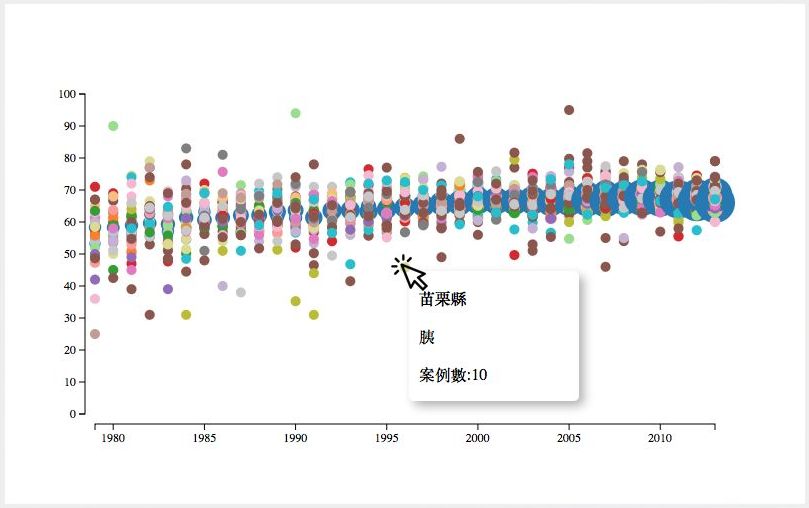
Data Visualization Foundation - D3.js

Michelle Chen

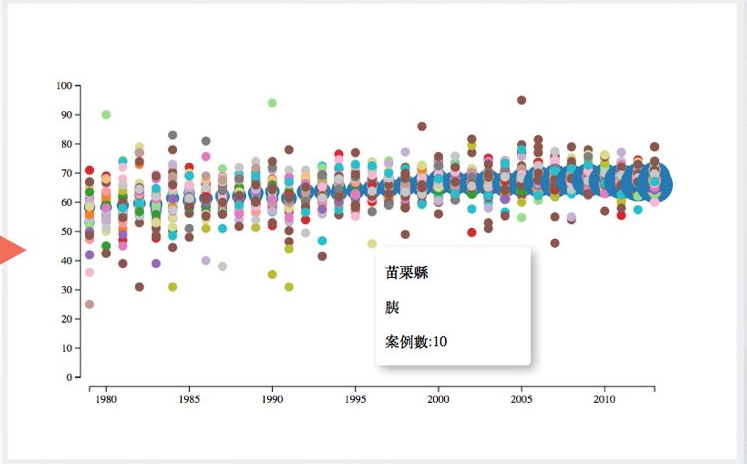
2018 / 07 / 12 Sharing

資料視覺化

一個把資料對應到視覺元素的過程



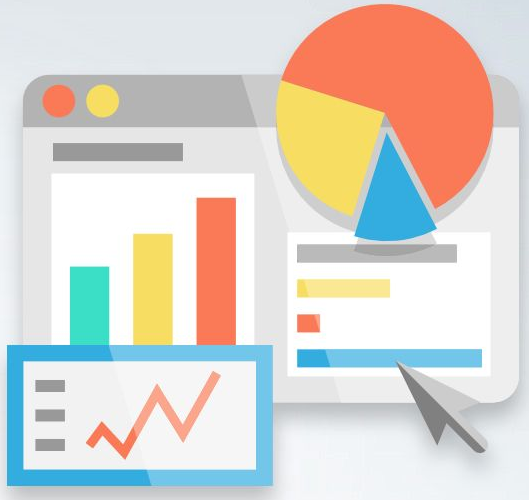
	A	B	C	D	E	F	G	H	I	J
1	癌症診斷年	性別	縣市別	癌症別	年齡標準化率	癌症發生數	平均年齡	年齡中位數	粗率 (每10萬人口)	
34	1979	男	宜蘭縣	口腔、口咽及下咽	4.73	9	52.44	52	3.88	
35	1979	男	苗栗縣	口腔、口咽及下咽	1.04	2	48.5	48.5	0.71	
36	1979	男	彰化縣	口腔、口咽及下咽	11.95	52	51.9	52.5	8.74	
37	1979	男	南投縣	口腔、口咽及下咽	4.61	10	52.7	52	3.66	
38	1979	男	雲林縣	口腔、口咽及下咽	3.57	11	51.09	50	2.65	
39	1979	男	嘉義縣	口腔、口咽及下咽	5.03	12	52.17	53	3.99	
40	1979	男	屏東縣	口腔、口咽及下咽	6.31	20	59.35	58	4.29	
41	1979	男	澎湖縣	口腔、口咽及下咽	4.18	2	56.5	56.5	3.52	
42	1979	男	花蓮縣	口腔、口咽及下咽	3.57	6	44	46	3.1	
43	1979	男	台東縣	口腔、口咽及下咽	9.97	16	55.27	57	10.37	
44	1979	女	台閩地區	口腔、口咽及下咽	1.32	75	53.84	56	0.89	
45	1979	女	台北市	口腔、口咽及下咽	3.36	23	53.26	56	2.18	
46	1979	女	台中市	口腔、口咽及下咽	0.51	3	45	43	0.4	
47	1979	女	台南市	口腔、口咽及下咽	0.35	2	50	50	0.27	
48	1979	女	高雄市	口腔、口咽及下咽	1.34	9	48.67	50	0.88	
49	1979	女	基隆市	口腔、口咽及下咽	1.02	1	63	63	0.61	
50	1979	女	新竹市	口腔、口咽及下咽	1.97	2	53.5	53.5	1.52	
51	1979	女	嘉義市	口腔、口咽及下咽	1.33	1	64	64	0.82	
52	1979	女	新北市	口腔、口咽及下咽	0.71	4	58.75	59	0.39	
53	1979	女	桃園市	口腔、口咽及下咽	2.32	6	55.83	57	1.26	
54	1979	女	宜蘭縣	口腔、口咽及下咽	0.71	1	58	58	0.48	
55	1979	女	苗栗縣	口腔、口咽及下咽	0.52	1	70	70	0.39	
56	1979	女	彰化縣	口腔、口咽及下咽	0.78	3	58.33	59	0.54	
57	1979	女	雲林縣	口腔、口咽及下咽	1.23	4	45	55	1.04	
58	1979	女	屏東縣	口腔、口咽及下咽	1.71	5	63.6	65	1.2	
59	1979	女	花蓮縣	口腔、口咽及下咽	2.82	3	55.67	55	1.88	
60	1979	女	台東縣	口腔、口咽及下咽	1.3	1	65	65	0.79	



資料連結

資料視覺化
D3.js

為什麼要做資料視覺化？

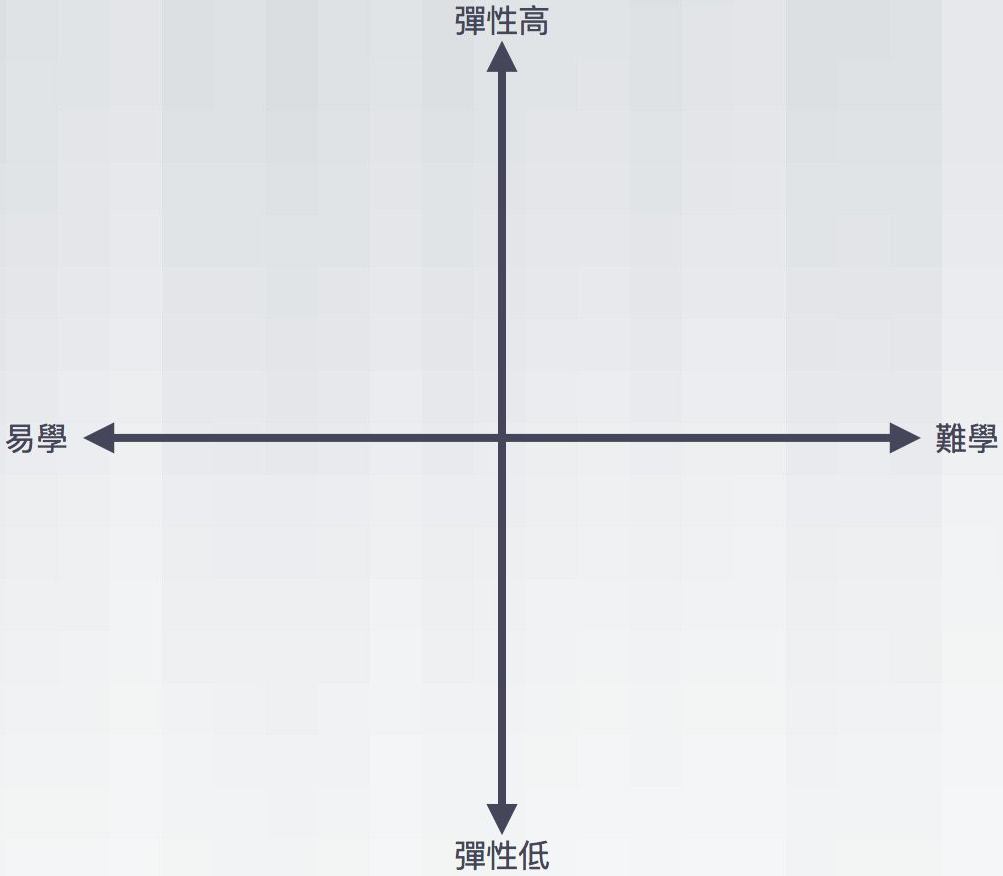


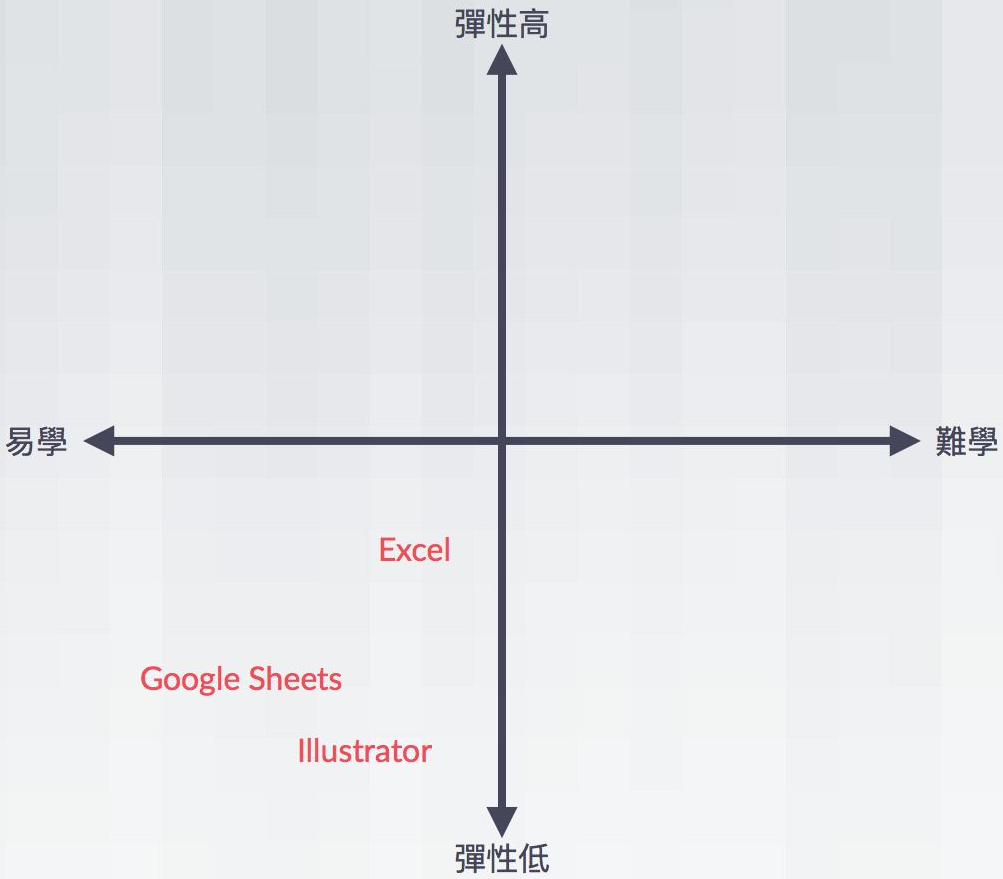
為什麼要做資料視覺化？

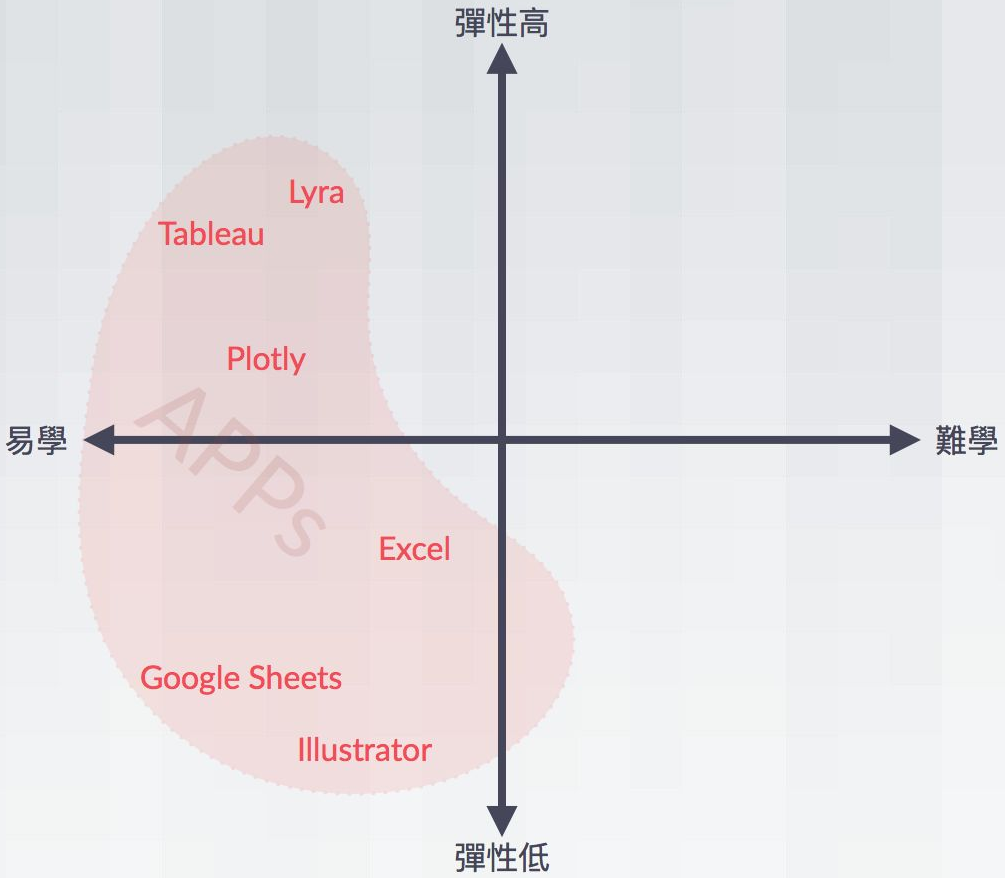
1. 圖形的傳達效率高
2. 理解数据的特性
3. 找出數據蘊含的模式
4. 建議建模的策略
5. 診斷分析中問題

