

Javascript 學徒試煉

Javascript 的前世今生

3-9 ID 寫法 & textcontent

函式

5-25 全域變數與區域變數

5-26 hoisting 概念

5-28 function 計算機(二)

陣列與物件

6-35 陣列簡報介紹

6-36 物件 簡報介紹

6-37 物件寫法

6-38 物件+function 寫法

6-39 物件 + 陣列 應用

控制判斷(運算子、if 、switch)

7-47 邏輯運算

7-52 else if 程式碼教學

7-54 switch - 程式教學

迴圈

8-58 for 寫法

8-59 for – array 寫法

8-60 for if 寫法

8-62 for 加總

8-63 for 與 break 用法

8-65 JSONview Chrome 插鍵

8-66 擷取 JSON 格式流程

8-67 json - for - open data 範例 (上)

8-68 json - for - open data 範例 (下)

DOM

9-72 querySelector - 選擇單一元素

9-73 querySelectorAll - 可重複選取多個元素

9-74 setAttribute - 增加標籤屬性

9-75 插入 HTML 標籤的兩種方法

9-76 innerHTML 寫法 (上)

9-77 innerHTML 寫法 (下)

9-78 innerHTML 與 for 運用

18 - 150 Template literals 輕鬆進行字串相加

9-79 createElement 寫法

9-80 createElement 與 for 用法

Event (事件)

10-85 event 物件 - 告知你當下元素資訊

10-86 各種事件綁定的差異

10-87 addEventListener - 監聽事件

10-88 綁定事件的語法差異

10-89 event bubbling(氣泡)、event capturing(捕捉) 差異

10-90 stopPropagation - 終止冒泡事件

10-91 preventDefault - 取消預設觸發行為

10-92 e.target - 了解目前所在元素位置

10-93 change - 表單內容更動內容時觸發

10-94 keyCode - 點擊鍵盤，發射火箭

10-95 blur - 離開焦點時進行事件觸發

10-96 mouse - 當滑鼠滑入指定內容時觸發

10-97 網頁座標 - 了解 screen、page、client 箇中差異

10-98 網頁座標 - 應用篇

10-99 事件監聽優化篇 - 從父元素來監聽子元素內容

localStorage - 瀏覽器資料存取

11-101 什麼是 localStorage

11-102 setItem、getItem 基本操作

11-103 JSON.parse、JSON.stringify 來編譯資料

11-104 data-* - 透過 dataset 讀取資料

AJAX

15-122 透過 XMLHttpRequest 物件跨瀏覽器撈資料

15-123 AJAX 非同步觀念(上)

15-124 AJAX 非同步觀念(下)

15-125 HTTP 狀態碼

15-126 Cross-Origin Resource Sharing (CORS)

15-127 傳統表單輸入介紹

15-128 AJAX POST 寫法

15-130 AJAX JSON 傳遞 (POST 寫法)

15-131 實務範例設計

Google Map API

16-137 Google Map API 原理

16-138 增加座標

16-139 增加多個標記

16-141 自製客製化 google map 樣式

16-140 標記 + 第三方 JSON 資料

ECMAScript 6 入門 – let 、 const

17-145 window 、 var 特性

17-146 let-if 、 function 用法

17-147 let-for 用法

17-148 const 特性

17-149 let 、 const 注意事項

3-9 ID 寫法 & textcontent

```
HTML
1 <h1 id="test"> 安安安大家好 </h1>

CSS

JS 43 unsaved changes X
1 document.getElementById('test').textContent='標籤h1被修改了';
```

標籤h1被修改了

5-25 全域變數與區域變數

1. 一般來說在函式設置的變數(var xxx)只要在函示裡執行完畢後就會銷毀，因此此範例如需印出最底下的 console.log(total)則必須在最一開始就設置一個全域變數，供函式裡的變數 total 使用。

```
HTML
CSS
JS
1
2 //全域變數
3 var total;
4
5 function test(t){
6     //區域變數
7     var price = 30 ;
8     total = price * t ;
9     document.write(total);
10 }
11
12 test(3);
13 console.log(total);
```

90

Console

90

2. 然而，當 test 函式裡的 total 前面也冠上 var 他將自行變成區域變數，便與全域變數的 total 毫無關係，因此最後的 console.log(total) 將會顯示 undefined 。

```
HTML
CSS
JS
1
2 //全域變數
3 var total;
4
5 function test(t){
6     //區域變數
7     var price = 30 ;
8     var total = price * t ;
9     document.write(total);
10 }
11
12 test(3);
13 console.log(total);
```

90

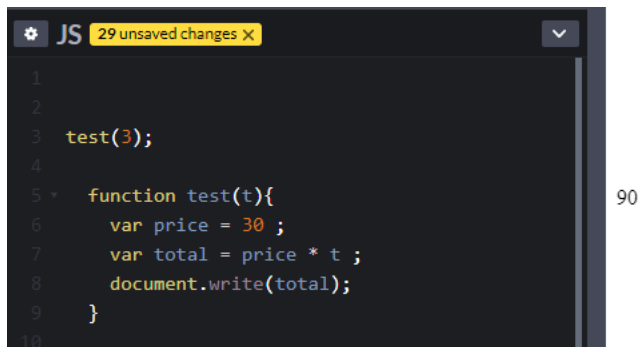
Console

undefined

5-26 hoisting 概念

原本都說網頁在載入後的渲染順序是由上至下做渲染，因此此案例再函式前就先行渲染，將會出錯，但因為函式有 **hosting** 的概念，會主動將函式提升至最上層，使其優先執行。

Hoiston 相關資料: <https://blog.techbridge.cc/2018/11/10/javascript-hoisting/>



```
1 test(3);
2
3 function test(t){
4   var price = 30;
5   var total = price * t;
6   document.write(total);
7 }
8
9
10
```

5-28 function 計算機(二)

1. Onclick 事件：



```
HTML
1 <input type="button" id="bbb" value="點我">
2
CSS
JS
1 document.getElementById('bbb').onclick = function(){
2   alert('你點擊了我');
3 }
```

2. .value & .type

Value 可將其值帶出來做運算，但她帶出來的值都會是 **string**，因此需要經過 **parseInt** 將其轉為數值方可運算。Type 則可以判斷出其屬性是什麼。



```
HTML
1 A欄位<input type="text" id="aaa"><br>
2 B欄位<input type="text" id="bbb"><br>
3 <input type="button" id="btn" value="請點我">
4 A+B 答案是:<p id="print"></p><br><br>
5
6 <input type="text" id="ccc"> → 此欄位屬性是 <p id="print2"></p>
7
CSS
JS
1 document.getElementById('btn').onclick = function(){
2   var showAaa = parseInt(document.getElementById('aaa').value);
3   var showBbb = parseInt(document.getElementById('bbb').value);
4   document.getElementById('print').textContent= showAaa + showBbb;
5 }
6
7 document.getElementById('print2').textContent = document.getElementById('ccc').type;
8
```

3. typeof() 查詢型別

```
> typeof(3)
< "number"
> typeof("hi")
< "string"
> typeof(function(){} )
< "function"
> typeof({name:"jiaren"})
< "object"
> typeof(false)
< "boolean"
```

4. 觀念釐清 (.value)

```
9  <body>
10  <input type="text" id="textId">
11  <input type="button" id="btn">
12  <script>
13    //以下可以正常執行 輸入東西後 按下btn可以alert出內容
14    var btn = document.getElementById('btn');
15    btn.onclick=function (){
16      var text = document.getElementById('textId').value;
17      alert(text);
18    }
19    //但下面這個不行，因為他取value的值的時機在於onclick之前，因此會undefined，抓不到值
20    var btn = document.getElementById('btn');
21    var text = document.getElementById('textId').value;
22    btn.onclick=function (){
23      alert(text);
24    }
25  </script>
26
27
```



6-35 陣列簡報介紹

1. 陣列案例解析：陣列可以減少變數的定義，如下有狗狗 a 、 狗狗 b 、 狗狗 c 等多種變數。

```
var dogA = 'jia';  
var dogB = 'john';  
var dogC = 'leon';
```

