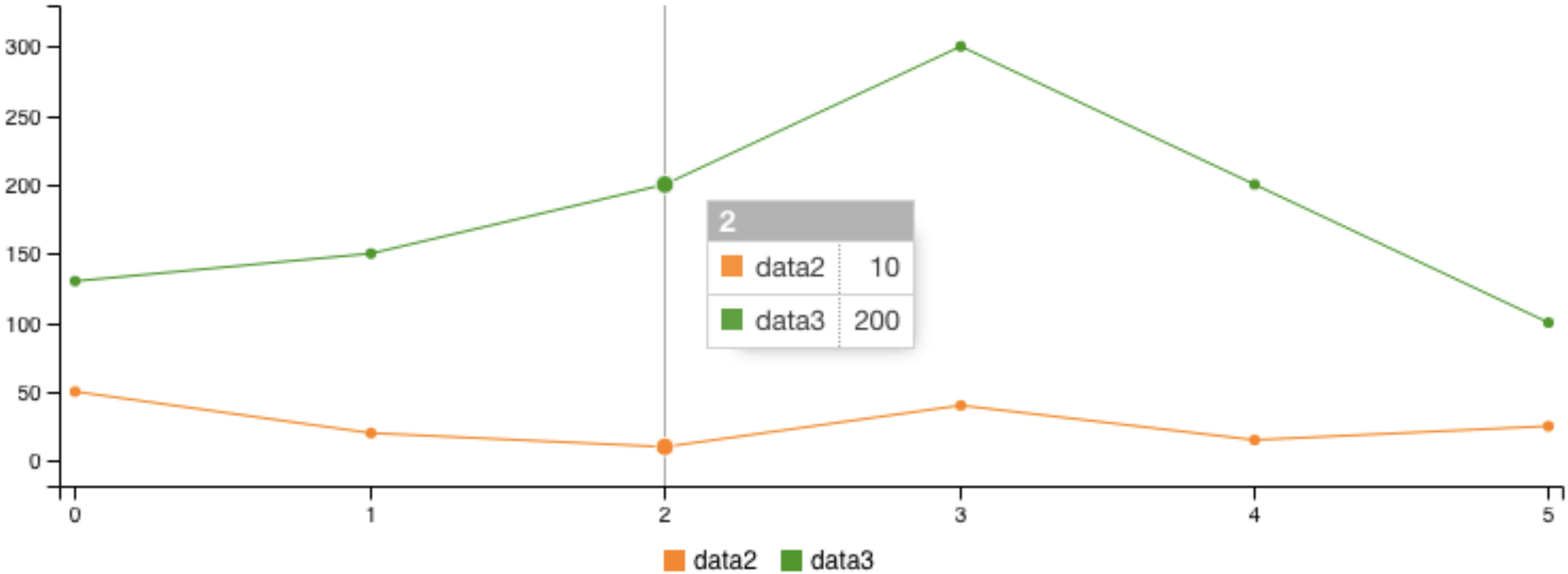
# 現有的圖表工具

- 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

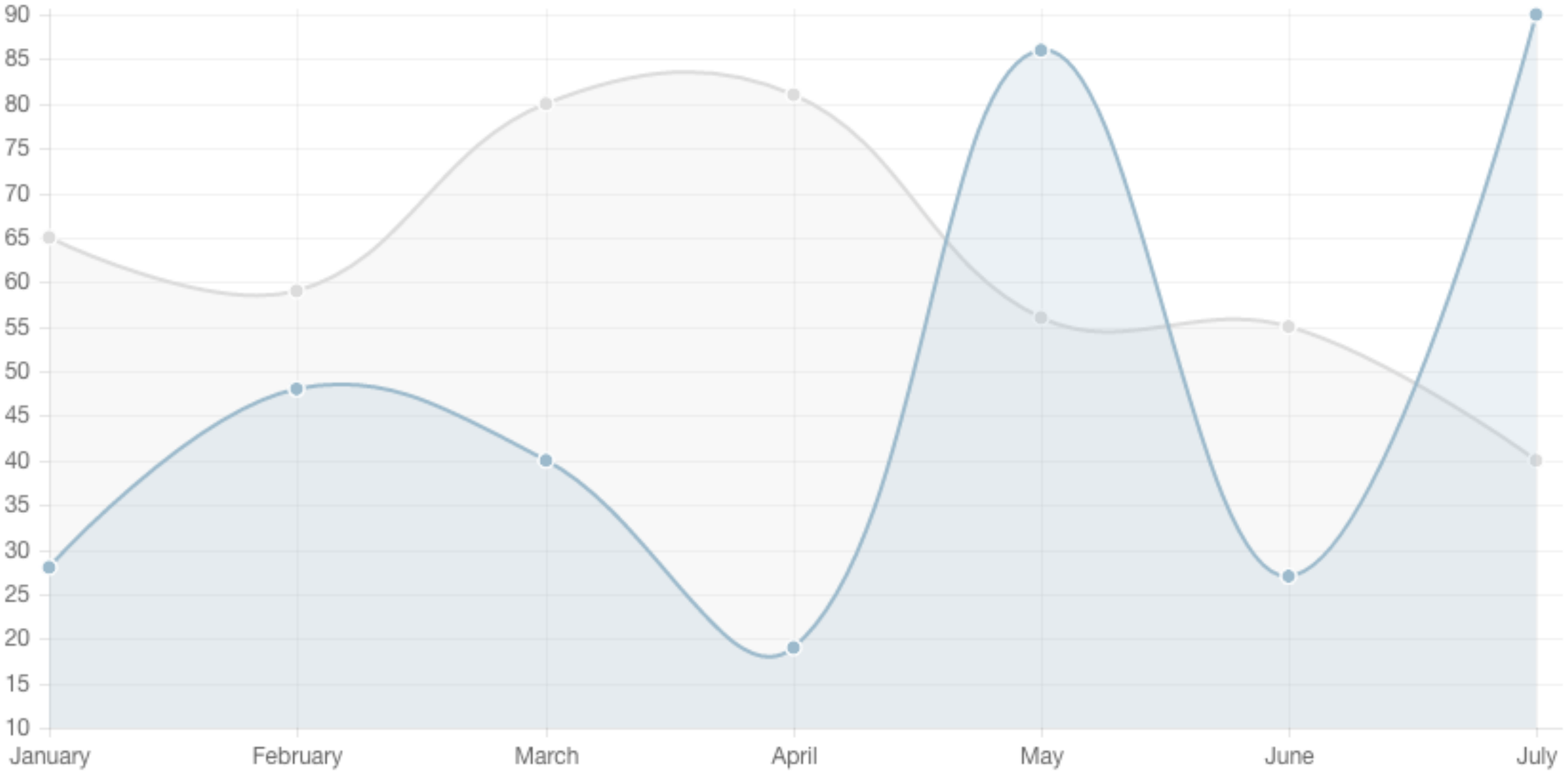
# C3.js


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

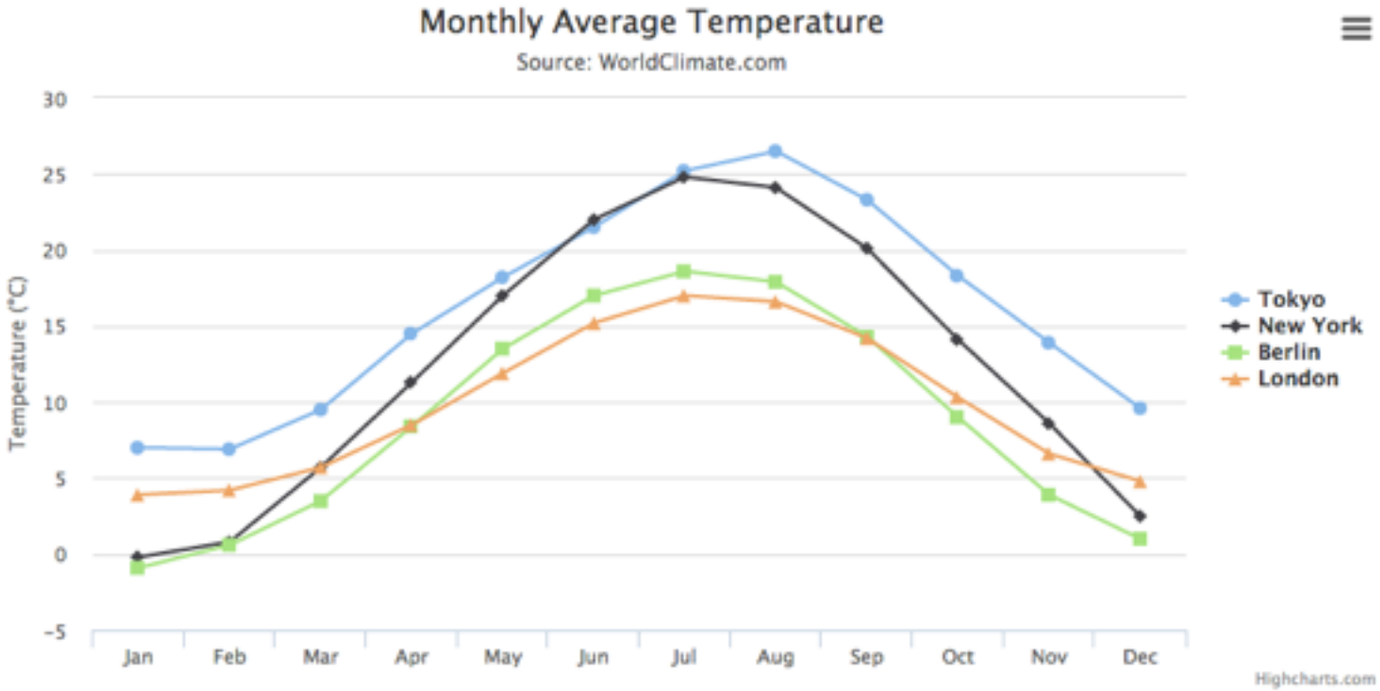
# n3-charts

Awesome charts for Angular

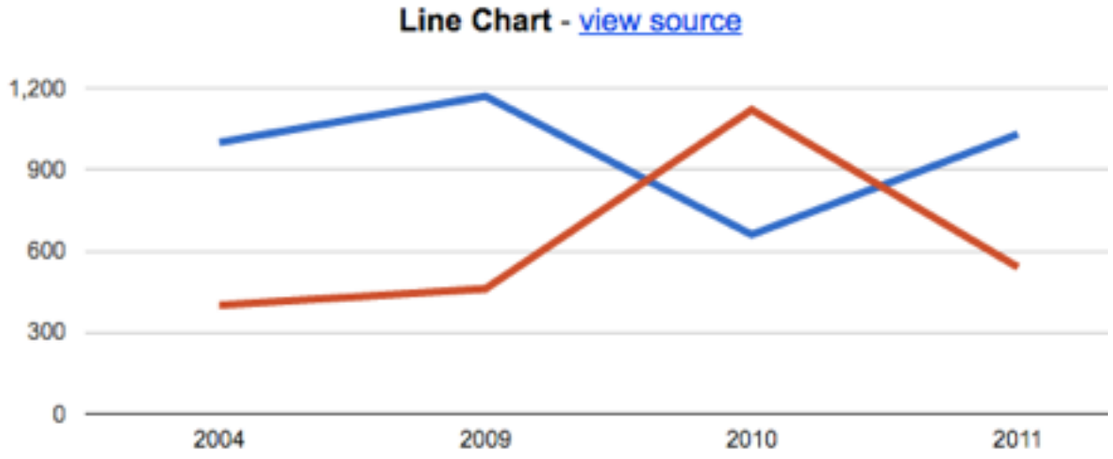
# Chart.js

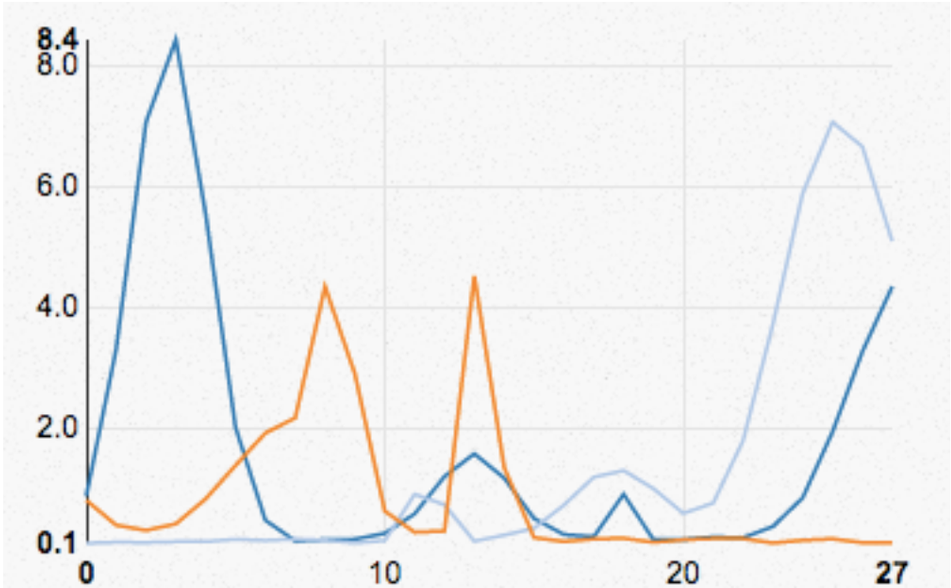

90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
January
February
March
April
May
June
July

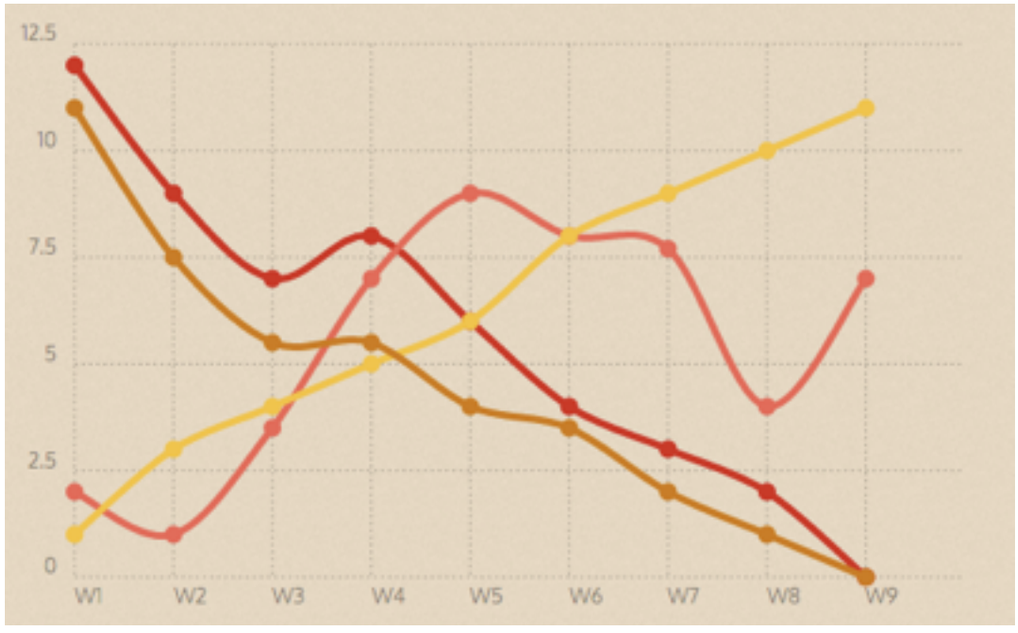
# Highcharts & Amcharts

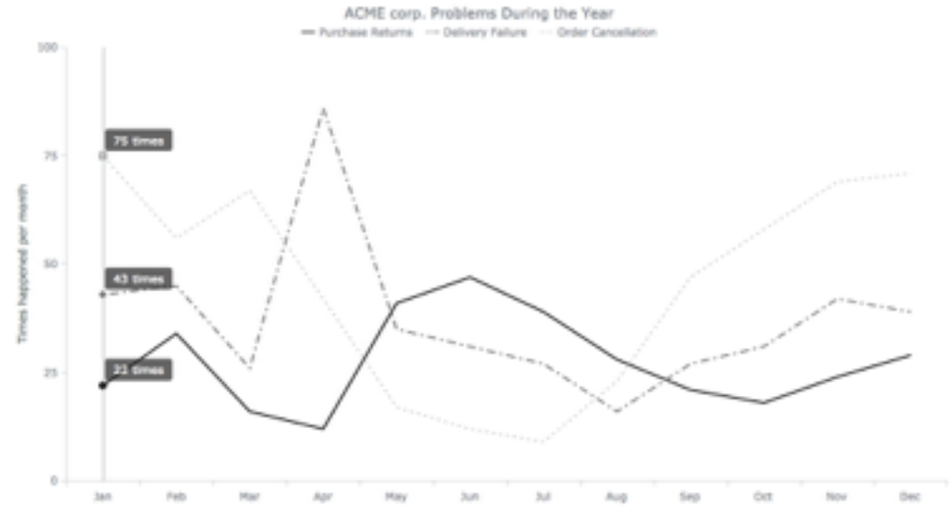



# Anychart, Google Chart, Chartist,  nvd3

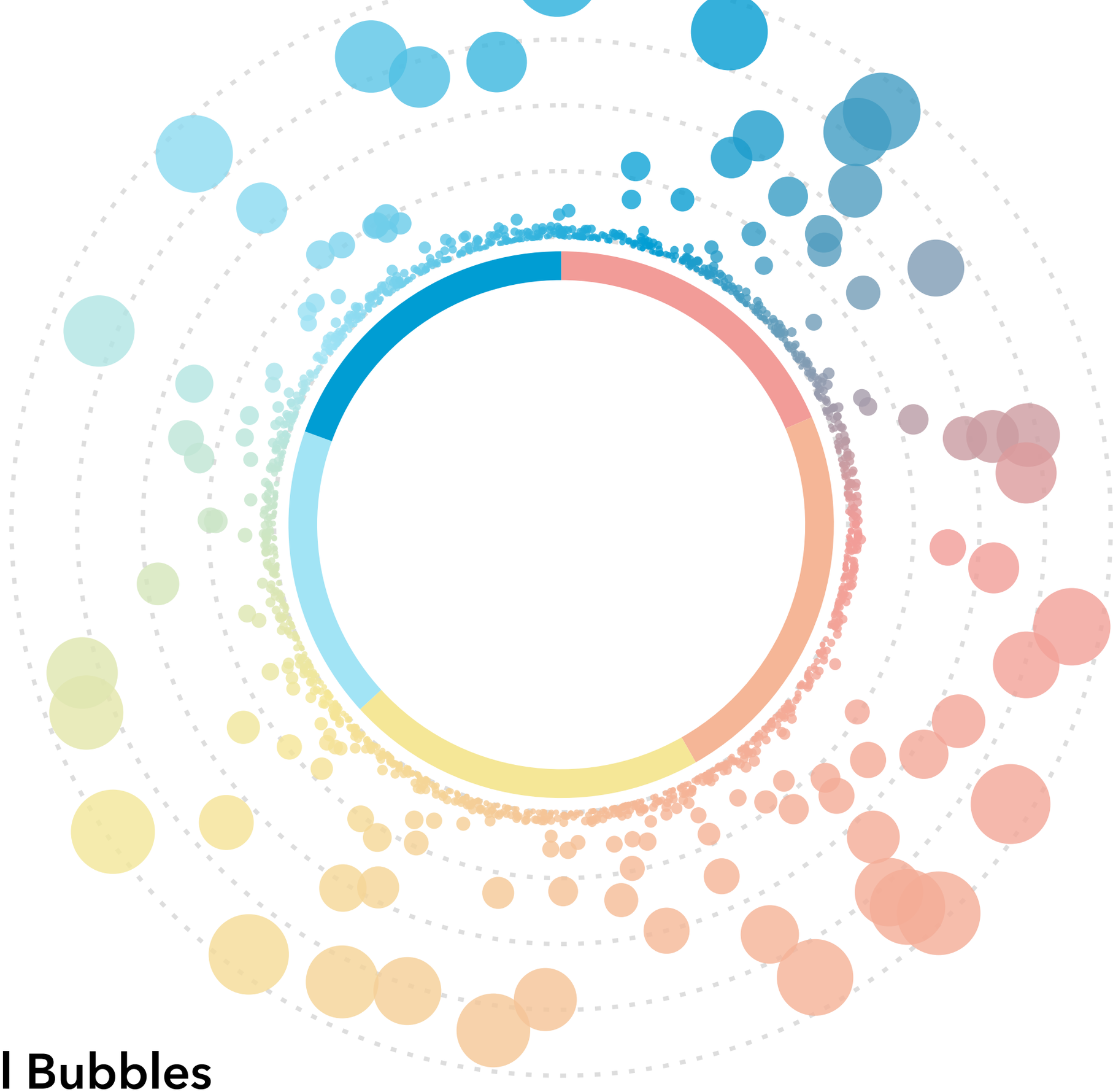


# 2. Flexibility

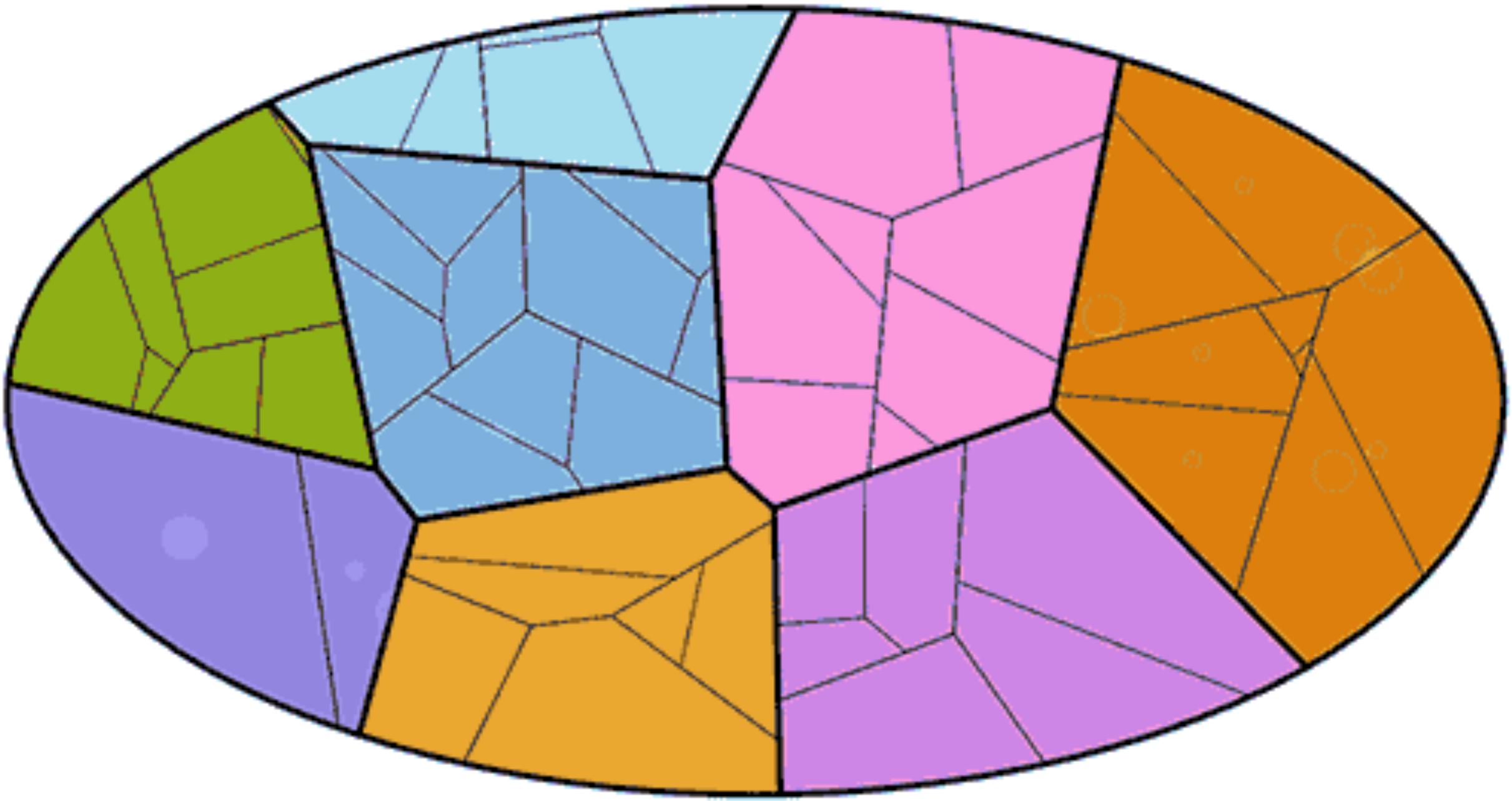


# Radial Bubbles

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

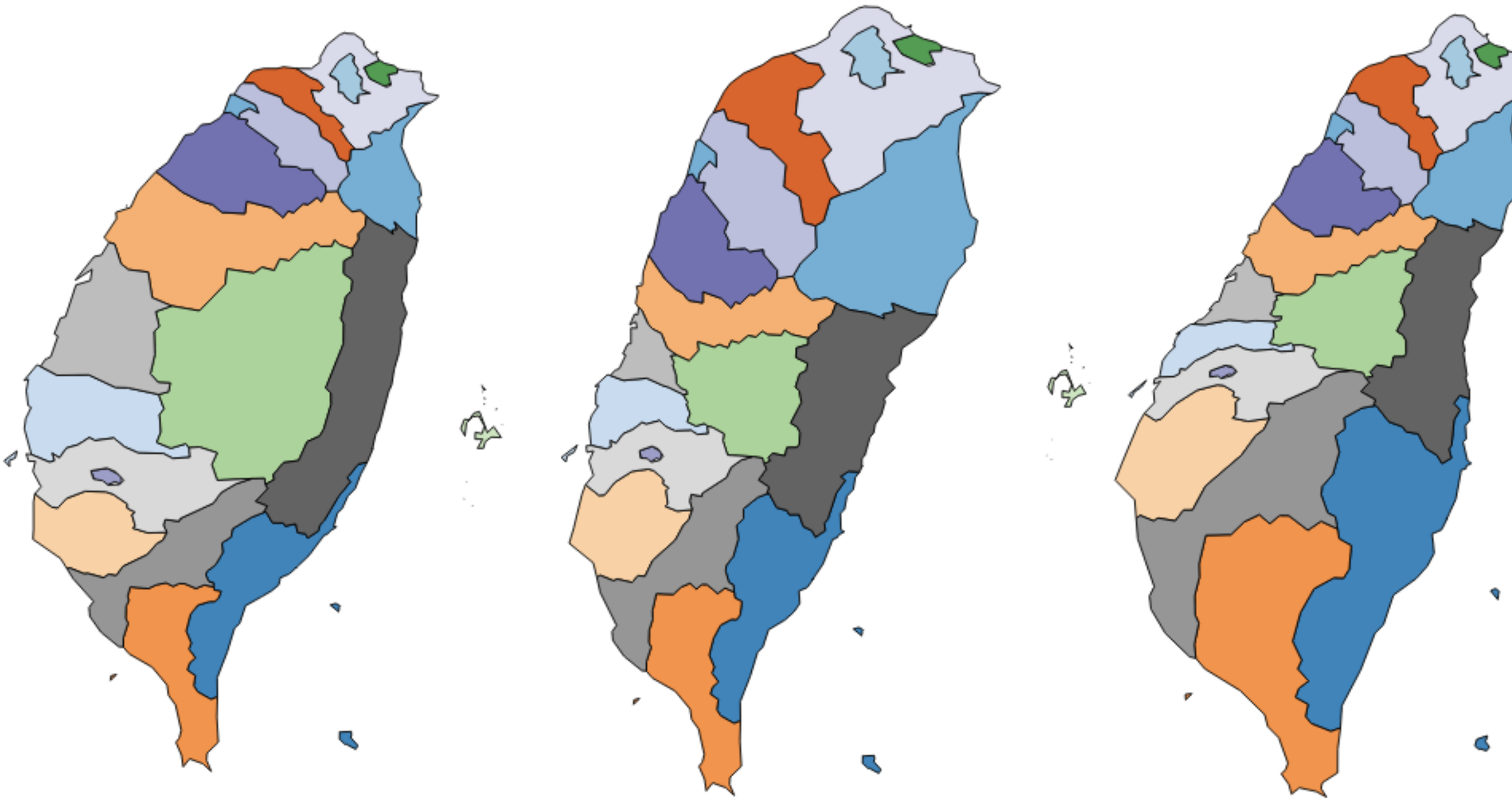

# Sticky Bubbles

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. Download/unzip D3.js

https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.16/d3.min.js

## 2. Create index.html and type:

```
<body></body>
<script src="d3.min.js"></script>
<script>
d3.select("body").text("Hello World!");
</script>
```

