第一章

簡介

D3.js（簡稱 D3，也就是 Data-Driven Documents 的縮寫）是一套用來操作 Document Object Model（DOM）樹狀資料，使其可以透過視覺化來表現資訊內容的 JavaScript 函式庫，它已然成為在網頁上將資料視覺化的業界標準了。

儘管 D3 很受歡迎，卻有著學習曲線陡峭的壞名聲。我不認為這是因為 D3 太過複雜（它的確不是），甚至不是因為它有非常大量的 API（它是有很多 API，但這些 API 有很好的結構與設計）。對於新手而言，我覺得許多不好的經驗是因為一些不正確的假設所造成的。因為 D3 經常被用於建立一些令人印象深刻的圖表，所以很容易地，甚至是自然而然被認為是一個「圖形函式庫」，它的能力是有助於處理圖形元件以及提供高階通用繪圖型態的支援。以此種期待來研究 D3，對於新手而言，會被許多冗贅的必要設定，例如元件的顏色這麼基礎的工作而感到不快，到底這是啥玩意兒？為什麼不能就簡單的提供像畫布這樣的元件，然後就在上面完成繪圖呢？

此種誤解肇因於 D3 並不是一個圖形函式庫：事實上它是一個用來操作 DOM 樹的 JavaScript 函式庫！它的基本建構方塊並不是圓和矩形，而是節點（node）和 DOM 元素（element）；而且它的典型作業中並不包含把圖形畫在畫布上，而是透過屬性（attribute）建立元素的樣式。「目前的位置」並不是透過畫布上的 xy 座標，而是透過在 DOM 樹中節點的選擇。

碁峰 www.gotop.com.tw

從前述的觀察讓我相信，對於許多新手的第二個主要的挑戰是：D3 是　個網頁技術，而且是建立在其他網頁技術集合上；DOM API 和事件模型（event model）、CSS（Cascading Style Sheet）選擇器（selector）和屬性項（property），JavaScript 物件模型（object model），當然還包括 SVG（Scalable Vector Graphics）。在許多的例子中，D3 提供相對輕巧的層以環繞這些技術，而且它自己的設計頻繁地反應了底層的 API。這樣的組合產生了　個大型的而且有那麼　些異質的環境。如果你已經熟悉現代網頁技術 HTML5 的所有技術端，那麼你很快就能上手，但如果你還不熟悉它們，那麼如果缺乏　個獨特的　致化抽象層，可能會變得非常令人迷惑。

所幸，你不需要深入研讀所有這些相依的底層技術：D3 已經將它們變得很易用，而且在其上提供相當程度的　致化和抽象化。無疑地，唯　　個還不足以即興發揮的區塊是 SVG。你必須要對於 SVG 有相當程度的理解，不僅僅是它的表現元素，還包括那些用來控制在圖形中如何組織資訊的結構化元素。我試著把這些必需的資訊整理在　起放在附錄 B。如果你並不是那麼熟悉 SVG，筆者建議你在開始學習本書接下來的內容之前，先閱讀附錄的部份，這將會對你很有幫助！

誰適合這本書？

本書的潛在讀者是那些想要在自己的工具箱中加入 D3 的程式設計師和科學家們。我假設你是　個具有還不錯的專業能力之程式設計師，也對於資料和繪圖的處理都還算上手，於此同時，我只期待你對於現代化專業級網頁開發有些粗略的瞭解。

閱讀本書所需要具備的知識和技能列示如下：

- 至少要有　種以上的程式語言知識（但不　定要是 JavaScript），而且有自信可以從程式語言參考資料中找出相關的語法。
- 熟悉現代化的程式設計概念，也就是，不單單只是迴圈、條件式、以及　般的資料結構，還包括閉包（closure）以及高階函式（higher-order functions）。
- 對於 XML 以及它施加在文件上的階層結構有基本的瞭解。我假設讀者們知道 DOM 以及它是如何把網頁中的元素當作是樹狀結構中節點的操作方式，但是我並不預期讀者會熟悉原始的 DOM API 或是任何　種現代化的取代機制（像是 jQuery）。
- 會使用簡單的 HTML 和 CSS（你應該可以識別以及使用 `<body>` 以及 `<p>` 標籤等等），以及對於 CSS 語法和它的機制有　些熟悉。