>>>

```
var dogs = ['jia','john','leon']
```

新增、讀取 陣列：

2. .push()：

加入新值進去陣列。 / 加入新值進去陣列方式也可使用 `a[3] = 23`; 例如此方法，直接指定陣列的第幾位數要加入甚麼數值。

3. .length：查詢該陣列的值有幾項。

```
> var a = [];  
a.push(3);  
console.log(a);  
▶ [3]  
← undefined  
> a.push(3);  
← 2  
> console.log(a)  
▶ (2) [3, 3]  
← undefined  
> a[0] = 12  
← 12  
> console.log(a)  
▶ (2) [12, 3]  
← undefined  
> a[2] = 3333  
← 3333  
> console.log(a)  
▶ (3) [12, 3, 3333]  
← undefined  
> a.length  
← 3
```

陣列相關資料: JavaScript Array 陣列操作方法大全 (含 ES6)

<https://www.oxxostudio.tw/articles/201908/js-array.html>

6-36 物件 簡報介紹

1. 物件則可以再對陣列做簡化並給上更精確的冠名。

物件的使用將可減少因需要定義多個變數來儲存陣列的情形例如。

物件的值 可以有字串、陣列、數值、函式

2. 案例解析：狗狗品種又分成 高貴狗、中等狗、下等狗。 屬性(property)：值(value)

```
var dogsLevel = {  
  level_1 : ['jia','john','leon'],  
  level_2 : ['givon','mark'],  
  level_3 : "candy",  
  totals : 6  
}
```

6-37 物件寫法

新增、讀取物件：

物件名稱.屬性 = 值；即可直接加入一個 eat 的屬性進去物件當中，如下案例。

查詢也是，底下案例即是 console.log 出 mark 這條狗狗出來。

```
8 * var dogsLevel = {  
9   level_1 : ['jia','john','leon'],  
10  level_2 : ['givon','mark'],  
11  level_3 : "candy",  
12  totals : 6  
13 }  
14  
15 dogsLevel.eat = "meat";  
16  
17 console.log(dogsLevel.level_2[1]);
```

6-38 物件+function 寫法

1. 範例：即可印出 狗狗回家吃飯 一個函式的動作

```
var dogsLevel = {  
  level_1 : ['jia','john','leon'],  
  level_2 : ['givon','mark'],  
  level_3 : "candy",  
  totals : 6,  
  goHome : function(){  
    document.write("狗狗們回家吃飯了");  
  }  
}  
  
//執行函式  
dogsLevel.goHome();
```

2. 範例：計算 列出計算的值

```
* var pets = {  
  catName : ['jia','john','leon'],  
  dogName : ['givon','mark'],  
  cats : 3,  
  dogs : 2,  
  petTotal : function(){  
    var total = pets.cats + pets.dogs  
    document.write("貓狗數量："+total+"隻");  
  }  
}  
  
//呼叫函式 列出 貓狗數量:5隻  
pets.petTotal();
```


6-39 物件 + 陣列 應用

範例：如需撈出 農場中農夫查理養的狗-皮皮，則需 鍵入 `farms[1].dogs[0]` 即可

```
13 var farms = [  
14   {  
15     farmer: '卡斯伯',  
16     dogs: ['張姆士', '龐的'],  
17     chick: 15,  
18     cornField: [8, 5, 6],  
19     broccoliField: [6, 6, 6, 6],  
20     scarecrow: 9  
21   },  
22   {  
23     farmer: '查理',  
24     dogs: ['皮皮'],  
25     chick: 30,  
26     cornField: [18, 12],  
27     broccoliField: [8, 8, 8],  
28     scarecrow: 6  
29   }  
30 ];  
31  
32 console.log(farms[1].dogs[0])
```

7-47 邏輯運算

1.

&& (And)：都需要是 true 才行

|| (OR)：其中一項是 true 即可

```
> var vip = false  
↵ undefined  
  
> var money = 1000  
↵ undefined  
  
> money > 900 || vip == true  
↵ true  
  
> money > 900 && vip == true  
↵ false
```

2. ! (NOT)：轉型，幫你把 true 變成 false，false 變成 true

```
> !(2>3)  
↵ true  
  
> !(3>2)  
↵ false
```

7-52 else if 程式碼教學

If 判別式會在 true 的條件下才往下繼續執行

案例：將根據飢餓程度秀出 要進食的食物或是不進食。

```
1 //飢餓程度 1~10
2 var hungry = 2;
3
4 //吃東西的函式動作
5 function eat(food){
6     document.write("我要吃"+food);
7 }
8
9 if(hungry<=3){
10     eat("比薩");
11 }else if(hungry>3 && hungry<=5){
12     eat("沙拉");
13 }else{
14     document.write("我現在完全不餓");
15 }
```

7-54 switch - 程式教學

當瀏覽器在渲染時，switch 效能較 if else(所有內容都會審視過一次)加，其原因是他會直接在 case 做比對，如果符合 case 內容就執行接續的動作，並且 break 跳出(每個 case 後面都會接 break 來阻止已完成的區塊繼續往後面執行)，不再進行其他比對度做。一旦不符合 case 就直接往下個做比對。

常用於已經確定要判斷東西的狀態是什麼

範例：

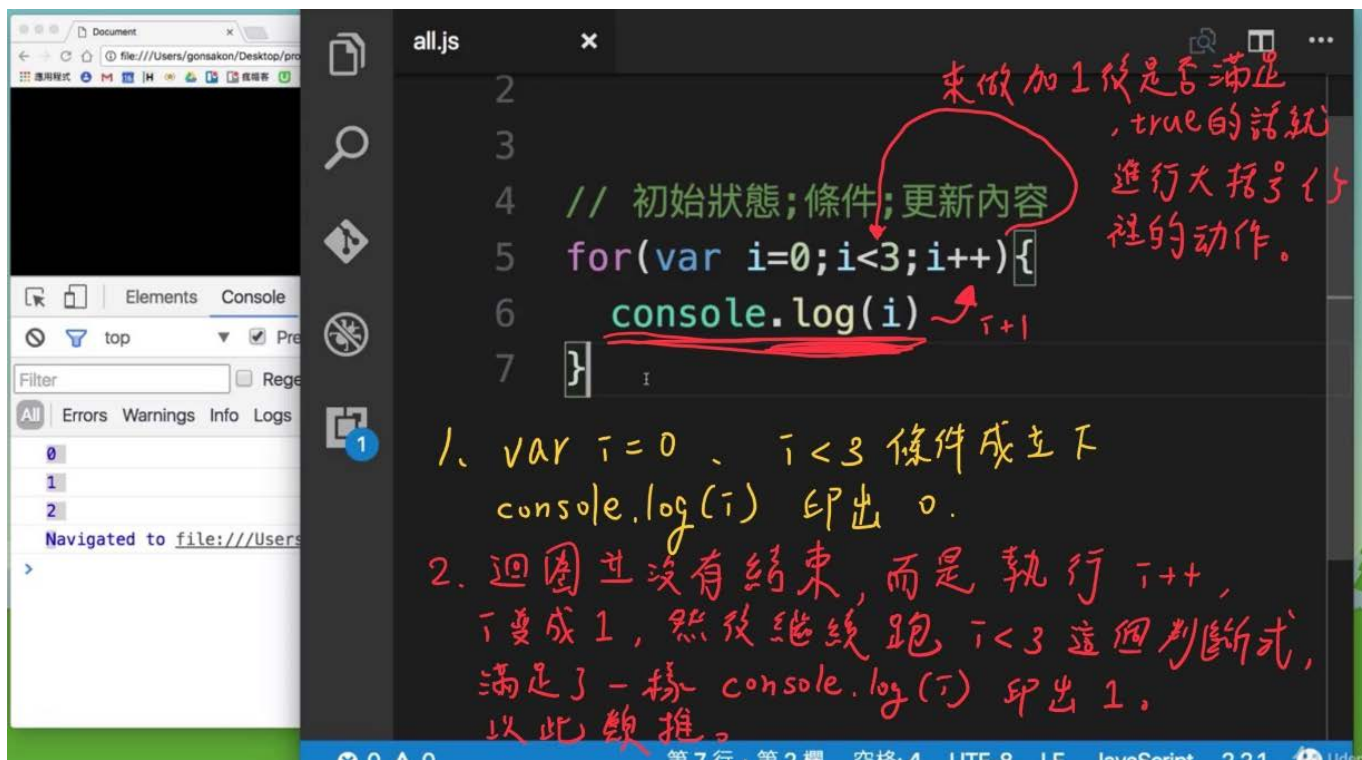
```
//變數：紅色、藍色、綠色 警戒等等
var light = "green";

// Action function
function lightAction(color){
    document.write("目前是"+color+"警戒");
}

// 變數判斷
switch (light) {
    case "blue":
        lightAction("藍色");
        break;
    case "green":
        lightAction("綠色");
        break;
    case "red":
        lightAction("紅色");
        break;

    // else的概念，但其位置可以隨意放置(放在開頭也可以)
    default:
        document.write("無任何警戒");
        break;
}
```