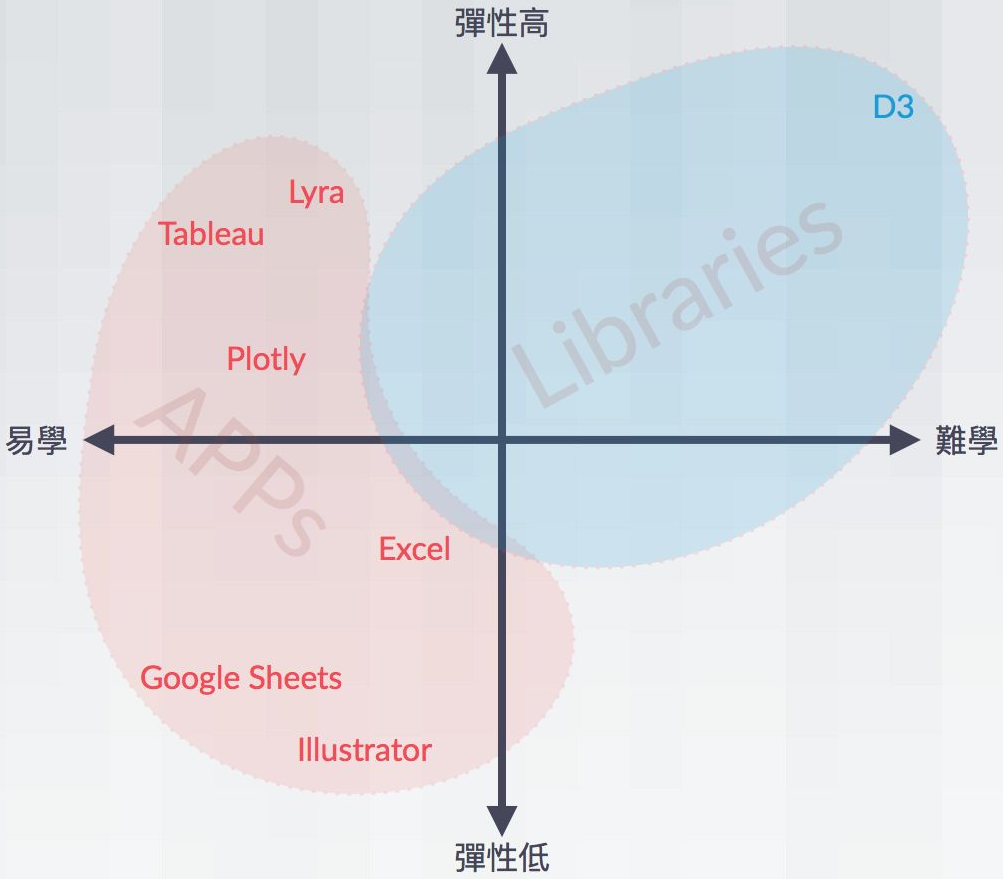
今天你手上有一份 X 資料

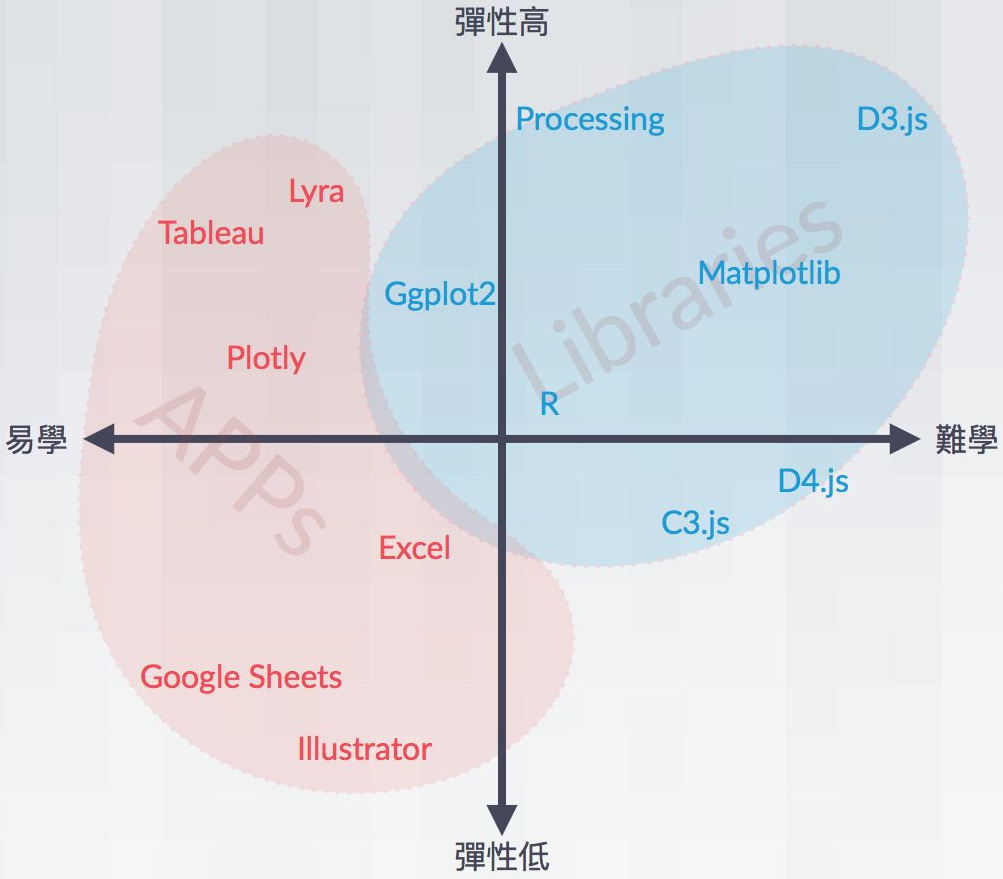
想做資料視覺化，你會怎麼做？


彈性高
易學
難學
彈性低


彈性高
易學
難學
Excel
Google Sheets
Illustrator
彈性低


彈性高
Lyra
Tableau
Plotly
APPS
易學
難學
Excel
Google Sheets
Illustrator
彈性低


彈性高
D3
Lyra
Tableau
Libraries
Plotly
易學
難學
APPs
Excel
Google Sheets
Illustrator
彈性低



from What I Learned Recreating One Chart Using 24 Tools - Lisa Charlotte Rost

沒有完美的工具，只有適合你的好工具

簡單方便的圖表

Illustrator
Plotly
C3.js

分析後做視覺化

Excel
Google Sheets
Matplotlib
Ggplot2

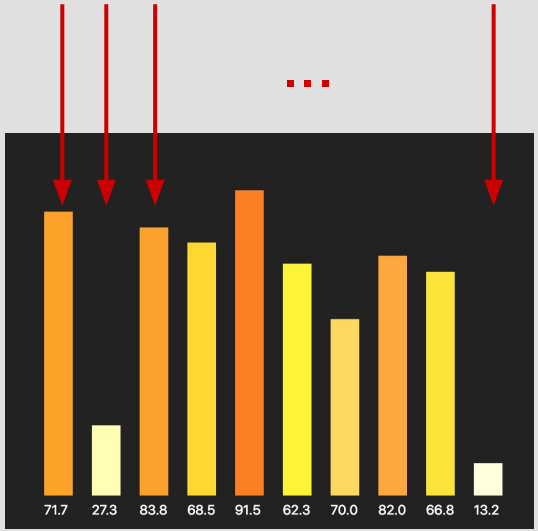
跳脫框架的圖表

Lyra
D3.js
Processing

Why using D3?