碁峰 www.gotop.com.tw

⓫ 最後，基於結合的資料點之屬性值設定了每　個圓形的位置。每　個 `<circle>` 元素的 `cx` 和 `cy` 屬性依照在資料檔案中的項目來設定：不使用提供固定值的方式，在這裡我提供的是**存取器函式**（*accessor function*），給它　個在資料檔中的項目（也就是，　列的紀錄），會傳回那個資料點的相對應值。

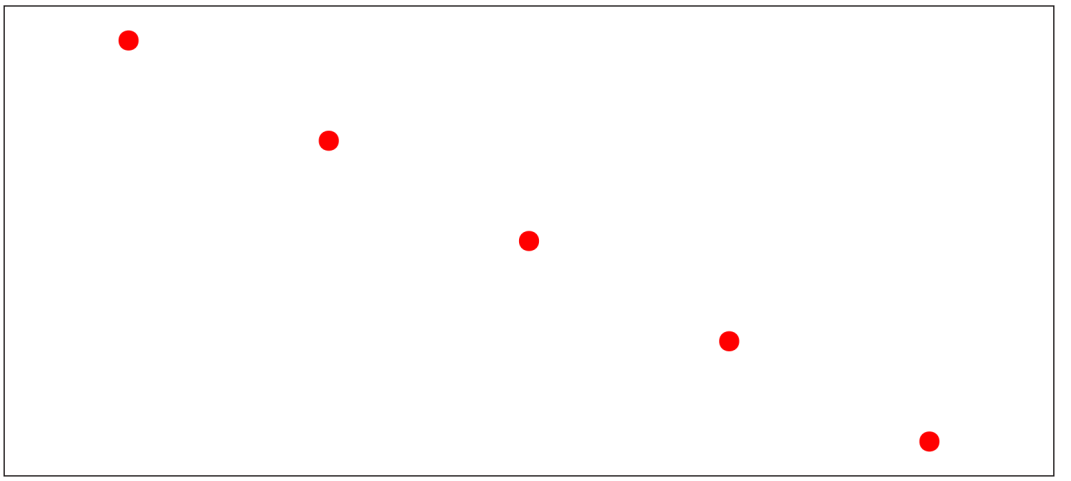

圖 2-1　一個簡單資料集所繪出的圖形（請參考範例 2-3）

說實在的，要畫出這麼簡單的圖形居然需要這麼多麻煩的作業。很顯然地在此也就可以很清楚地看出了：D3 並不是　個圖形函式庫，更不是　個繪圖的工具，而是　個用來操作 DOM 樹的函式庫，讓它可以把其中的資訊進行視覺化的作業。你將會發現自己好像總是在操作 DOM 樹的某些部份（透過 selection）。你也將會注意到 D3 並不會節省你的打字數，它需要使用者去操作屬性值和　步步去編寫存取的函式。同時，我認為可以公平地說，這些程式碼雖然冗長，但卻是乾淨且相當直接的。

如果你實際練習了這些例子，可能會遇到　些意料之外的細節。例如，`tsv()` 函式很挑剔：欄**必須**以定位符號作為區隔，空白**不能**被忽略，**必須**要有標題列等等。最後，仔細觀察資料集和圖形，你應該會注意到這張圖並不正確—它是上下顛倒的！這是因為 SVG 使用的「圖形座標」如平常　樣水平軸是從左到右，但是垂直軸卻是由上而下。

在有了　開始的印象之後，就讓我們持續往下進行針對第二個例子的探索。

碁峰 www.gotop.com.tw

❿ factory 函式 `d3.axisRight( scale )` 傳回　個函式物件，它會產生具有所有部件的軸。它需要　個 SVG 容器（通常都是 `<g>` 元素）作為參數，然後將建立所有軸的元素作為此容器元素的子項目。換句話說，它是　個已經在前面定義好的**元件**（*component*，詳細的資訊請參閱第 7 章）。