8-58 for 寫法



The screenshot shows a code editor with the following JavaScript code:

```
2  
3  
4 // 初始狀態;條件;更新內容  
5 for(var i=0;i<3;i++){  
6   console.log(i)  
7 }
```

Handwritten annotations in red and yellow explain the execution:

- Red text: "來做加1後是否滿足, true的話就進行大括號{}裡的動作。" (To do the +1 and check if it satisfies, if true, then execute the actions inside the curly braces.)
- Red arrow: Points from the `i++` part of the loop header to the `console.log(i)` statement.
- Yellow text: "1. var i=0, i<3 條件成立下 console.log(i) 印出 0." (1. var i=0, i<3 condition is true, console.log(i) prints 0.)
- Yellow text: "2. 迴圈並沒有結束, 而是執行 i++, i 變成 1, 然後繼續跑 i<3 這個判斷式, 滿足了一樣 console.log(i) 印出 1. 以此類推。" (2. The loop hasn't ended, it executes i++, i becomes 1, then continues to run the i<3 condition, which is satisfied, so console.log(i) prints 1. This continues.)

For 觀念釐清

分別在 for 迴圈裡印出的結果沒有 4 & 3, 但在之後呼叫查詢時卻是 4 & 3 原因是當變成 4 & 3 已不符合迴圈裡的判斷了, 因此離開迴圈後便是此結果, 而非回圈內最後印出的值。

```
> for(var i=0 ; i<3 ; i+=2){  
  console.log(i);  
}  
0  
2  
< undefined  
> i  
< 4  
> for(var i=0 ; i<3 ; i++){  
  console.log(i);  
}  
0  
1  
2  
< undefined  
> i  
< 3
```

8-59 for – array 寫法

範例：可以透過 `.length` 去撈出有幾筆的農夫資料 印出：
第 1 個農場主人是卡斯柏 第 2 個農場主人是包柏

```
//農場裡的農夫們
var farms = [
  {
    farmer: '卡斯柏',
    field: 6
  },
  {
    farmer: '包柏',
    field: 5
  }
]

for(var i=0; farms.length; i++){
  document.write("第"+(i+1)+"個農場主人是"+farms[i].farmer);
}
```

8-60 for if 寫法

範例：印出農場小雞數量超過 100 隻的農夫

```
//農場裡的農夫們
var farms = [
  {
    farmer: '卡斯柏',
    field: 6,
    chick: 110
  },
  {
    farmer: '包柏',
    field: 5,
    chick: 99
  }
]

for(var i=0; i<farms.length; i++){
  if(farms[i].chick>100){
    document.write(farms[i].farmer+"農場小雞數量超過100隻");
  }
}
```

8-62 for 加總

案例：計算整個農場小雞數量 印出：農場小雞數量總共 242 隻

```
//農場裡的農夫們
var farms = [
  {
    farmer: '卡斯柏',
    field: 6,
    chick: 110
  },
  {
    farmer: '包柏',
    field: 5,
    chick: 99
  },
  {
    farmer: '卡利',
    field: 4,
    chick: 33
  }
]
var farmersTotal = farms.length; //農夫人數
var chickTotal = 0; //小雞初始值
for(var i=0; i<farmersTotal; i++){
  chickTotal = chickTotal + farms[i].chick;
}
document.write("農場小雞數量總共"+chickTotal+"隻");
```

8-63 for 與 break 用法

範例：今天嬾嬾要去農場買小雞 50 隻，任何農場都可以，只要達到數量即可，一買到就不再往下一家農場購買、挑選了。 因此使用 break 讓購買數量成立後就跳出迴圈，

印出：卡斯柏的農場小雞夠我買，並且買完後剩下 3 隻小雞

```
//農場裡的農夫們
var farms = [
  {
    farmer: '卡斯柏',
    field: 6,
    chick: 99
  },
  {
    farmer: '包柏',
    field: 5,
    chick: 103
  },
  {
    farmer: '卡利',
    field: 4,
    chick: 30
  }
]
var farmersTotal = farms.length; //農夫人數
var chickTotal = 0; //小雞初始值
for(var i=0; i<farmersTotal; i++){
  if(farms[i].chick>100){
    document.write(farms[i].farmer+"的農場小雞夠我買，");
    farms[i].chick -= 100 ; // farms[i].chick = farms[i].chick - 100
    document.write("並且買完後剩下"+farms[i].chick+"隻小雞");
    break; //跳離整個迴圈
  }
}
```

8-65 JSONview Chrome 插鍵

今天如果要清楚解析某開放原始碼的 json 資料格式可以透過 JSONview 這款 chrome 插鍵將原始碼做清楚的排列 (但如果裝上插鍵後又想複製原始碼則另外用無痕視窗開啟即可)

```
[{"Title": "創新高雄 產業轉型開展城市經濟新版圖", "Link": "https://kcginfo.kcg.gov.tw/Publish_Content.aspx?n=3D7C9BFC4F86BF4A&s=1DC11B9A0083D59B", "Image": "https://kcginfo.kcg.gov.tw/01.jpg&w=100", "Description": "氣爆事件造成的嚴重災情，突顯出高雄作為石化、重工業城市以「高雄轉型」作為施政的核心價值，要讓高雄擺脫工業城市帶來的陰霾，最終要達成產業創新、宜居環境、人本社福、安全城市四大目標。", "pubDate": "2014-12-02T01:47:17"}, {"Title": "躍升高雄 見證亞洲新灣區大實現", "Link": "https://kcginfo.kcg.gov.tw/Publish_Content.aspx?n=3D7C9BFC4F86BF4A&s=9070DC77151202A3", "Image": "https://kcginfo.kcg.gov.tw/02.jpg&w=100", "Description": "2014年9月，眾人期盼已久的輕軌列車運抵高雄，簡潔的流線型車體宛如一道曙光，照亮了「綠色高雄」的新里程，而高雄水岸輕軌正是串連「亞洲新灣區」各項重大建設的火車頭，連結起新近完工的高雄展覽館、高雄市新圖書館總館，也開創了高雄港埠轉型的新契機。", "pubDate": "2014-11-11T09:15:06"}, {"Title": "再造宜居家園 災區重建ING", "Link": "https://kcginfo.kcg.gov.tw/Publish_Content.aspx?n=3D7C9BFC4F86BF4A&s=E604BEC02D4251EF", "Image": "https://kcginfo.kcg.gov.tw/03.jpg&w=100", "Description": "2014年7月31日深夜至8月1日凌晨時分，接連幾聲巨響劃開了夜空的寂靜，高雄發生了一場震驚全台的氣爆災害，市政府旋即成立災害應變中心緊急動員救災，並於8月4日展開氣爆災害重建工程。", "pubDate": "2014-10-13T02:15:49"}]
```