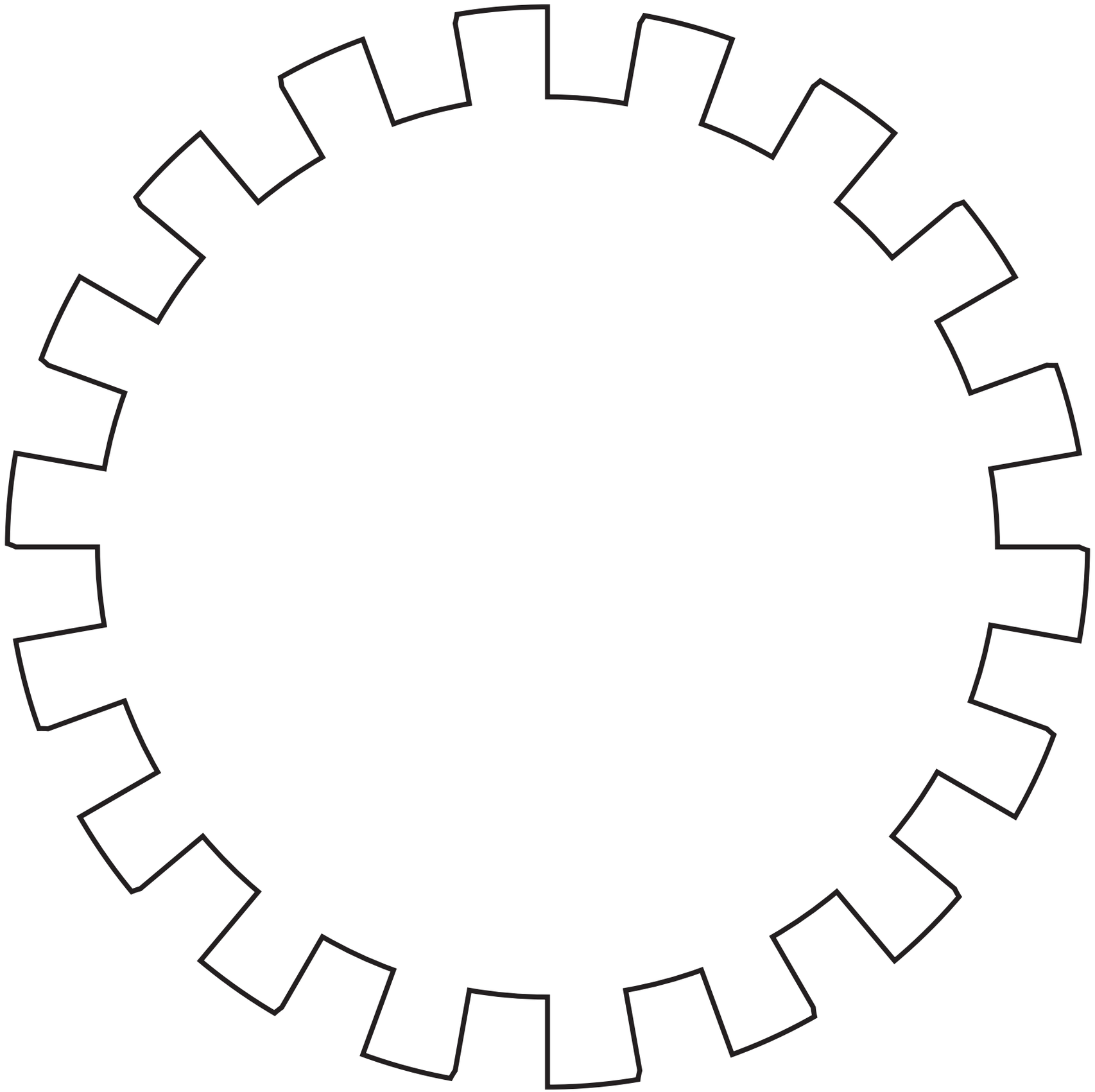

SVG

```
<body>
<svg width="100%" height="100%">
</svg>
</body>
<script src="d3.min.js"></script>
<script></script>
```

# 練習 #01 利用 SVG 畫一個紅色的圓

```
<body>
<svg width="100%" height="100%">
<circle
   cx="200"
   cy="200"
   r="100"
   fill="red"/>
</svg>
</body>
<script src="d3.min.js"></script>
<script></script>
```

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



<?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>

```
d3.select("svg")
    .append("circle")
    .attr({
      cx: 200,
      cy: 200,
      r: 100,
      fill: "red"
});
```

```
d3.select("svg")
    .append("circle")
    .attr({
        cx: 200,
        cy: 200,
        r: 100,
        fill: "red"
});
```

functional style

（注意傳回值）

config with object

# 練習 #02 利用 D3.js 畫一個紅色的長方形

```
<body>
<svg width="100%" height="100%">
</svg>
</body>
<script src="d3.min.js"></script>
<script>
d3.select("svg")
  .append("rect")
  .attr({
    x: 10,
    y: 10,
    width: 100,
    height: 20,
    fill: "red"
  });
</script>
```


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

node.text("Hello World!")

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

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

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

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

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

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

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

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

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

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

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

```
node.style(
    {
        "width": "100%",
        "height": "100%",
        "background: "red"
    }
)
```

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

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

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

return "div"

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

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

# 練習 #02 利用 D3.js 畫一個紅色的長方形

```
<body>
<svg width="100%" height="100%">
</svg>
</body>
<script src="d3.min.js"></script>
<script>
d3.select("svg")
  .append("rect")
  .attr({
    x: 10,
    y: 10,
    width: 100,
    height: 20,
    fill: "red"
  });
</script>
```

我要打十個

# 暴力法

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

x 10

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

# 使用迴圈

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

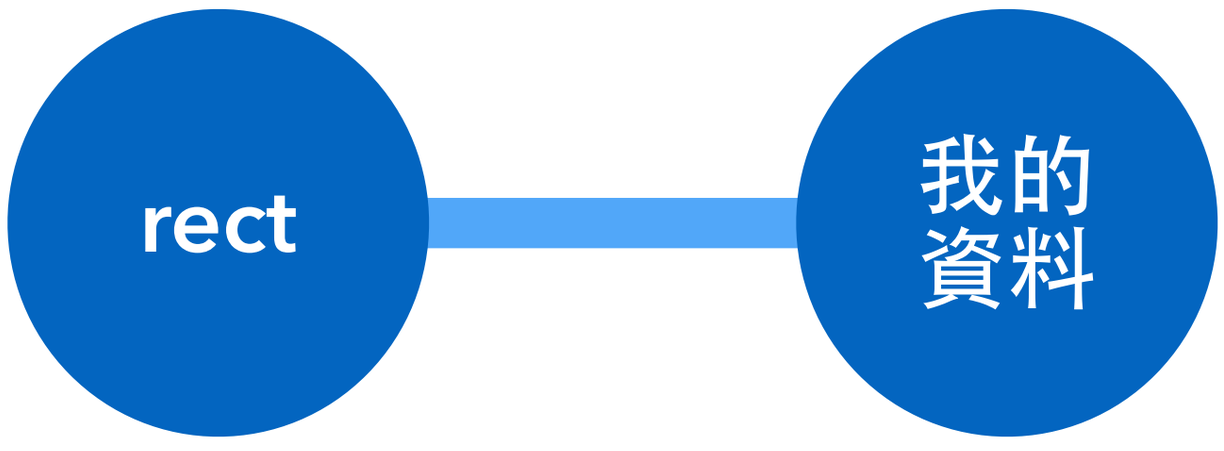
# 使用迴圈

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


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

# Data Binding

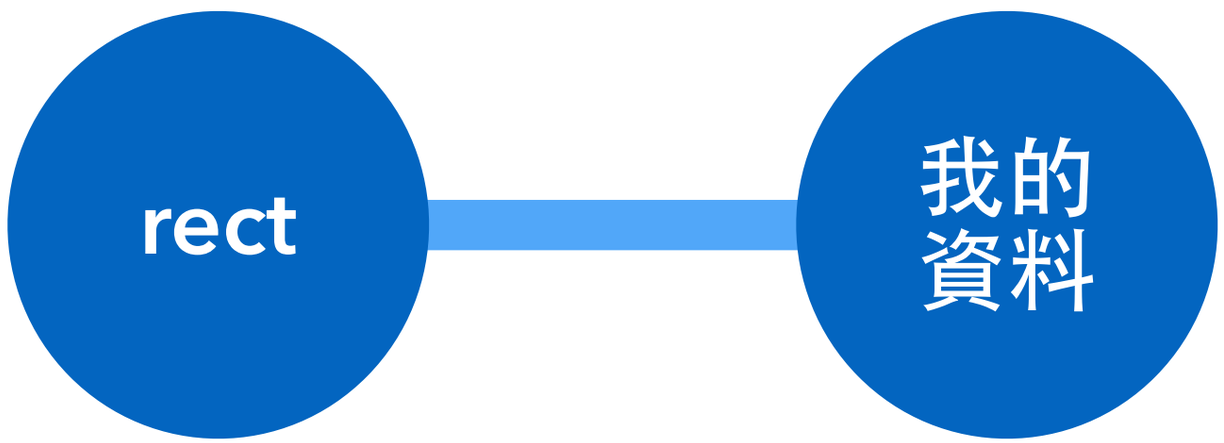
```
d3.select("rect")
    .datum( 我的資料 )
```



# Data Binding

d3.select("rect").datum( 我的資料 )

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



# 使用迴圈

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

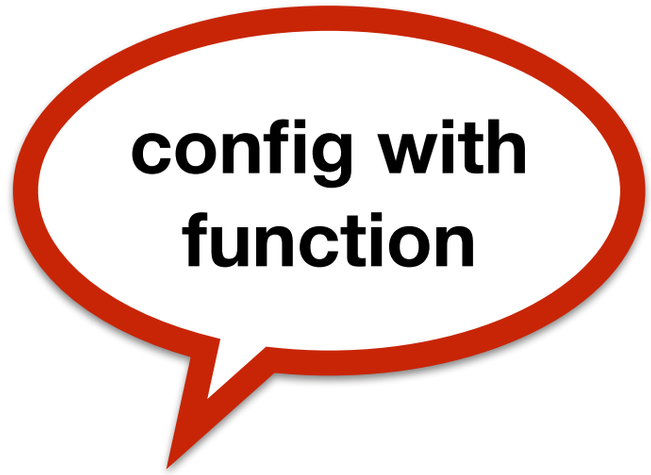
reorder?

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

data?

**.datum()**

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




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

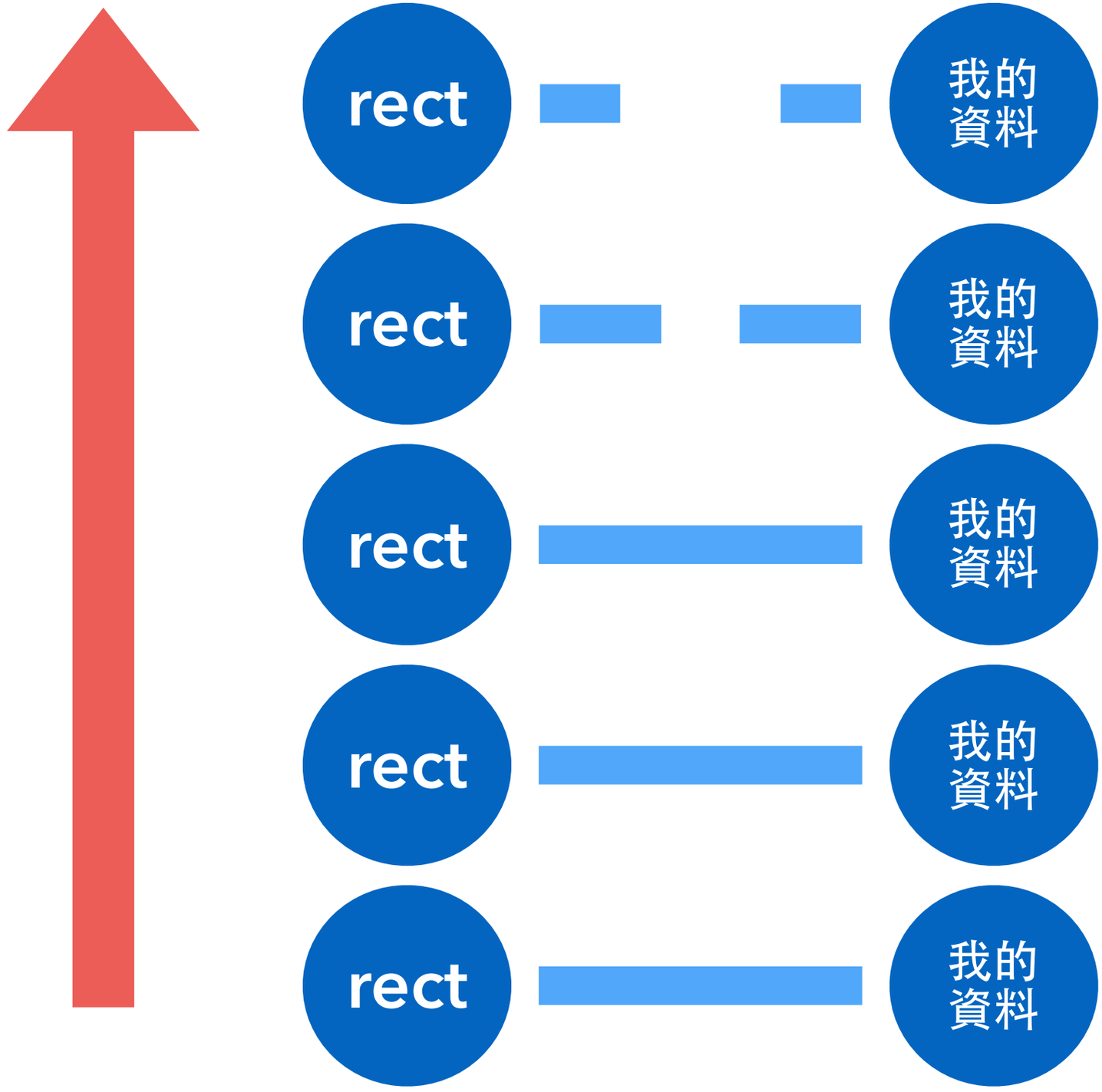
```
for(var i =1; i<=10;i++) {
   d3.select("svg")
      .append("rect")
      .datum(data[i])
      .style({
         width: function(d,i) {
            return d * 10;
         }
      });
}
```