⓫ 對於在圖形右邊的軸，容器元素必須被移動到正確的位置上。此點可以使用 SVG 的 `transform` 屬性來完成。

⓬ 此點是新的：不使用 `axMkr` 函式明確地把 `<g>` 元素作為參數，取而代之的是把 `axMkr` 函式作為參數傳遞到 `call()` 函式中。`call()` 函式是 `Selection` API（請參閱第 3 章）的　部份；它有和 JavaScript 語言中類似的功能模式。它呼叫它的參數（必須是函式），並提供目前的 selection 作為其參數。此種型式的「合成（synthetic）」函式呼叫在 JavaScript 中相當常見，最好能夠習慣它。此種類型的程式設計方式的其中　個優點是，它支援方法鏈（method chaining），正如你將在下　個項目中會看到的使用方式。

⓭ …我們在圖形的底部添加水平軸。為了顯示可能性，函式呼叫的順序已經互換：先呼叫 axis 元件，然後再執行轉換。此時，我們也免除了 `axMkr` 的輔助變數。這可能是編寫這個程式的最慣用的方式[3]。

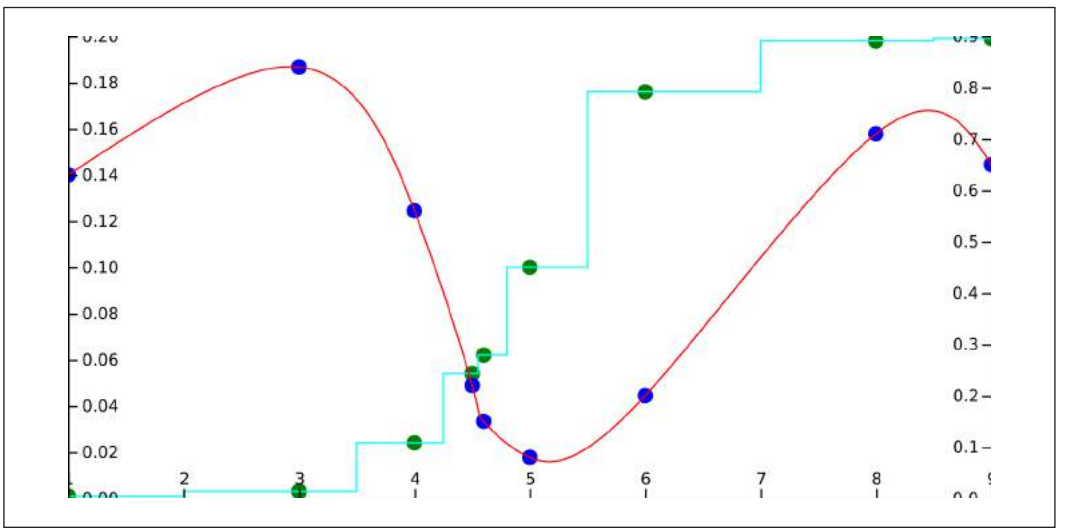

圖 2-3　一個改良過的圖形，包含座標軸並使用不同類型的曲線（和圖 2-2 以及範例 2-6 比較）

3　事實上，`drawData()` 函式一般都是使用此種方式呼叫：`g1.call( draw Data, d=> scY1(d["y1"]), d3.curveStep )`

碁峰 www.gotop.com.tw

產生的圖形如圖 2-3 所示。它並不完美但卻具有許多的功能，分別使用不同的曲線展現了兩個資料集。要改良此圖形外觀的第一步可以先從在 SVG 區域中加入一些留白圍繞在實際資料的周圍，以讓軸和刻度標籤之間多一些空間。（試試看！）

第三個範例：讓清單項目動起來

我們的第三個也是最後一個例子看起來可能比其他兩個例子更加地奇特，但是它仍說明了以下兩個重點：

- D3 不只能產生 SVG。它可以用來操作 DOM 樹的任何部份。在這個例子中，我們將使用 D3 操作一般的 HTML 清單項目。
- D3 使得建立響應式以及動態的文件變得容易：也就是說，文件可以依據使用者的事件（例如滑鼠點擊）來加以回應，並可隨時間而改變它們的外觀。在此，我們將只會提供那些所有驚人的可能性中的一個簡要的預覽：我們將會在第 4 章中更詳細地說明這個主題。

利用 D3 建立 HTML 元素

為了做些變化，而且因為實際上腳本非常地短，這次的 JavaScript 命令將會被直接包含在此頁中，而不會把它放在另外獨立的檔案裡。本範例的全部，包括 D3 指令，都被放在範例 2-7（也請同時參閱圖 2-4）中。