(原始碼，去除空白較能節省位元)



```
[
  {
    Title: "創新高雄 產業轉型開展城市經濟新版圖",
    Link: "https://kcginfo.kcg.gov.tw/Publish_Content.aspx?n=3D7C9BFC4F86BF4A&s=1DC11B9A0083D59B",
    Image: "https://kcginfo.kcg.gov.tw/Common/DisplaySmallPicture.ashx?p=Upload/Publish/oldCover/01-01.jpg&w=100",
    Description: "氣爆事件造成的嚴重災情，突顯出高雄作為石化、重工業城市的宿命。持續重建的同時，也適逢縣市合併屆滿四年，高雄市政府再度提出以「高雄轉型」作為施政的核心價值，要讓高雄擺脫工業城市帶來的陰霾，最終要達成產業創新、宜居環境、人本社福、安全城市四大目標。",
    pubDate: "2014-12-02T01:47:17"
  },
  {
    Title: "躍升高雄 見證亞洲新灣區大實現",
    Link: "https://kcginfo.kcg.gov.tw/Publish_Content.aspx?n=3D7C9BFC4F86BF4A&s=9070DC77151202A3",
    Image: "https://kcginfo.kcg.gov.tw/Common/DisplaySmallPicture.ashx?p=Upload/Publish/oldCover/01-02.jpg&w=100",
    Description: "2014年9月，眾人期盼已久的輕軌列車運抵高雄，簡潔的流線型車體宛如一道曙光，照亮了「綠色高雄」的新里程，而高雄水岸輕軌正是串連「亞洲新灣區」各項重大建設的火車頭，連結起新近完工的高雄展覽館、高雄市新圖書館總館，也開創了高雄港埠轉型的新契機。",
    pubDate: "2014-11-11T09:15:06"
  },
  {
    Title: "再造宜居家園 災區重建ING",
    Link: "https://kcginfo.kcg.gov.tw/Publish_Content.aspx?n=3D7C9BFC4F86BF4A&s=E604BEC02D4251EF",
    Image: "https://kcginfo.kcg.gov.tw/Common/DisplaySmallPicture.ashx?p=Upload/Publish/oldCover/01-03.jpg&w=100",
    Description: "2014年7月31日深夜至8月1日凌晨時分，接連幾聲巨響劃開了夜空的寂靜，高雄發生了一場震驚全台的氣爆災害，市政府旋即成立災害應變中心緊急動員救災，並於8月4日展開氣爆災害重建工程。",
    pubDate: "2014-10-13T02:15:49"
  }
]
```

(整理過叫好閱讀)

8-66 擷取 JSON 格式流程

讀取 json 物件總數有幾筆：自訂一變數將整個陣列形式的 json 放入其中

```
var data = [{"Title": "創新高雄 產業轉型開展城市經濟新版圖", "Link": "https://kcginfo.kcg.gov.tw/Publish_Content.aspx?n=3D7C9BFC4F86BF4A&s=1DC11B9A0083D59B", "Image": "https://kcginfo.kcg.gov.tw/Common/DisplaySmallPicture.ashx?p=Upload/Publish/oldCover/01-01.jpg&w=100", "Description": "氣爆事件造成的嚴重災情，突顯出高雄作為石化、重工業城市的宿命。持續重建的同時，也適逢縣市合併屆滿四年，高雄市政府再度提出以「高雄轉型」作為施政的核心價值，要讓高雄擺脫工業城市帶來的陰霾，最終要達成產業創新、宜居環境、人本社福、安全城市四大目標。", "pubDate": "2014-12-02T01:47:17"}, {"Title": "躍升高雄 見證亞洲新灣區大實現", "Link": "https://kcginfo.kcg.gov.tw/Publish_Content.aspx?n=3D7C9BFC4F86BF4A&s=9070DC77151202A3", "Image": "https://kcginfo.kcg.gov.tw/Common/DisplaySmallPicture.ashx?p=Upload/Publish/oldCover/01-02.jpg&w=100", "Description": "2014年9月，眾人期盼已久的輕軌列車運抵高雄，簡潔的流線型車體宛如一道曙光，照亮了「綠色高雄」的新里程，而高雄水岸輕軌正是串連「亞洲新灣區」各項重大建設的火車頭，連結起新近完工的高雄展覽館、高雄市新圖書館總館，也開創了高雄港埠轉型的新契機。", "pubDate": "2014-11-11T09:15:06"}, {"Title": "再造宜居家園 災區重建ING", "Link": "https://kcginfo.kcg.gov.tw/Publish_Content.aspx?n=3D7C9BFC4F86BF4A&s=E604BEC02D4251EF", "Image": "https://kcginfo.kcg.gov.tw/Common/DisplaySmallPicture.ashx?p=Upload/Publish/oldCover/01-03.jpg&w=100", "Description": "2014年7月31日深夜至8月1日凌晨時分，接連幾聲巨響劃開了夜空的寂靜，高雄發生了一場震驚全台的氣爆災害，市政府旋即成立災害應變中心緊急動員救災，並於8月4日展開氣爆災害重建工程。", "pubDate": "2014-10-13T02:15:49"}]

console.log(data.length);
```

8-67 json - for - open data 範例 (上)

以下有一筆政府開原始碼資料，要撈出裏頭是岡山區的資料。

Open1999派工受理案件即時600筆資料

前往存取資源

資料下載網址 <https://soweb.kcg.gov.tw/open1999/ServiceRequestsQuery.aspx/ServiceRequestsQuery?startdate=&enddate=>

Seq(序號)、FileNo(案號)、Cre_Date(反映日期)、Save_Date(通報日期)、ReplyUnit(權責單位代碼)、UnitName(權責單位)、Zipcode(行政區域代碼_郵政區號)、ZipName(行政區域)、Address(維修地點)、Informdesc(通報事項)、Beforedesc(反映內容)、Status(管制狀況代碼)、StatusName(管制狀況)、Close_Date(結案日期)、Afterdesc(處理結果)、Memo(備註)、Lat_(緯度)、Lng_(經度)

將資料匯入後 存在一個變數 **data** 之中，透過 **for** 迴圈和 **if** 判斷 即可印出物件當中的地址資料出來

```
var data = [
{
  Seq: 1,
  FileNo_: "A-IN-2019-227034",
  Status_: "2",
  Cre_Date_: "2019/12/24 上午 11:30:04",
  ReplyUnit_: "397150000I",
  zipcode_: "6400600000",
  ZipName_: "新興區",
  address_: "新興區七賢一路71號",
  UnitName_: "環保局",
  StatusName_: "待確認",
  InformDesc_: "異味污染_有機氣體(含溶劑)或化學物質",
  BeforeDesc_: "渠反映該址目前進行噴漆作業產生異味致空污，建請查處不需電覆。",
  AfterDesc_: "",
  Memo_: "",
  Save_Date_: "2019/12/24 上午 11:31:31",
  Close_Date_: "",
  Lat_: "22.633491",
  Lng_: "120.315336"
},
{
  Seq: 2,
  FileNo_: "A-IN-2019-227028",
  Status_: "2",
  Cre_Date_: "2019/12/24 上午 11:20:15",
  ReplyUnit_: "397110300G",
  zipcode_: "6400800000",
  ZipName_: "苓雅區",
  address_: "苓雅區三多一路上東向",
  UnitName_: "環保局",
  StatusName_: "待確認",
  InformDesc_: "異味污染_有機氣體(含溶劑)或化學物質",
  BeforeDesc_: "渠反映該址目前進行噴漆作業產生異味致空污，建請查處不需電覆。",
  AfterDesc_: "",
  Memo_: "",
  Save_Date_: "2019/12/24 上午 11:21:15",
  Close_Date_: "",
  Lat_: "22.633491",
  Lng_: "120.315336"
}
```

...

```
AfterDesc_: "",
Memo_: "",
Save_Date_: "2019/12/21 上午 11:56:05",
Close_Date_: "2019/12/21 下午 04:10:57",
Lat_: "22.732088",
Lng_: "120.347234"
}
]

var total = data.length; //資料總筆數
for(var i=0 ; i < total ; i++){
  if(data[i].ZipName_ == "岡山區"){
    console.log(data[i].address_)
  }
}
```

Console.log(data[i].address_) 的結果

Console

"岡山區仁壽里中華路7"

"岡山區華崗里大埔二街"

"岡山區信義里信義街3"

"岡山區台上里成功路-"

"岡山區河堤路二段上."