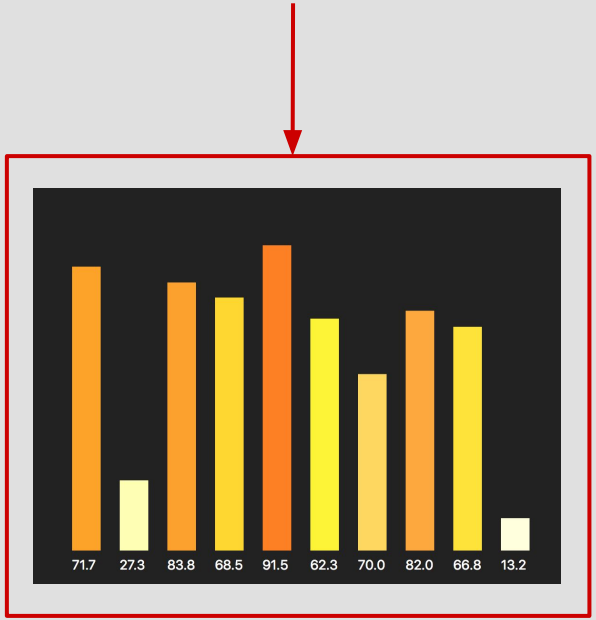
jQuery

操作DOM與資料處理



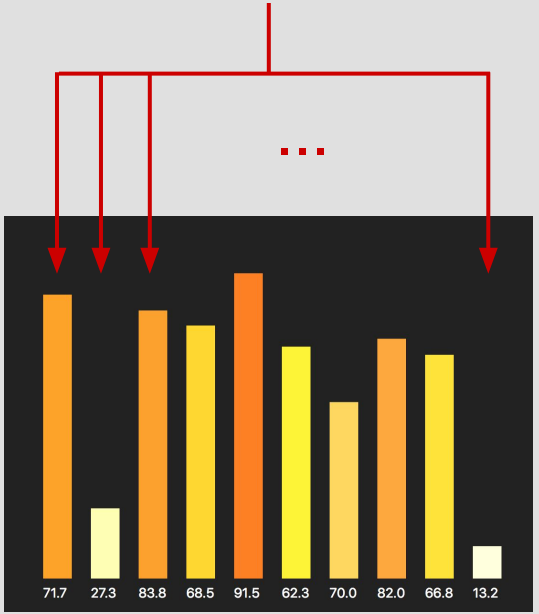
Vue.js

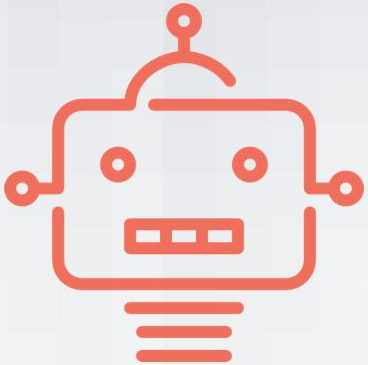
計算資料，自動指定，
自動產生



D3.js

操作DOM(高級版)，
函式多元





彈性高

互動性

跨平台

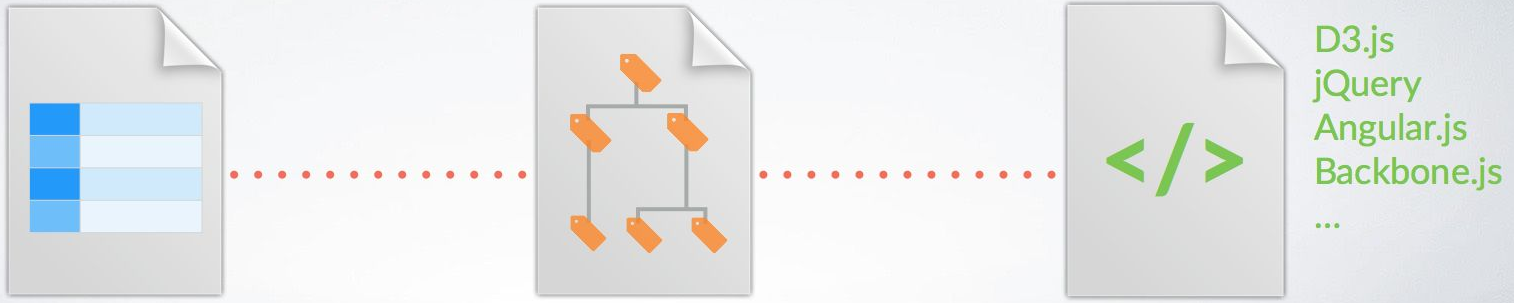
資料視覺化
D3.js

資料視覺化

D3.js

Web Concept

網頁構成要素



CSS

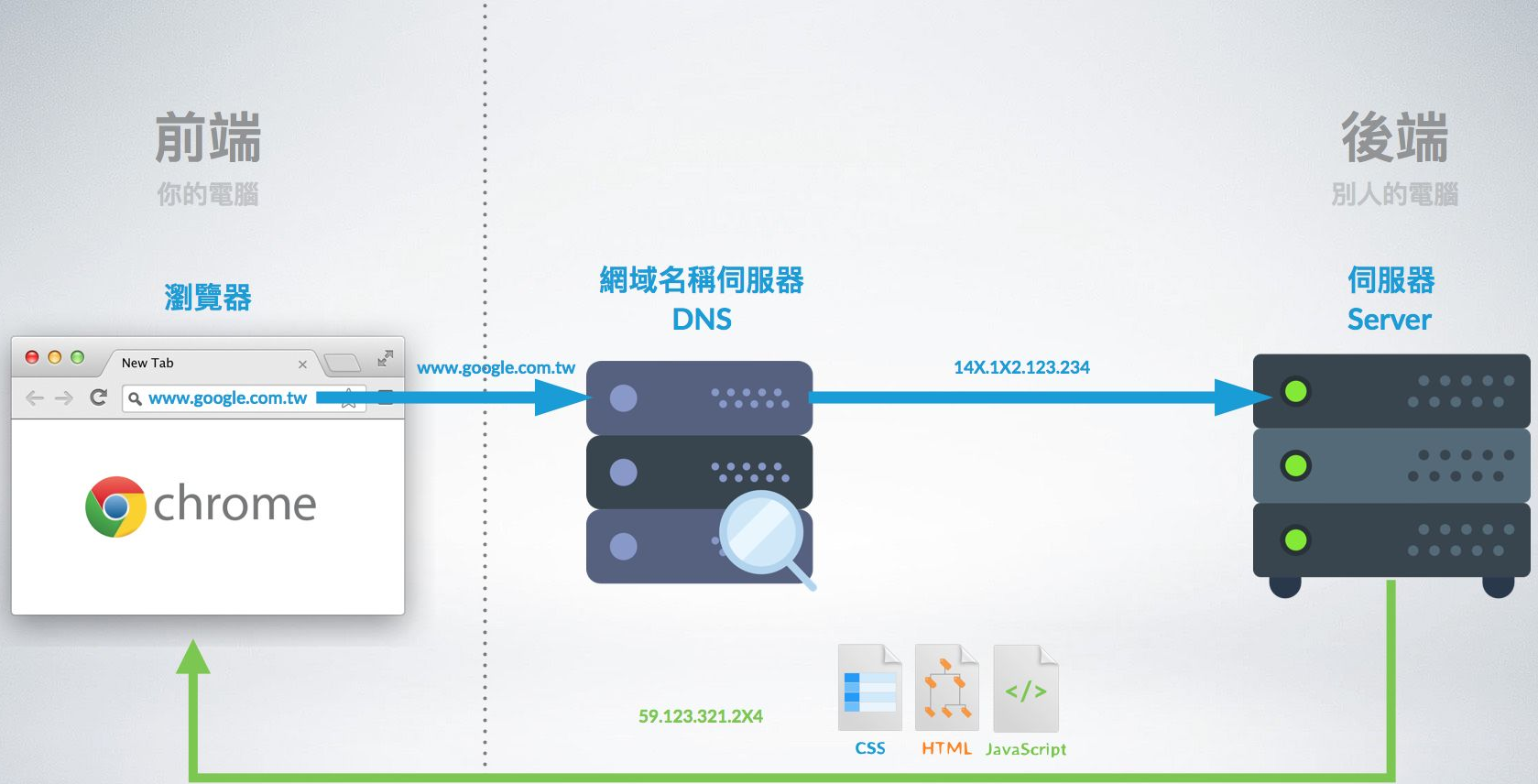
外觀

HTML

框架 / 內容

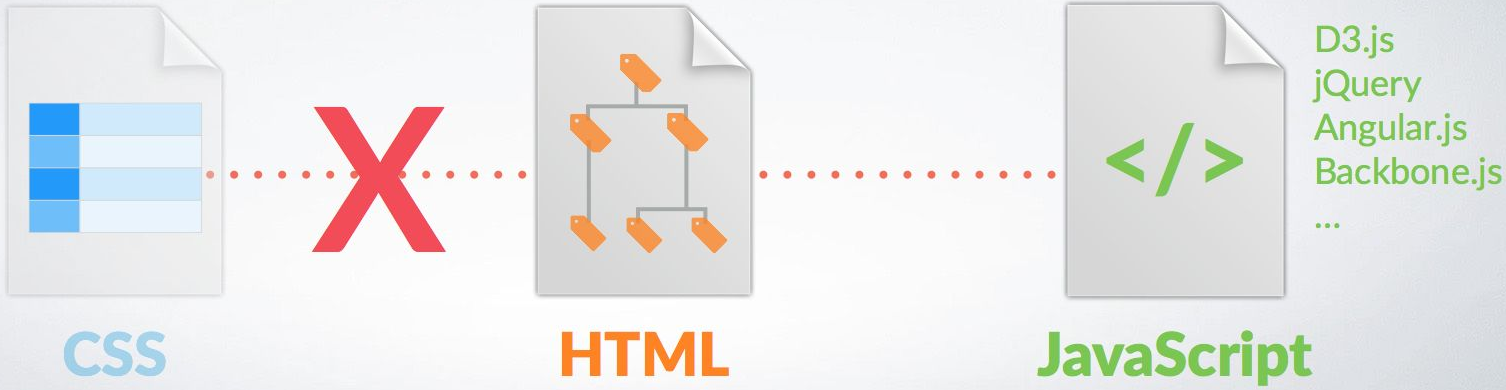
JavaScript

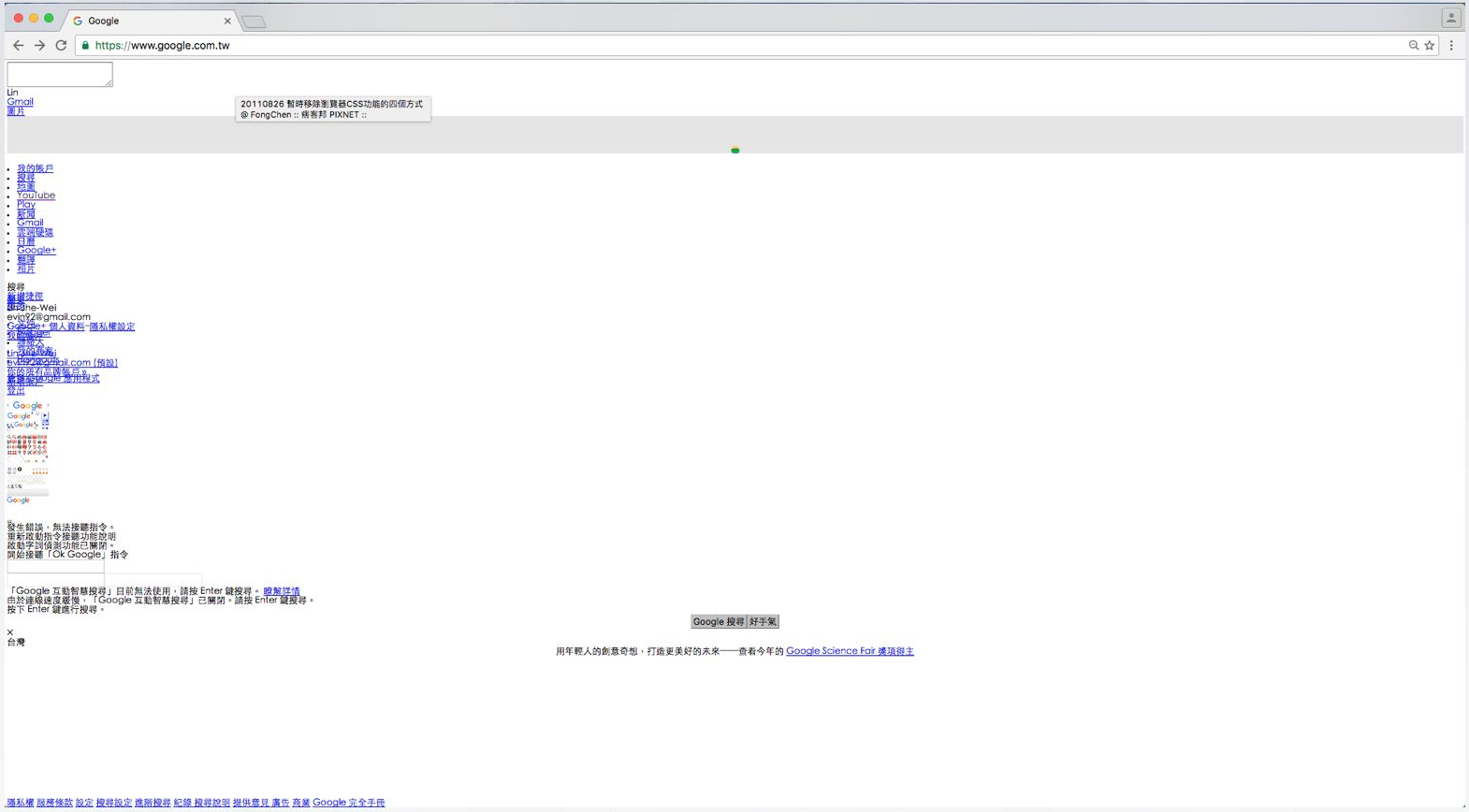
互動


前端
你的電腦
後端
別人的電腦
瀏覽器
網域名稱伺服器
DNS
伺服器
Server
New Tab
www.google.com.tw
chrome
www.google.com.tw
14X.1X2.123.234
59.123.321.2X4
CSS
HTML
JavaScript

資料視覺化
D3.js

網頁構成要素


D3.js
jQuery
Angular.js
Backbone.js
...
CSS
HTML
JavaScript
外觀
框架/內容
互動

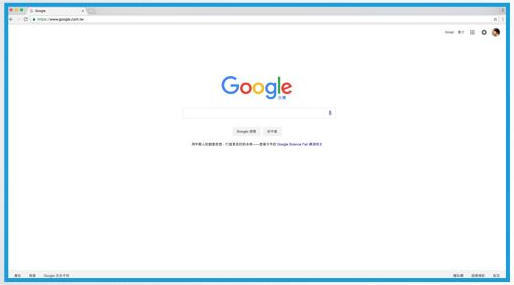


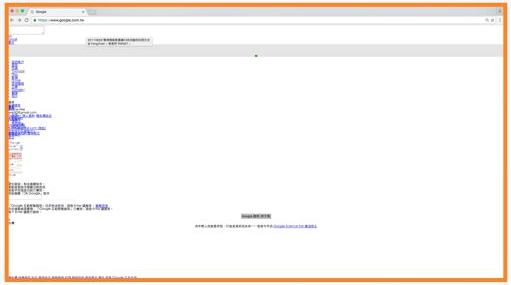
資料視覺化
D3.js

咒語:

```
javascript:for(i=0;i<document.styleSheets.length;i++){void(document.styleSheets.item(i).disabled=true);}
el=document.getElementsByTagName('*');for(i=0;i<el.length;i++){void(el[i].style.cssText='');}
```

網頁構成要素





1. 送出搜尋文字
2. 接收後端資料
3. 呈現搜尋結果

CSS

外觀

HTML

框架 / 內容

JavaScript

互動

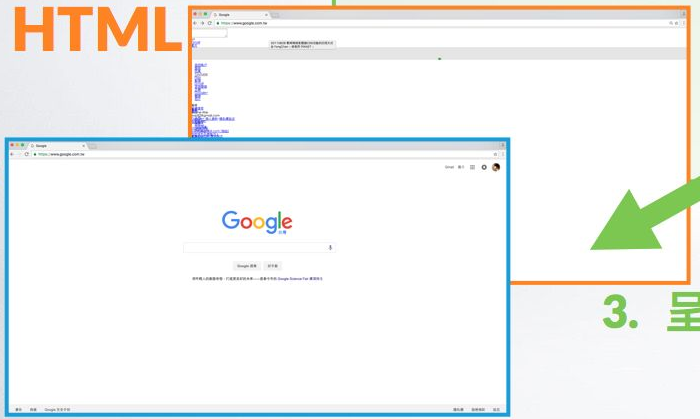
前端

後端

JavaScript

HTML

CSS



1. 送出搜尋文字

2. 接收後端資料

3. 呈現搜尋結果

D3 - 資料驅動的文件

Data-Driven Documents

Data-Driven Documents

D3 - 資料驅動的文件

[白話]

把資料與文件中的視覺元素綁定在一起

JS函式庫:
指用JavaScript語言開發的函式庫，多應用在網頁上，
可以幫助網頁開發者用更快速地撰寫網頁

D3.js - 資料驅動的文件

[白話]

用JS函式把資料與文件中的視覺元素綁定在一起

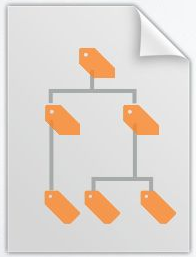
Data-Driven Documents

D3.js - 資料驅動的文件

[白話]

用JS函式庫把資料與文件中的視覺元素綁定在一起

HTML

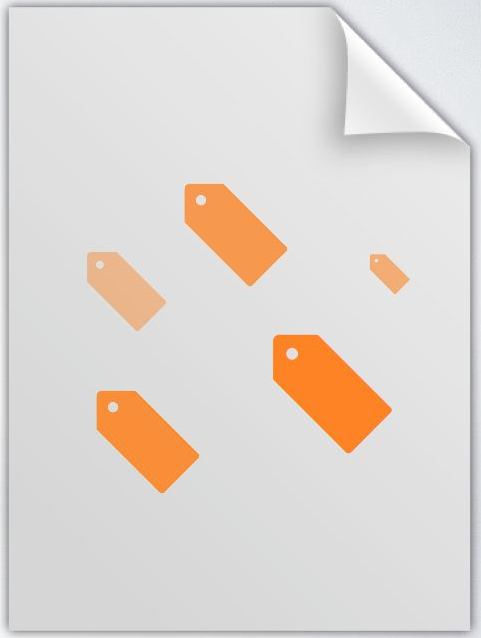


HTML - 超文本標籤語言

HyperText Markup Language

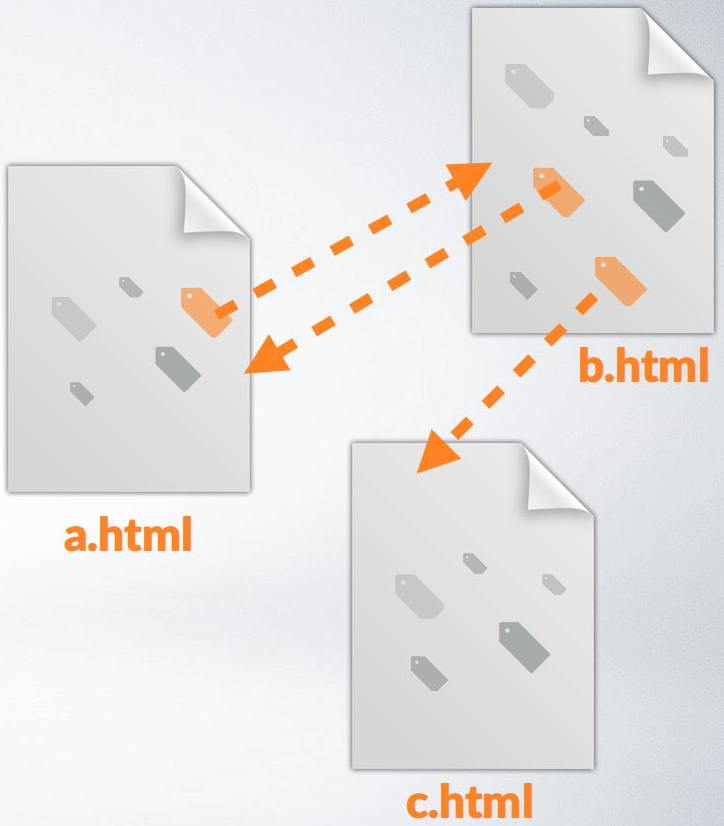
HTML
超文本標籤語言

Ex. <a>,
, <hr>...