範例 2-7 使用 D3 用來進行非圖形 HTML 的操作（參閱圖 2-4）

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <script src="d3.js"></script>
  <script>
function makeList() {
    var vs = [ "From East", "to West,", "at Home", "is Best" ];   ❶

    d3.select( "body" )                                           ❷
        .append( "ul" ).selectAll( "li" )                         ❸
        .data(vs).enter()                                         ❹
        .append("li").text( d => d );                             ❺
}
  </script>
</head>

<body onload="makeList()" />
</html>
```


碁峰 www.gotop.com.tw

就結構上來講，此範例幾乎是和本章到目前為止的所有範例相同（尤其是範例 2-3），但是在細節上還是有些差異：

❶ 資料集並沒有定義在外部檔案，而是直接放在程式碼中。

❷ HTML 的 `<body>` 元素被選擇作為目標對象的最外層容器。

❸ 此程式碼附加一個 `<ul>` 元素，然後建立一個用來放置清單項目的空的預留位置（使用 `selectAll( "li" )`）。

❹ 像之前一樣，資料集被綁定在 selection，並且取得沒有匹配 DOM 元素的資料點集合。

❺ 最後，一個清單項目被附加到每一個資料點，而其內容（也就是，在此範例中，是清單項目的文字）就被從資料集中把值套用到其中。

以上所有的內容都和之前做過的相當地類似，除了最後的結果頁是一個純文字的 HTML 之外。基於此點，D3 也被證明是一個用來操作 DOM 非常好的泛用（也就是非圖形）工具。

- From East
- to West,
- at Home
- is Best

圖 2-4 在 HTML 中的清單（參閱範例 2-7 和 2-8）

建立一個簡單的動畫

讓此文件回應使用者事件並不會太費力。請將在範例 2-7 中的 `makeList()` 函式替換成在範例 2-8 中的函式。你可以藉由點擊任一清單項目來將其文字顏色在黑色和紅色之間進行切換。此外，這個改變不會立即生效：文字的顏色將會在 2 秒之間漸進地變更。

碁峰 www.gotop.com.tw

❹ 取得 `<svg>` 元素作為 `Selecton`，並指定到變數中以供接下來的調用。

❺ 建立文字標籤…

❻ … 以及這些圓作為它們的初始位置。

❼ 在 `click` 事件處理器中，新的資料集被綁定到 `circle` 元素的 selection 上。請留意，`data()` 函式的第二個參數：這個函式定義了將要被加入的資料項目的 *key*。

❽ 從舊位置到新位置的平滑動態轉換。

❾ `exit()` selecton 現在被填充到 Mary 的節點，因為在最後呼叫到 `data()` 時，沒有任何資料點是被綁定到此節點，我們給這個圓　個不同的顏色讓它看起來更醒目。

這個例子展現了如何實現使用 key 來連接：只要簡單地提供第二個參數給 `data(data, key)` 函式即可。這個額外的參數必須是　個存取器函式，它可以為每　個節點或資料點傳回想要的 key 值作為字串。此函式將會在資料集中的每　個項目進行計算，然後符合 key 的項目就會彼此相互綁定。沒有匹配到的項目，不論是在資料集或是 selection 中，就像是之前提到的，都被放入到 `enter()` 以及 `exit()` selection 中。如果有重複的 key，不管是在資料集或是在 selection 裡面，只有首次出現的 key（以集合中的順序）被綁定。在資料集中重複的部份會被放到 `enter()` selection 中，而在目前 selection 重複的項目則會被放到 `exit()` selection 中（參閱圖 3-4）。