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

到底要多少個？

多出來的
怎麼辦？

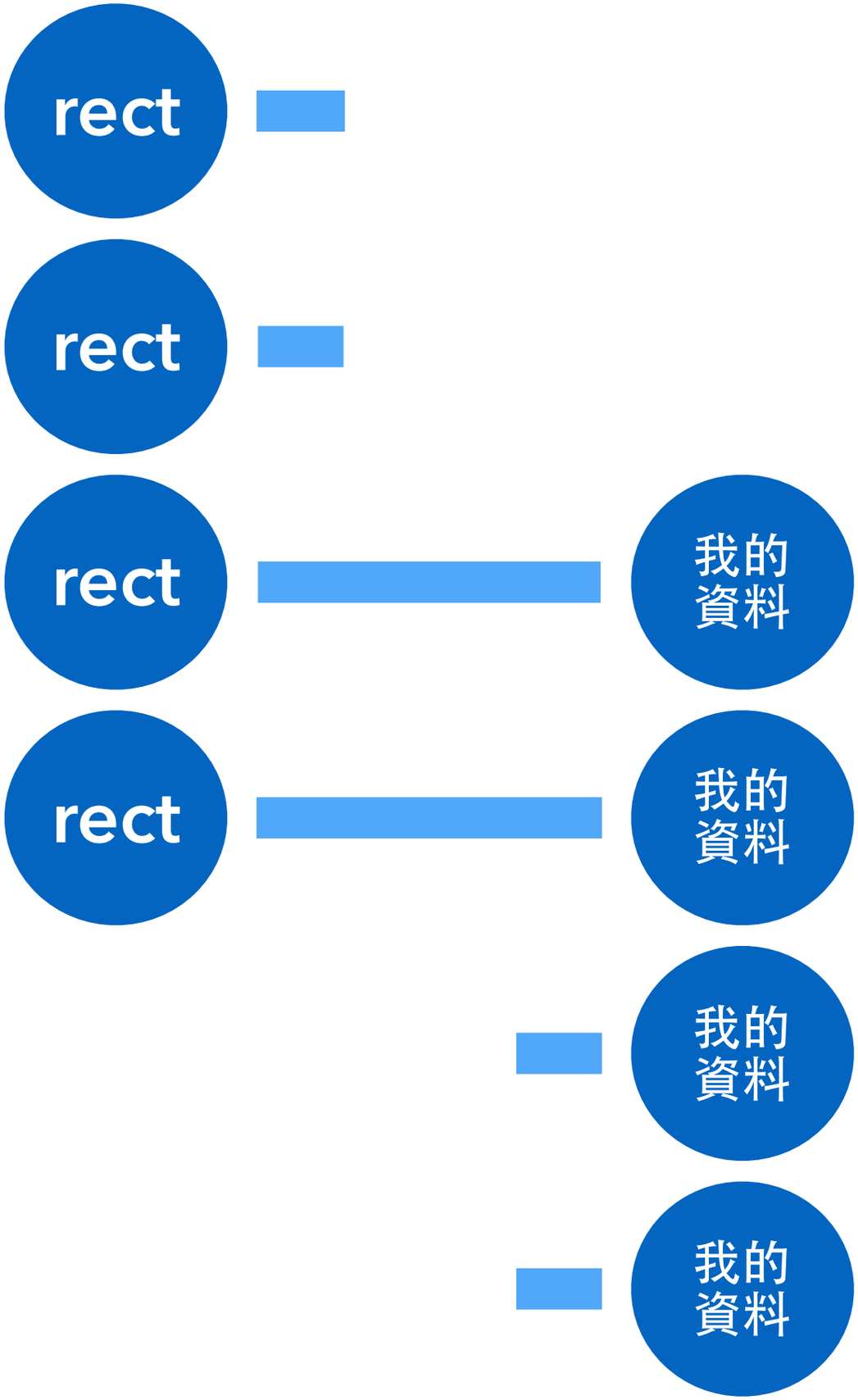
# 幫我自動綁定資料



資料已經
不見的

之前已經
綁好的

還沒建立
元素的

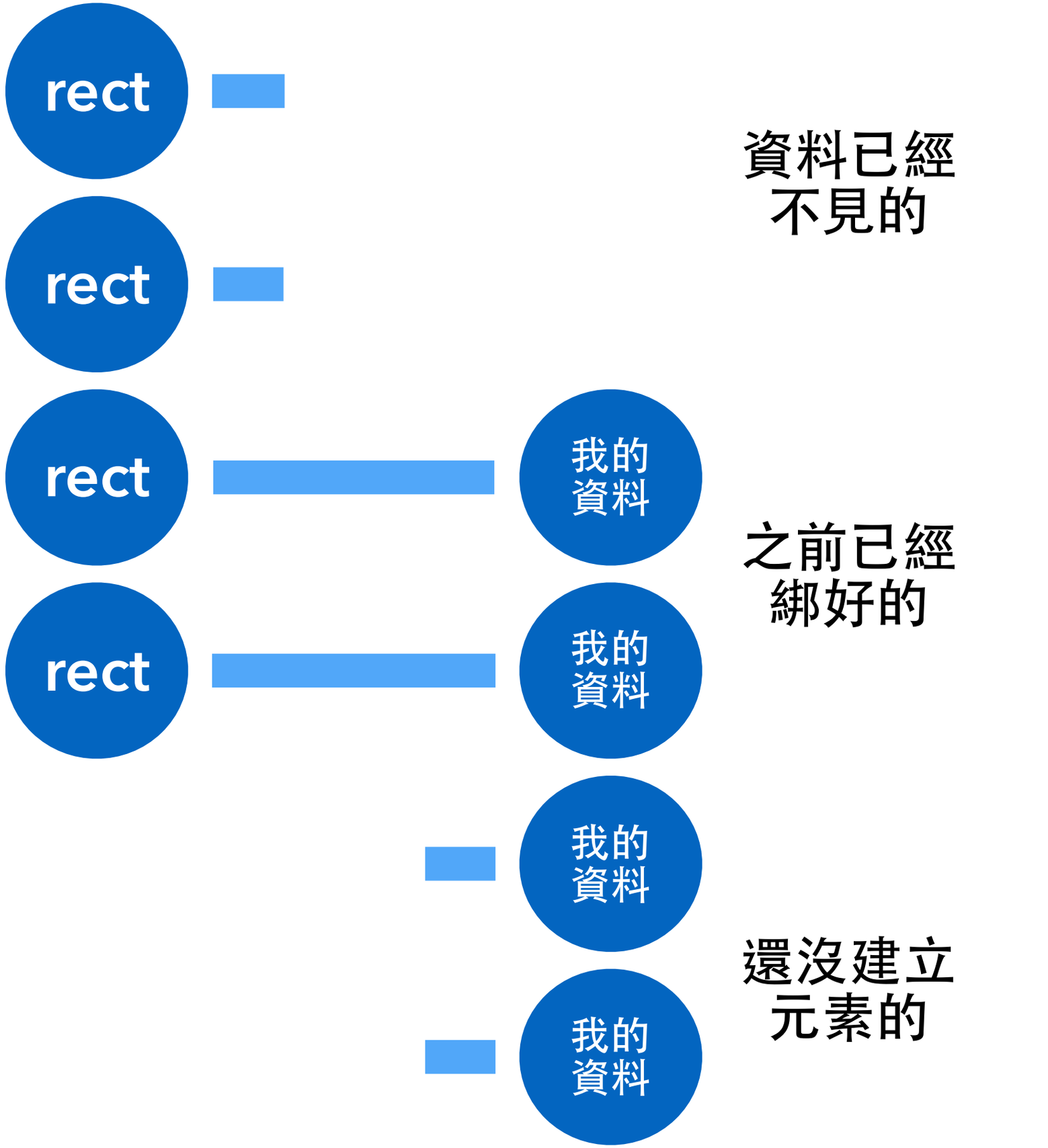


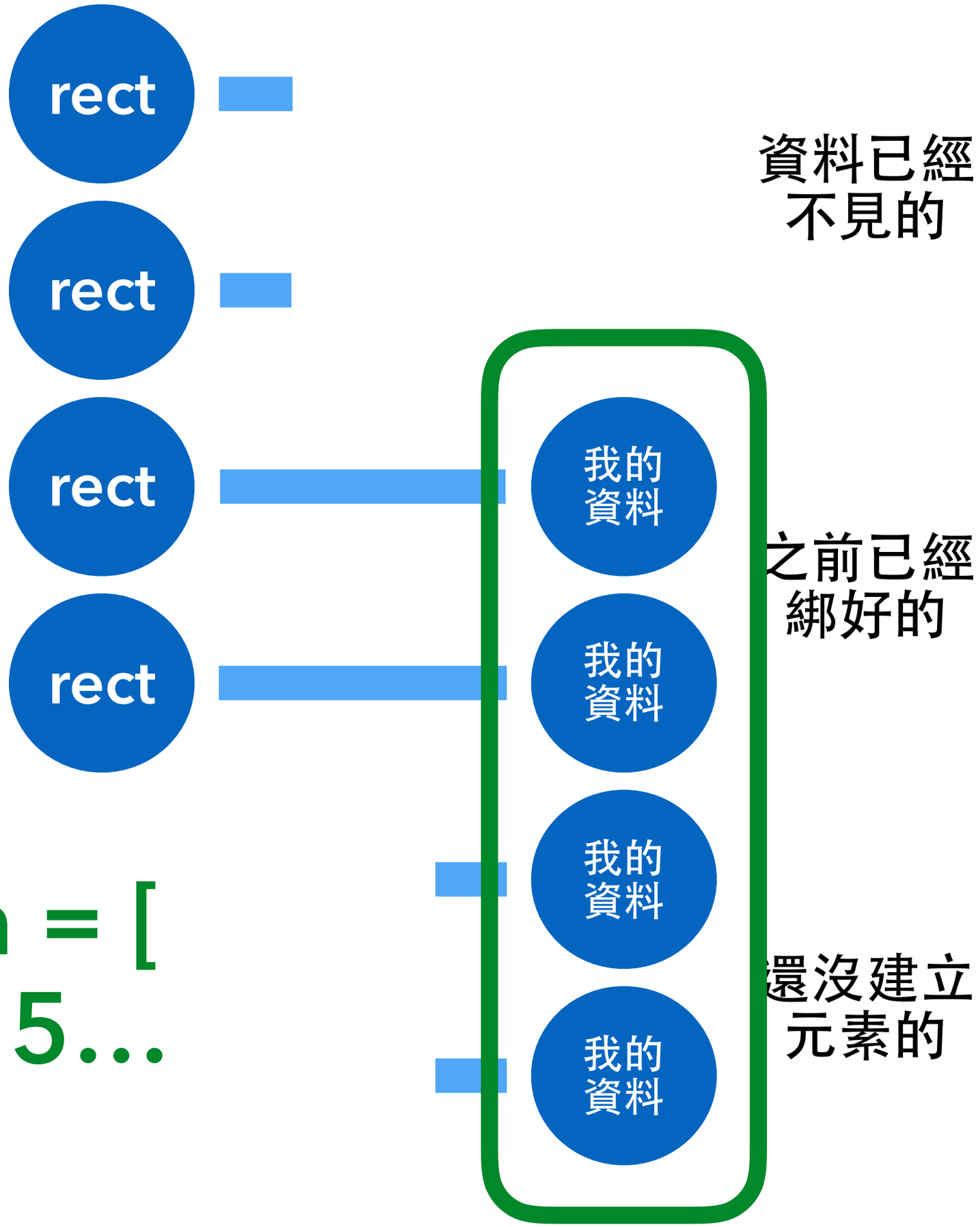
# 我們需要

元素陣列
資料陣列
綁定動作

# 我們想要

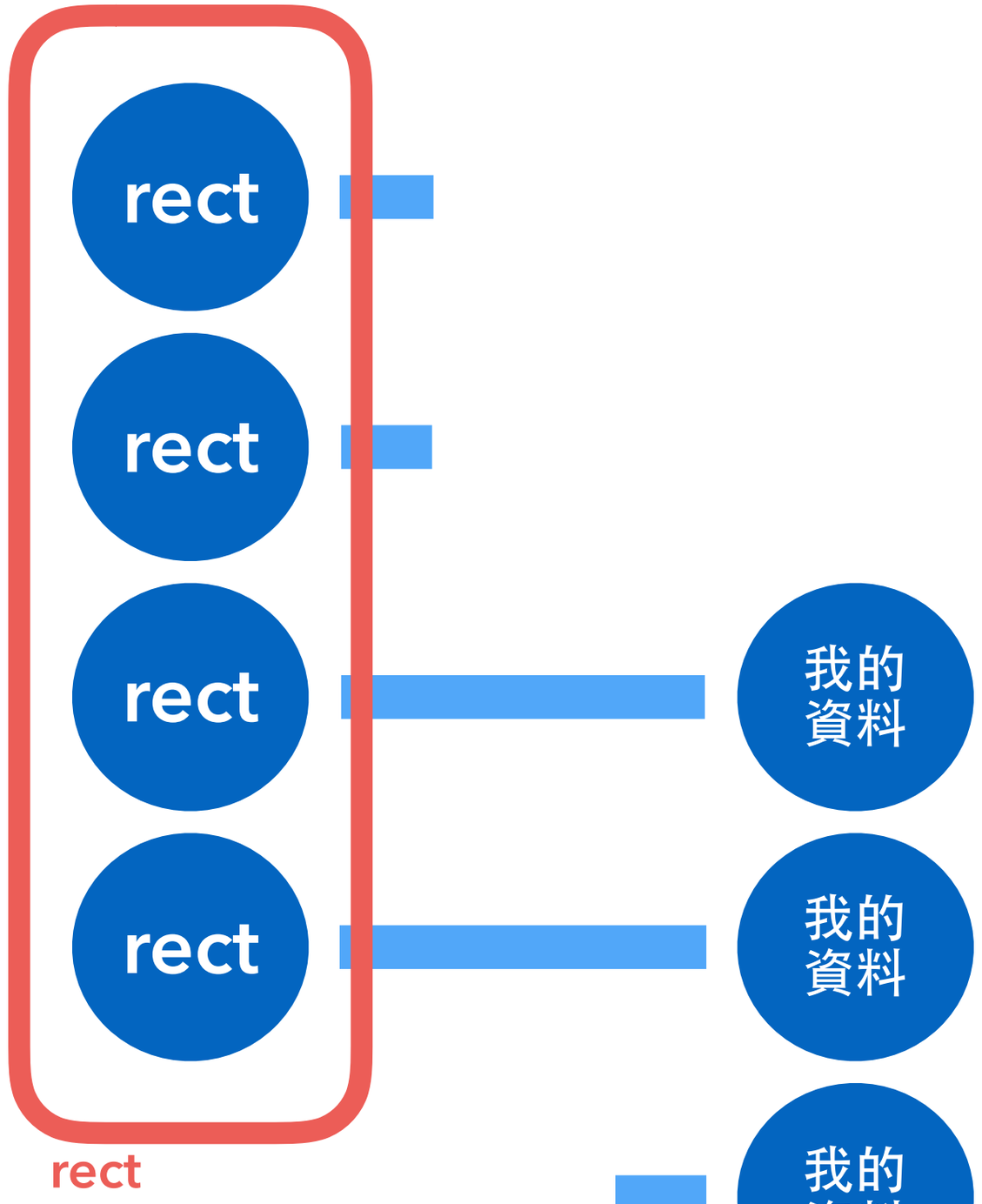
A 集合
B 集合
C 集合






```
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

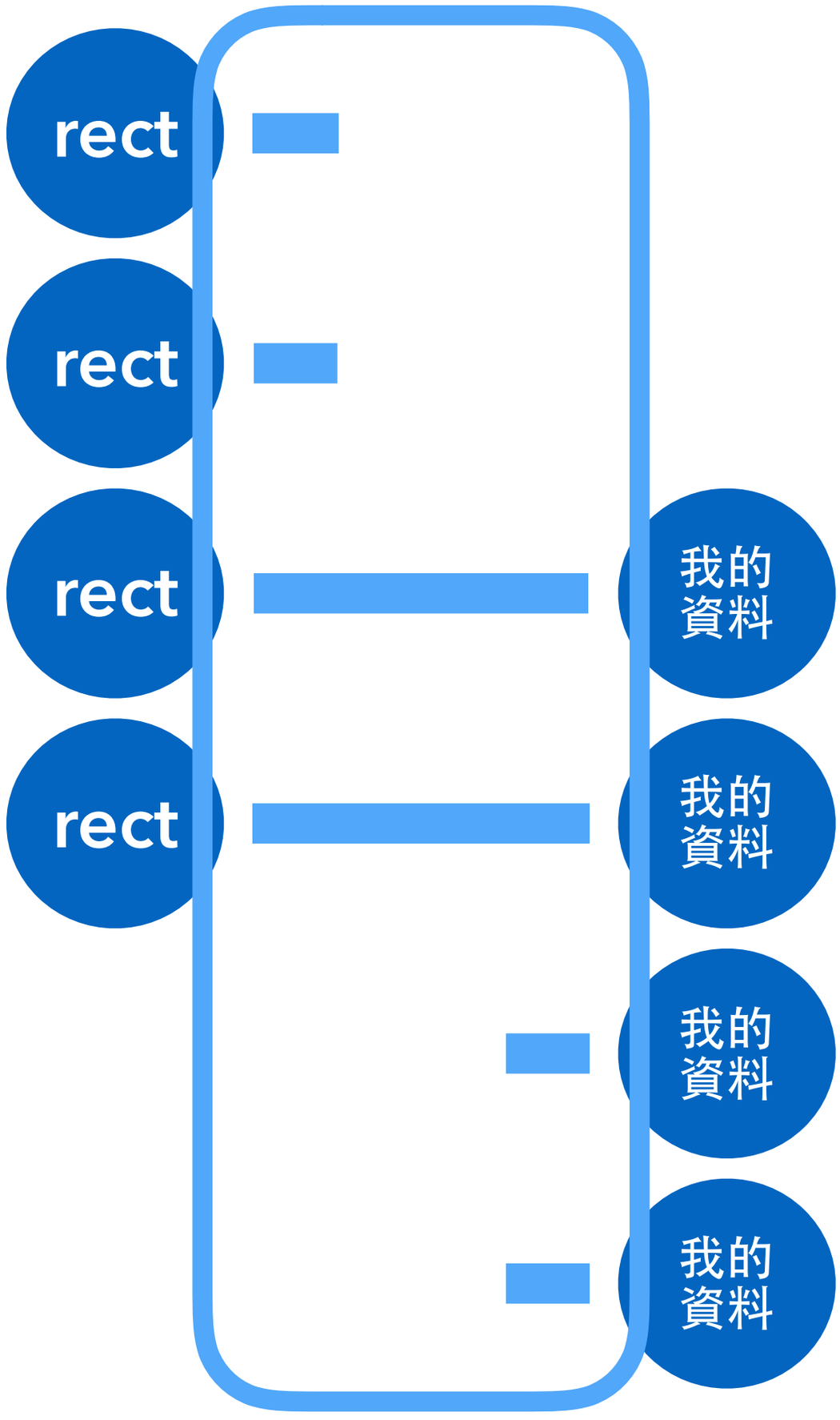
還沒建立元素的

d3.selectAll("rect")

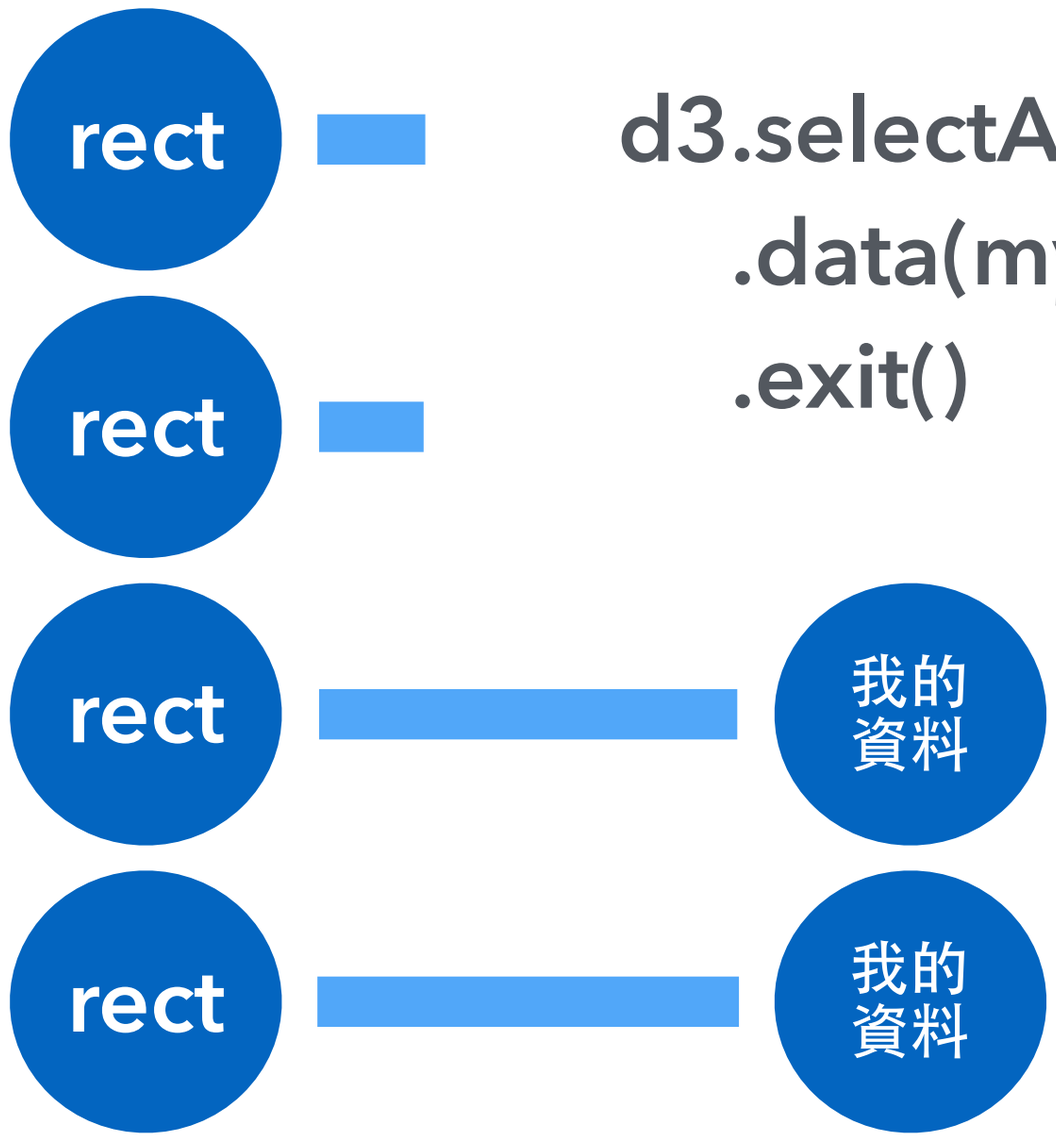
```
var mydata = [
   3, 1, 4, 1, 5...
];
```

d3.selectAll("rect")

.data(mydata)



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

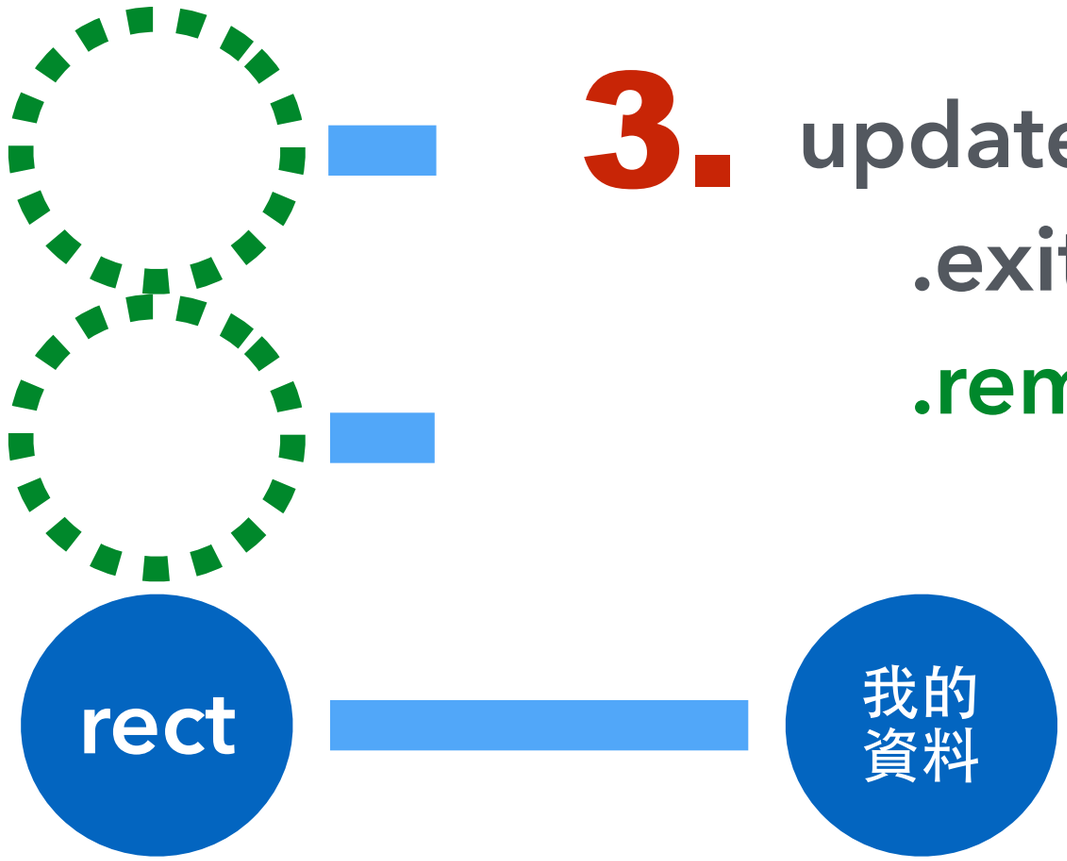



d3.selectAll("rect")
.data(mydata)
.exit()

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

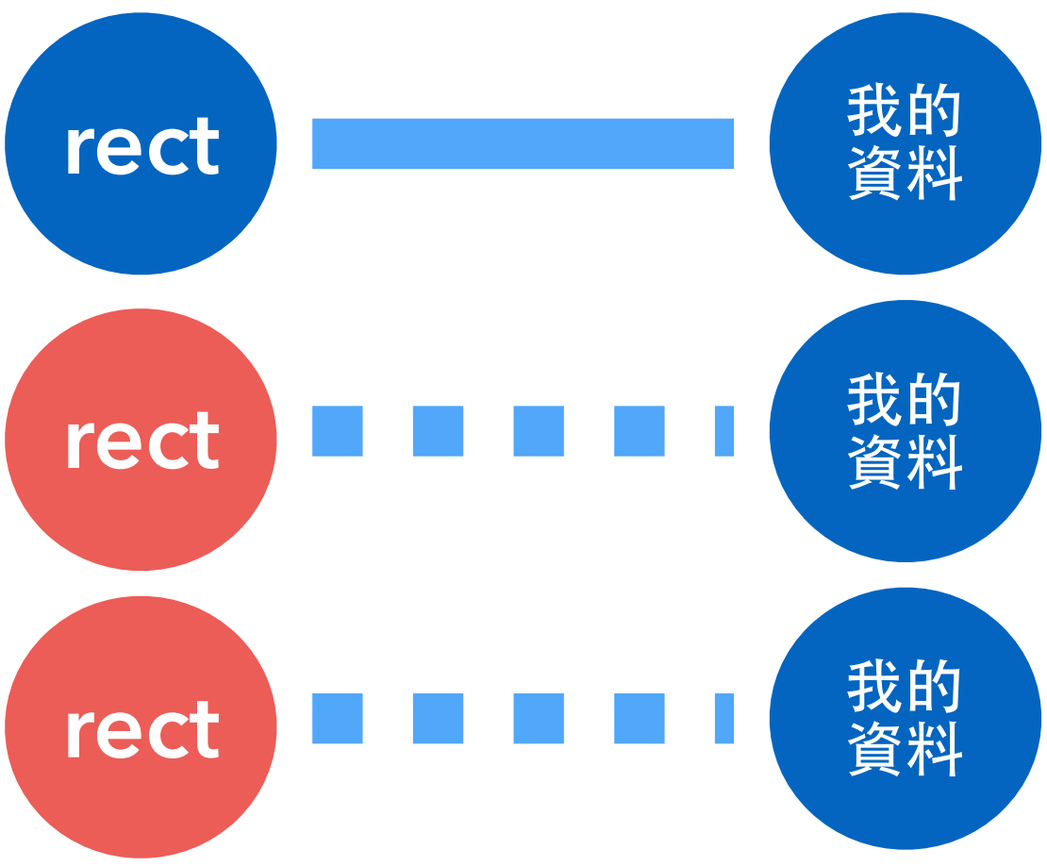
**1.**

```
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")
   .attr({
      width: function(d,i) {
         ....
      },
      ....
   });
```

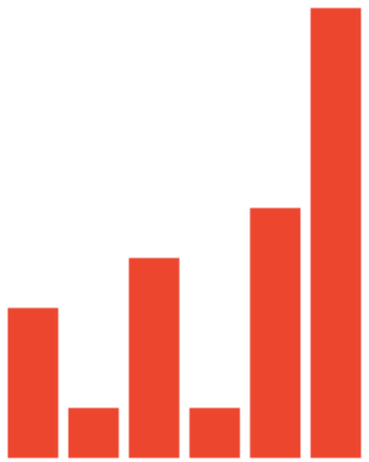
```
var update = d3.select("svg")
  .selectAll("rect").data([3,1,4,1,5,9]);
update.enter().append("rect");
update.exit().remove();
d3.select("svg").selectAll("rect").attr({
  x: 10,
  y: function(d,i) {
    return i * 12;
  },
  width: function(d,i) {
    return d * 10;
  },
  height: 10,
  fill: "red"
});
```

07.html
08.html

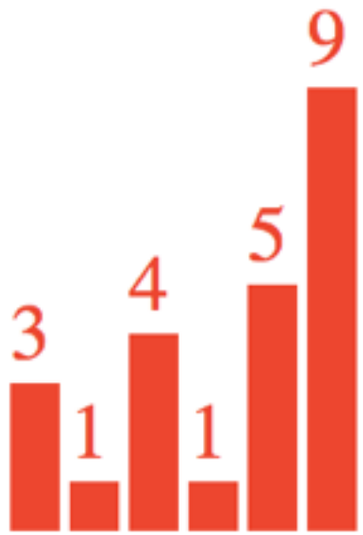
練習 #03 畫一個垂直的長條圖

範例如下:

加分題1



加分題2



# Recap

# 1. Invoke with Functions

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

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

綁定的資料

元素的順序

設定的值

元素本身

# 2. Chaining Method Call

```
d3.select("body")
  .style("width", "100%")
  .text("hello world!")
  .attr( … )
```

# Return Values

```
d3.select(“body”)      - 選取 body …
   .selectAll(“div”)   - 下的所有 div
   .attr(…);           - 並設定屬性
// 傳回值為「所有div」Selection
```

# 3. Invoke with Object

```
d3.select("body")
  .style({
    "width": "100%",
    "height": "100%"
});
```

# D3 Selection

選取 div (只取第一個 )

```
d3.select("body")
```

選取 div (全部)

```
d3.selectAll("body")
```

選取 body 下的所有 div

```
d3.select("body").selectAll("div")
```

# Update Selection

```
d3.select("body")      - 選取 body …
   .selectAll("div")   - 下的所有 div
   .data(…);           - 並綁定資料
```

//傳回值為「所有~~div~~」「Update」Selection

資料已經
不見的

rect

rect

rect

.exit() 給你元素

我的
資料

之前已經
綁好的

rect

我的
資料

我的
資料

我的
資料

還沒建立
元素的

.enter() 給你棒子

# Recap

d3.

select
selectAll

make selection

data binding

data
datum

update selection

enter
exit

attr
style
text / html

alter attributes

append
remove

manipulate node

# 來點動畫吧！

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

# 來點動畫吧！

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

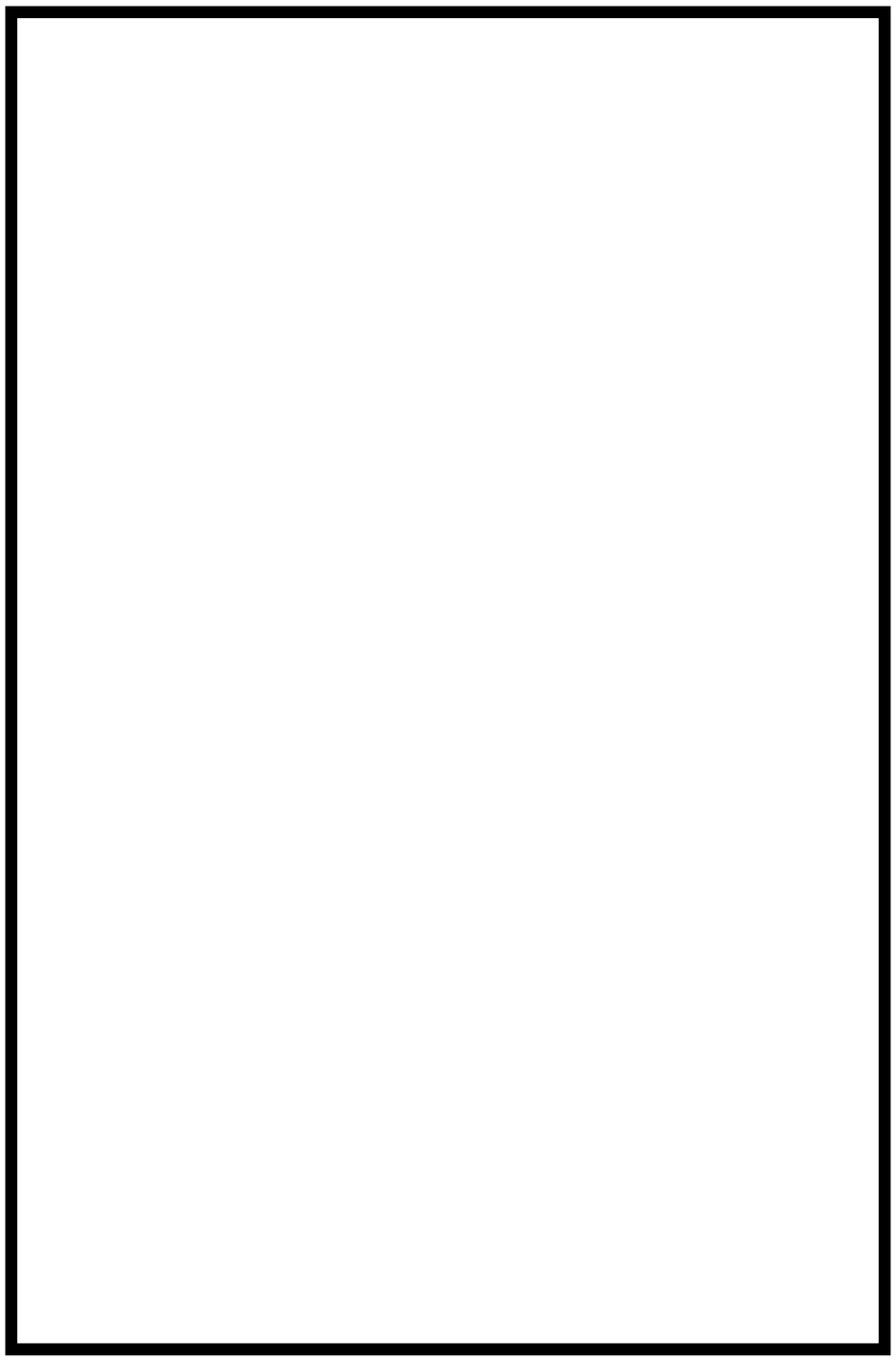
接下來的設定
都要做動畫

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

# 分開寫也行

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

# 練習 #04 畫一個長出來的長條圖



加分題

```
transition()
  .duration(ms)
  .delay(ms)
```

可以改變動畫長度跟延遲,
試著用用看吧！

(單位是千分之一秒)

# Good Pattern

## Binding 與 Styling

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

```
d3.selectAll(“rect”)
   .attr({
      …
   }).style({
      …
   })
```

## 要分開

# Good Pattern

## Binding

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

## Styling