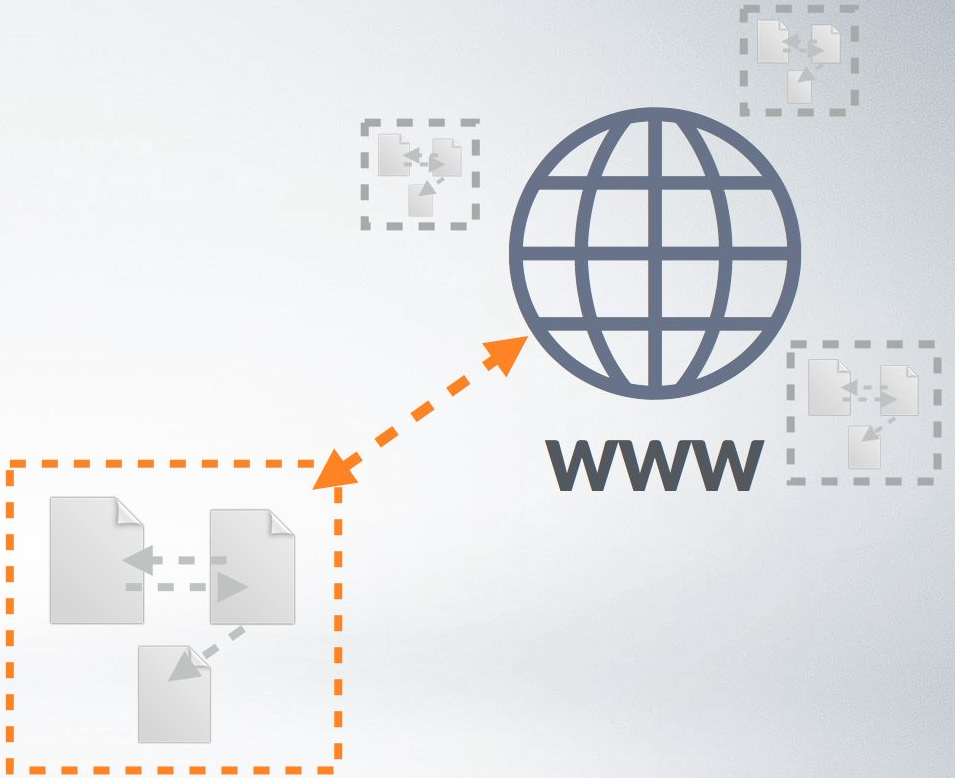
HTML
超文本標籤語言

Ex. a.html <- - -> b.html



HTML(s) -> 網站

網站由許多網頁們所構成

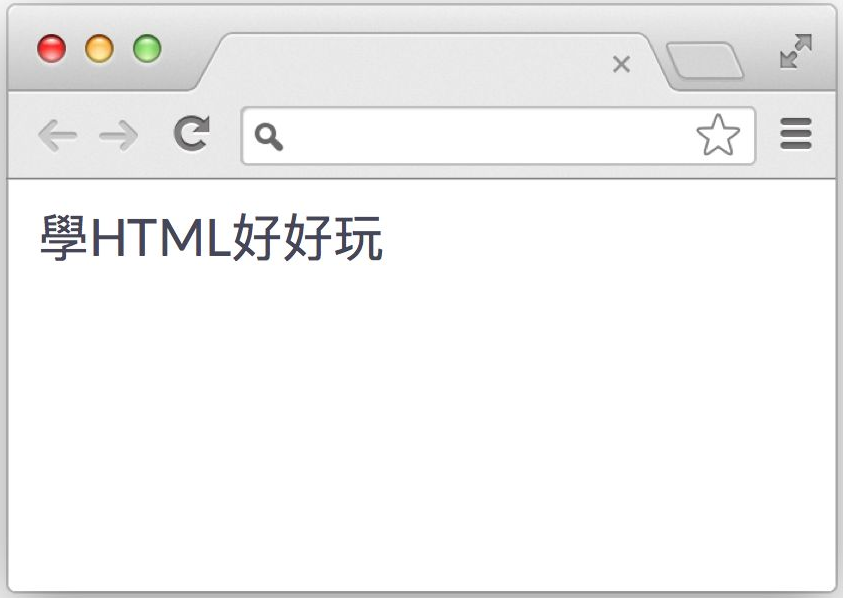


一切從 HTML 的標籤開始

標籤

 起始標籤

 結束標籤

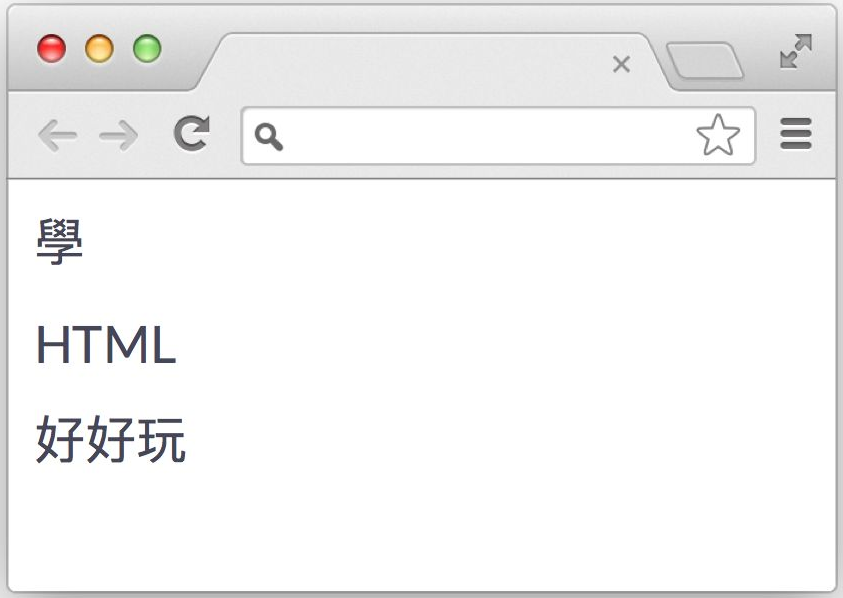


標籤含內容

 學HTML好好玩

- 起始標籤
- 內容
- 結束標籤

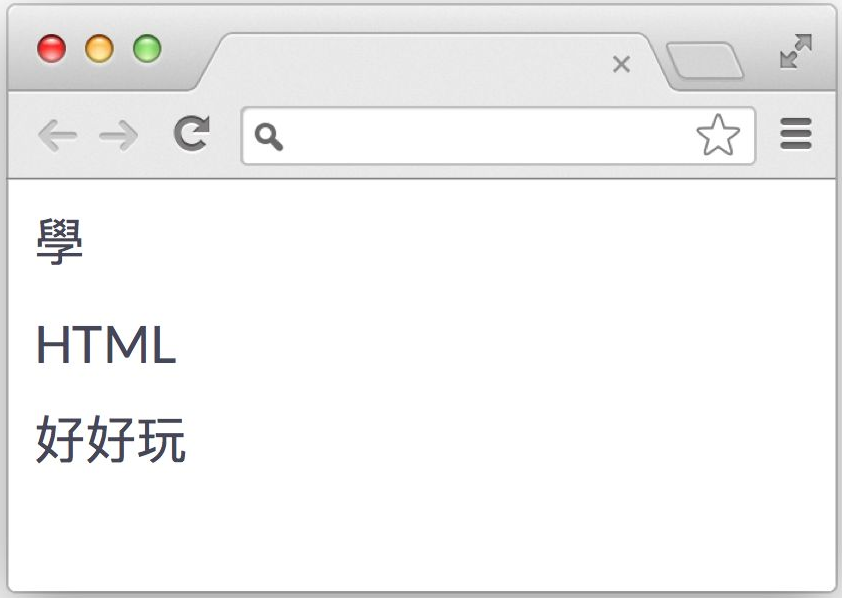
巢狀標籤




```
<div> 學<p>HTML</p>好好玩 </div>
```

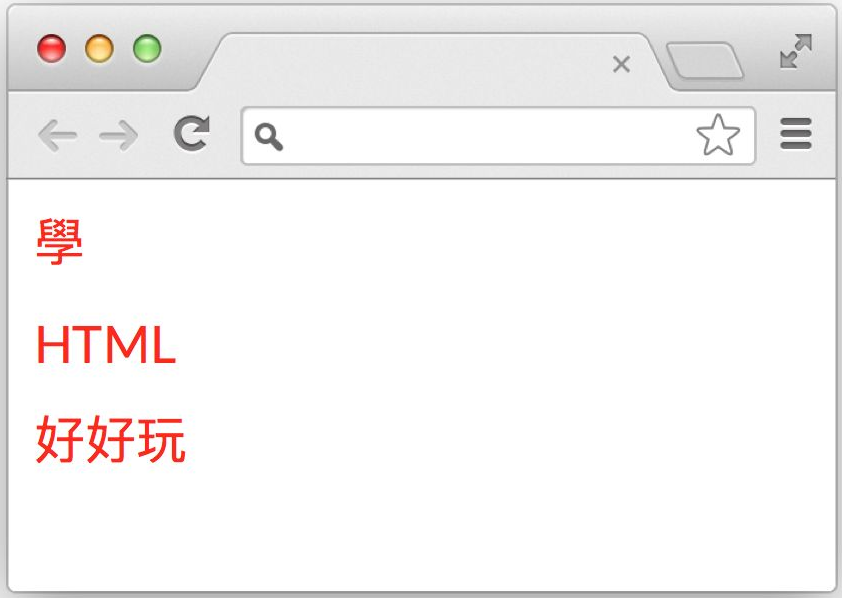
<p> 標出「段落」

排版



排版不影響呈現結果

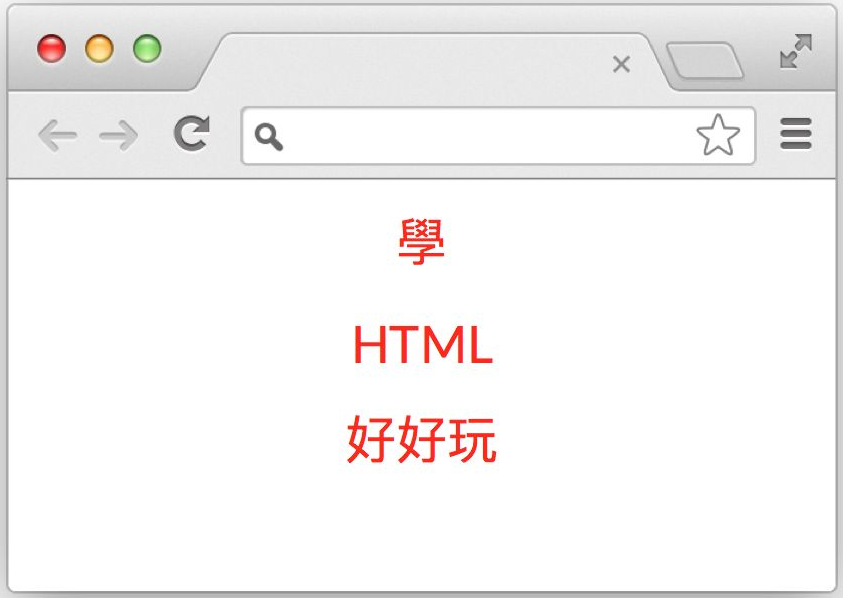
```
<div>
    學
    <p>HTML</p>
    好好玩
</div>
```



屬性　　值

```
<div color="#f00">
    學
    <p>HTML</p>
    好好玩
</div>
```

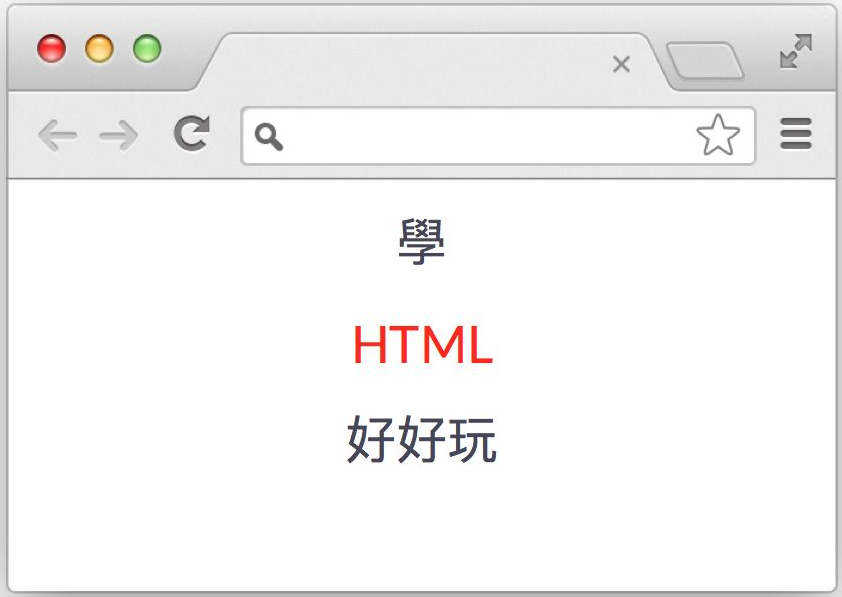

注意:在此之color屬性已不適用於HTML5中，僅因舉例容易!



屬性1 | 值1 | 屬性2 | 值2

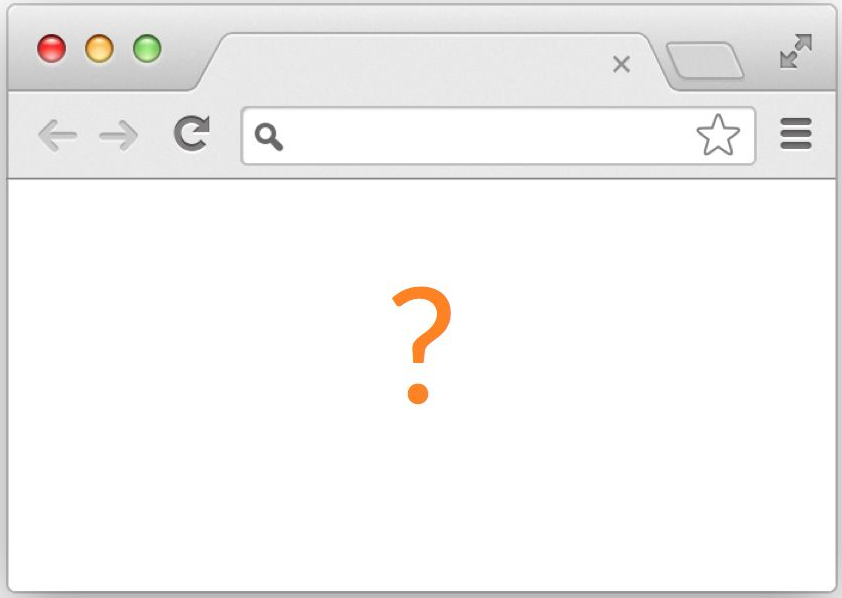
```
<div color="#f00" align="center">
    學
    <p>HTML</p>
    好好玩
</div>
```


注意:在此之color, align屬性已不適用於HTML5中，僅因舉例容易!




```
<div align="center">
    學
    <p color="#f00" >HTML</p>
    好好玩
</div>
```

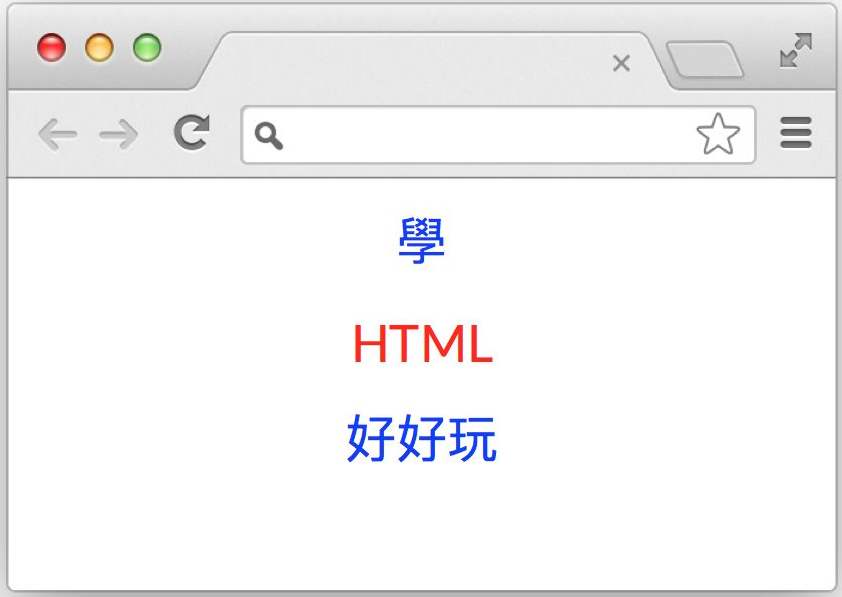

注意:在此之color, align屬性已不適用於HTML5中，僅因舉例容易!