"岡山區大仁北路與竹園"

"岡山區竹園國小東北側"

"岡山區大遼中路114"

"岡山區崙仔頂路1之2"

"岡山區嘉峰里"

"岡山區嘉峰里"

"岡山區潭底里路燈編號"

"岡山區岡山里路燈編號"

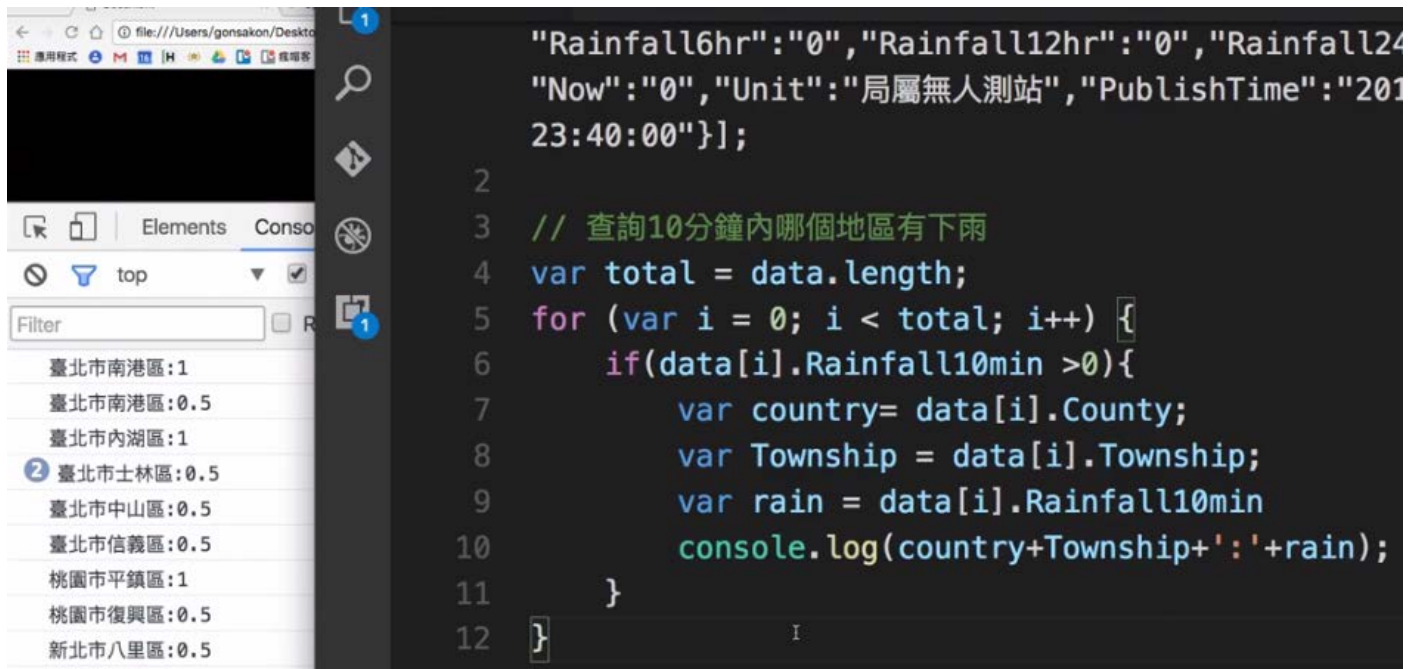
8-68 json - for - open data 範例 (下)

進行數字上的判斷 (即便是"223"這類的 string 也可以當成 number 進行 if 判斷)

```
[
  {
    SiteId: "C11230",
    SiteName: "九份二山",
    County: "南投縣",
    Township: "國姓鄉",
    TWD67Lon: "120.8368",
    TWD67Lat: "23.9637",
    Rainfall10min: "0",
    Rainfall1hr: "0",
    Rainfall3hr: "0",
    Rainfall6hr: "0.5",
    Rainfall12hr: "0.3",
    Rainfall24hr: "4",
    Now: "4",
    Unit: "局屬無人測站",
    PublishTime: "2017-03-16 23:20:00"
  },
  {
    SiteId: "466940",
    SiteName: "基隆",
    County: "基隆市",
    Township: "仁愛區",
    TWD67Lon: "121.7322",
    TWD67Lat: "25.1351",
    Rainfall10min: "0.5",
    Rainfall1hr: "3.5",
    Rainfall3hr: "6",
    Rainfall6hr: "7.5",
    Rainfall12hr: "12.5",
    Rainfall24hr: "27",
    Now: "27",
    Unit: "局屬氣象測站",
    PublishTime: "2017-03-16 23:20:00"
  },
  {
    SiteId: "466900",
```

(json data)

以下進行 印出 Rninfall10min (十分鐘內下雨量)大於 0 的地區



The screenshot shows a web browser window with a list of weather stations. The list includes locations like 臺北市南港區, 臺北市內湖區, 臺北市士林區, etc., with their corresponding 10-minute rainfall values. The JavaScript console on the right shows a loop that iterates through the data array, checks if the 10-minute rainfall is greater than 0, and logs the location and rainfall value.

```
1 "Rainfall6hr":"0","Rainfall12hr":"0","Rainfall24
2 "Now":"0","Unit":"局屬無人測站","PublishTime":"201
3 23:40:00"}];
4
5 // 查詢10分鐘內哪個地區有下雨
6 var total = data.length;
7 for (var i = 0; i < total; i++) {
8   if(data[i].Rainfall10min > 0){
9     var country= data[i].County;
10    var Township = data[i].Township;
11    var rain = data[i].Rainfall10min
12    console.log(country+Township+':'+rain);
13  }
14 }
```


9-72 querySelector - 選擇單一元素

相較於 getElementById 等 更有彈性，且可針對元素內的東西在進行操控

但是其特性是假如在撈取之下有多元素時，便只能撈取其第一筆資料。詳情看到 10-99 章節

```
HTML
1 <h1 class="tt">
2   123
3 </h1>
4 <p class="p">安安你好<em>嘉仁</em></p>

CSS

JS 61 unsaved changes X
1 var el = document.querySelector('.tt');
2 el.textContent = "12345";
3
4 var el2 = document.querySelector('.p em');
5 el2.textContent = "帥哥";
```

12345

安安你好帥哥

9-73 querySelectorAll - 可重複選取多個元素

適合撈多筆資料使用

與 querySelector 差別在於，如果你抓的是多筆資料，他會回傳陣列給你。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, ...>
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
  <h1 class="tittleClass"><em></em></h1>
  <h1 class="tittleClass"><em></em></h1>
</body>

<script>
  var el = document.querySelectorAll('.tittleClass');
  console.log(el);
</script>
```

194 x 372

Elements Console

top

NodeList(2) [h1.tittleClass, h1.tittleClass]

因此需要用陣列形式進行動作，例如 textContent

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, ...>
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
  <h1 class="tittleClass"><em></em></h1>
  <h1 class="tittleClass"><em></em></h1>
</body>

<script>
  var el = document.querySelectorAll('.tittleClass');
  el[0].textContent = "1234";
  el[1].textContent = "789";
</script>
```

194 x 372

1234

789

使用 for 執行多筆資料

```
<meta charset="UTF-8">
<meta name="viewport" con
<meta http-equiv="X-UA-Co
<link rel="stylesheet" hr
<title>Document</title>
</head>
<body>
  <h1 class="titleClass"><e
  <h1 class="titleClass"><e
  <script src="js/all.js"><
</body>
</html>
```

```
2
3 var el = document.querySelectorAll('.titleClass em');
4
5 |
6 el[0].textContent = '123';
7 el[1].textContent = '123';
8
9 var elLen = el.length;
10
11 for(var i = 0;i<elLen;i++){
12   el[i].textContent = i+'123';
13 }
```

9-74 setAttribute - 增加標籤屬性

透過 setAttribute 動態新增、修改 dom

範例：修改標題顏色、新增 href 網址進去

```
<title>Document</title>
<style>
  #colors{
    color: red;
  }
</style>
</head>
<body>
  <h1 class="tittleClass"><a href="#">Ttttle</a></h1>
<script>
  var el = document.querySelectorAll('.tittleClass a');
  // 屬性， 值
  el[0].setAttribute('id','colors');
  el[0].setAttribute('href','http://www.yahoo.com.tw');
</script>
```

Google 翻譯

← → ↺ ⓘ

應用程式 f Fa

Ttttle

也可透過 getAttribute 查詢屬性的值是什麼

```
<h1 class="tittleClass"><a href="#">Ttttle</a></h1>
<script>
  var el = document.querySelector('.tittleClass a');
  // 屬性， 值
  el.setAttribute('id','colors');
  el.setAttribute('href','http://www.yahoo.com.tw');

  var elSearch = document.querySelector('.tittleClass a');
  elSearch.getAttribute('href');
  console.log(elSearch);
</script>
```

90 x

Console >>

top

1225.html:24

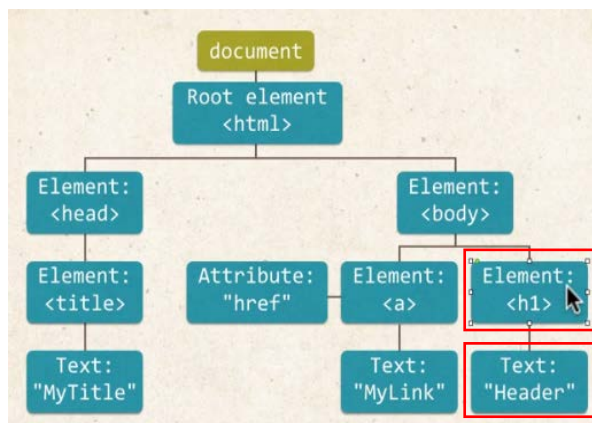
```
<a href="http://
www.yahoo.com.tw" id=
"colors">Ttttle</a>
```

Live reload 1225.html:55
enabled.