```
function render() {
    d3.selectAll(“rect”)
      .attr({
          …
      }).style({
        …
      })
}
```

包裝起來，可重複使用

# Good Pattern

## Binding

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

## Styling

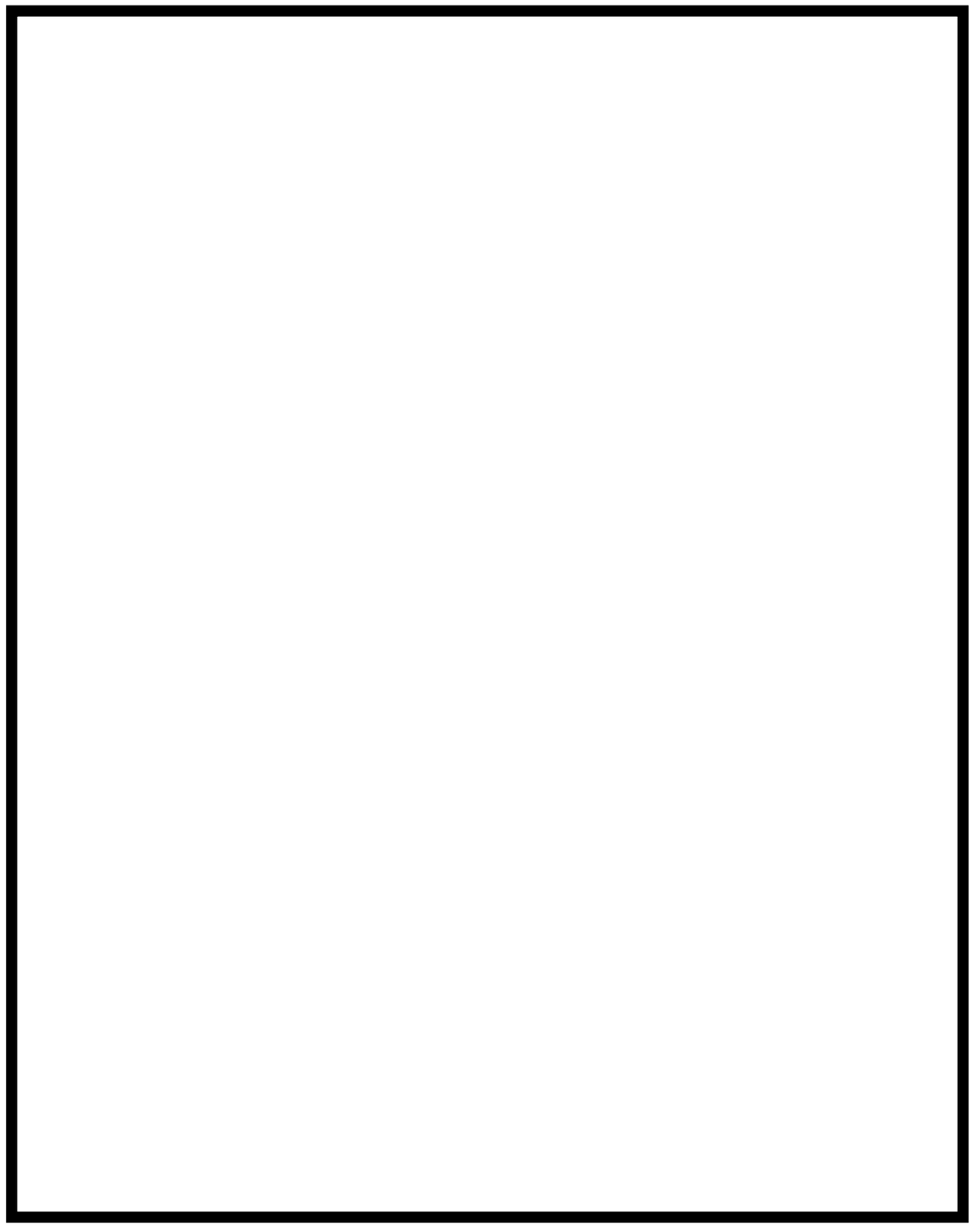
```
function render(delay) {
    d3.selectAll(“rect”)
      .transition()
      .delay(delay)
      .attr({
          …
      }).style({
        …
      })
}
```

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);
```

練習 #05 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

一開始似乎閃了一下黑色
你能不能讓他一開始就是紅色呢？

```
function bind(data) {
  var update = d3.select("svg").
    selectAll("rect").data(data);
  update.enter().append("rect");
  update.exit().remove();
}

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

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

# #05 解答

```
function bind(data) {
  var update = d3.select("svg").
    selectAll("rect").data(data);
  update.enter().append("rect")
    .attr({/* 初始化設定 */});
  update.exit().remove();
}
function render(delay) {
  d3.select("svg").selectAll("rect").attr({
    /* 不想做動畫的設定 */
  });
  d3.select("svg").selectAll("rect")
    .transition().delay(delay).attr({
      x: function(d,i) { return i * 12; },
      y: 10, width: 10,
      height: function(d,i) { return d * 10; },
      fill: "red"
    });
}
bind([3,1,4,1,5,9]);
render(0);
bind([9,5,1,4,1,3]);
render(1000);
```

# #05 加分題

# Quiz

字串參數可以做動畫嗎？

```
d3.selectAll("rect")
  .attr("fill", "red")
  .transition().duration(100)
  .attr("fill", "blue");
```

# 動畫的原理

t = 0

t = 1

A

B

初始值

中間值

目標值

**i(A,B,t)**

# 動畫的原理

## 線性內插

t = 0 **i = 5*(1- t) + 9 * t** t = 1

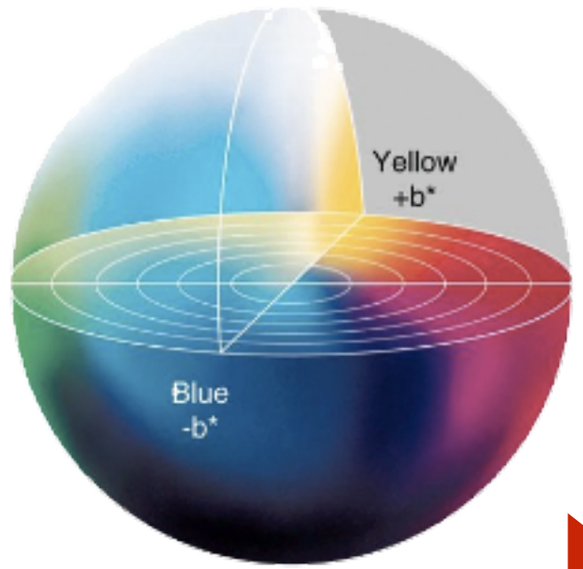
5 9

初始值 中間值 目標值

**i(5,9,t)**

# 動畫的原理

t = 0




t = 1

red → blue

初始值　　中間值　　目標值

i("red","blue",t)

"red" → "blue"

# d3.interpolate("red","blue")

↓

# d3.interpolators

↓

d3.interpolateRGB d3.interpolateHSL

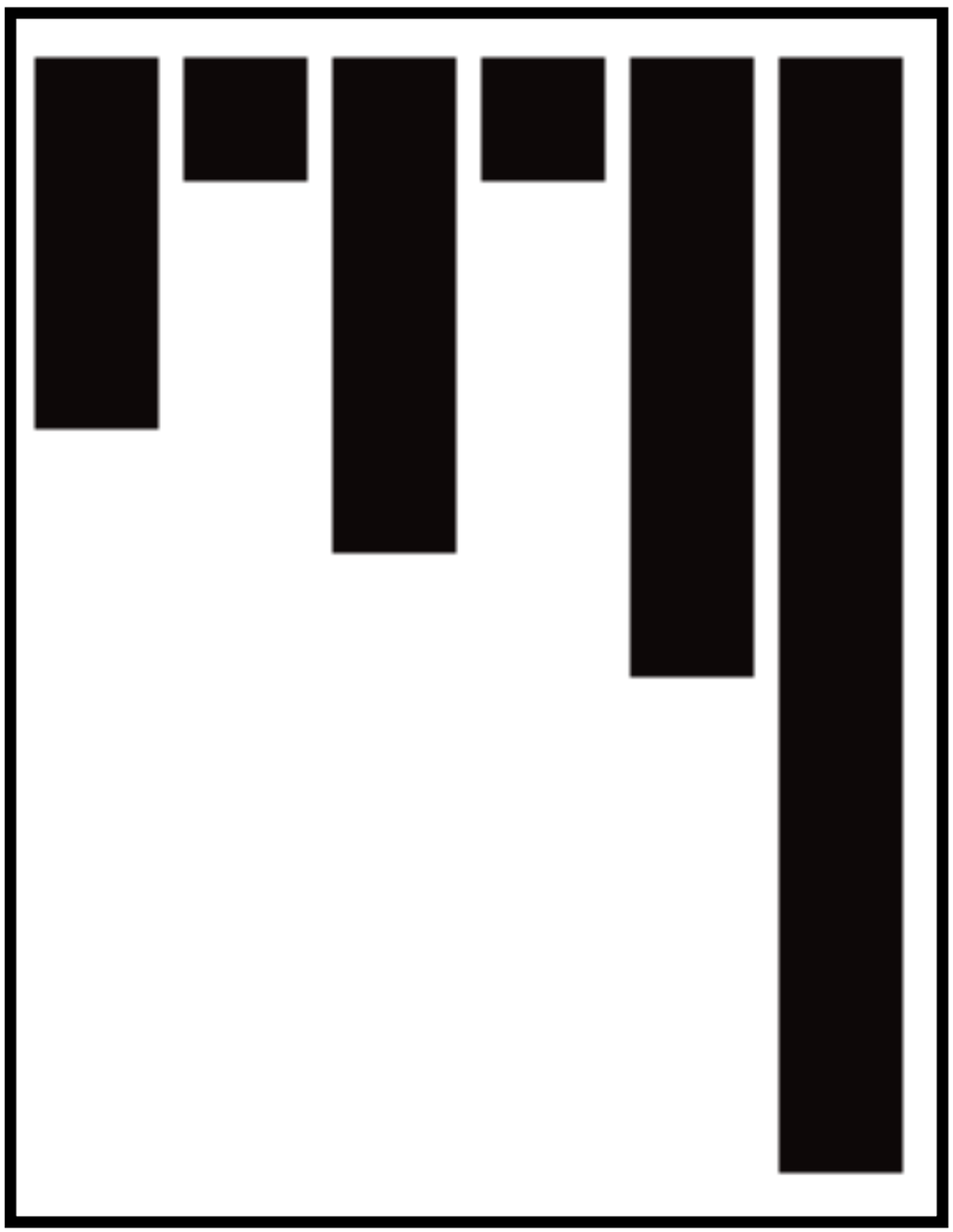
d3.interpolateString d3.interpolateObject

d3.interpolateArray d3.interpolateTransform

# Simple Number Interpolator

```
d3.interpolators.push(function(a, b) {
  if(typeof(a)==“number” &&
     typeof(b)==“number) {
     return function(t) {
       return (1 - t) * a + t * b;
     };
  } else return false;
});
```

練習 #06 畫一個長條圖，從 [3,1,4,1,5,9] 變形到 [9,5,1,4,1,3]
同時依數字設定顏色



## 小秘訣

可以利用 **d3.rgb** 設定顏色

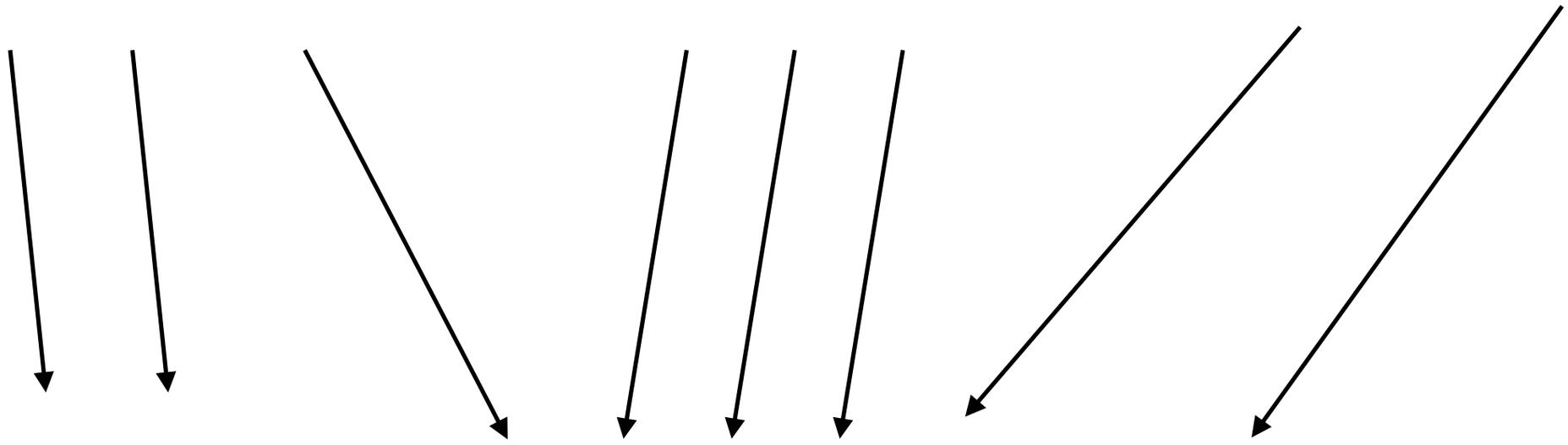
```
function(d,i) {
   return d3
      .rgb(255,d*25,d*25)
      .toString();
}
```

# Quiz

字串參數可以做動畫嗎？

```
d3.selectAll("rect")
    .attr("fill", "Sharkspear")
    .transition().duration(100)
    .attr("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

# SVG Path

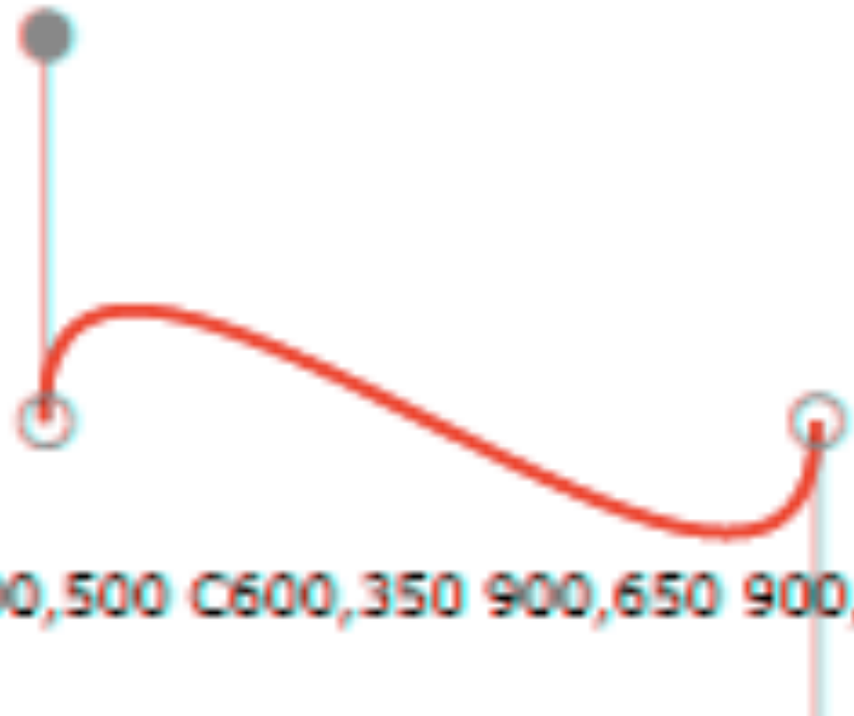
```
<path
fill="red" stroke="black"
d="M10 10L20 20"/>
```