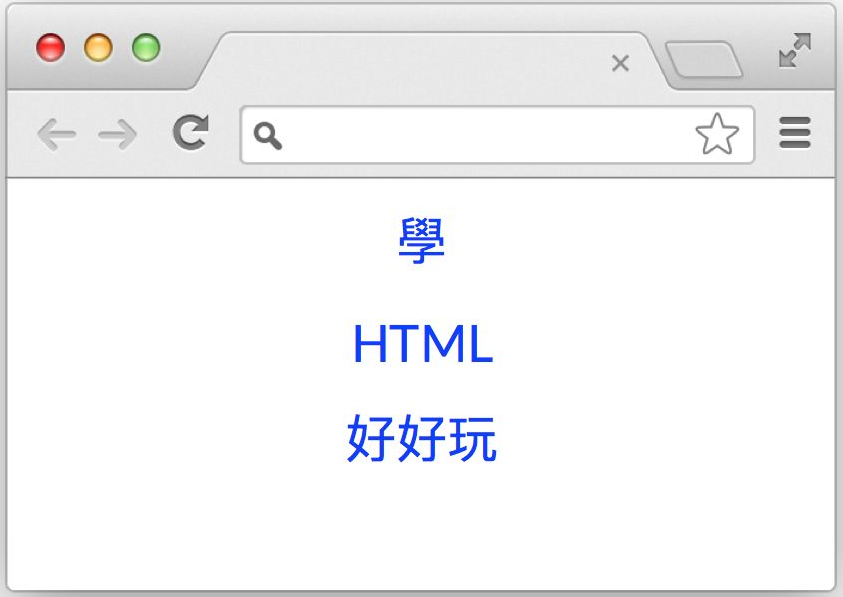


1 2

```
<div color="#00f" align="center">
    學
    <p color="#f00" >HTML</p>
    好好玩
</div>
```


注意:在此之color, align屬性已不適用於HTML5中，僅因舉例容易



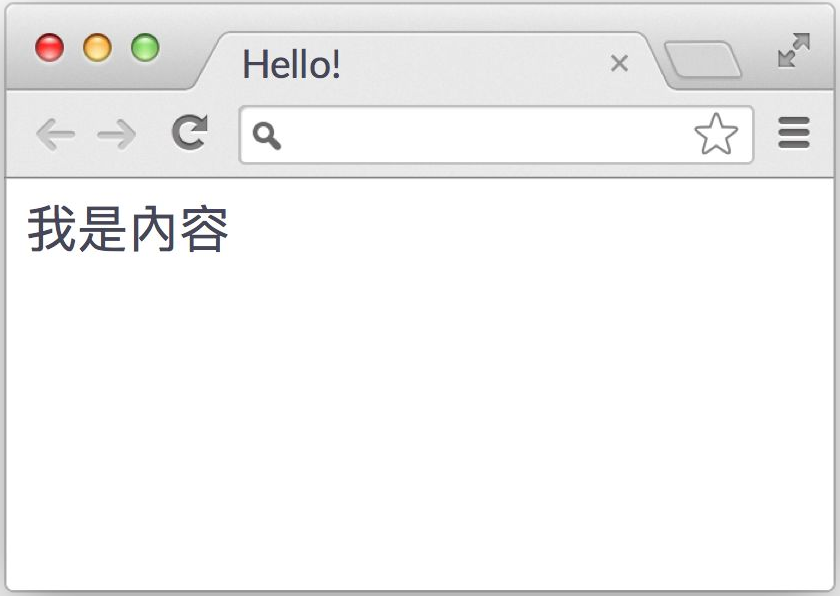


A

```
<div color="#00f" align="center">
        學
        <p color="#f00" >HTML</p>
        好好玩
</div>
```

B

HTML文件的架構



架構(4部份)

```
<!doctype html>   1.宣告文件類型:HTML
<html>            2.根元素
  <head>          3.標頭
    <meta charset="UTF-8">
    <title>Hello! </title>
  </head>
  <body>          4.主體
    我是內容
  </body>
</html>
```

Data-Driven Documents

D3.js - 資料驅動的文件

[白話]

用JS函式庫把資料與文件中的視覺元素綁定在一起

SVG

產生視覺元素 - SVG

SVG: 可縮放向量圖形
Scalable Vector Graphics

SVG可顯示:

1.向量圖形: 矩形、圓、橢圓、多邊形、直線、任意曲線

2.內嵌圖像，包括PNG、JPEG、SVG等

3.文字物件

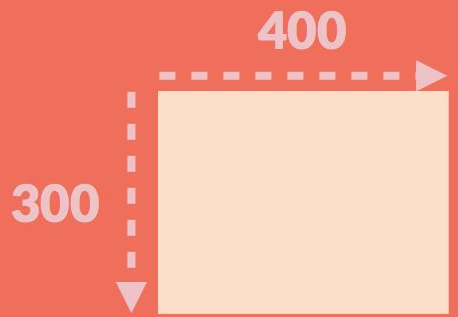
產生視覺元素 - SVG

SVG: 可縮放向量圖形
Scalable Vector Graphics

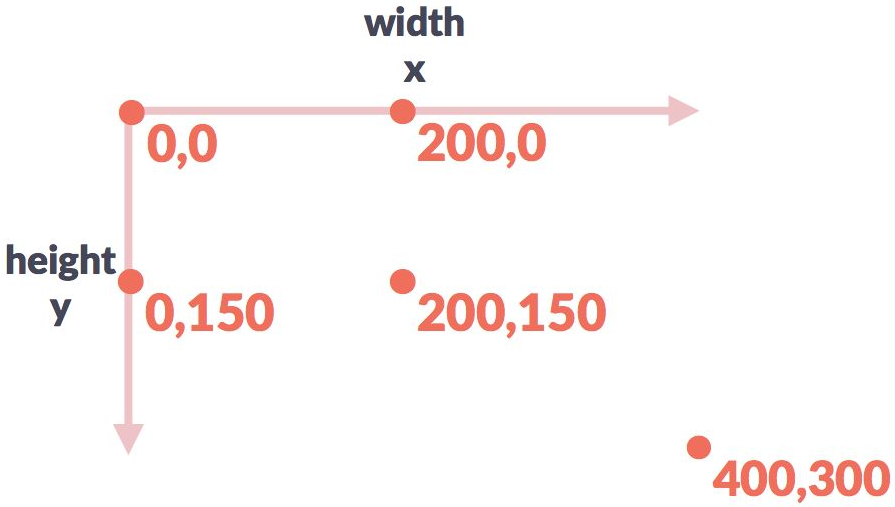
SVG元素

標籤　　屬性

```
<svg width="400" height="300">
</svg>
```



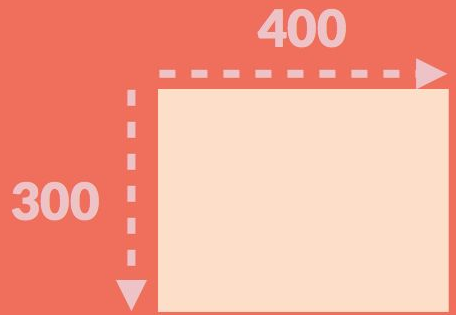
SVG 的座標系統



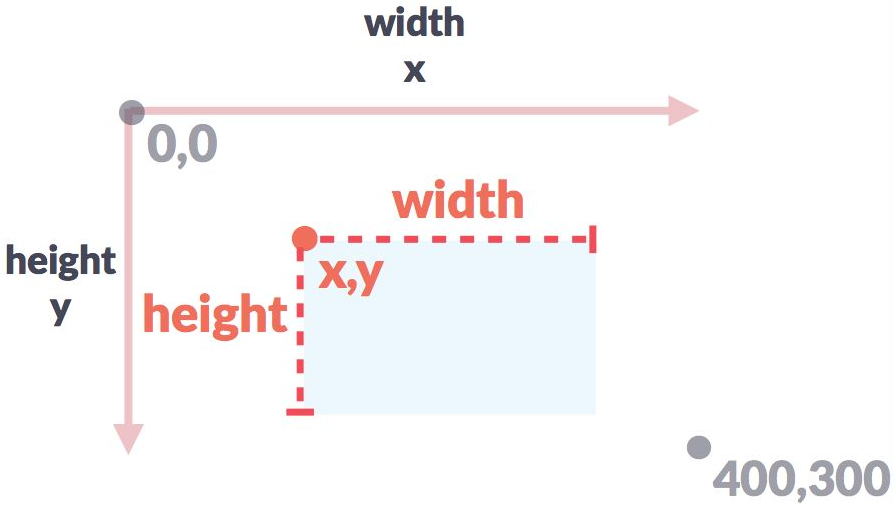
<svg>

標籤　　屬性

```
<svg width="400" height="300">
</svg>
```



用SVG 畫矩形(RECT)

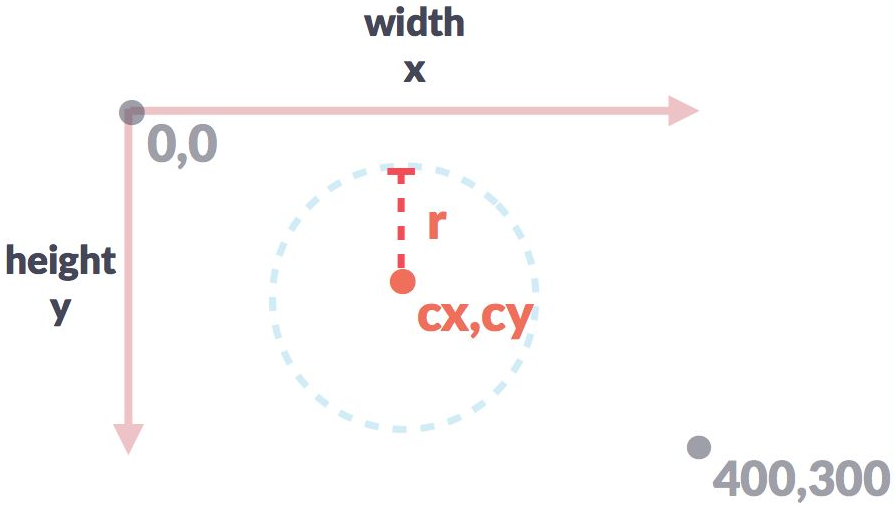


<rect>

```
<svg width="400" height=“300”>
   <rect x="100" y="100"
   width="150" height=“50">
   </rect>
</svg>
```

x: (左上)起始點 x
y: (左上)起始點 y
width:　寬度
height:　高度

用SVG 畫圓形

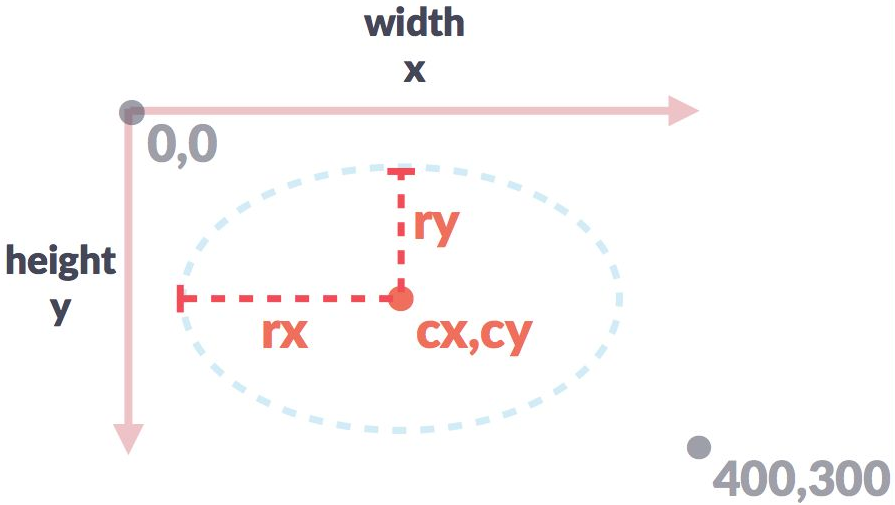


<circle>

```
<svg width="400" height=“300”>
   <circle cx="100" cy="100"
   r="50"></circle>
</svg>
```

cx: Center-X 中心點 x
cy: Center-Y 中心點 y
r: Radius 半徑長

用SVG 畫橢圓形



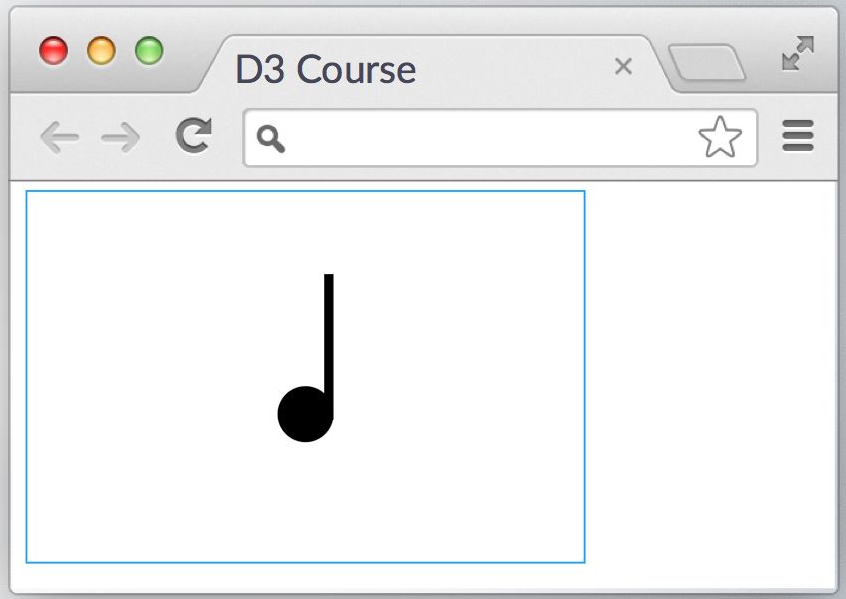
<ellipse>

```
<svg width="400" height=“300”>
   <ellipse cx="100" cy="100"
   rx="150" ry="50"></ellipse>
</svg>
```

cx: Center-X 中心點 x
cy: Center-Y 中心點 y
rx: Radius-X 半徑x長
ry: Radius-Y 半徑y長

動手時間

-靜態SVG使用法-



使用SVG語法畫出來

[溫馨提示]
1.svg畫布寬高: 300 x 200
2.圓形-中心: 150, 120 半徑: 15
3.矩形-起始:160, 45 寬高: 5, 78

D3語法 - 屬性器

Select

```
<body>
  <svg width="400" height="300"></svg>
</body>
```

Append

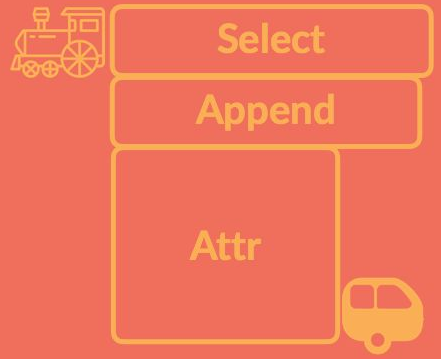
```
<body>
  <svg width="400" height="300">
    <circle></circle>
  </svg>
</body>
```