Ttttle

9-75 插入 HTML 標籤的兩種方法

與 `textContent` 差別在於他是插入節點(權限較大)，並非只是控制像是 `Text` 這樣的元素而已。



用 JavaScript 操控 HTML 的方法

innerHTML

- 方法：組完字串後，傳進語法進行渲染
- 優點：效能快
- 缺點：資安風險，要確保來源沒問題

createElement

- 方法：以 DOM 節點來處理
- 優點：安全性高
- 缺點：效能差

9-76 innerHTML 寫法 (上)

```
<body>
  <div class="main">
  </div>
  <script>
    var el = document.querySelector('.main');
    el.innerHTML = '<h1> 我是新增出來的節點 </h1>' ;
  </script>
</body>
```

我是新增出來的節點

`innerHTML` 特性：如原先標籤裏頭有資料，他會將其資料清空，在塞入你賦予的值。

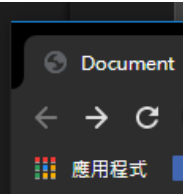
```
<body>
  <div class="main">
    <h2>
      原始資料
    </h2>
  </div>
  <script>
    var el = document.querySelector('.main');
    el.innerHTML = '<h1> 我已經取代掉原始資料 </h1>' ;
  </script>
</body>
```

我已經取代掉原始資料

9-77 innerHTML 寫法 (下)

```
<body>
  <ul class="list"></ul>
</body>

<script>
  var link = 'http://google.com';
  var web = '谷歌';
  var el = document.querySelector('.list');
  el.innerHTML = '<li><a href="'+link+'">'+web+'</a></li>' ;
</script>
</body>
```



- 谷歌

9-78 innerHTML 與 for 運用

範例：印出農場主人的名字

定義一全域變數的空字串來記錄 li 的資料，避免 innerHTML 最新資料會覆蓋舊資料的特性(如不這麼做將只會印出包柏的名字)

```
<body>
  <ul class="list"></ul>
</body>

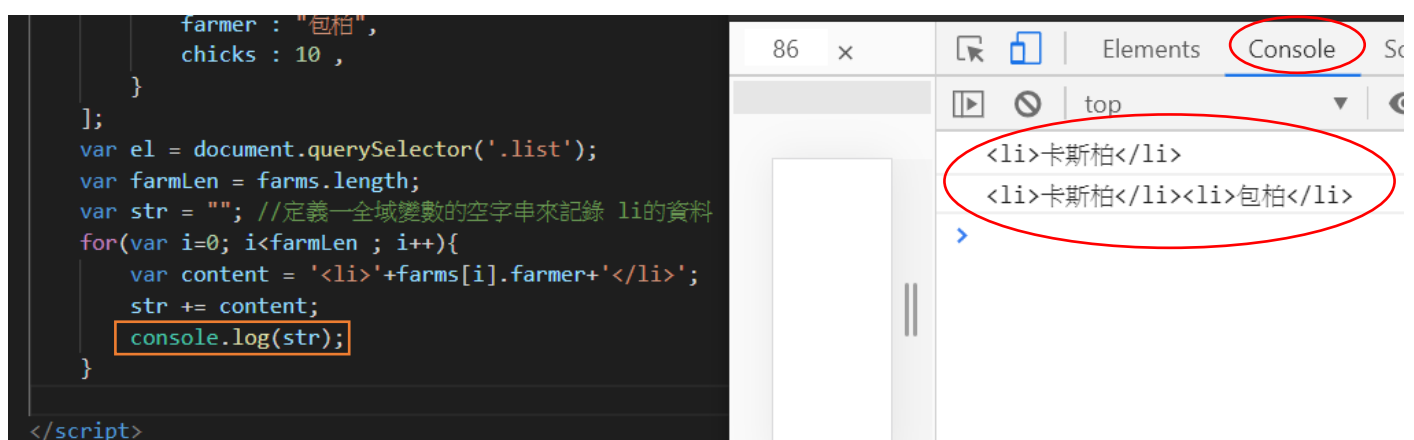
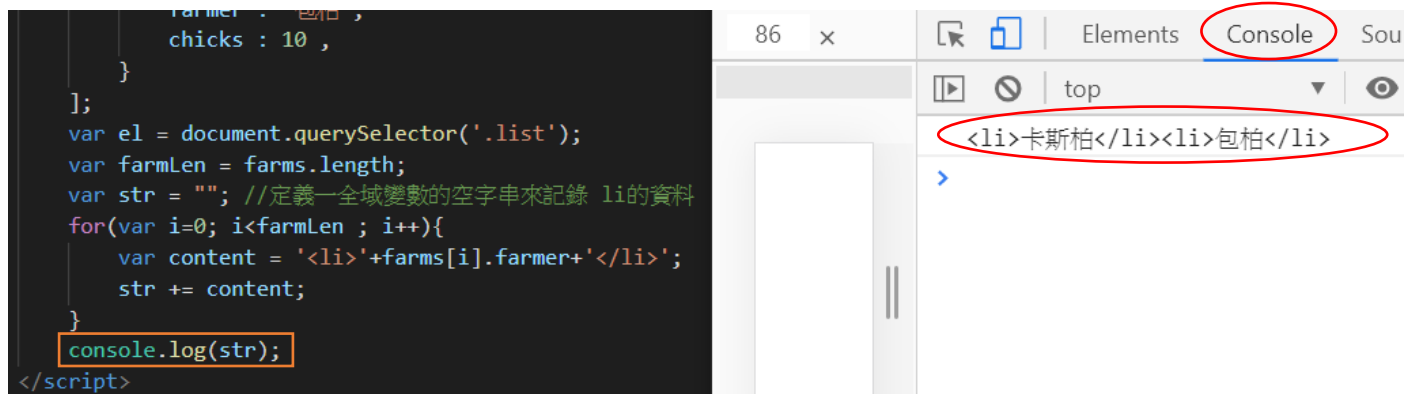
<script>
  var farms = [
    {
      farmer : "卡斯柏",
      chicks : 6 ,
    },
    {
      farmer : "包柏",
      chicks : 10 ,
    }
  ];
  var el = document.querySelector('.list');
  var farmLen = farms.length;
  var str = ""; //定義一全域變數的空字串來記錄 li的資料
  for(var i=0; i<farmLen ; i++){
    var content = '<li>'+farms[i].farmer+'</li>';
    str += content;
  }
  el.innerHTML = str;
</script>
</body>
```



- 卡斯柏
- 包柏

額外補充: 最後的 `el.innerHTML = str`; 不管放在 `for` 迴圈之中或之外都可以的。

透過以下兩張圖看到, `console.log` 放在 `for` 裡面和外面的差異, 但因為是碰到 `innerHTML` 的特性, 因此印出來的結果才都一樣(只有印出最後的資料), 遇到這種狀況就只能新增一全域變數來記錄每次 `for` 迴圈產出的東西。



18 - 150 Template literals 輕鬆進行字串相加

ES6 的新 `innerHTML` 組字串的寫法: 利用 “`” 將字串包住, 其中在把變數塞入 “`{}`” 裏頭即可。



9-79 createElement 寫法

1. 與 innerHTML 用法的差別: 先透過.createElement('') 新增一元素, 再透過.appendChild 動態的在 javascript 元素(子節點)再動態增加一新子節點。
2. 優點: 不會覆蓋掉原本的內容, 可以增加在原本內容後方(appendChild) or 前方(prependChild)