# SVG Path

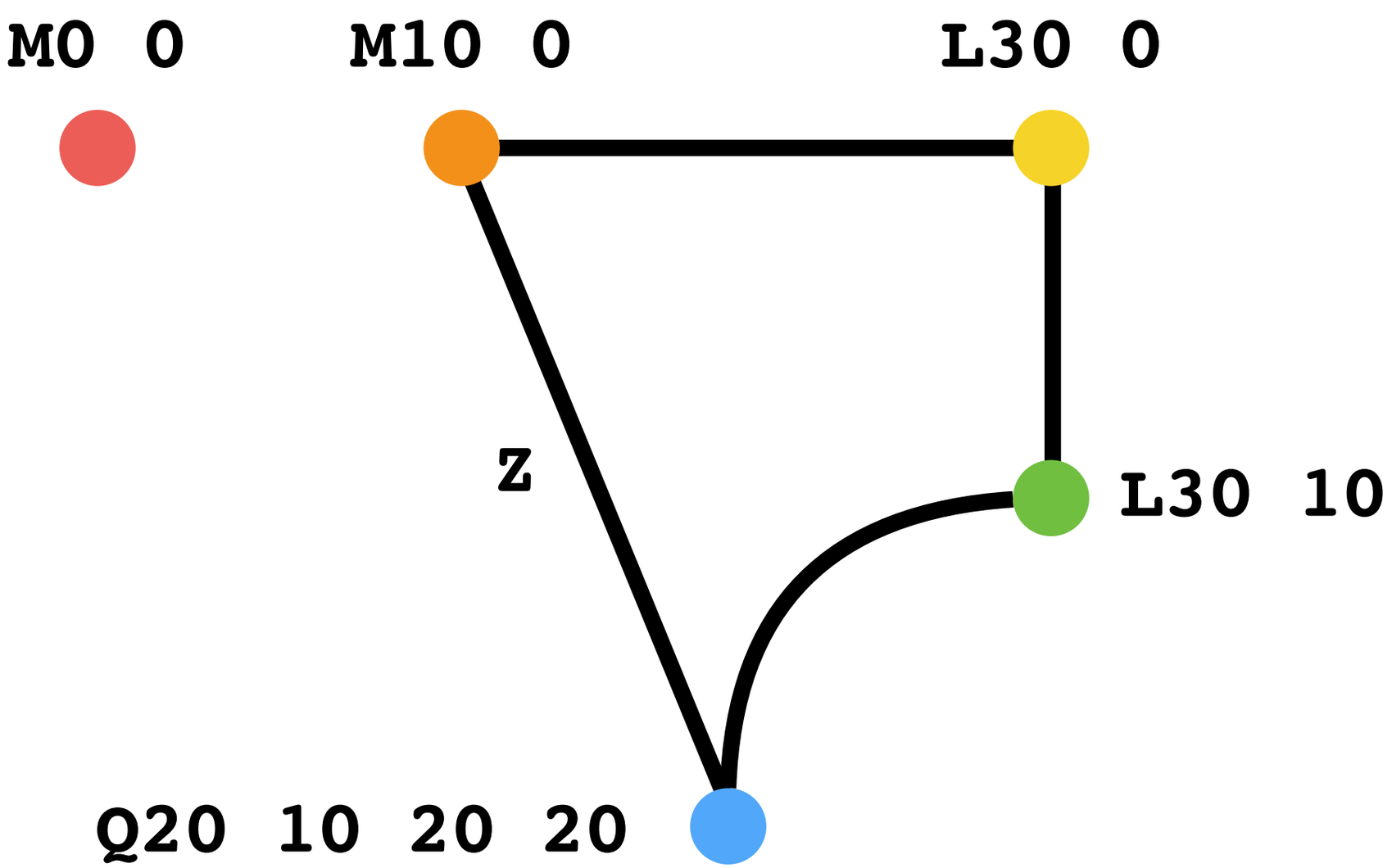
M100,200 C100,100 400,100 400,200

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

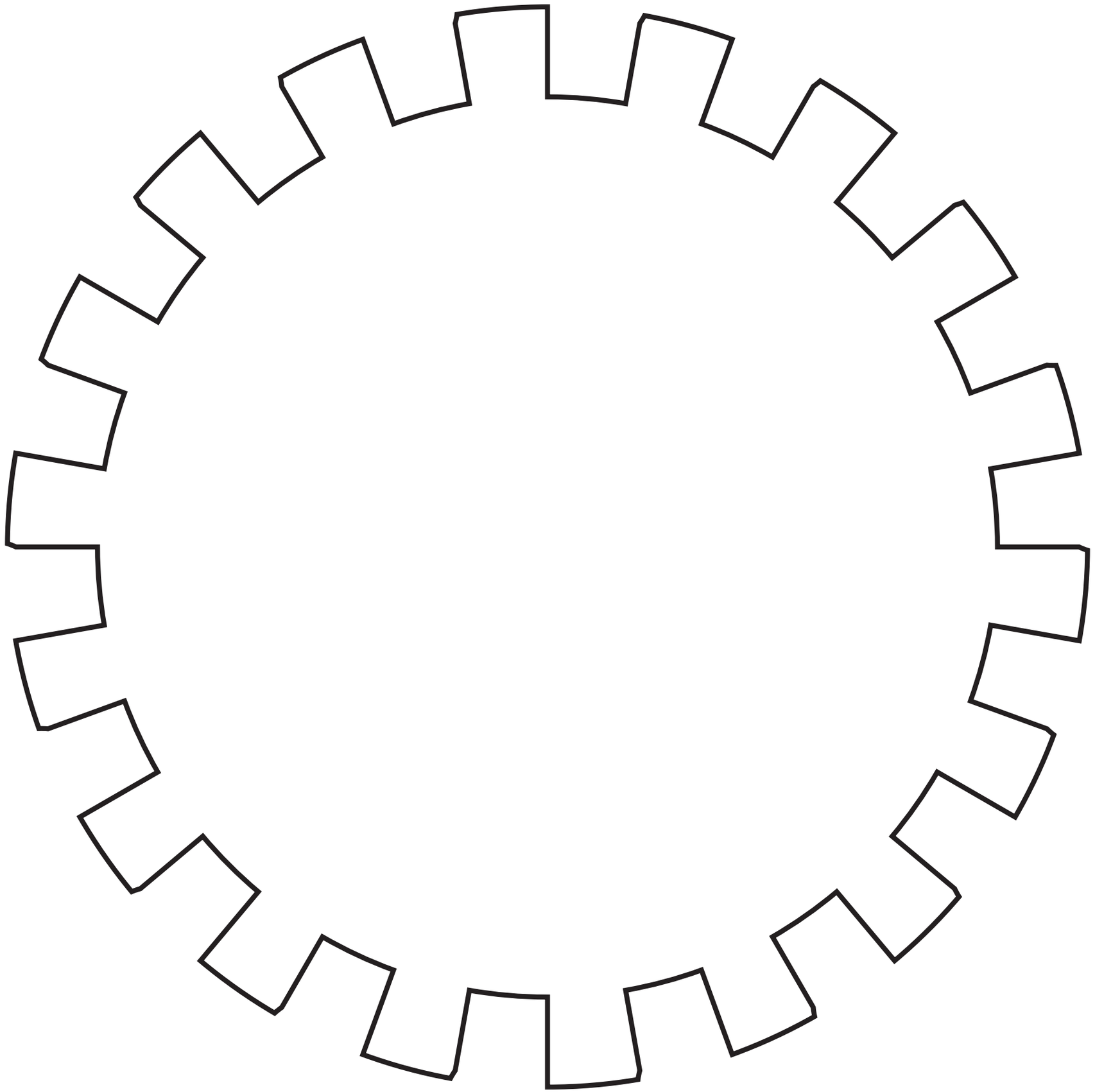
M100,500 C25,400 475,400 400,500



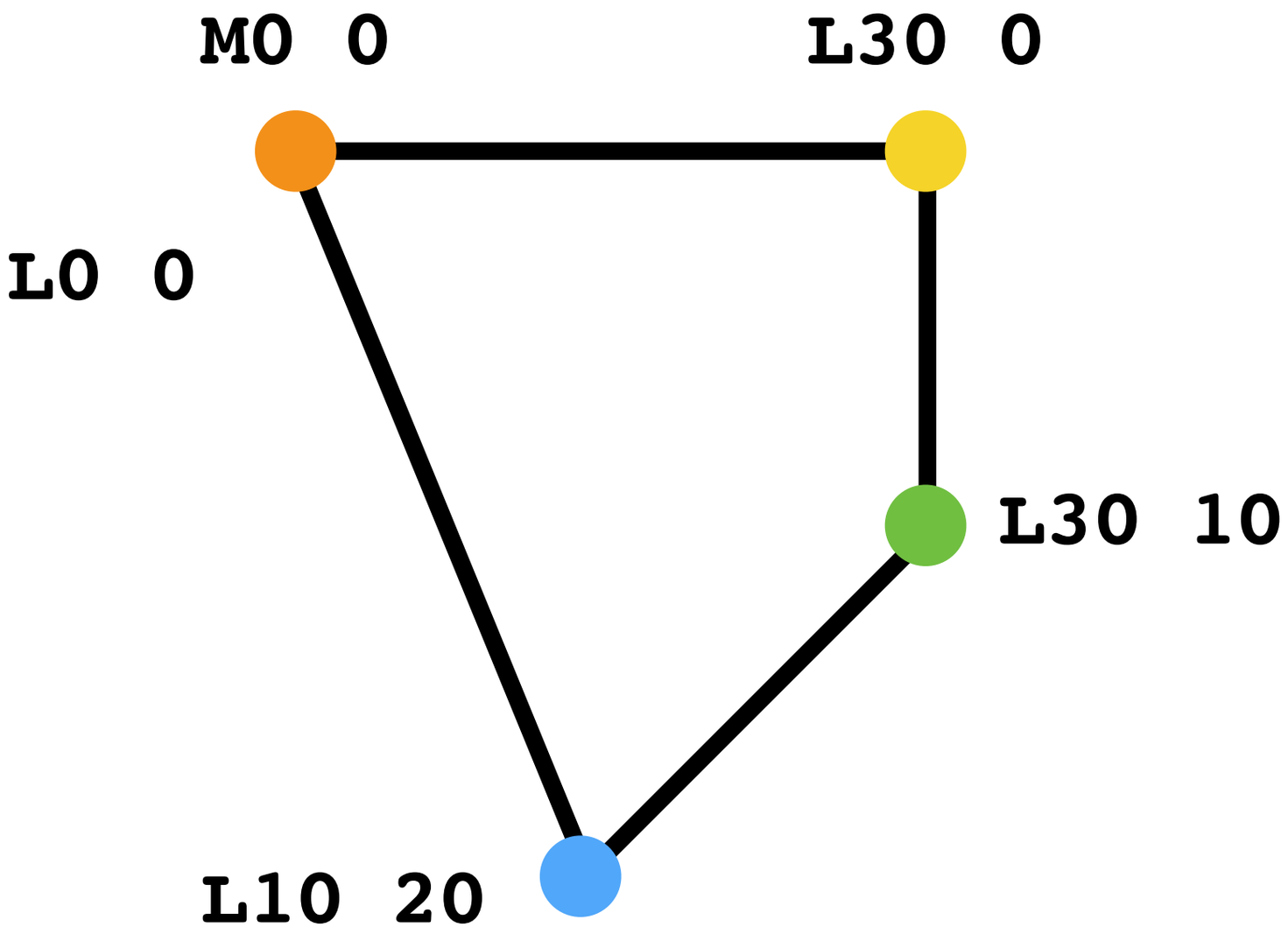
M600,500 C600,350 900,650 900,500



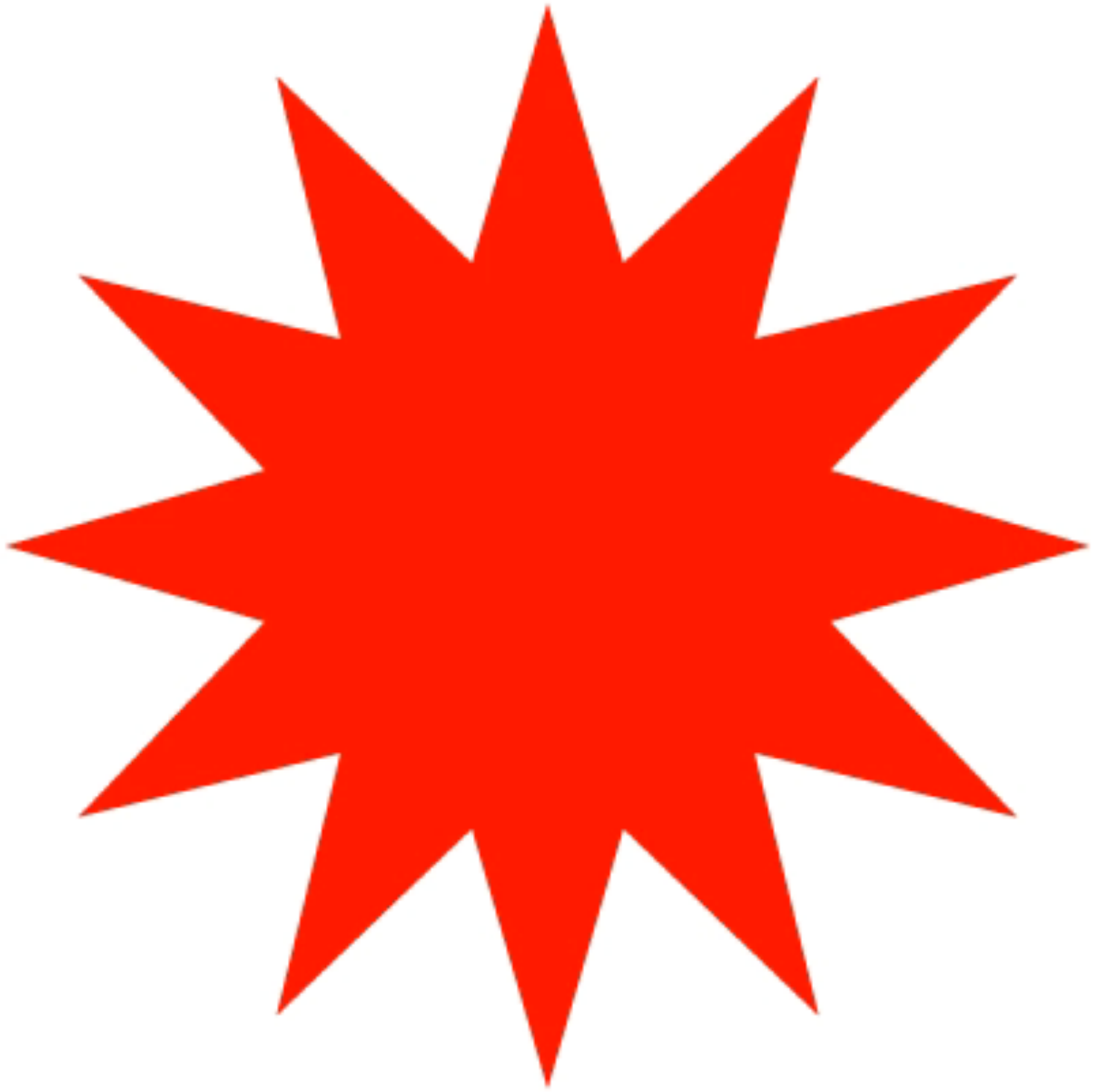

M0 0 M10 0 L30 0 L 30 10 Q20 10 20 20 Z



<?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>



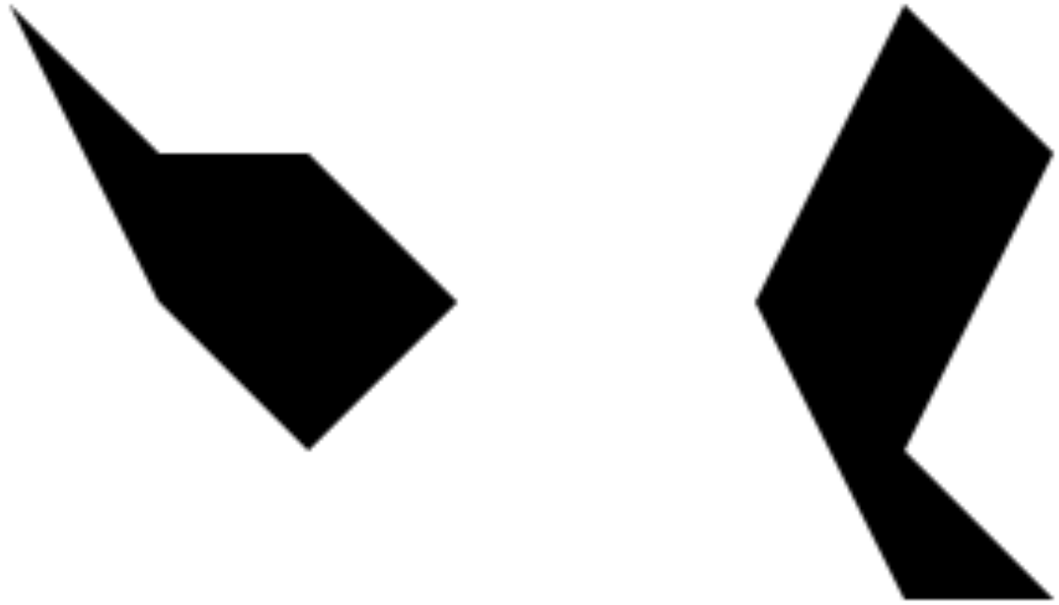

```
<path d="M0 0 L30 0 L 30 10 L10 20 L0 0"/>
```



練習 #07

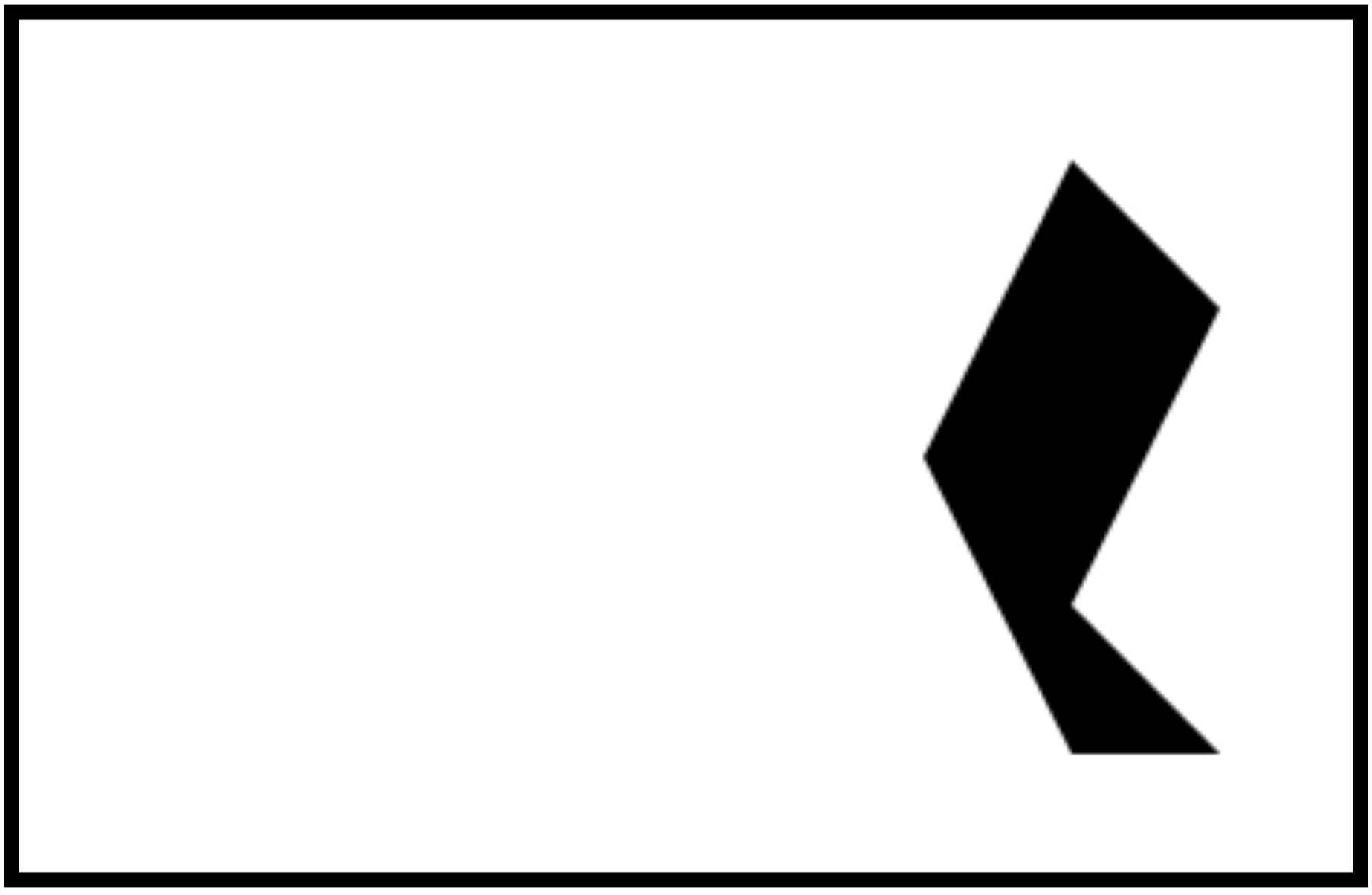
# 利用 SVG Path 隨便畫兩個不一樣的六邊形

```
<svg width="100%" height="100%">
<path d="M10 10 L20 20 L30 20 L40 30 L30 40 L20 30 L10 10"/>
<path d="M70 10 L80 20 L70 40 L80 50 L70 50 L60 30 L70 10"/>
</svg>
```



練習 #08

# 將練習 #07 中的第一個六邊形利用 D3.js 變形到第二個六邊形



提示: 建立一個 path, 先設定為第一個六邊形，再用 transition 設定為第二個六邊形

# use d3.svg.area

```
builder = d3.svg.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]);
```

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

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

# 練習 #09　製作動態區域圖，從 [3,1,4,1,5,9] 變到 [9,5,1,4,1,3]

## 加分題

讓動畫不斷重覆

# Quiz

多個 transition 會怎樣？

```
d3.selectAll(“rect”)
    .transition().duration(1000)
        .attr(“fill”, “#f00”)

d3.selectAll(“rect”)
  .transition().duration(1000)
      .attr(“fill”, “#0f0”);
```

# Chaining Transition

```
d3.selectAll(“rect”)
    .transition().duration(1000)
        .attr(“fill”, “#f00”)
.transition().duration(1000)
        .attr(“fill”, “#0f0”)
```

# Chaining Transition

```
d3.selectAll(“rect”)
    .transition()
        .delay(1000)
        .duration(1000)
            .attr(“fill”, “#f00”)
.transition()
        .delay(2000)
        .duration(1000)
            .attr(“fill”, “#0f0”)
```

<- 將覆蓋自動計算的 delay

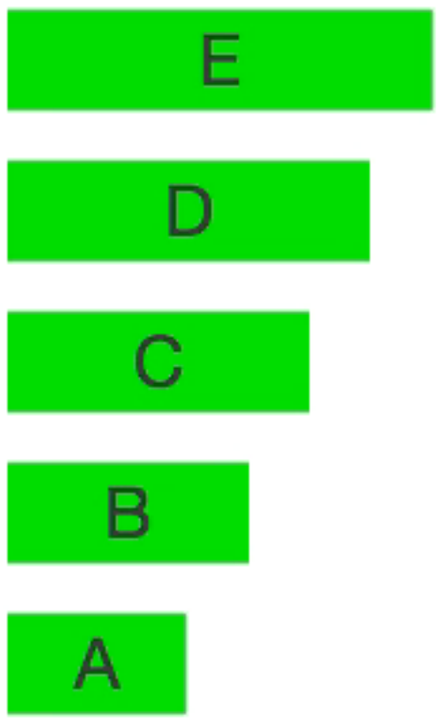
# Chaining Transition

```
d3.selectAll(“rect”)
    .transition().duration(1000)
        .attr(“fill”, “#f00”)
   .transition().duration(1000)
        .attr(“fill”, “#0f0”)
   .transition().duration(1000)
        .attr(“fill”, “#0f0”)

     ……..

   .transition().duration(1000)
        .attr(“fill”, “#0f0”)
   .transition().duration(1000)
        .attr(“fill”, “#0f0”)
```

# Chaining Transition

```
function doTransition(node,n) {
    return node.transition.duration(1000)
        .attr(“fill, newValue)
}
```

```
node = doTransition(
  d3.select(“body”).selectAll(“div”)
);
node = doTransition(node);

for(var i = 1; i < 100; i++) {
    node = doTransition(node);
    /* update newValue */
}
```




```
function doTransition() {
  node
    .transition()
    .duration(1000)
    .style( … );
}

setInterval(function() {
  render();
}, 1000);
```

# 利用 end 事件串連動畫

```
.transition()
  .each("end", callback);

function callback() {
  var node = d3.select(this);
  (function repeat() {
    node.transition().attr(….);
  })();
}
```

# Parallel Transition

```
d3.selectAll(“rect”)
    .transition(“name1”).duration(1000)
        .attr(“fill”, “#f00”)
```

```
d3.selectAll(“rect”)
  .transition(“name2”).duration(1000)
      .attr(“stroke”, “#0f0”)
```

# 動畫的原理


初始值 **A**　　內插函式 **i**

目標值 **B**　　時間函式 **t**

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

# Customize Tweening

## transition.ease(type)
## d3.ease(type)

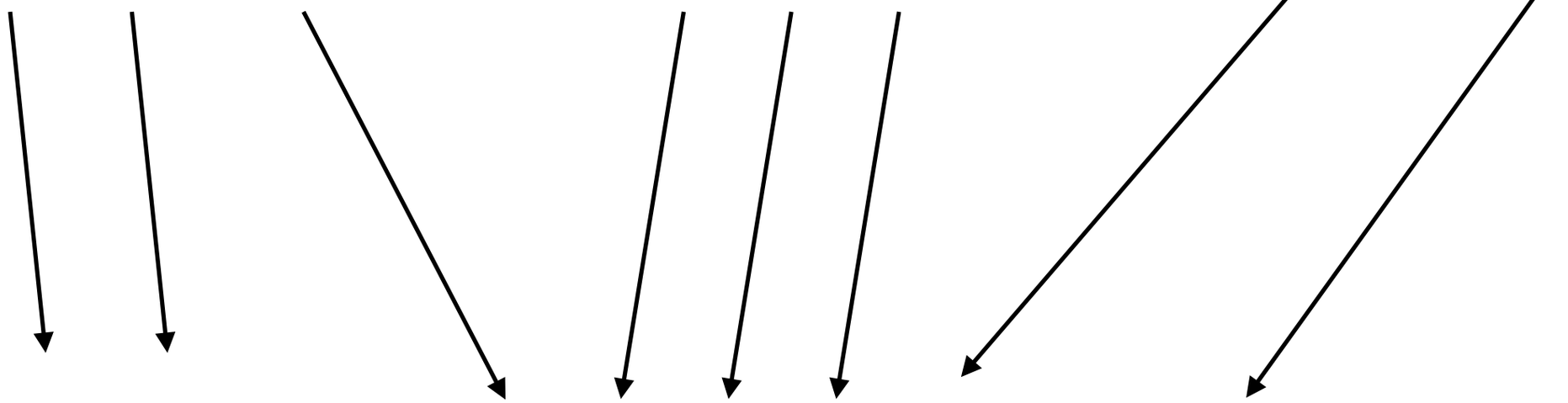
| | | | |
|---|---|---|---|
| linear | poly(k) | | in |
| quad | sin | + | out |
| cubic | exp | | in-out |
| bounce | circle | | out-in |

```
…transition()
        .ease(“bounce-out”) …
```

# 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)
      );
    };
  });
```

# Recap

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

# 資料到座標需要轉換

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

# 資料到座標需要轉換

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

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

d3.selectAll("circle").attr({
  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").attr({
  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)

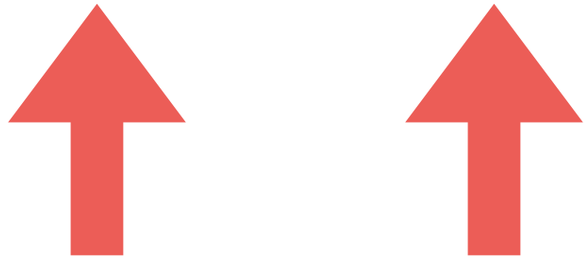
# D3 Scales

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

```
scale(0.5);
```

# D3 Scales

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



**start  end**

# D3 Scales

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

scale(4);
```

練習 #10 將 1 ~ 9 的資料對應到 100 ~ 800 的座標
然後求資料 "4" 對應的座標值

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

scale(4);
```

加分題： 若對應到 800 ~ 100, 答案又是多少呢？

# D3 Scales

## Qualitative Scale

continuous domain

d3.scale.linear
d3.scale.pow
d3.scale.log
d3.scale.threshold
d3.scale.quantize
d3.scale.quantile

## Ordinal Scale

discrete domain

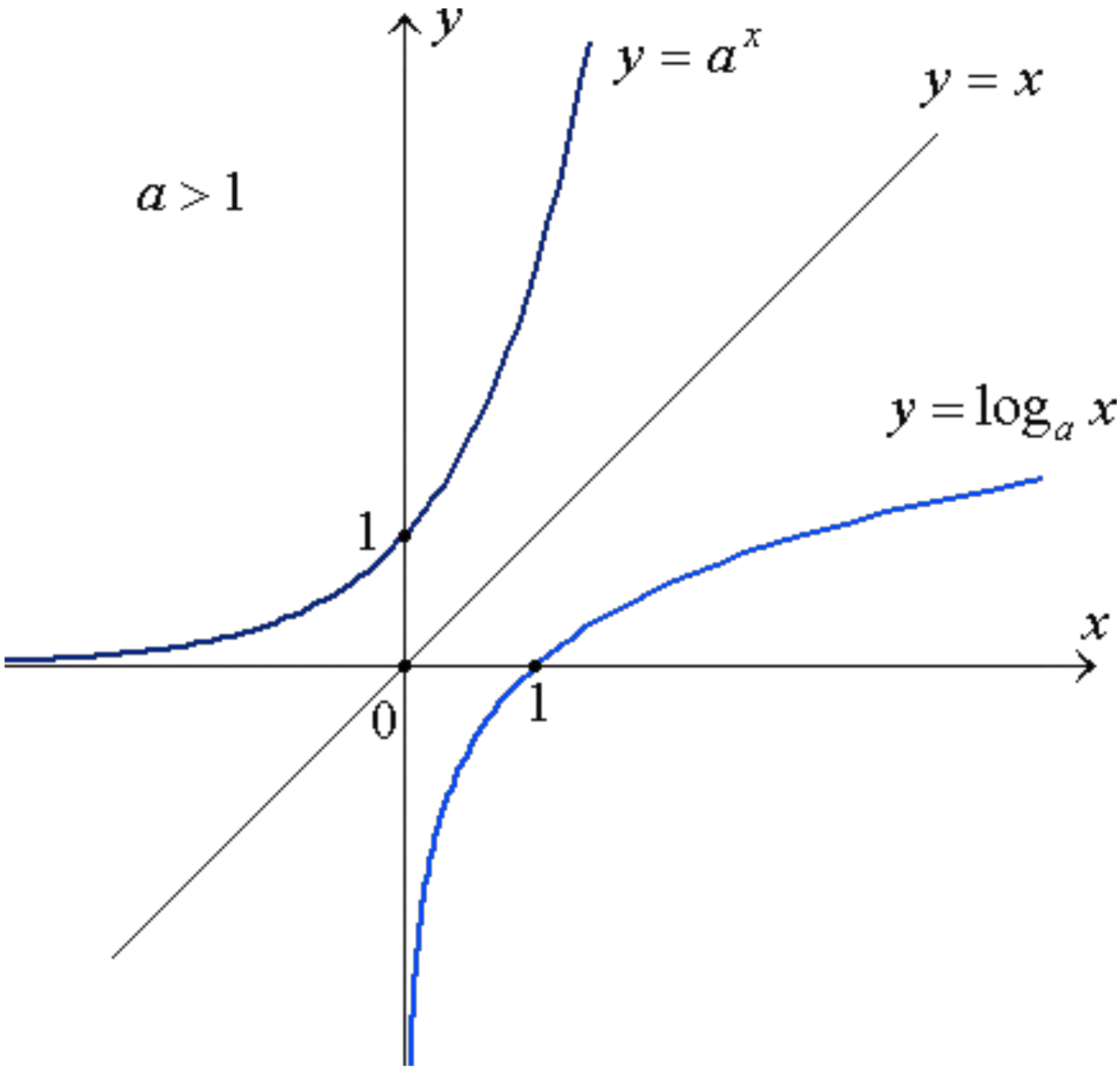
d3.scale.ordinal
d3.scale.category10
d3.scale.category20
d3.scale.category20b
d3.scale.category20c

## Time Scale

linear scale for time

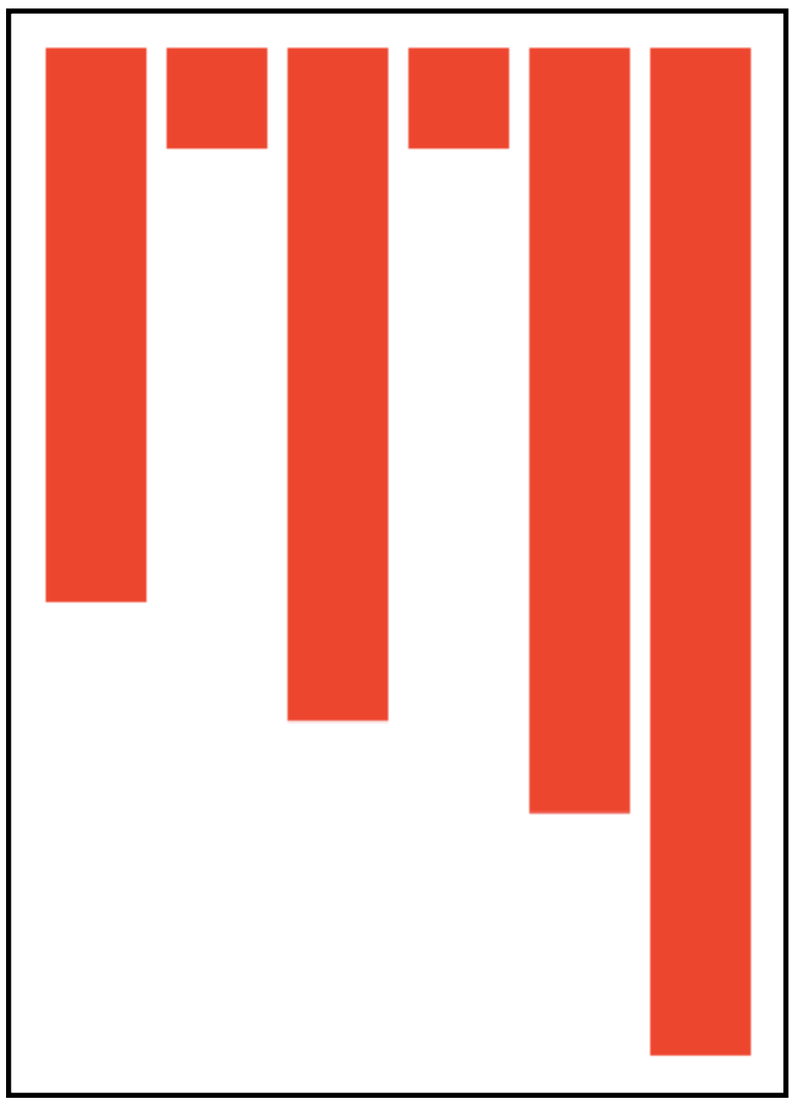
d3.time.scale

```
xscale = d3.scale.log()
   .domain([1, 10])
   .range([0, 790]);

yscale = d3.scale.log()
   .domain([1, 10])
   .range([590, 0]);
```


$a > 1$
$y$
$y = a^x$
$y = x$
$y = \log_a x$
$x$
1
0
1

# 練習#03 的垂直的長條圖

# 練習 #11 將練習 #03 的長條圖改用 Linear Scale 表現



提示:

利用 d3.scale.linear 建立一個轉換函式

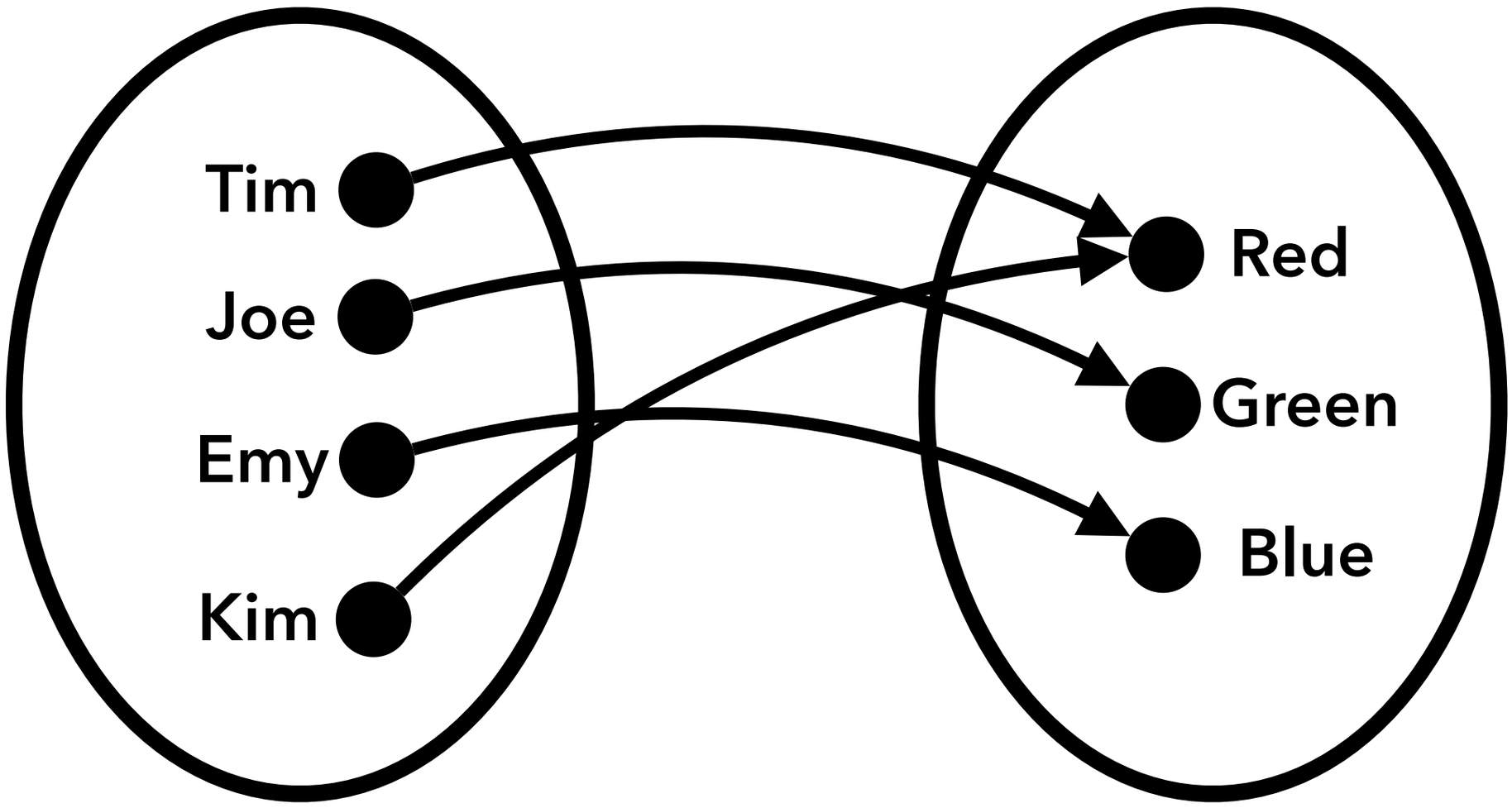
利用轉換函式計算高度的傳回值

加分題

玩玩看 d3.scale.pow 、 d3.scale.log

再加個動畫吧！

# d3.scale.ordinal



```
var scale = d3.scale.ordinal()
    .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.scale.category20()

## d3 categorical colors & Google colors

d3 category10

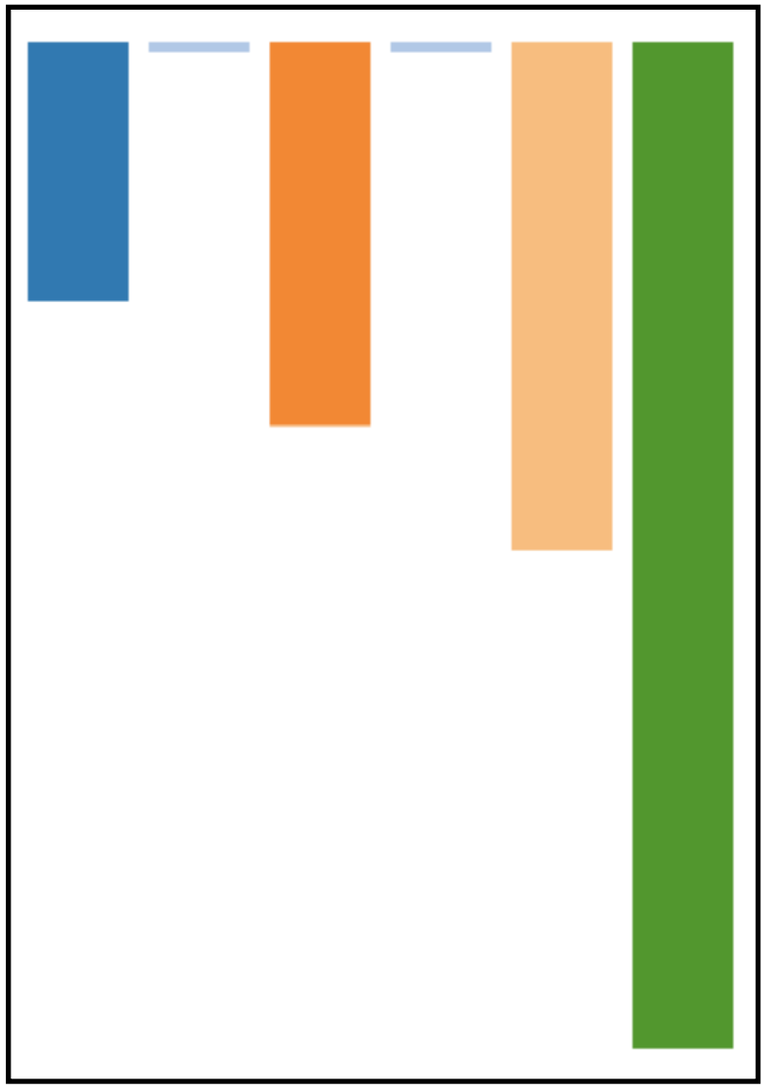
google 10c

d3 category20

d3 category20b

d3 category20c

# 練習 #12　將練習 #1 的長條圖利用 Category20 上色



加分題
自己選一些色盤來試試看吧！

# 有用的知識

## d3.scale 是靠 d3.interpolators 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.scale.linear()
    .domain([1, 9])
    .range([10, 790]);
```

placeholder?