Attr

```
<body>
  <svg width="400" height="300">
    <circle cx="100" cy="100" r="50"></circle>
  </svg>
</body>
```

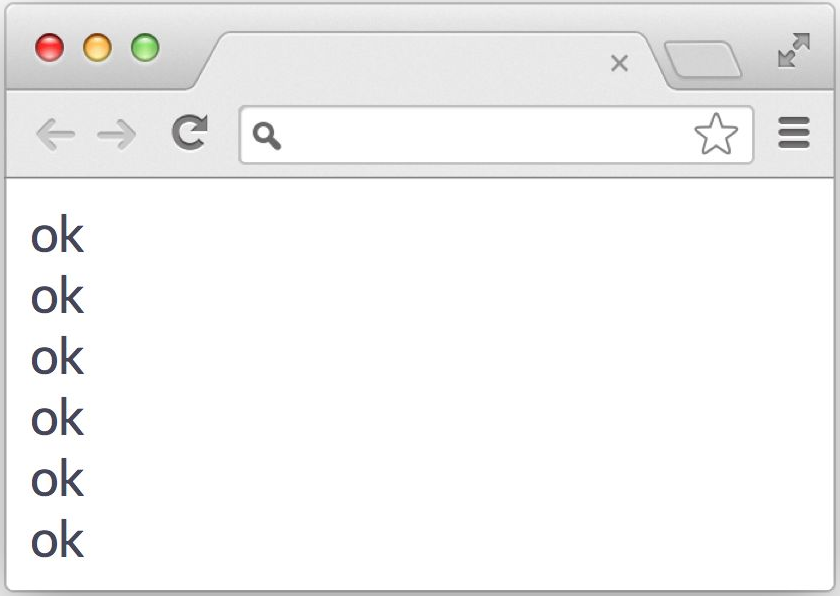
在標籤中增加屬性與值
.attr(屬性, 值)

```
d3.select("svg")
    .append("circle")
    .attr("cx", 100)
    .attr("cy", 100)
    .attr("r", 50);
```



Data Binding

綁定資料到單個視覺元素

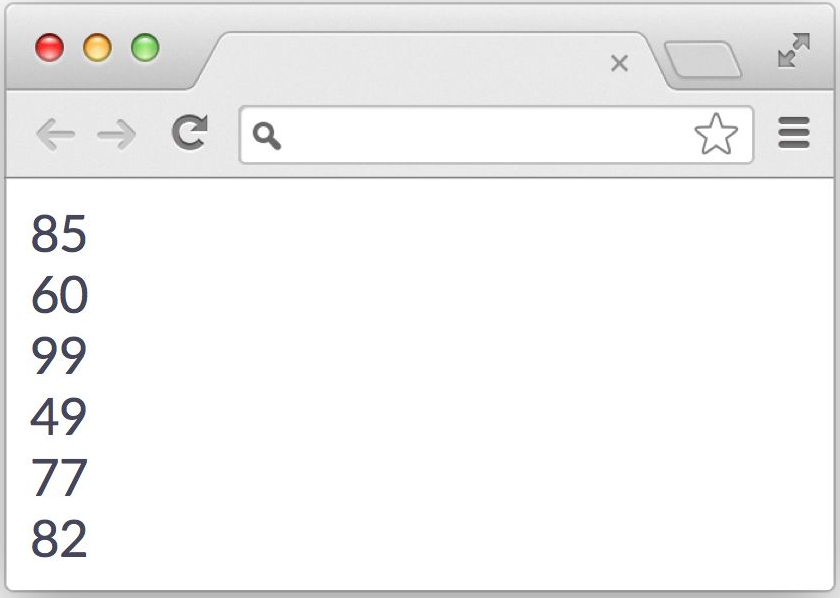


.datum() 單筆資料(變數)

```
var arr = [85, 60, 99, 49, 77, 82];

for(var i=0; i<arr.length; i++){

    d3.select("body")  // 回傳body
      .append("div")   // 回傳body>div
      .datum(arr[i])   // 回傳body>div
      .text("ok");     // 回傳body>div

}
console.log(selectAll("div") );
```

印出所有div看看！

綁定資料到單個視覺元素



取出已綁定資料

```
var arr = [85, 60, 99, 49, 77, 82];

for(var i=0; i<arr.length; i++){

    d3.select("body")       // 回傳body
        .append("div")      // 回傳body>div
        .datum(arr[i])      // 回傳body>div
        .text(function(d){  // 回傳body>div
            return d;
        });

}
```

綁定資料到多個視覺元素

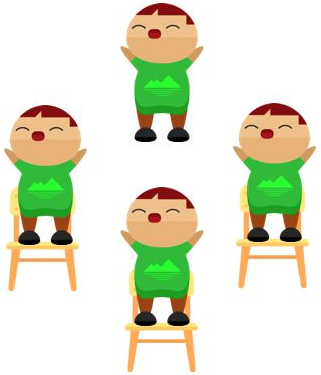
.data() 多筆資料(陣列)

```
var arr = [85, 60, 99, 49, 77, 82];

for(var i=0; i<arr.length; i++){

    d3.select("body")
        .selectAll("div")
        .data(arr)
        .text(function(d){
            return d;
        });

}
```

??

圖解D3綁定

D3綁定像什麼?



圖解D3綁定

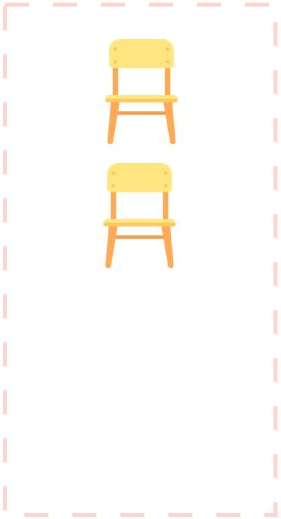
綁定資料到視覺元素上

- 1 -

椅子少

人多

SVG



DATA

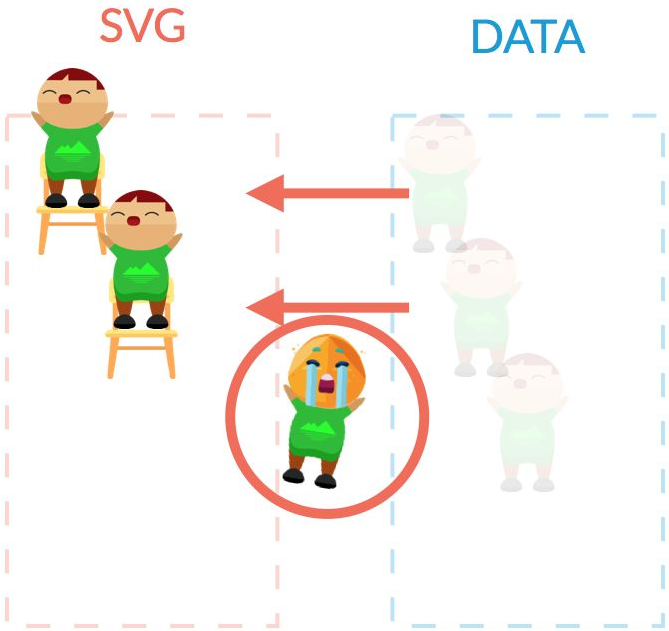
圖解D3綁定

綁定資料到視覺元素上

- 1 -

椅子少

人多



圖解D3綁定

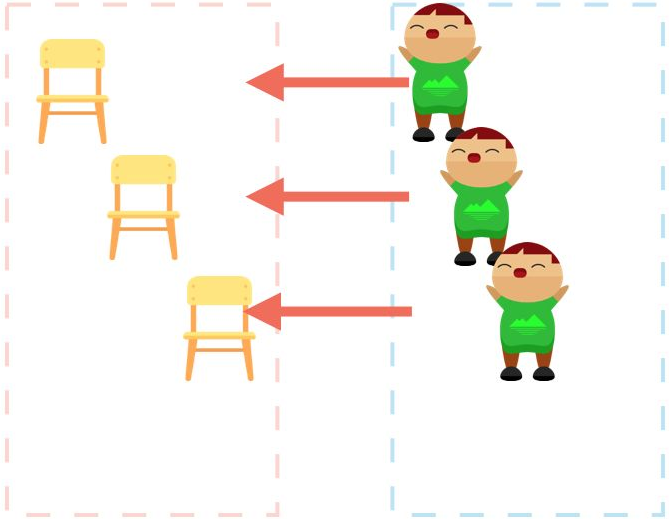
綁定資料到視覺元素上

- 2 -

椅子&人

一樣多

還有什麼情況？



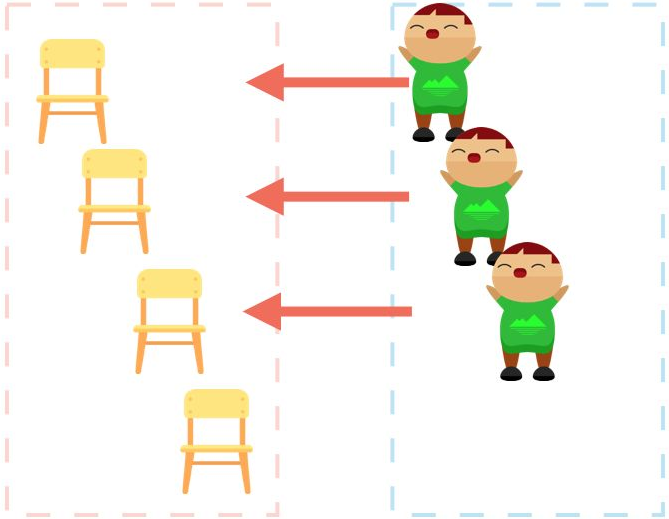
圖解D3綁定

綁定資料到視覺元素上

- 3 -

椅子多

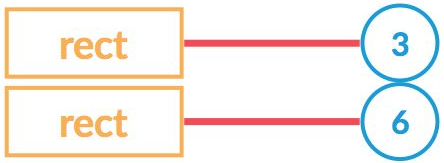
人少



圖解D3綁定

資料與視覺元素間三種狀況

-1-
有元素-有資料

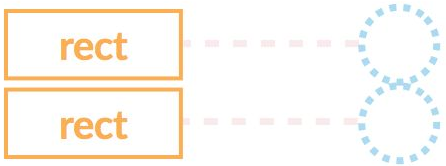


椅子&人
一樣多

-2-
無元素-有資料

椅子少
人多

-3-
有元素-無資料



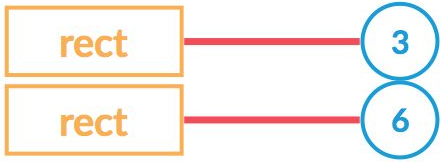
椅子多
人少

圖解D3綁定

資料與視覺元素間三種狀況

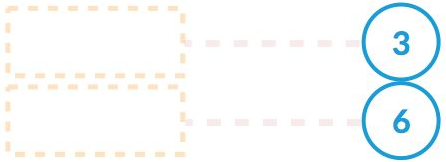
-1-
有元素-有資料

剛剛好(已滿足)



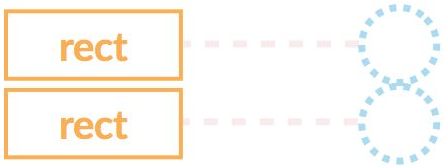
-2-
無元素-有資料

增加足量元素



-3-
有元素-無資料

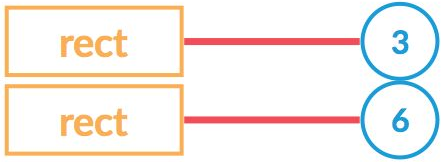
移除多餘元素



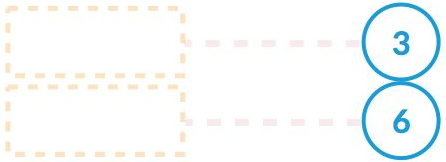
圖解D3綁定

資料與視覺元素間三種狀況

剛剛好(已滿足)
selection



增加足量元素
enter(進入可視化)



移除多餘元素
exit(離開可視化)


```
var selection = d3.select("svg")
                  .selectAll("rect")
                  .data("dataSet");
```

```
selection.enter();
```

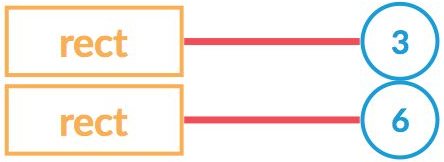
```
selection.exit();
```

圖解D3綁定

資料與視覺元素間三種狀況

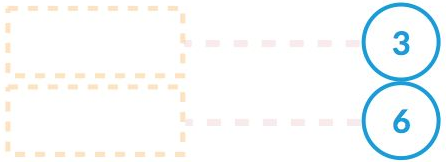
大部分情況！

剛剛好(已滿足)
selection




```
var selection = d3.select("svg")
                  .selectAll("rect")
                  .data("dataSet");
```