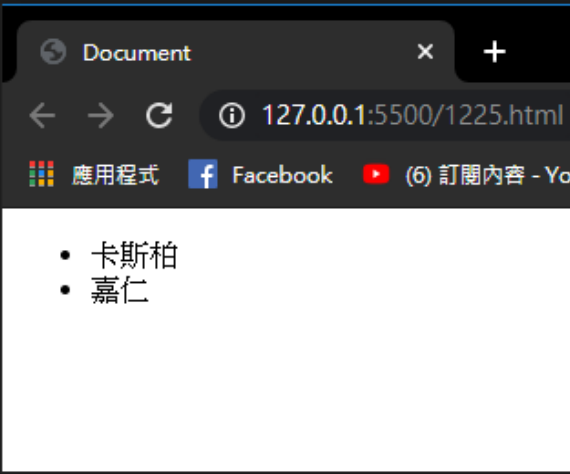
```
<title>Document</title>
<style>
  .text-color{
    color: red;
  }
</style>
</head>
<body>
  <div class="creat">
    <em>1234</em>
  </div>
<script>
  var el = document.createElement('em'); // 先製造一元素
  el.textContent = '5678'; // 填入值進去元素中
  document.querySelector('.creat').appendChild(el); //選擇要加入元素的子節點並加進去
  el.setAttribute('class', 'text-color'); // 另外也可再setAttribute進去做變化~
</script>
```



9-80 createElement 與 for 用法

範例: 印出農夫名字

```
<body>
  <ul class="list"> </ul>
<script>
  var farms = [
    {
      farmer: "卡斯柏",
      chicks: 30,
    },
    {
      farmer: "嘉仁",
      chicks: 32,
    }
  ];
  var el = document.querySelector('.list');
  for(var i=0; i<farms.length; i++){
    var str = document.createElement('li'); //定義str並製造li
    str.textContent = farms[i].farmer; //加入農夫名字進li
    el.appendChild(str); //把str加進子節點
  }
</script>
```



10-85 event 物件 – 告知你當下元素資訊

This screenshot illustrates the basic setup for an event listener on a button. The HTML code defines a button with the class 'btn'. The JavaScript code uses `document.querySelector` to find the button and attaches an `onclick` event listener that logs the event object `e` to the console. The browser's developer console shows the resulting `MouseEvent` object with properties like `isTrusted`, `screenX`, `screenY`, `clientX`, `clientY`, `ctrlKey`, `shiftKey`, `altKey`, and `metaKey`.

```
</head>
<body>
  <input type="button" value="click" class="btn">
</body>

<script>
  var el = document.querySelector('.btn');
  //透過此參數e可以獲取事件的詳細資料，
  //用法:ex: console.log(e) (e算是event的意思)
  el.onclick = function(e){
    console.log(e);
  }
</script>
</body>
```

MouseEvent {isTrusted: true, screenX: 1034, 237, clientX: 36, clientY: 19, ...}

- isTrusted: true
- screenX: 1034
- screenY: 237
- clientX: 36
- clientY: 19
- ctrlKey: false
- shiftKey: false
- altKey: false
- metaKey: false

範例：抓取 `altkey` 事件，使使用者按住 `alt` 不放+點擊 `click` 按鈕時會觸發 `alert` 函式。

This screenshot shows a more complex event listener that checks if the `alt` key is pressed during a click. The code uses `event.altKey` to determine this. When the condition is met, it triggers an `alert` function. The browser's developer console shows the `MouseEvent` object with `altKey` set to `true`. A red box highlights the `if(key == true)` condition in the code and the `altKey: true` property in the console, with a red arrow pointing from the code to the console.

```
<body>
  <input type="button" value="click" class="btn">
</body>

<script>
  var el = document.querySelector('.btn');

  el.onclick = function(e){
    var key = event.altKey;
    if(key == true){ //沒按住alt則會顯示false
      alert("你按住了alt + 滑鼠");
      console.log(e);
    }
  };
</script>
```

127.0.0.1:5500/1225.html 顯示
你按住了alt + 滑鼠

MouseEvent {isTrusted: true, screenX: 704, screenY: 488, clientX: 26, clientY: 22, ctrlKey: false, shiftKey: false, altKey: true, metaKey: false, button: 0}

10-86 各種事件綁定的差異

1. 較早期寫法

```
<body >
| <input class="btn" onclick="cli()" type="button">
</body>
<script>
|     function cli(){
|         alert("hello");
|     };
</script>
```

2. 常見寫法

```
<body >
| <input class="btn" type="button">
</body>
<script>
|     var el = document.querySelector('.btn');
|     el.onclick = function(){
|         alert("hello");
|     };
</script>
```

```
<body >
| <input class="btn" type="button">
</body>
<script>
|     var el = document.querySelector('.btn');
|     el.onclick = cli; //注意在此不能寫 cli();
|                     //將出錯，為點擊就觸發函式
|     function cli(){
|         alert("hello");
|     };
</script>
```

3. 監聽 寫法

```
<body >
| <input class="btn" type="button">
</body>
<script>
|     var el = document.querySelector('.btn');
|
|     el.addEventListener('click', cli, false); //監聽
|
|     function cli(){
|         alert("hello");
|     };
</script>
```



10-87 addEventListener – 監聽事件

`addEventListener (事件(語法), 函式 , false)` 最後的 `false` 不寫也可以，因為預設就是 `false`。

10-88 綁定事件的語法差異

`onclick` 不可同時綁定兩個事件，其永遠只會讀取最後一筆資料，來覆蓋掉前面的資料。
(靈活性較低)

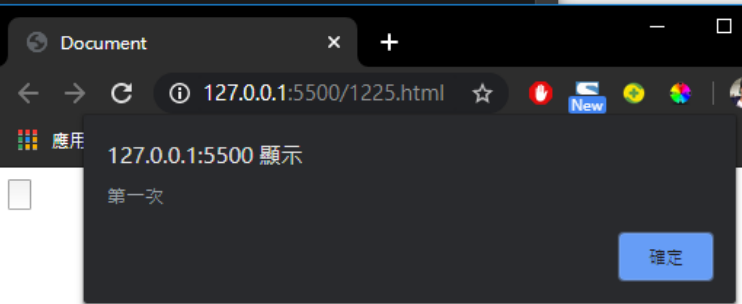
```
<body>
  <input class="btn" type="button">
</body>
<script>
  var el = document.querySelector('.btn');
  el.onclick = function(){
    alert('第一次');
  }
  el.onclick = function(){
    alert('第二次');
  }
</script>
```



`addEventListener` 將可成功執行綁定的次數 (以下範例將執行兩次)

```
<body>
  <input class="btn" type="button">
</body>
<script>
  var el = document.querySelector('.btn');
  el.addEventListener('click',cli1,false);
  el.addEventListener('click',cli2,false);

  function cli1(){
    alert("第一次");
  };
  function cli2(){
    alert("第二次");
  };
</script>
```



10-89 event bubbling(氣泡)、event capturing(捕捉) 差異

解釋 addEventListener 內要指定 false 的原因：

False (事件氣泡)：從 標籤(指定元素)內 往標籤外找(子層往父層)(DOM 的最上層網最下層找)。
因此一般都是需要此需求，針對使用者點擊的當下元素來觸發事件。

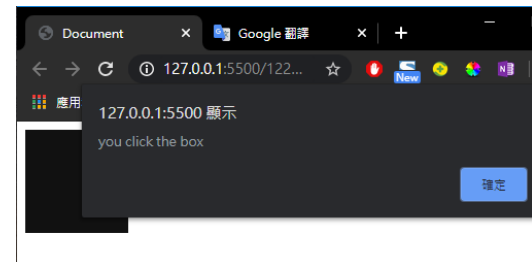
True (事件捕捉)：從 標籤(指定元素)外 往標籤內找 (父層往子層)(DOM 的最下層網最上層找)。
但有時會需要先讀取最外層資料，再網內讀取的情況。

範例：以下將會有穿透的效果發生 (即 點擊 box 之後 底下的 body 也會被點到，而跟著作動)。

1. false：點擊 box 顯示 click box，再來才顯示 click body

```
<body class="body" >
  <div class="box">
  </div>
  <script>
    var box = document.querySelector('.box');
    box.addEventListener('click',cliBox,false); //box clicked
    var body = document.querySelector('.body');
    body.addEventListener('click',cliBody,false); //body clicked

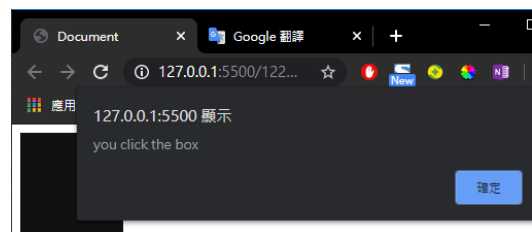
    function cliBox(){
      alert('you click the box');
    };
    function cliBody(){
      alert('you click the body');
    };
  </script>
</body>
```