```
[1,45,132,200, 4943, … ]
```

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

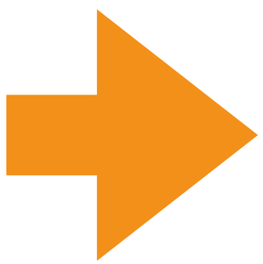
# d3.extent(Array)

```
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()
   };
});
```

練習 #13

# 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 ~ 9

分成三類

```
var data = d3.range(20).map(function(it) {
  return {
    value: parseInt(Math.random() * 9 + 1),
    category: "ABC".charAt(
      parseInt(Math.random() * 3)
    )
  };
});
```

# #13 解答

```
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]
```

想知道各類
有哪些資料

想知道有
哪幾類

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

# d3.nest()

group array into hierarchy structure

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

把 a 轉成 key/value 陣列

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

建立 nest 物件

從 cat 取 key

把 a 轉成「物件」

```
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, d3.map).keys()
```

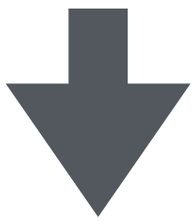
```
[ "A", "B", "C" ]
```

# 練習 #14 利用練習 #13 產生的資料與 d3.nest

求出資料的 category 陣列

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

**d3.nest()**



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

A = [85, 47, 77, 76, 46, ...]

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

converted = {
   "4": [46, 47, …],
   "7": [76, 77, …],
   "8": [85, …]
}
```

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


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

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

```
converted = [
   [{value: 1,…}, {value: 2,…}],
   [{value: 2,…}, {value: 3,…}],
   [{value: 3,…}, {value: 4,…}],
   [{value: 4,…}, {value: 5,…}],
]
```

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

```
d3.pairs(a) => [
  [{value: 1,…}, {value: 2,…}],
  [{value: 2,…}, {value: 3,…}],
  [{value: 3,…}, {value: 4,…}],
  [{value: 4,…}, {value: 5,…}],
]
```

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]

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

配對

雜湊

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

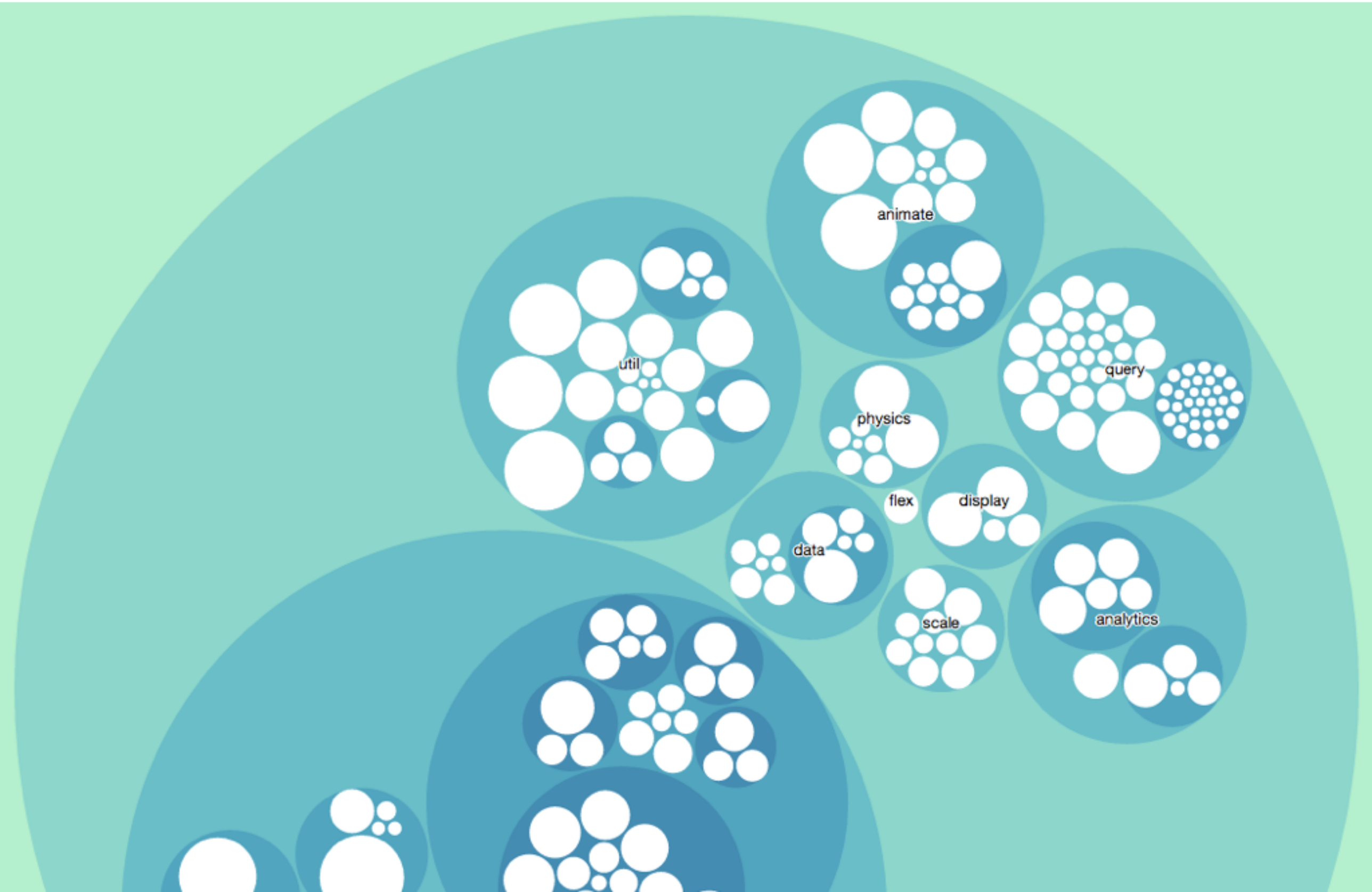
階層陣列

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

雜湊陣列

```
{ 1: [Obj],
  2: [Obj, Obj],
  3: [Obj, Obj, Obj]}
```

# Circular Treemap



# d3.layout.pack

```
d3.layout.pack()
    .size([width, height])
    .children( … )
    .nodes(data)
```

# data structure for Pack Layout

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

```
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);
```

```
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"},
  ]}
]
```

```
entries = d3.nest()
  .key(function(it) {it.cat; })
  .entries(a);
root = {values: entries};

d3.layout.pack()
.size([800, 400])
.children(function(it) {
  return it.values;
})
.nodes(root)
```

注意！
傳回值為資料
而非 pack()

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

```
d3.select("svg")
   .selectAll("circle")
   .data(a)
   .enter().append("circle")
     .attr({
     cx: function(it) { return it.x; },
     cy: function(it) { return it.y; },
     r:  function(it) { return it.r; },
     fill: "none",
     stroke: "#000"
   });
```

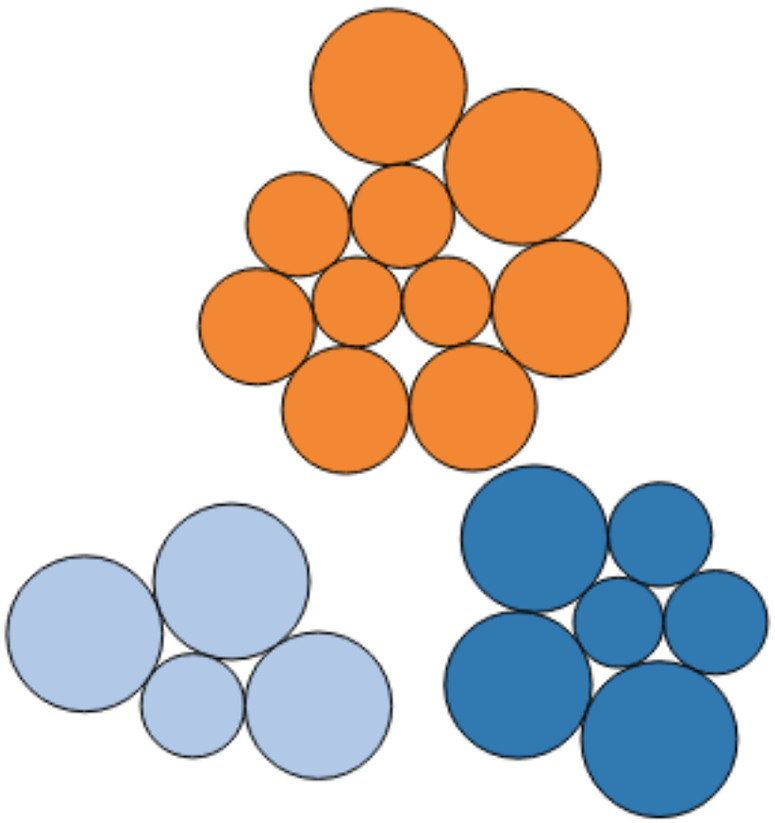
# 製作步驟

- 準備資料 -  d3.range + d3.nest()
- 建立 pack layout 物件
- 利用 pack layout 更新資料
- 利用 selectAll + Data Binding 建立元素並繪製

# 實作練習 #15

# 利用 #13 的結果做出泡泡圖

## 並利用 d3.scale.category20 著色



**[ hint ]**

利用下列函式：

- d3.range
- d3.layout.pack
- d3.nest
- d3.scale.category20

```
var data = d3.range(20).map(function(it) {
  return {
    value: parseInt(Math.random() * 9 + 1),
    category: "ABC".charAt(parseInt(Math.random() * 3))
  };
});
var entries = d3.nest().key(function(it) {
  return it.category; }).entries(data,d3.map);
var root = {values: entries};
var nodes = d3.layout.pack()
   .size([800, 400])
   .children(function(it) {
     return it.values;
   })
   .nodes(root)
var color = d3.scale.category20();

d3.select("svg")
  .selectAll("circle")
  .data(data)
  .enter().append("circle")
    .attr({
      cx: function(it) { return it.x; },
      cy: function(it) { return it.y; },
      r:  function(it) { return it.r; },
      fill: function(d,i) { return color(d.category); },
      stroke: "#000"
    });
```

# #15 解答

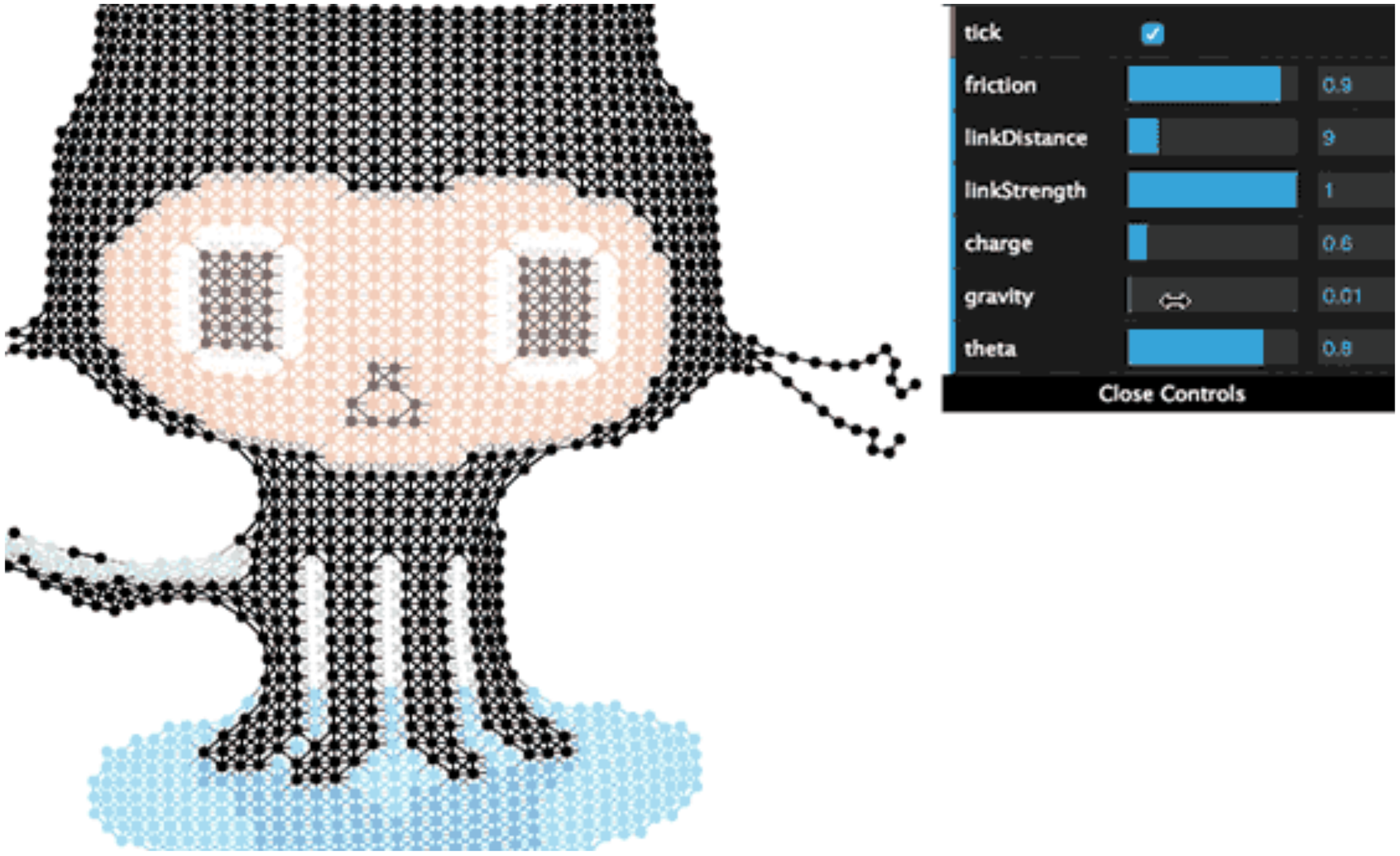
# d3.layout.force

```
d3.layout.force()
    .size([width, height])
    .nodes(data)
```

# d3.layout.force

```
d3.layout.force()
    .size([width, height])
    .friction()  – 弾力
    .charge()    – 斥力 (負値)
    .gravity()   – 引力
    .nodes(data)
    .start()
```

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



```
[
  {…},
  {
    cat: “A”,
    value: 0.447
    r: 5
    x: 248.867
    y: 414.396
    px: 354.939
    py: 303.569
    weight: 0,
    index: 1,
  }, {…}, …
]
```

# 利用 tick 更新畫面

```
force.on("tick",
  function() {
    ......
  });
```

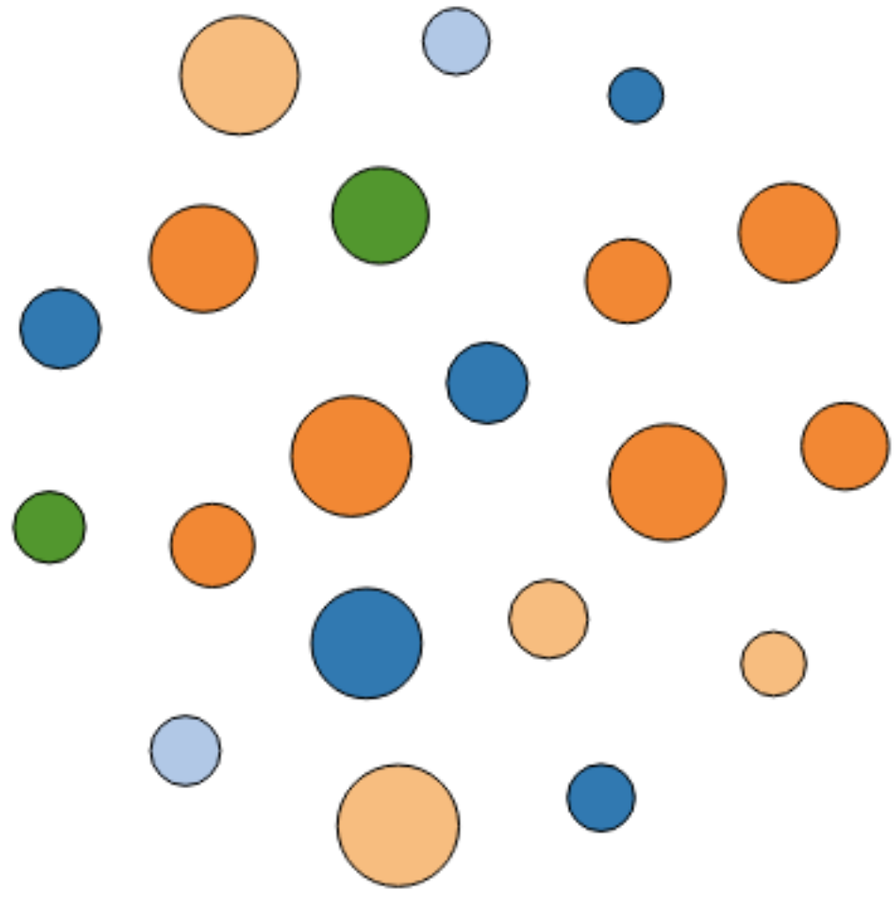
# 製作步驟

- 準備資料 -  d3.range + d3.nest()
- 建立 force layout 物件
- 利用 force layout 更新資料
- 利用 selectAll + Data Binding 建立元素
- 利用 tick event 做繪製

# 實作練習 #16

# 利用 #13 的結果做出彈力圖

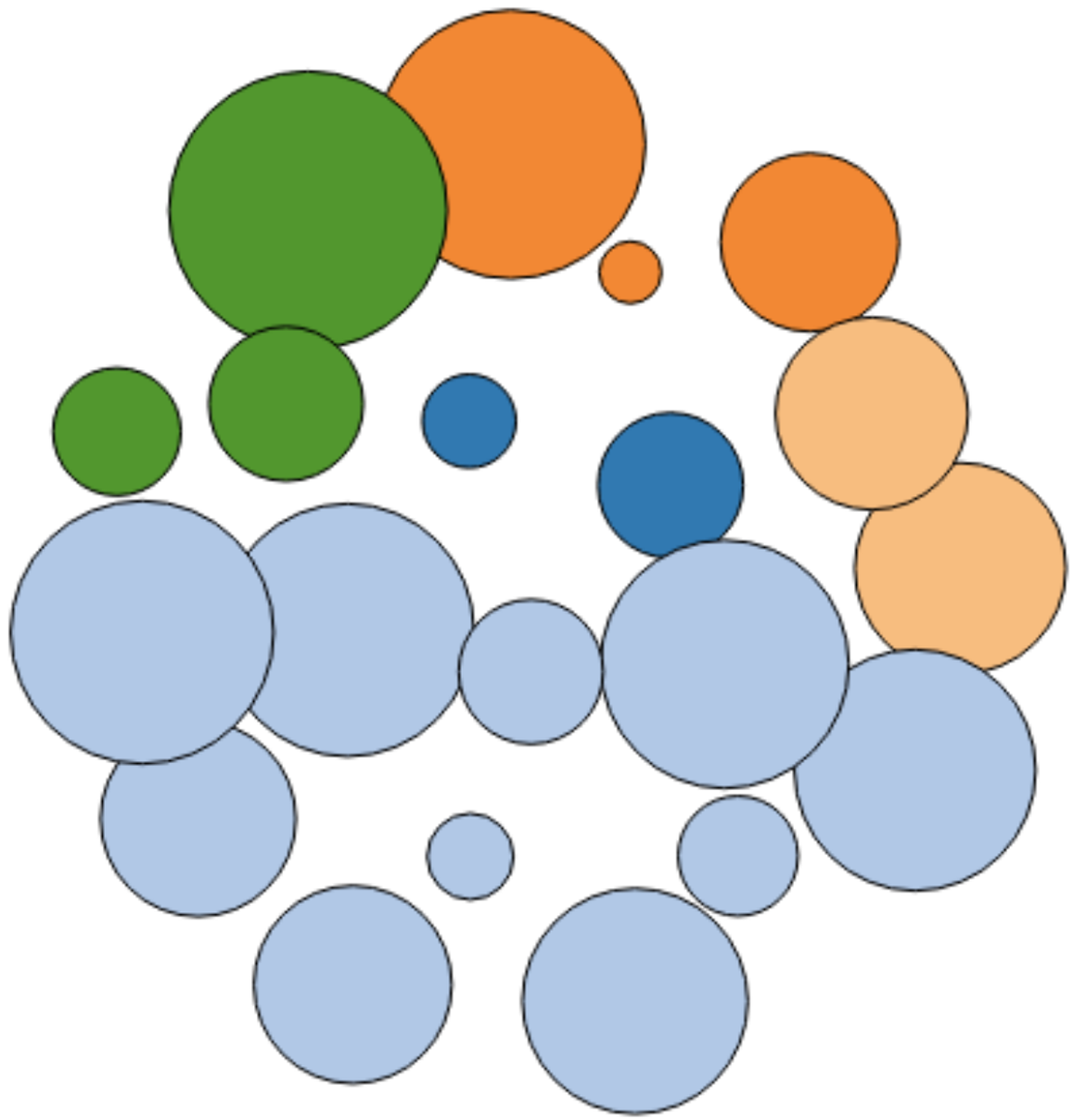
## 並利用 d3.scale.category20 著色



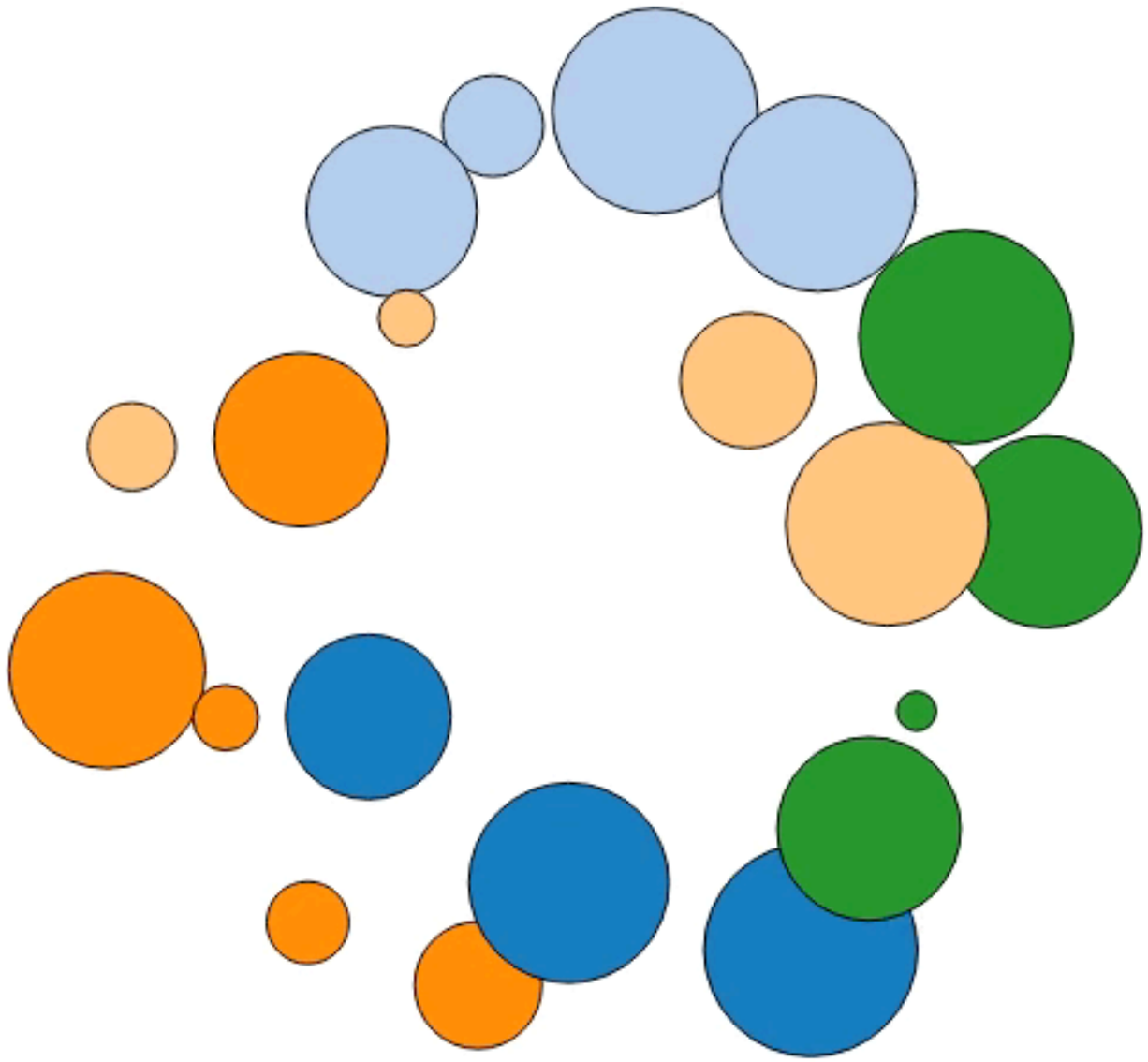
```
var data = d3.range(20).map(function(it) {
  return {
    value: parseInt(Math.random() * 9 + 1),
    category: "ABC".charAt(parseInt(Math.random() * 3))
  };
});

var color = d3.scale.category20();

d3.select("svg")
  .selectAll("circle")
  .data(data)
  .enter().append("circle")

function render() {
  d3.select("svg")
    .selectAll("circle")
      .attr({
        cx: function(it) { return it.x; },
        cy: function(it) { return it.y; },
        r:  function(it) { return it.value; },
        fill: function(d,i) { return color(d.category); },
        stroke: "#000"
      });
}

var force = d3.layout.force()
  .size([800, 400])
  .nodes(data)
  .on("tick", render)
  .start();
```