增加足量元素
enter

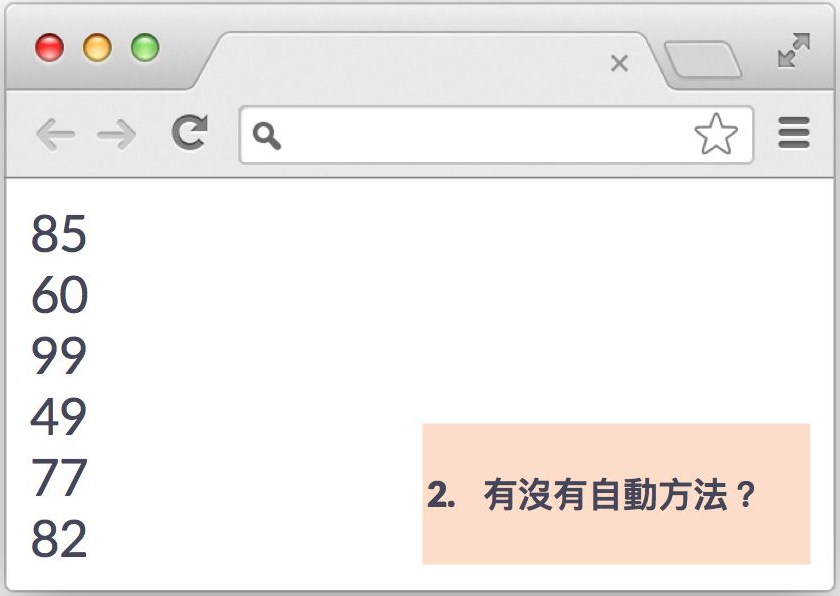



```
selection.enter()
         .append("rect");
```

移除多餘元素
exit


```
selection.exit()
         .remove();
```

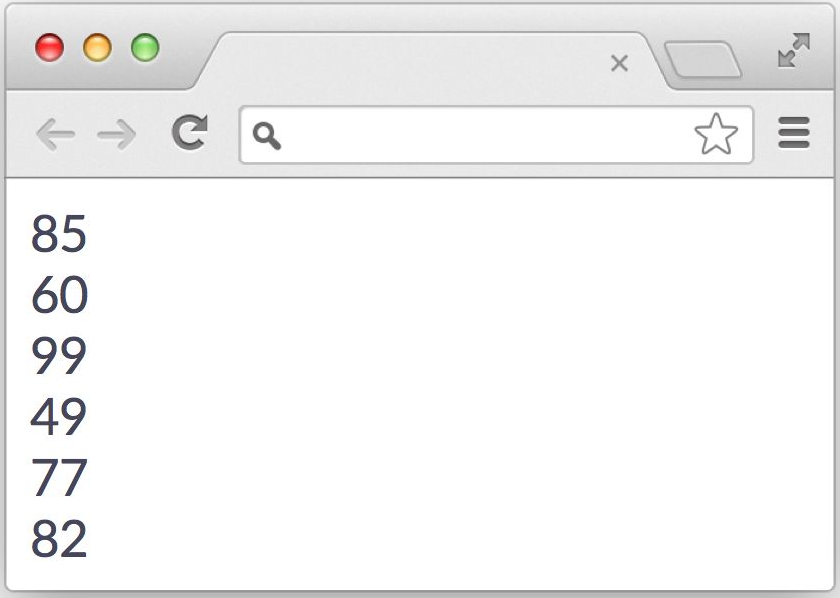
綁定資料到多個視覺元素



.enter() 與 exit()

```
var arr = [85, 60, 99, 49, 77, 82];

var selection = d3.select("body")
    .selectAll("div")
    .data(arr);

 selection.???;
 selection.enter().???;
 selection.exit().???;
```

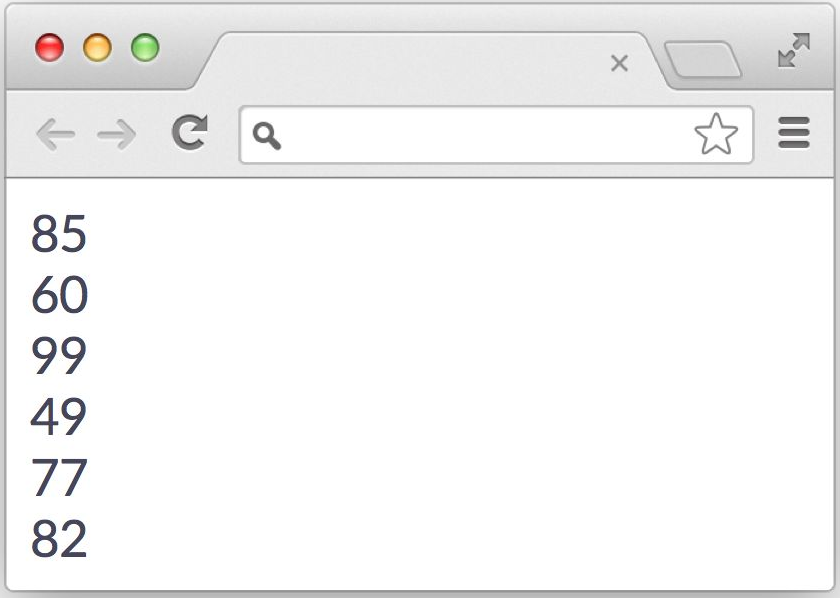
綁定資料到多個視覺元素
各司其職



分類: bind(), render()

```
var arr = [85, 60, 99, 49, 77, 82];

bind(???);
render();

function bind(data){
   var selection = d3.select("body")
                     .selectAll("div")
                     .data(data);
   selection.enter().???;
   selection.exit().???;
}
function render(){
   ???.text(function(d){
      return d;
   })
}
```

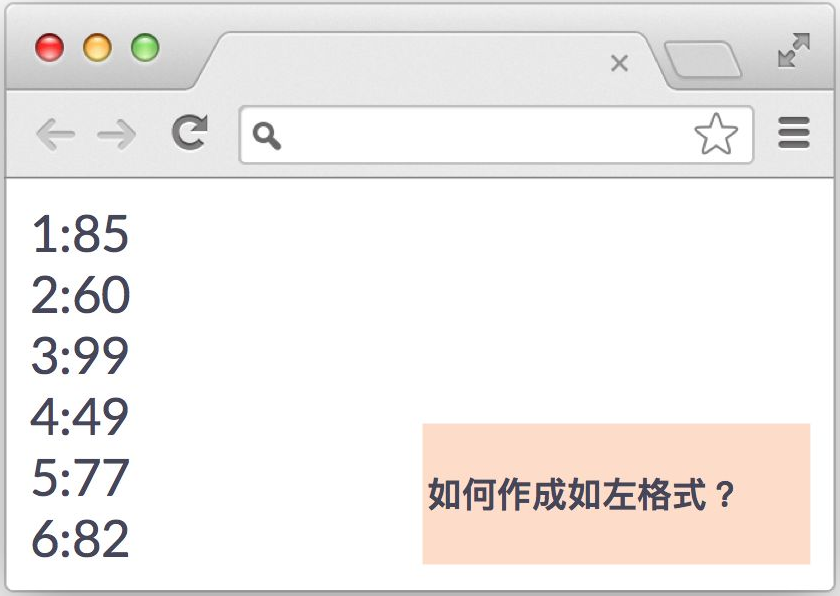
綁定資料到多個視覺元素
各司其職



分類: bind(), render()

```
var arr = [85, 60, 99, 49, 77, 82];

bind(arr);
render();

function bind(data){
   var selection = d3.select("body")
                     .selectAll("div")
                     .data(data);
   selection.enter().append("div");
   selection.exit().remove();
}
function render(){
   d3.selectAll("div").text(function(d){
      return d;
   })
}
```

綁定後的資料操作
如何得知單筆資料順序？



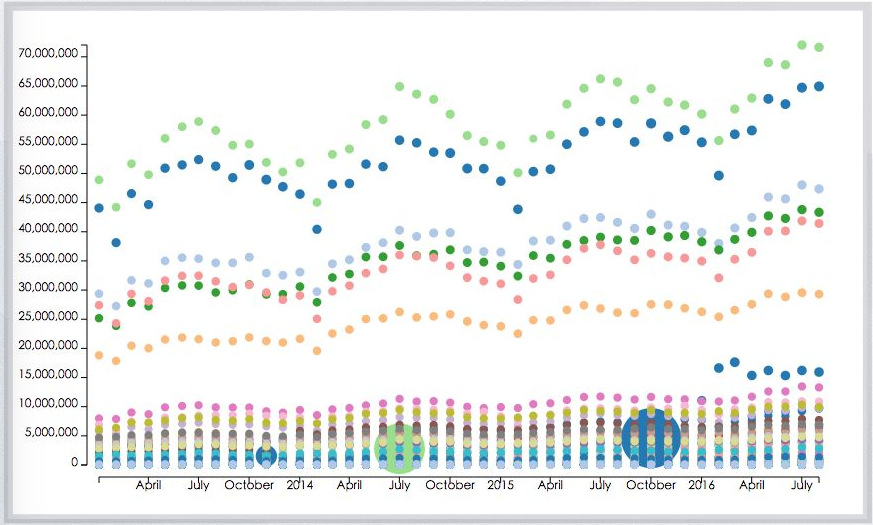
分類: bind(), render()

```
var arr = [85, 60, 99, 49, 77, 82];

bind(arr);
render();

function bind(data){
   var selection = d3.select("body")
                     .selectAll("div")
                     .data(data);
   selection.enter().append("div");
   selection.exit().remove();
}
function render(){
   d3.selectAll("div").text(function(d, i){
      return ???;
   })
}
```

該換真實資料上場了

發票發行
日期(x-axis)與數量(y-axis)
2D散佈圖

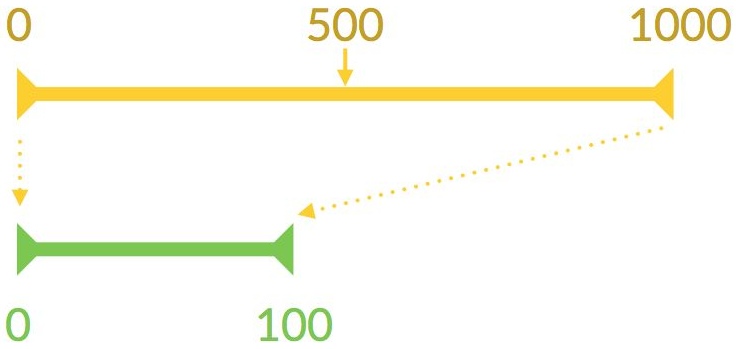
先了解: 1.比例尺 & 2.軸線



Scale & Axis

線性-比例尺

拿來做範圍變換-大變小



線性：d3.scale.linear()

```
var xScale = d3.scale.linear()
        .domain([0, 1000])
        .range([0, 100]);
```

D: 輸入

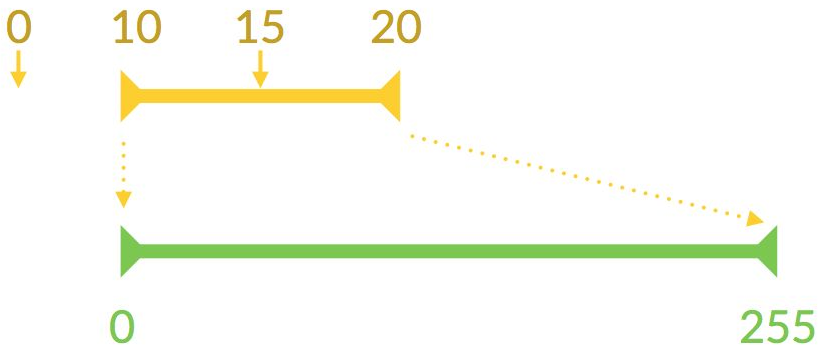
R: 輸出

在console中，用 xScale(455) 試試看

按我連結

線性-比例尺

拿來做範圍變換-小變大



線性：d3.scale.linear()

```
var xScale = d3.scale.linear()
        .domain([10, 20])
        .range([0, 255]);
```

D: 輸入

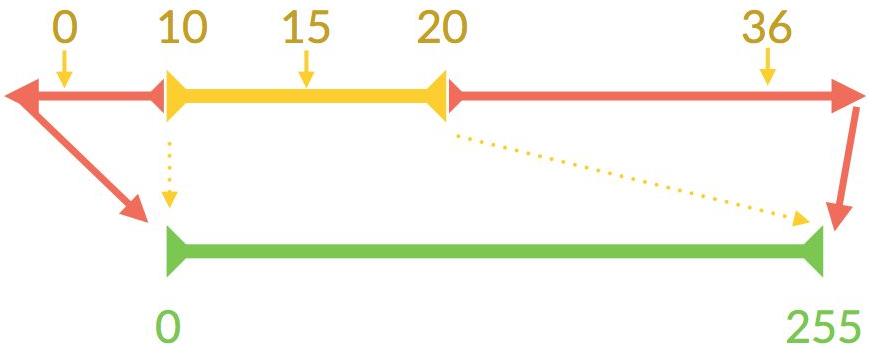
R: 輸出

在console中，用 xScale(30) 試試看

按我連結

線性-比例尺

clamp(true) 左右夾緊



線性：d3.scale.linear()

```
var xScale = d3.scale.linear()
        .domain([10, 20])
        .range([0, 255])
        .clamp(true);
```

D: 輸入

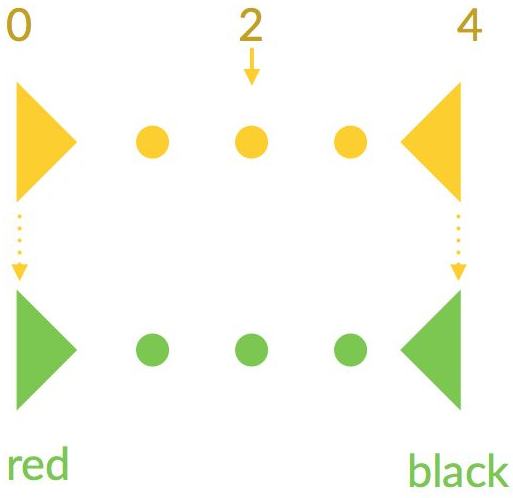
R: 輸出

在console中，用 xScale(30) 試試看

按我連結

序數-比例尺

拿來做範圍變換-不連續對應



序數：d3.scale.ordinal()

```
var index = [0, 1, 2, 3, 4];
var color  = ["red", "blue", "green", "yellow", "black"]

var xScale = d3.scale.ordinal()
        .domain(index)
        .range(color);
```

D: 輸入

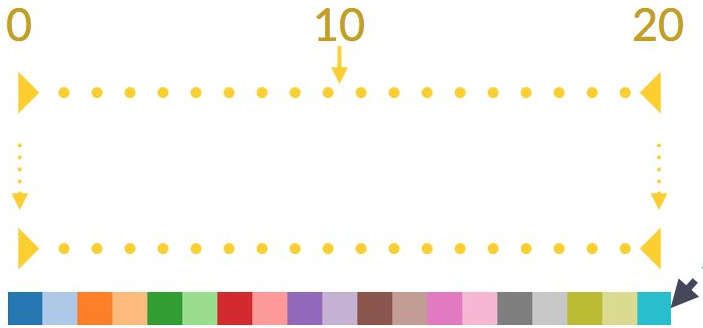
R: 輸出

在console中，用 xScale(3) 試試看

按我連結

序數-比例尺

拿來做範圍變換-內建填色



用console試試看

d3內建填色序數：d3.scale.category20()

var fScale = d3.scale.category10();