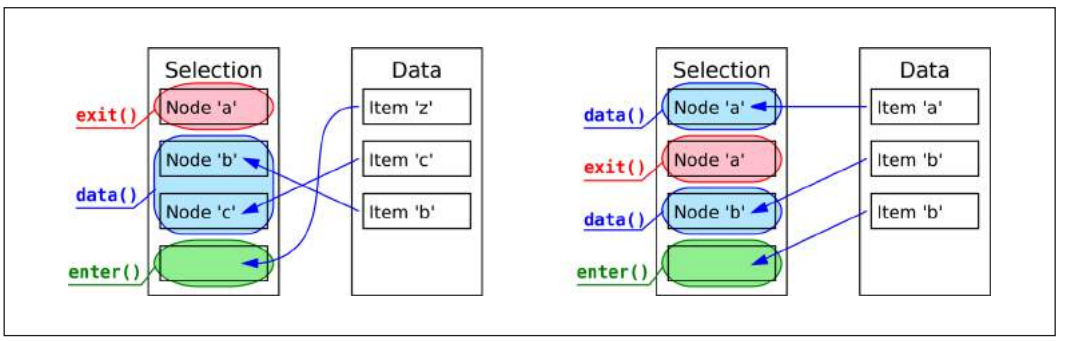

圖 3-4　使用 key 綁定資料不完全匹配的情況。這三個 selection：data()、enter() 以及 exit() 會依不同情況被填入資料（左側）。如果有重複的鍵，不管是在資料集或是 selection 中，相對應的項目會被放置在 enter() 或是 exit() selection 裡（右側）

碁峰 www.gotop.com.tw

- *ease* 可以控制改變的速率讓它不同於 transition 的 duration（做出「ease into」以及「out of」的動畫效果）。在預設的情況下，ease 可以做出三次曲線方程式的漸變效果（「slow-in, slow-out」）。
- *interpolator* 用來計算出中間值（這其實很少用，因為預設的 inerpolator 處理程式可以自動地應付大部份的情況）。
- *event handler* 可以在 transition 開始、結束、或是中斷時執行自訂的程式。

表 4-4 用來配置 transition 或是在沒有加上引數呼叫時取得目前設定的函式（trans 是 Transition 物件）

函式	說明
`trans.delay( value )`	在 selection 中的每個元素之 transition 開始前設定 delay 值（毫秒）；預設值是 0。delay 可以給定　個常數或是　個函式。如果提供的是函式，此函式將會在每　個元素中，transition 開始之前被呼叫　次，所以函式在被呼叫時要傳回　個想要的 delay。此函式會接收到元素的資料，取得它在 selection 中的資料 `d` 和它的索引值 `i`。
`trans.duration( value )`	以毫秒作為單位設定在 selection 中每　個元素 transition 的 duration；預設值是 250 毫秒。duration 的引數可以是常數也可以是函式。如果提供的是函式，此函式將會在每　個元素中，在 transition 開始之前被呼叫　次，所以函式在被呼叫時要傳回　個想要的 duration。這個函式會接收到元素的資料，取得它在 selection 中的資料 `d` 和它的索引值 `i`。
`trans.ease( fct )`	為每　個選取到的元素設定 ease 函式。ease 函數的引數需要是　個函式，此參數介於 0 和 1 之間，並傳回也是介於 0 到 1 之間的數值。預設的 ease 函數是 `d3.easeCubic`（它是　個分段定義的立方多項式，提供「slow-in, slow-out」行為）。
`trans.on( type, handler )`	在 transition 上新增　個事件處理程式。type 必須是 `start`、`end`、或是 `interrupt`。此事件處理程式會在 transition 生命週期中的適當處呼叫（此函式的行為和 `Selection` 物件上的 `on()` 函式類似）。

使用 Transition

`Transition` API 複製了大部份的 `Selection` API。尤其是，所有來自於表 3-2 的函式（也就是 `select()`、`selectAll()`、`filter()`）都是可用的。表 3-4 中，`attr()`、`style()`、`text()`、以及 `each()` 也可以，還有表 3-5 中除了 `append()`、`insert()`、以及 `sort()` 之外的所有函式（正如我們在之前指出的，在 transition 中共用的所有元素必須在 transition 建立之前就存在了。基於同樣的理由，表 3-3 中用來綁定資料的函式對於 transition 來說都是不存在的）。

碁峰 www.gotop.com.tw

範例 *4-6* 即時動畫（圖 *4-4* 的左側）（續）