# #16 解答

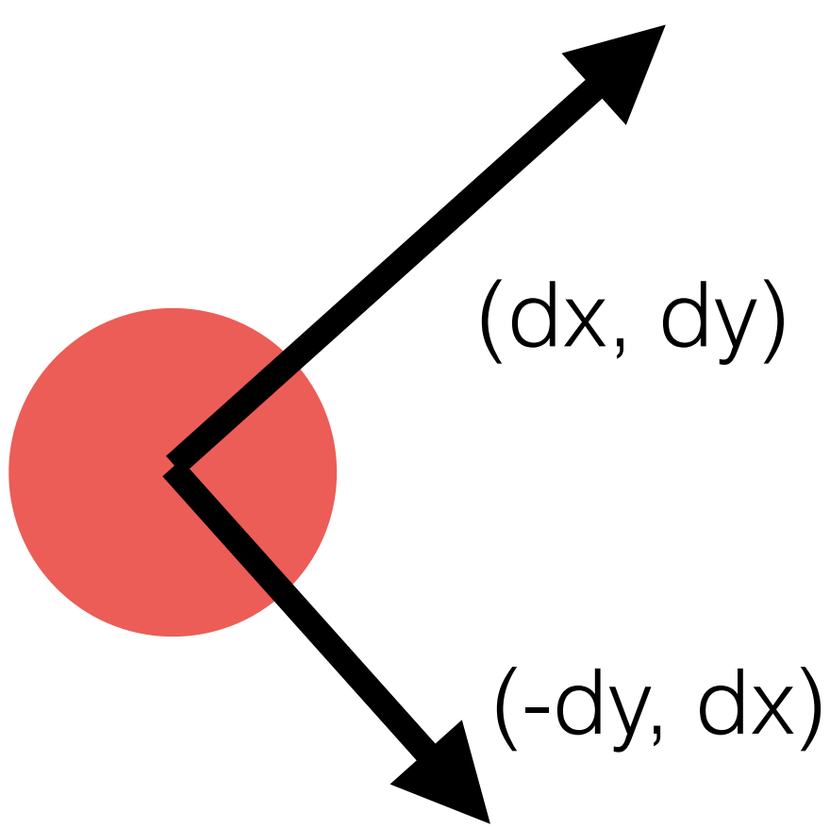
# Pack Layout + Force Layout



# Pack Layout + Force Layout and Rotate!

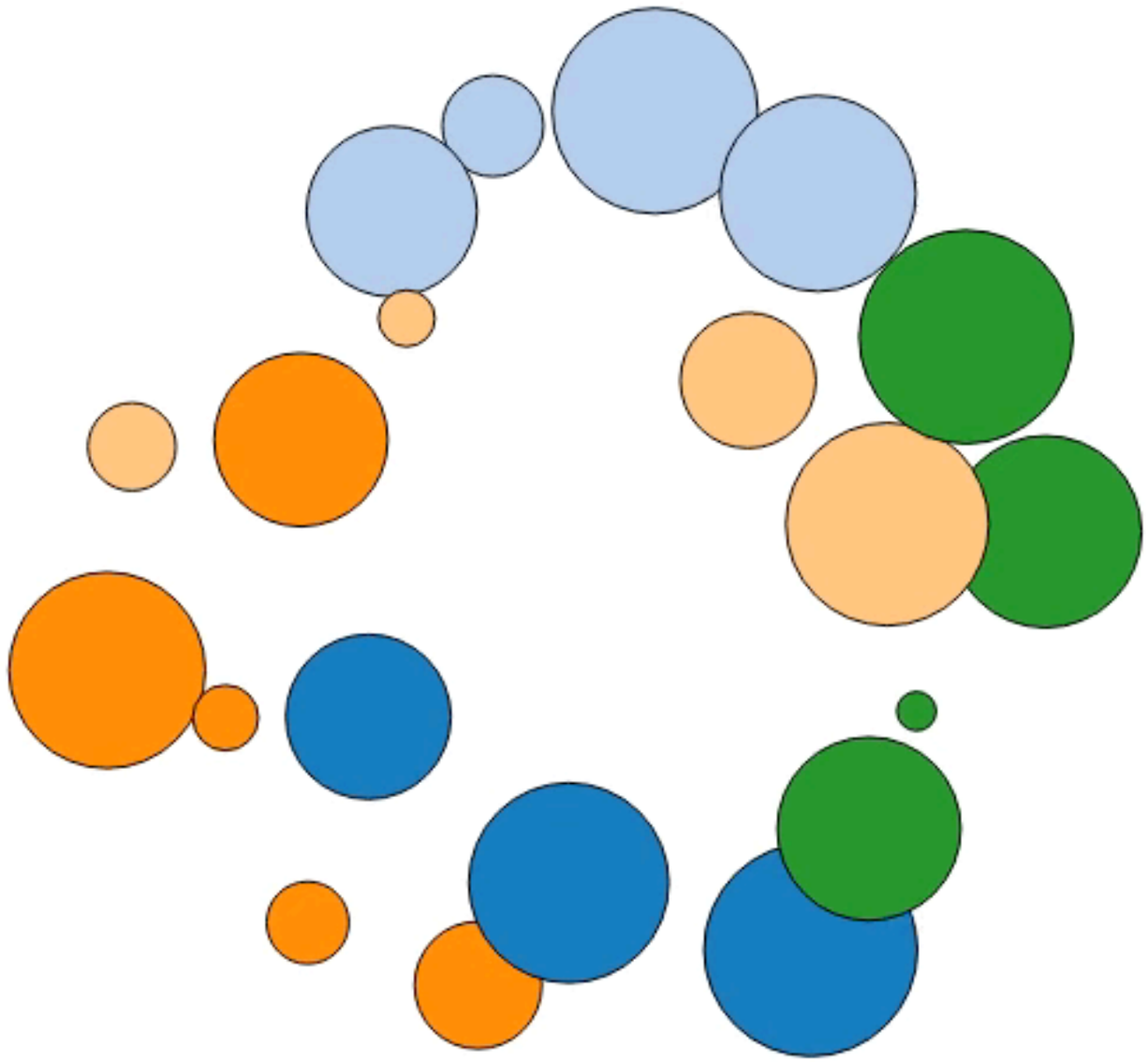


```
function tick() {
  for(var i=0;i<nodes.length;i++) {
    n = nodes[i];
    dx = 300 - n.x;
    dy = 300 - n.y;
    len = Math.sqrt(dx * dx + dy * dy );
    n.x -= dy / len;
    n.y += dx / len;
    }
}
```

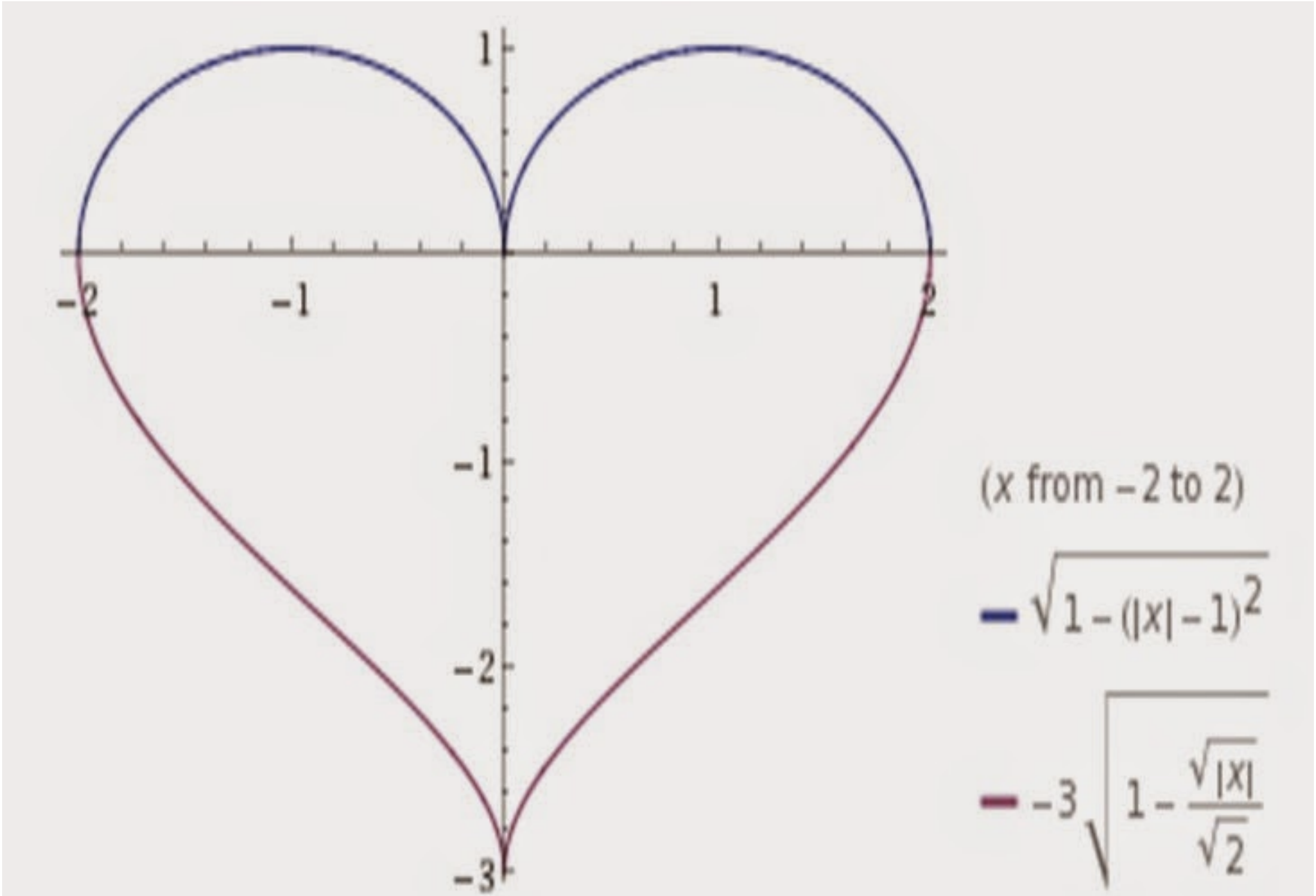


# 實作練習 #17

# 讓 #16 的結果旋轉！

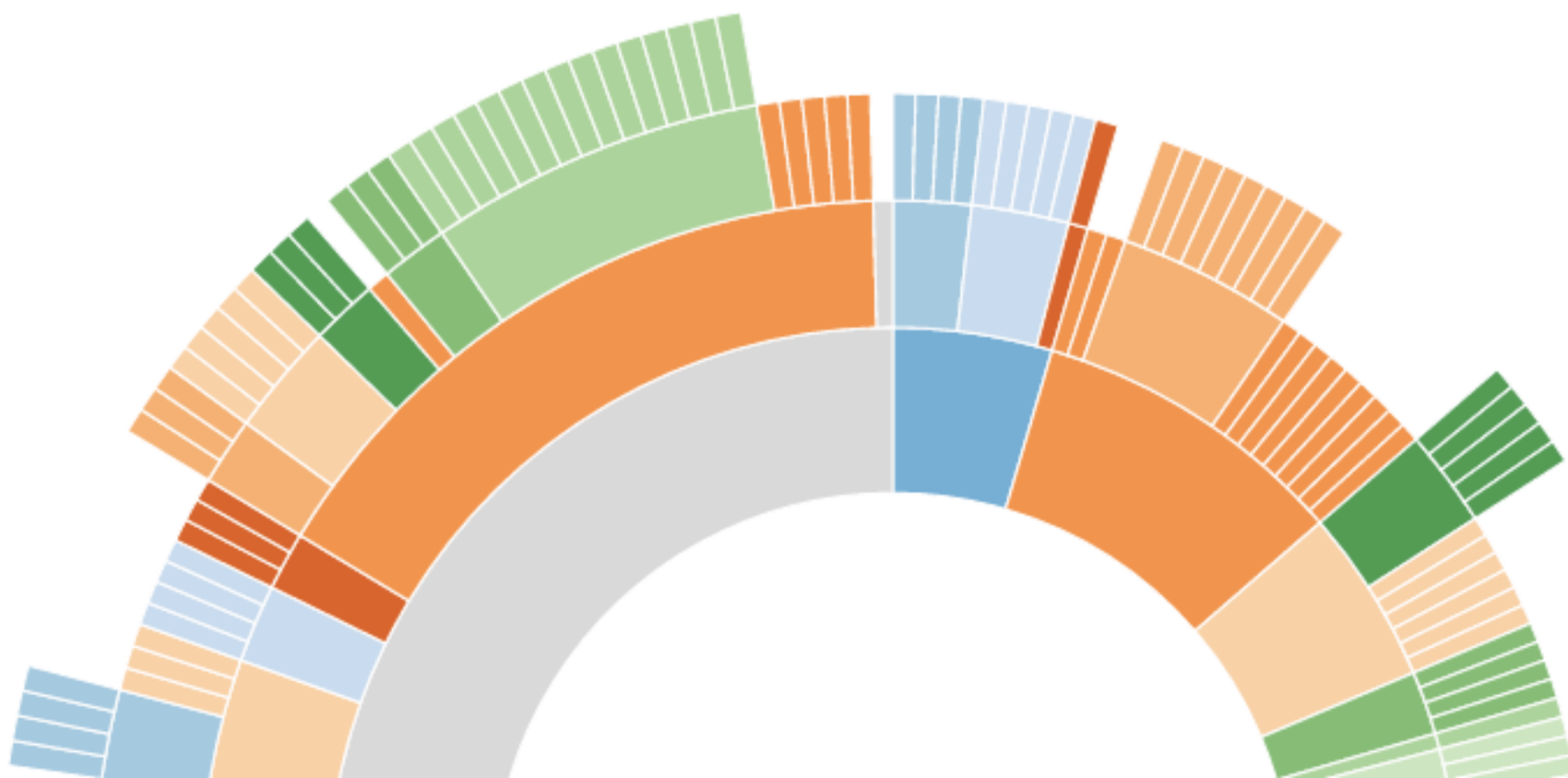


重新執行


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

# d3.layout.partition

```
d3.layout.partition()
    .size([width, height])
    .children( … )
    .nodes(data)
```

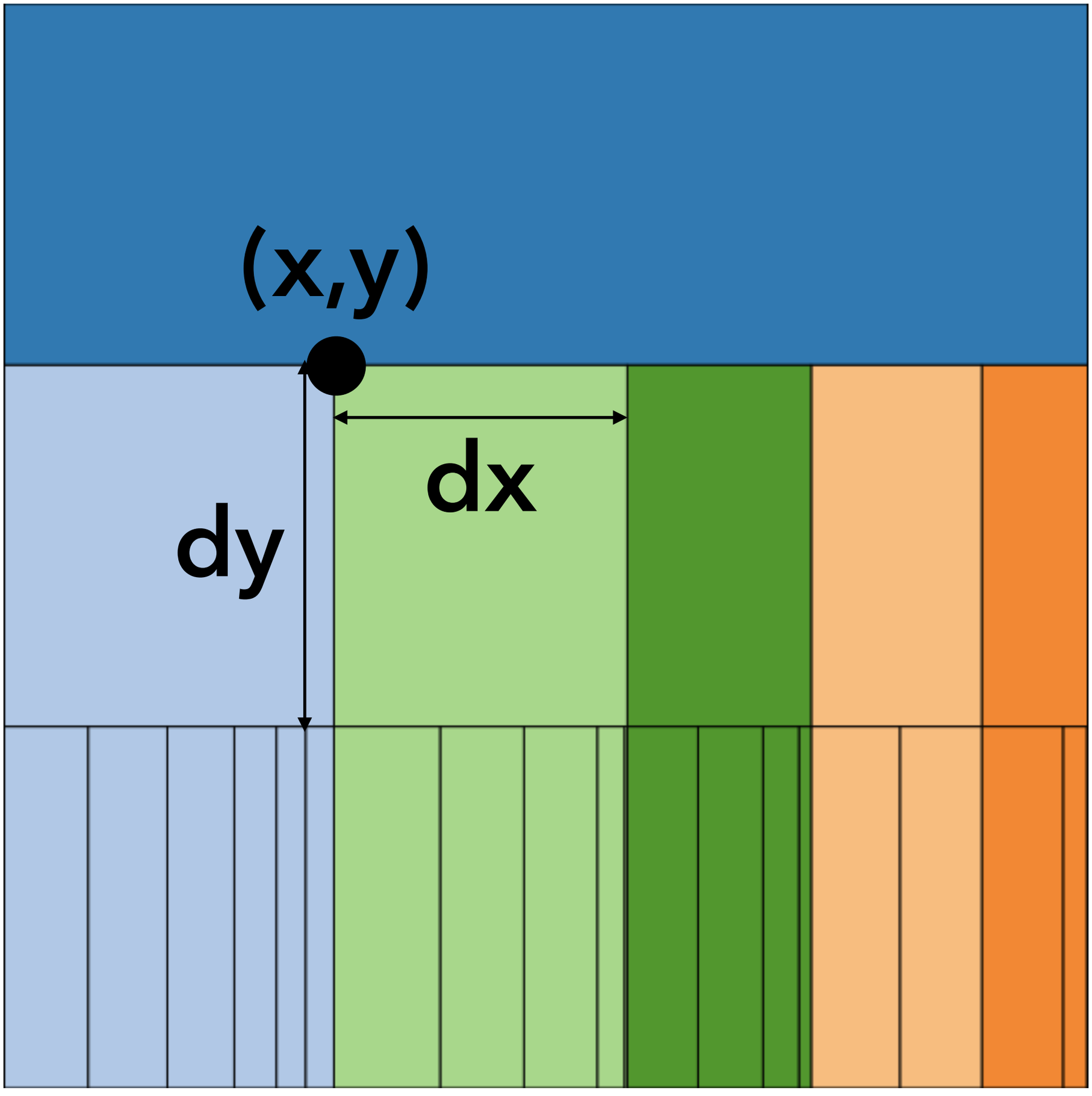


# Partition Layout

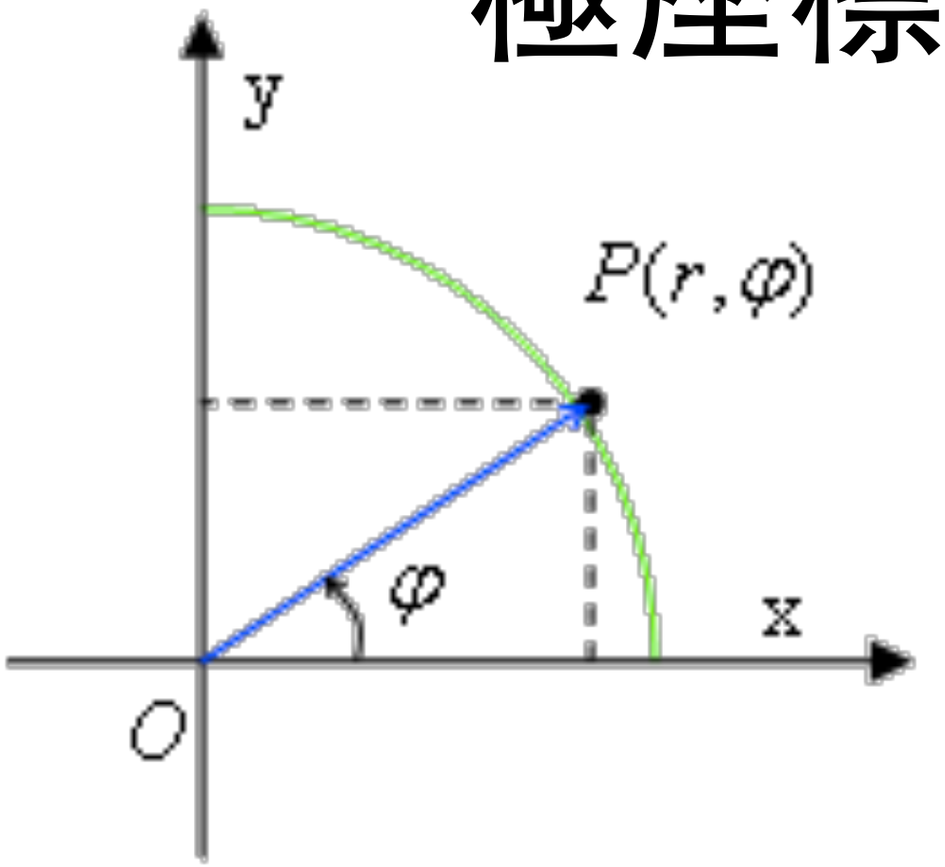
```
var nodes = d3.layout.partition()
.children(function(it) {
  return it.cat;
}).size([600,600]).nodes(data);
```

```
[
  {…},
  {
    cat: "A", // depends
    value: 0.447,
    x: 248.867,
    y: 414.396,
    dx: 354.939,
    dy: 303.569,
    depth: 0
  }, {…}, …
]
```

```
function tick() {
  d3.select("svg").selectAll("rect").data(nodes)
  .enter().append("circle").attr({
    x: function(it) { return it.x; },
    y: function(it) { return it.y; },
    width: function(it) { return it.dx; },
    height: function(it) { return it.dy; },
    fill: function(it) { return color(it.cat); },
    stroke: "#000",
  });
}
```


(x,y)
dx
dy

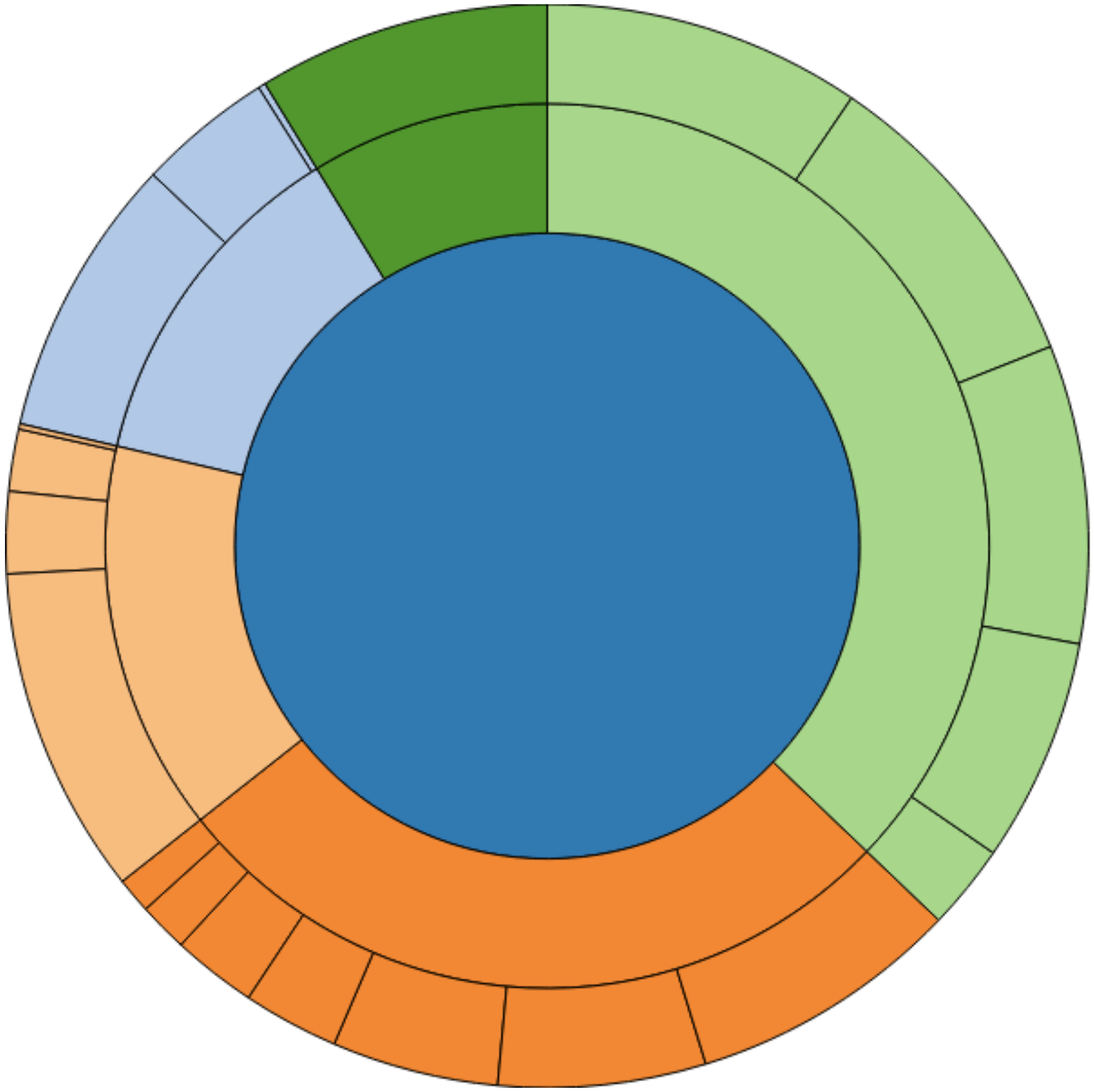
# 極座標轉換




```
var nodes = d3.layout.partition()
.children(function(it) {
  return it.cat;
}).size([2 * Math.PI,300]).nodes(data);
```