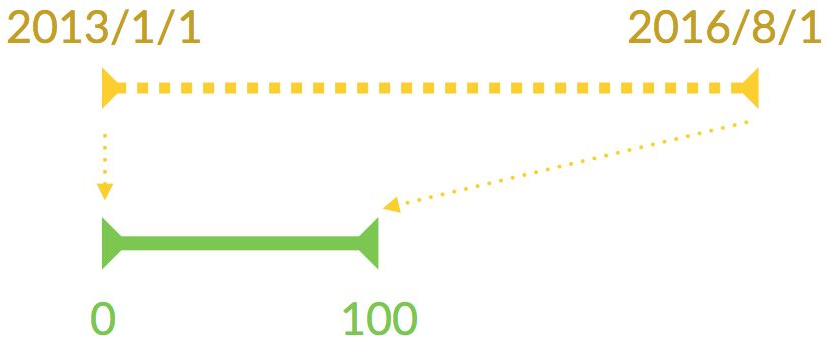
d3.scale.category20();

d3.scale.category20b();

d3.scale.category20c();

日期-比例尺

拿來做範圍變換-時間對應



日期：d3.time.scale()

```
var xScale = d3.time.scale()
        .domain([
             new Date("2013-01-01"),
             new Date("2016-08-01")
        ])
        .range([0, 100]);
```

D: 輸入

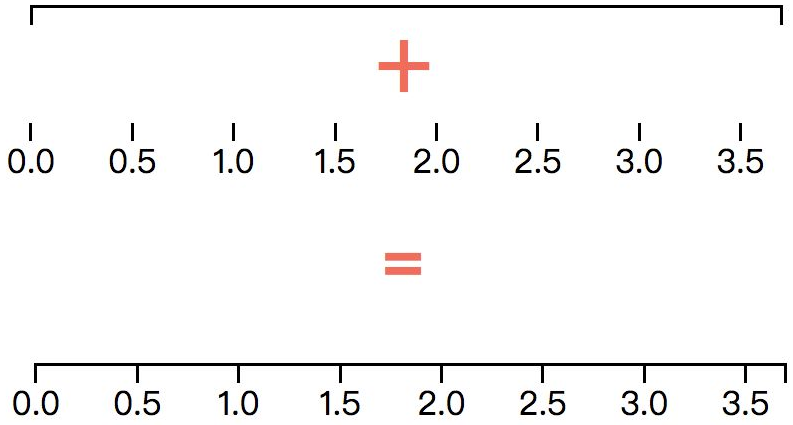
R: 輸出

在console中，

用 xScale(new Date("2014-10-30")) 試試看

Axes-座標軸

在svg畫布中出現比例尺



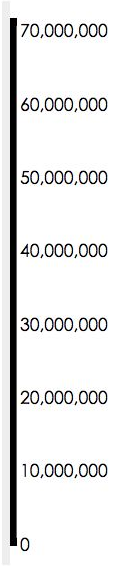
查看範例

```
<g>
    <!-- 第一個刻度 -->
    <g>
    <line></line>  <!-- 第一個刻度的直線 -->
    <text></text>  <!-- 第一個刻度的文字 -->
    </g>
    <!-- 第二個刻度 -->
    <g>
    <line></line>  <!-- 第二個刻度的直線 -->
    <text></text>  <!-- 第二個刻度的文字 -->
    </g>
    ...
    <!-- 坐標軸的軸線 -->
    <path></path>
</g>
```

<g>：分組(group)元素，是 SVG 中的元素。
此元素是將其他元素進行組合的容器。

Axes-座標軸 - 五步驟

在svg畫布中出現比例尺



第1步-產生軸線:

```
var xAxis = d3.svg.axis()
        .scale(xScale)
        .orient("right");
```

針對svg設計的

bottom(預設): 刻度在底部

top: 刻度在頂部

left: 刻度在左方

right: 刻度在右方

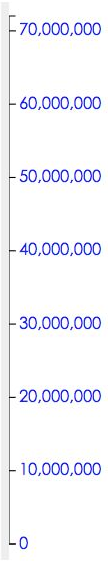
第2步-畫在svg上:

```
d3.select("svg").append("g").call(xAxis);
```

d3.svg.axis()

Axes-座標軸 - 五步驟

在svg畫布中出現比例尺



第3步-css調整樣式:

本身由path, line, text 構成

```
.axis path, .axis line{
    fill: none;
    stroke: black;
    shape-rendering: auto;
}
```

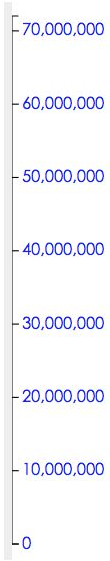
```
.axis text{
    font-size: 11px;
    fill: blue;
}
```

```
.attr("class" ,"axis")
```

```
d3.select("svg")
    .append("g")
    .attr("class" ,"axis")
    .call(yAxis);
```

Axes-座標軸 - 五步驟

在svg畫布中出現比例尺



第4步-刻度數量(參考值):

第5步-軸線位移:

.ticks( )

```
var xAxis = d3.svg.axis()
        .scale(xScale)
        .orient("bottom")
        .ticks(5);
```

.attr("transform", "translate("+...)

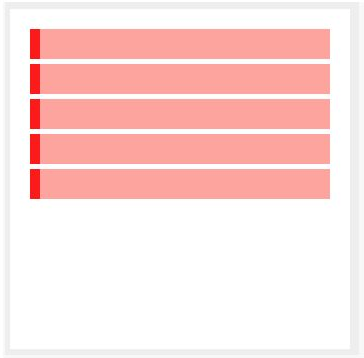
```
d3.select("svg")
    .append("g")
    .attr("class" ,"axis")
    .attr("transform" ,"translate("+x+","+y+")")
    .call(xAxis);
```

用D3玩動畫

讓視覺元素動起來

基本D3動畫

開始與結束兩個狀態



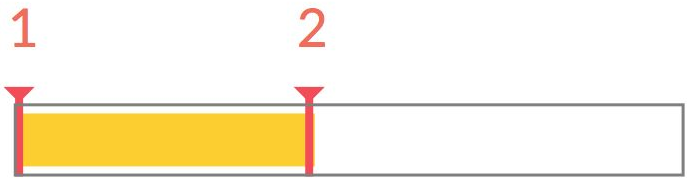
[注意]屬性需有具體值才會有效果

d3.selection.transition()

```
d3.selectAll(“rect#bar")
    .attr({width: 10})
    .transition()
    .attr({width: 300});
```

基本D3動畫

開始與結束兩個狀態



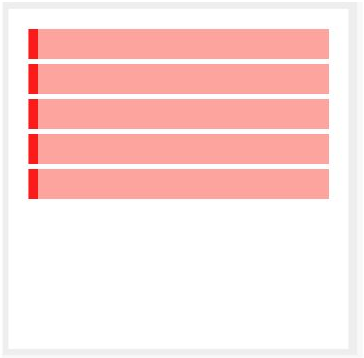
d3.selection.transition()

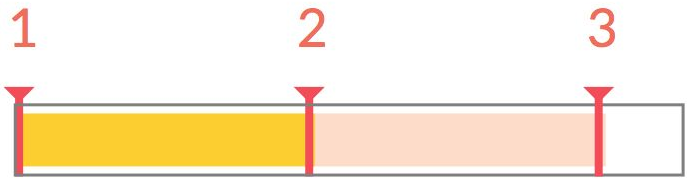
也可以分開設定

```
d3.selectAll(“rect#bar")
1 ⟶ .attr({width: 10})

d3.selectAll(“rect#bar")
    .transition()
2 ⟶ .attr({width: 300});
```

基本D3動畫

開始與結束兩個狀態





d3.selection.transition()

可以接續串接成連續動畫

```
d3.selectAll("rect#bar")
    .transition().attr({width: 300})
    .transition().attr({width: 100});
```

基本D3動畫

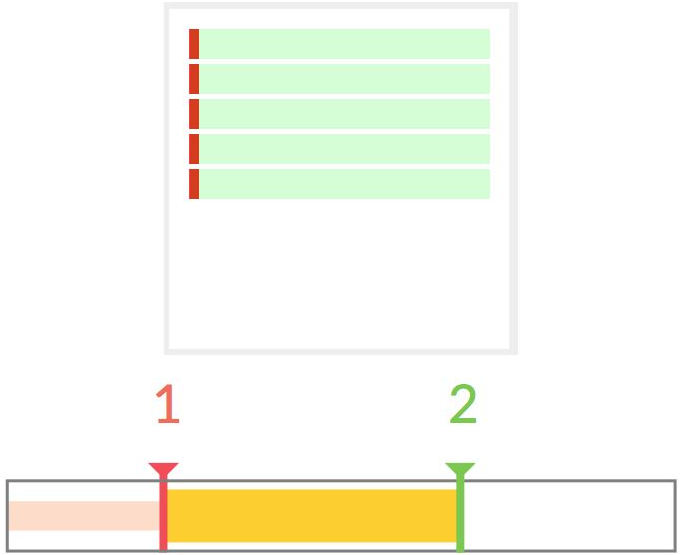
可設定動畫時間長度


```
d3.selection
.transition().duration(ms)
```

```
d3.selectAll("rect#bar")
    .transition().duration(1000)
    .attr({fill: "rgb(0,255,0)"});
```

基本D3動畫

可設定延遲播放時間

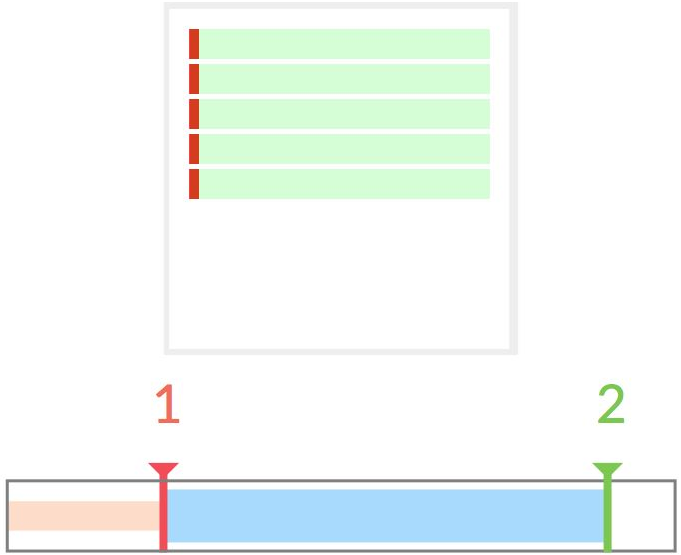



```
d3.selection
.transition().delay(ms)
```

```
d3.selectAll("rect#bar")
    .transition().delay(1000)
    .attr({fill: "rgb(0,255,0)"});
```

基本D3動畫

同時設定延遲與長度

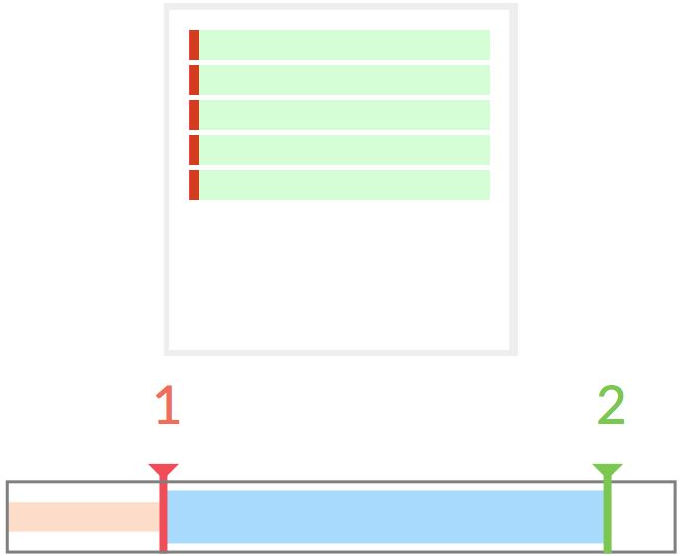


d3.selection
.transition().delay(ms)

```
d3.selectAll("rect#bar")
    .transition()
    .duration(1000).delay(1000)
    .attr({fill: "rgb(0,255,0)"});
```

基本D3動畫

同時設定延遲與長度-時間差



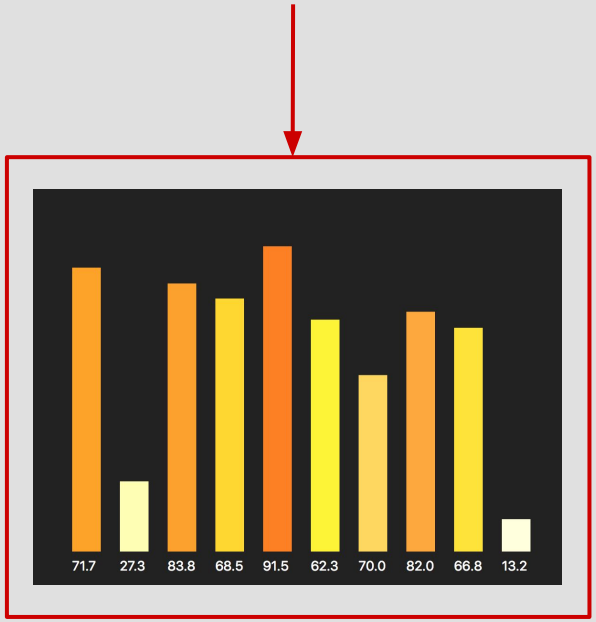
d3.selection
.transition().delay(ms)

括號可以放函式(取得綁定資料)

```
d3.selectAll("rect#bar")
    .transition()
    .duration(1000)
    .delay(function(d, i){
        return i*500;
    })
    .attr({fill: "rgb(0,255,0)"});
```

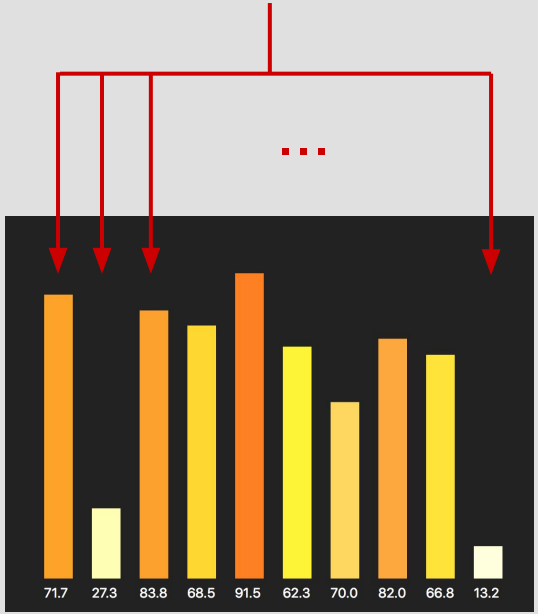
Vue.js

計算資料，自動指定，
自動產生



D3.js

操作DOM(高級版)，
函式多元




Thank you for listening