```
var crrX = 150+100, crrY = 150+0;
var prvX = crrX, prvY = crrY;

var timer = d3.timer( function(t) {
    phi = omega*t;
        crrX = 150+100*Math.cos(a*phi);
        crrY = 150+100*Math.sin(b*phi);

        svg.selectAll( "line" )
            .each( function() { this.bogus_opacity *= .99 } )
            .attr( "stroke-opacity",
                   function() { return this.bogus_opacity } )
            .filter( function() { return this.bogus_opacity<0.05 } )
            .remove();

        svg.append( "line" )
            .each( function() { this.bogus_opacity = 1.0 } )
            .attr( "x1", prvX ).attr( "y1", prvY )
            .attr( "x2", crrX ).attr( "y2", crrY )
            .attr( "stroke", "green" ).attr( "stroke-width", 2 );

        prvX = crrX;
        prvY = crrY;

        if( t > 120e3 ) { timer.stop(); } // after 120 seconds
    } );
}
```

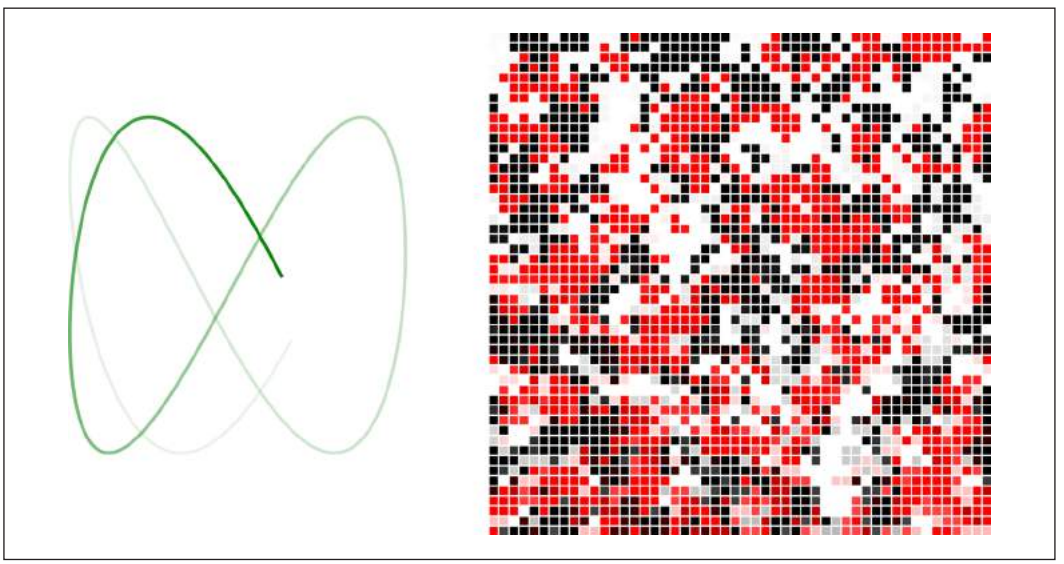

圖 4-4 動畫：李沙育圖（左側，參考範例 4-6），以及 voter 模型（右側，參考範例 4-7）