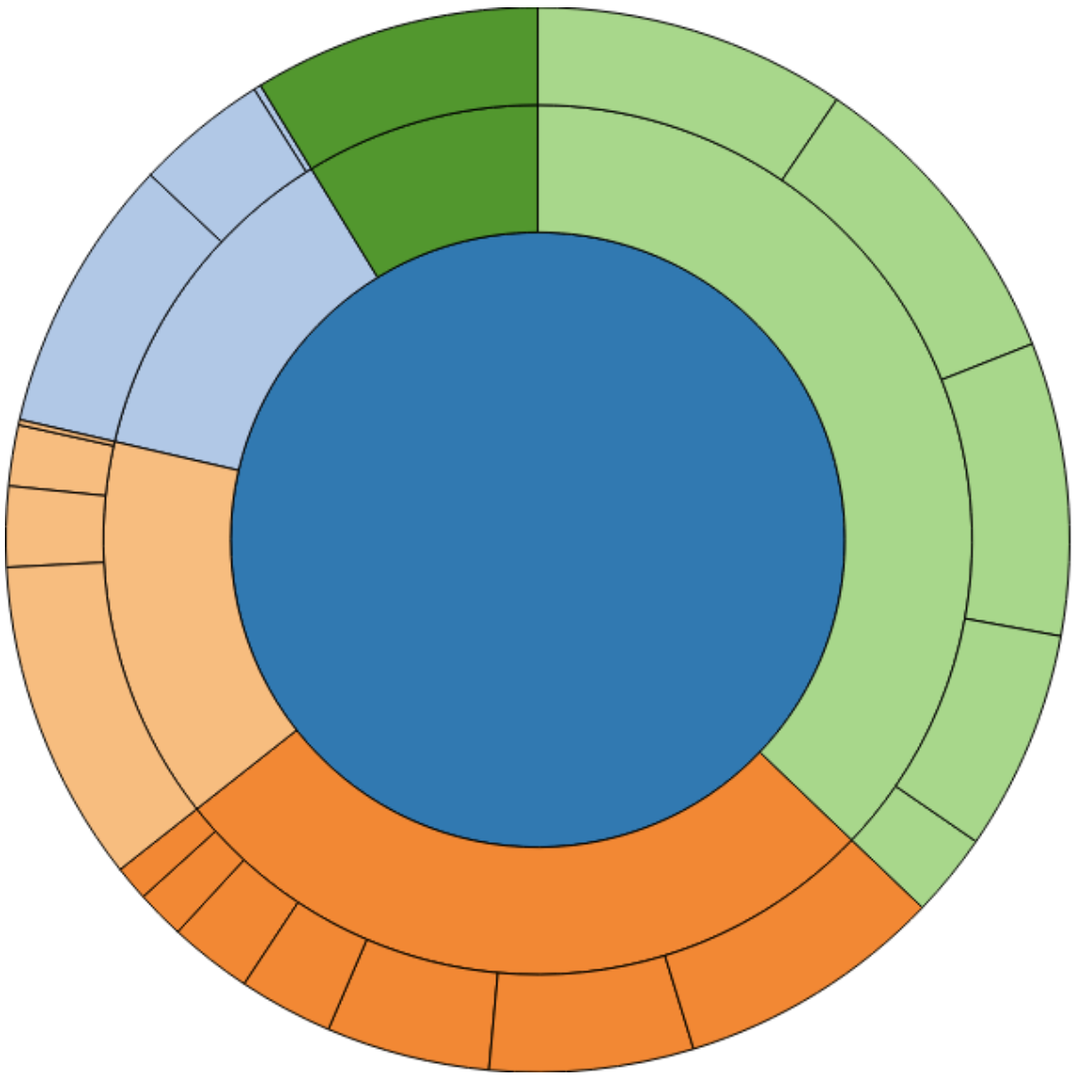
# use d3.svg.arc

```
arc = d3.svg.arc()
  .startAngle(function(it) { return it.x; })
  .endAngle(function(it) { return it.x + it.dx;)
  .innerRadius(function(it) { return it.y; });
  .outerRadius(function(it) {
    return it.y + it.dy;
  });

function tick() {
  d3.select("svg").selectAll("path").data(nodes)
  .enter().append("path").attr({
    d: arc
    fill: function(it) { return color(it.cat); },
    stroke: "#000",
  });
}
```



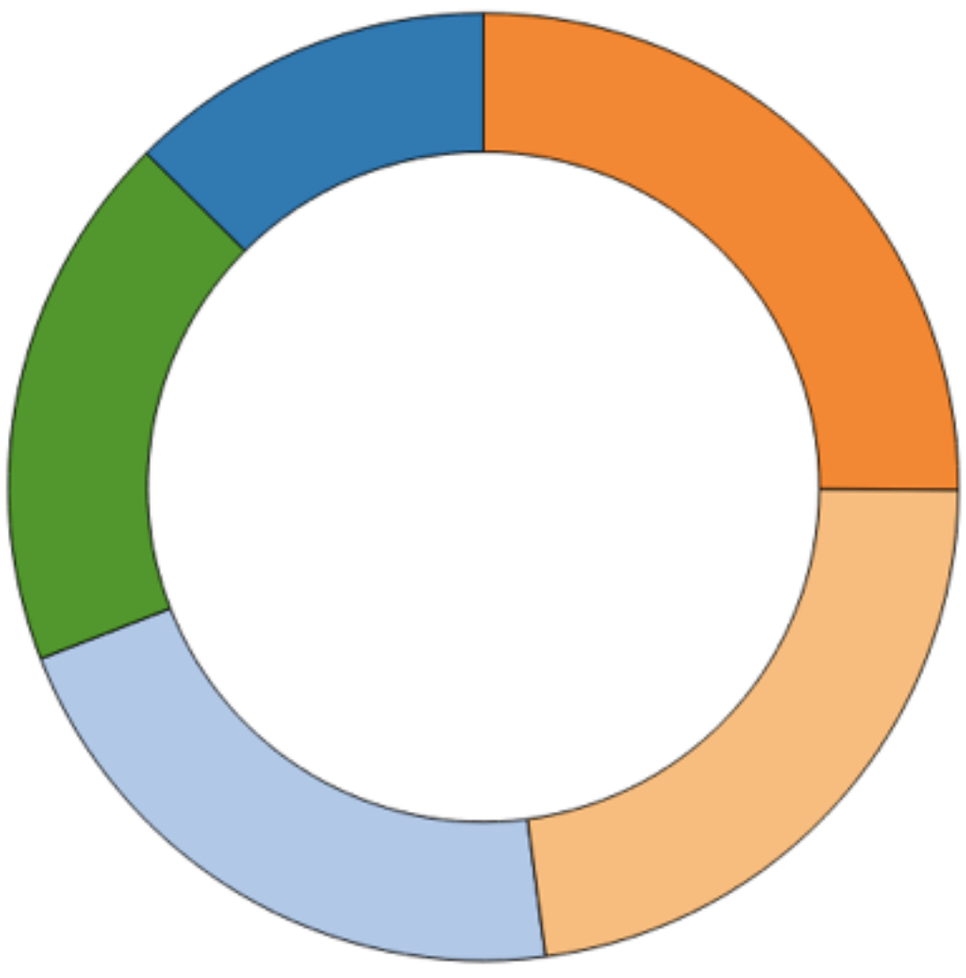
# 實作練習 #18

# 利用 #13 的結果畫出 Sunburst 圖



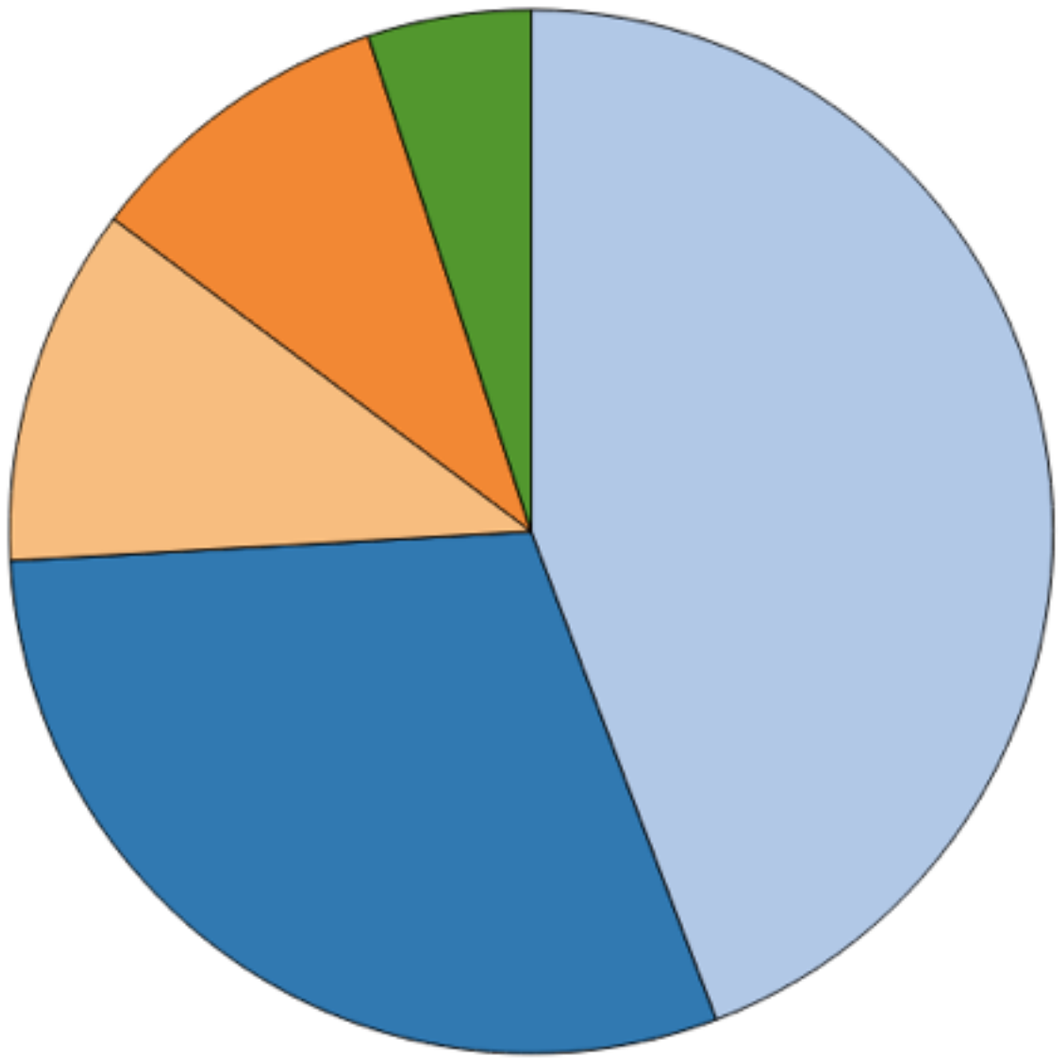
# Donut Chart Ready.

```
nodes = nodes.filter(function(it) {
  return it.depth == 1;
});
```



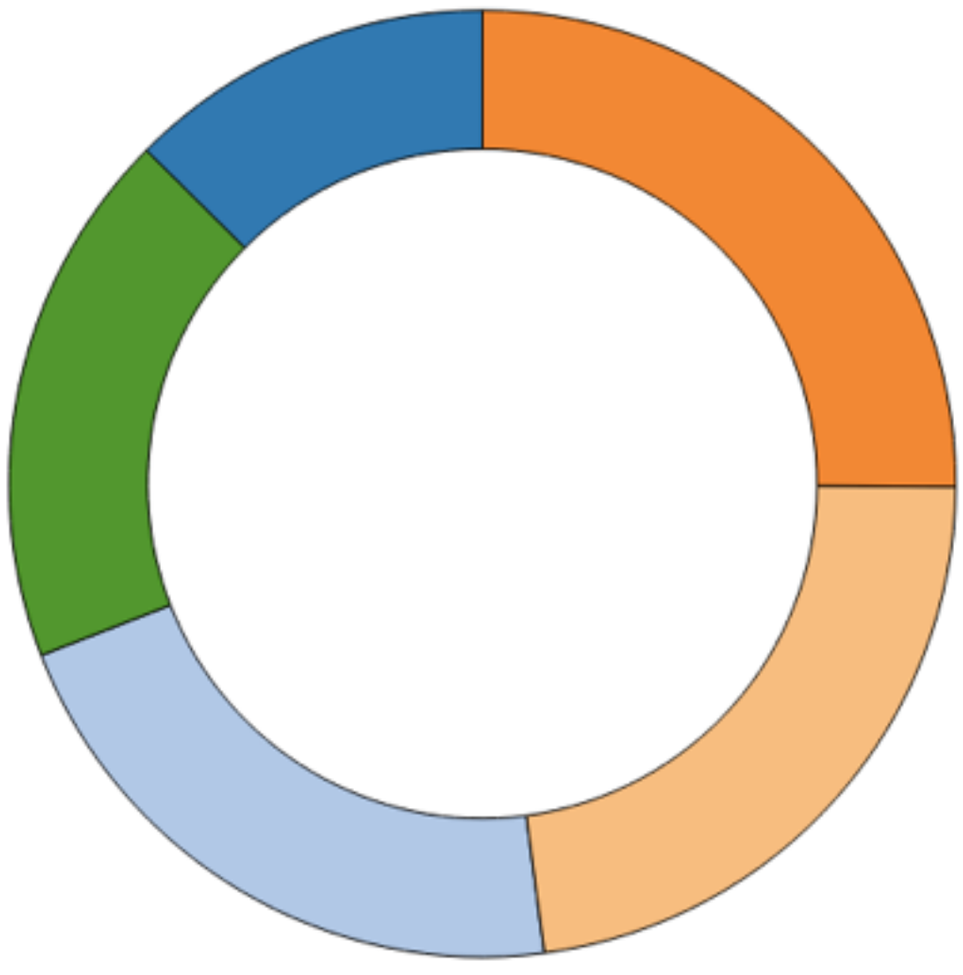
# Pie Chart Done.

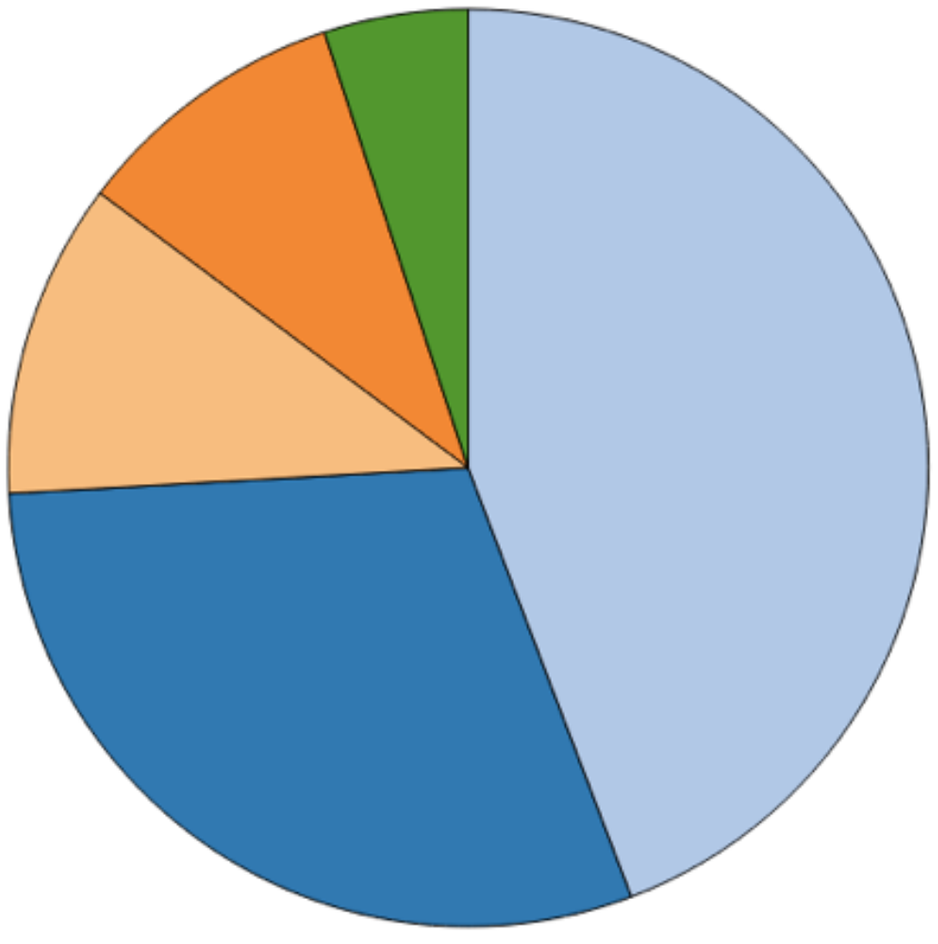
```
arc = d3.svg.arc()
  .innerRadius(function(it) { return 0; });
```



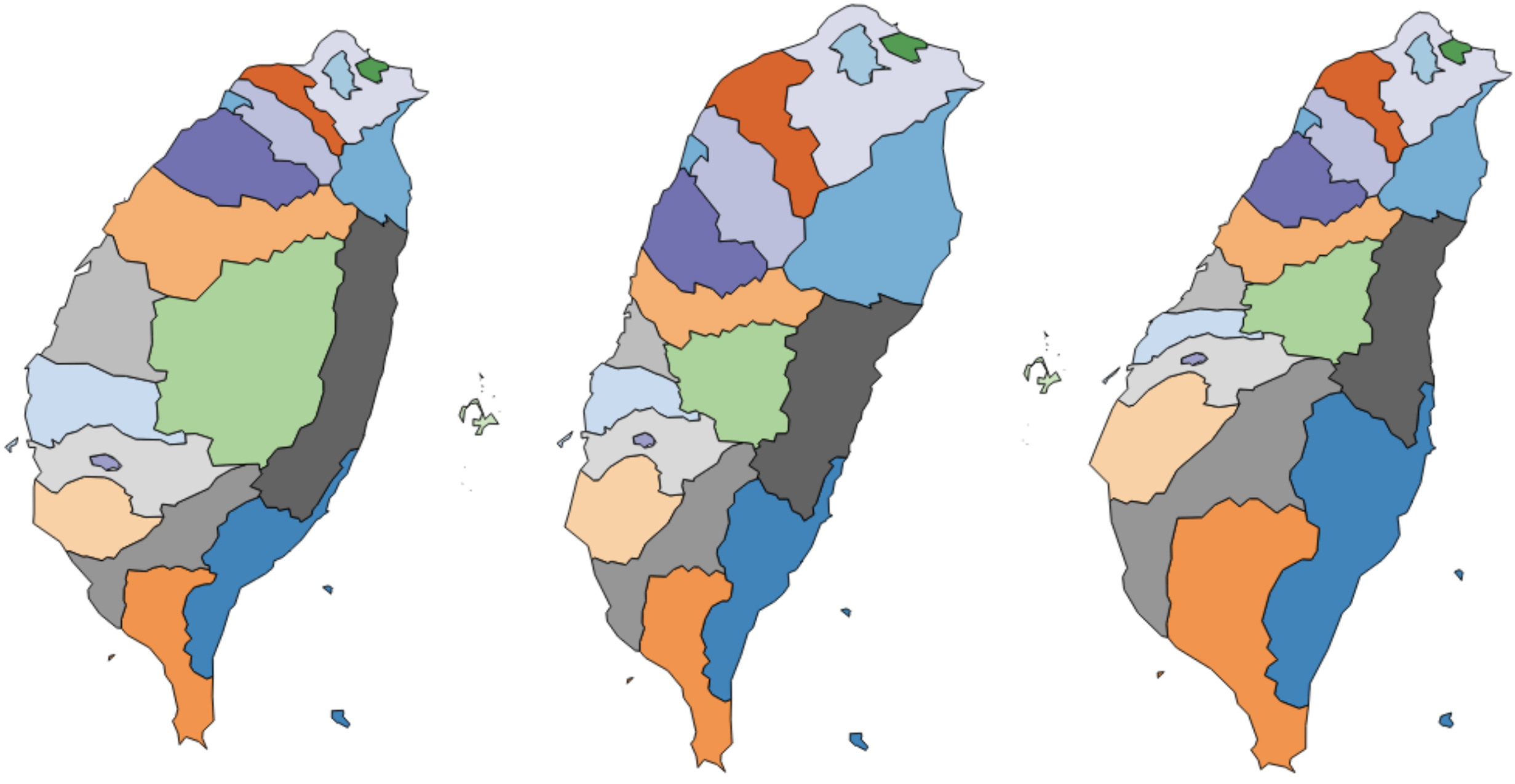
## 實作練習 #19

# 利用 # 18 的結果畫出
# Donut Chart & Pie Chart



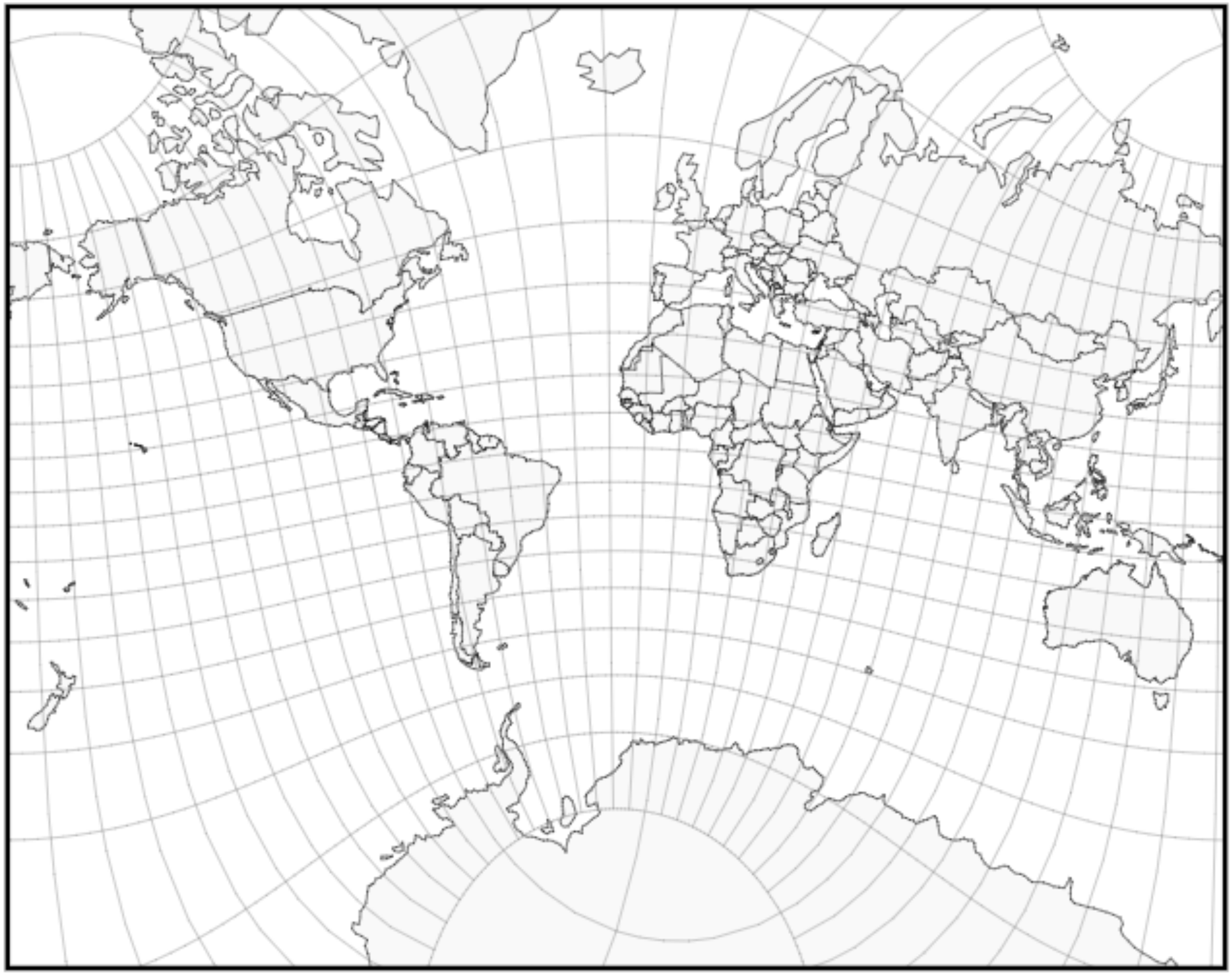


# Let’s Play With Map



# Prerequisite

- 地理區塊資料檔 — twTown1982.topojson
- 地理區塊解析程式 — topojson
- 地理區塊描繪器 — Path
- 地理區塊投影器 — Projection



https://www.jasondavies.com/maps/transition/

## include topojson.min.js

```
<script src="topojson.min.js"></script>
```

## load the topojson file

```
<script src="twTown1982.topo.json"></script>
```

```
projection = d3.geo.mercator()
   .center([121,24]).scale(6000);

path = d3.geo.path()
   .projection(projection);
```

```
projection = d3.geo.mercator()
  .center([121,24]).scale(6000);

path = d3.geo.path()
  .projection(projection);
```

```
features = topojson.feature(
    twTown, twTown.objects["layer1"]
).features;

d3.select("svg").selectAll("path")
  .data(features).enter().append("path")
  .attr({d: path });
```

# 實作練習 #15

# 隨機為鄉鎮區塊著上顏色

```
projection = d3.geo.mercator()
  .center([121,24]).scale(6000);

[x,y] = projection(
  [longitude,latitude]
);
```

```
path = d3.geo.path()
.projection(projection);
```

```
path = d3.geo.path()
.projection(function(latlng) {
  pts = projection(latlng);
  return pts;
});
```

# Include fisheye.js

```
<script src="fisheye.js"></script>
```

```
fish = d3.fisheye.circular()
  .radius(200).focus([300,300]);

path = d3.geo.path()
.projection(function(latlng) {
  pts = projection(latlng);
  ret = fish({x: pts.x, y: pts.y});
  return [ret.x, ret.y];
});
```

https://github.com/d3/d3-plugins/tree/master/fisheye

# 實作練習 #16

# 隨機為鄉鎮區塊著上顏色

條件：製作魚眼特效

馬祖

金門