碁峰 www.gotop.com.tw

請記住，D3 是一個用來操作 DOM 樹的 JavaScript 函式庫，不是繪圖套件。它不畫像素點，它是在 DOM 元素上面作業的。因此，一點都不令人驚訝的是，D3 在繪圖的功能是以組織及串流的方式在 DOM 元素上進行處理。

要產生複雜的圖形，D3 使用三個不同風格的 helper function（協助函式）。這些函式可以它們作用的規模來區分：*generator* 產生個別的屬性，*component* 建立一整個 DOM 元素，而 *layout* 則用來決定整張圖表的排列（參考圖 5-1）。請留意，因為它們之中沒有任何一個會做到你預期的所有作業。

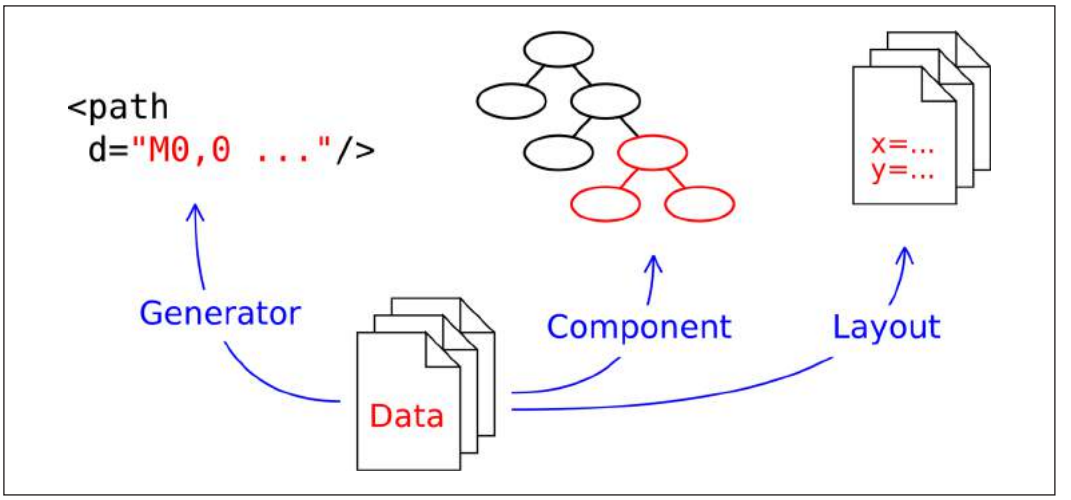

圖 5-1 不同的 D3 建構出視覺化資料：generator 建立一個 <path> 用的命令列字串，component 修改 DOM 樹，而 layout 則是新增像素座標和其他資訊到資料集本身

Generator（產生器）

產生器是 `<path>` 元素的海龜繪圖命令語言之包裝器：它們讓程式設計者可以自由地編寫那些神秘的命令字串。產生器會使用資料集然後傳回一個適合於 *<path>* 元素的屬性值。但是它們不會產生 `<path>` 元素：這需要透過其他的程式碼以確保這個元素是存在的。我們將會在本章和第 9 章中看到一些產生器的例子。

Component（元件）

元件是用來把新產生的元素注入 DOM 樹的函式。它們總是把一個 `Selection` 作為參數，此參數用來指示這個新的元素會被放入 DOM 樹的位置。通常 selection 都會是 `<g>` 元素，它作為一個新建 DOM 元素的容器。元件不會回傳任何東西：它們通常是

碁峰 www.gotop.com.tw

範例 5-6 定義　個資料集用來描述的　個選舉結果：每　個候選人，都有名字以及得票數。圖 5-7 利用圓餅圖來呈現這個資料集。

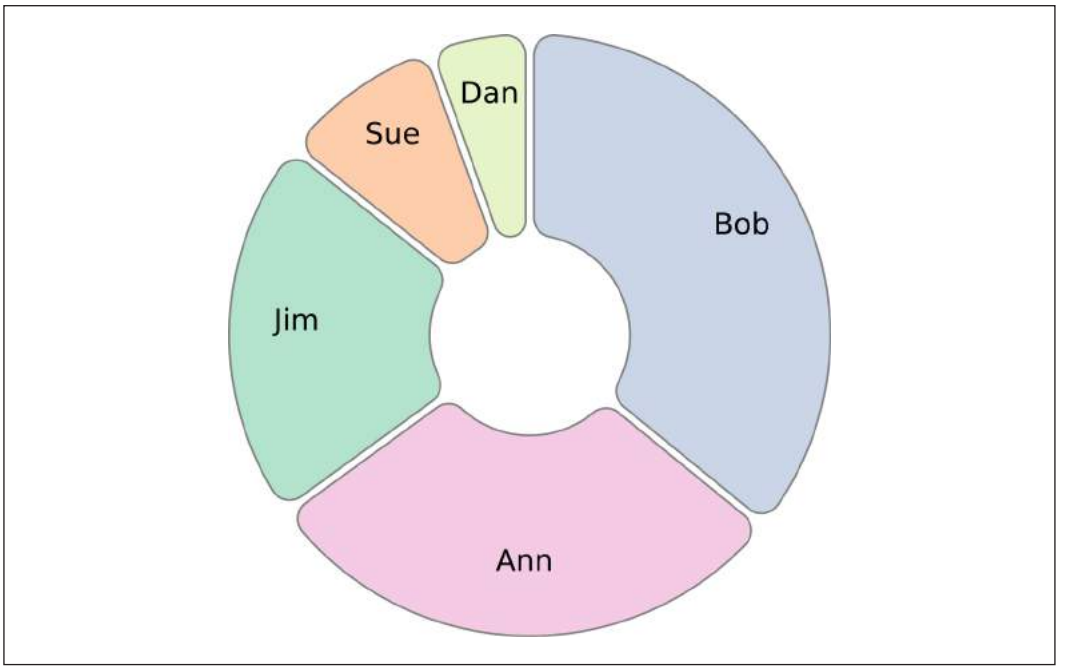

圖 5-7　圓餅圖（參考範例 5-6）

範例 *5-6* 使用圓餅圖 *layout* 以及 *arc* 產生器（參考圖 *5-7*）

```
function makePie() {
    var data = [ { name: "Jim", votes: 12 },
                 { name: "Sue", votes:  5 },
                 { name: "Bob", votes: 21 },
                 { name: "Ann", votes: 17 },
                 { name: "Dan", votes:  3 } ];

    var pie = d3.pie().value(d=>d.votes).padAngle(0.025)( data ); ❶

    var arcMkr = d3.arc().innerRadius( 50 ).outerRadius( 150 )    ❷
        .cornerRadius(10);

    var scC = d3.scaleOrdinal( d3.schemePastel2 )                 ❸
        .domain( pie.map(d=>d.index) )                            ❹

    var g = d3.select( "#pie" )                                   ❺
```


碁峰 www.gotop.com.tw

由這三個工廠函式所建立的 scale 物件使用三種不同的方法理解 `domain()` 函式的引數：`d3.scaleQuantize()` 作為輸入域的程式，`d3.scaleThreshold()` 作為臨界值來區分箱子，而 `d3.scaleQuantile()` 則是作為資料集中的紀錄。

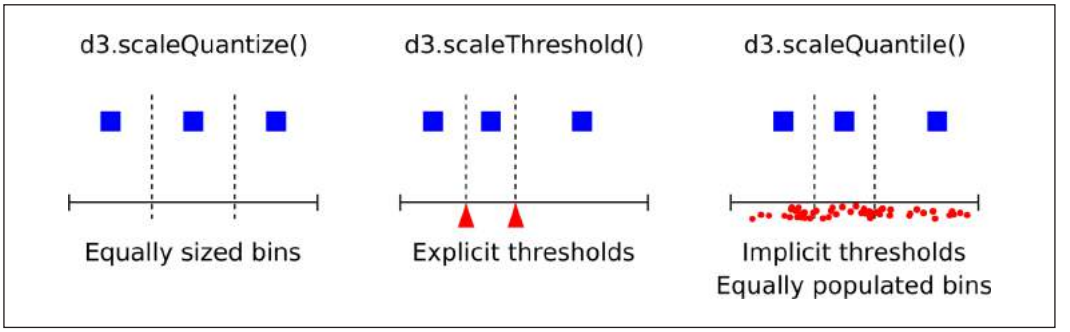

圖 7-1 分箱 scale 使用 3 種不同的方式來把輸入域區分到不同的箱子

分箱 scale 提供一個比用來對應到連續性範圍還要小範圍的 API（參考表 7-2）。特別是分箱 scale 並不支援 clamping：值如果超出域會被對應到第一個或是最後一個箱子中。你可以從以下的工廠函式中得到一個分箱的 scale 物件：

`d3.scaleQuantize()`

- 回傳一個可被區分之連續性數值的輸入域到一組相等大小的箱子中。
- 輸入域必須被指定作為數值的一對。預設是 `[0, 1]`。
- 輸出範圍元素被使用 `range()` 方法函式定義。預設值是 `[0, 1]`。
- 傳回的 scale 物件提供了 `nice()`、`tick()`、以及 `tickFormat()` 函式，它們和在連續性 scale 中所提供的是相同的（參考表 7-1）。

`d3.scaleThreshold()`

- 基於在輸入域中明確的臨界值傳回一個分箱 scale。這些臨界值並不一定要是數字，但是它們必須是可以排序的。
- 這些臨界值被使用 `domain()` 方法定義，到 `domain()` 的引數必須是一個值的陣列，而且要以升序的方式做過排序。
- 輸出範圍的元素被使用 `range()` 方法定義。範圍元素的數目必須比臨界值元素的數量還多。

碁峰 www.gotop.com.tw