2. true：點擊 box 顯示 click box，再來才顯示 click body

```
<body class="body" >
  <div class="box">
  </div>
  <script>
    var box = document.querySelector('.box');
    box.addEventListener('click',cliBox,true); //box clicked
    var body = document.querySelector('.body');
    body.addEventListener('click',cliBody,true); //body clicked

    function cliBox(){
      alert('you click the box');
    };
    function cliBody(){
      alert('you click the body');
    };
  </script>
</body>
```



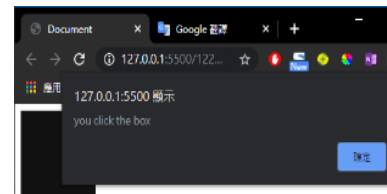
此事件表示當我們點擊某元素時，其實也會連他的父層一起點擊到。

10-90 stopPropagation – 終止冒泡事件

透過.stopPropagation()這個函式 終止氣泡往父層找。

範例：在此情況下 假如案到 box，即只執行 box，後就停止往下執行 body 的動作。

```
<body class="body" >
  <div class="box">
  </div>
  <script>
    var box = document.querySelector('.box');
    box.addEventListener('click',cliBox,false); //box clicked
    var body = document.querySelector('.body');
    body.addEventListener('click',cliBody,false); //body clicked
    console.log(body);
    function cliBox(e){          // 加入參數- 撈出e事件裡詳細資訊
      alert('you click the box');
      e.stopPropagation();       //透過此語法阻止氣泡往外層標籤找(終止動作)
    };
    function cliBody(){
      alert('you click the body');
    };
  </script>
</body>
```



氣泡終止參考資料：<https://ithelp.ithome.com.tw/articles/10198999>

10-91 preventDefault – 取消預設觸發行為

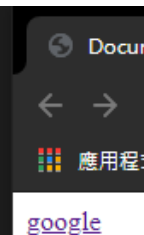
此功能常用於表單寄送，例如 submit 按鈕案的當下原本會直接傳到後端去，但透過 preventDefault 來取消這項默認，以使用 js 先檢查資料(表單、欄位等)是否有錯或未填寫，再用 post 來傳送。

範例：原先點選<a>標籤會進行網址跳轉，但透過 preventDefault 來取消，按下 google 將無反應。

```
</head>
<body>
  <a href="https://www.google.com/" class="link">google</a>
  <script>
    var stopGoogle = document.querySelector('.link');
    stopGoogle.addEventListener('click',stop,false);

    function stop(e){
      e.preventDefault(); //終止、取消預設功能
    }

  </script>
</body>
```



10-92 e.target - 了解目前所在元素位置

此方法可以透過 if else 等等判斷後，讓我們的使用者確切的點擊到我們想要的東西才執行動作。

1. 查明: e.target.nodeName: target 裡的 nodeName

```
<body>
  <ul class="link">
    <li>
      <a href="#">google</a>
    </li>
  </ul>
<script>
  var el = document.querySelector('.link');
  el.addEventListener('click',target,false);

  function target(e){
    console.log(e);
  }
</script>
```

- google

Styles Event Listeners DOM Break

Filter :hov .cls +

Console What's New Anim

top

onwebkitfullscreenchange
onwebkitfullscreenerror:
nodeType: 1
nodeName: "A"
baseURI: "http://127.0.0.0
isConnected: true

2. 案例：點擊到兩次<a> 與一次

```
</style>
</head>
<body>
  <ul class="link">
    <li>
      <a href="#">google</a>
    </li>
  </ul>
<script>
  var el = document.querySelector('.link');
  el.addEventListener('click',target,false);

  function target(e){
    console.log(e.target.nodeName); //當選擇e.target時會秀出整個標籤區域
  } //因此再往target底層去找nodeName將可直接選到該節點名稱
</script>
```

- google

Styles Event List

Filter :ho

element.style {

Console W

top

2 A
LI

3. 案例：點即到部分才會顯示 you clicked ，點擊其他地方都無反應。

```
<body>
  <ul class="link">
    <li>
      <a href="#">google</a>
    </li>
  </ul>
<script>

  var el = document.querySelector('.link');
  el.addEventListener('click',target,false);

  function target(e){ //點到<li>才觸發
    if(e.target.nodeName == "LI"){
      alert('you clicked <li>');
    };
  }
</script>
```

- google

Document

127.0.0.1:5500/122

應用程式

127.0.0.1:5500 顯示

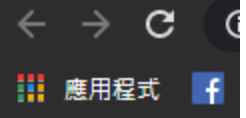
you clicked

10-93 change – 表單內容更動內容時觸發

範例：更換地區表單 底下顯示岡地區住戶。

```
<body>
  <select name="" id="areaId">
    <option value="岡山區">岡山區</option>
    <option value="橋頭區">橋頭區</option>
  </select>
  <ul class="list"></ul>
<script>
  var area = document.getElementById('areaId');
  var list = document.getElementsByClassName('list');
  var country = [
    {
      name : "查理",
      place : "岡山區",
    },
    {
      name : "卡司博",
      place : "橋頭區"
    },
    {
      name : "陳嘉仁",
      place : "岡山區",
    }
  ];
  area.addEventListener('change',updateList,false);

  //此函式功能：組字串<li>後再innerHTML帶到<ul>之中
  var countryLen = country.length;
  function updateList(e){
    var select = e.target.value;
    console.log(select);
    var str = "";
    for(var i=0; i<countryLen ;i++){
      if(select == country[i].place){
        str += '<li>'+country[i].name+'</li>';
      }
    }
    list[0].innerHTML = str;
    //這裡使用list[0]原因是上方採用getElementsByClassName的關係
  };
</script>
```



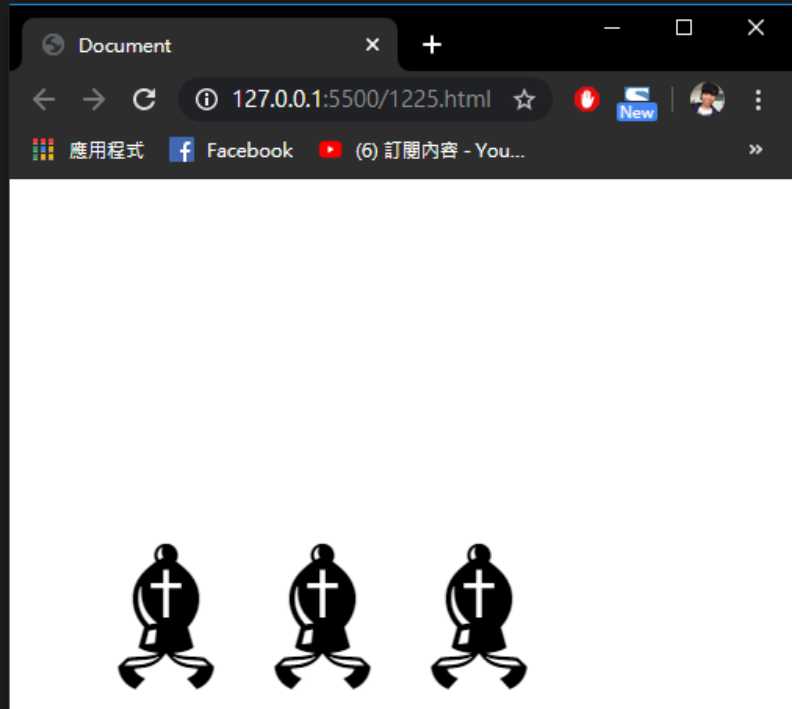
岡山區 ▼

- 查理
- 陳嘉仁

10-94 keyCode - 點擊鍵盤，發射火箭

透過 keydown 觸發鍵盤按鈕 再由 e.keyCode 抓取資料 比對兩者是否吻合，便更改有動畫的 css

```
<style>
  .rocket{
    font-size: 100px;
    position: absolute;
    bottom: 0px;
    transition: all 3s cubic-bezier(0.36, 0.05, 0.73, 0.22);
  }
  .rocket-1{
    left: 10%;
  }
  .rocket-2{
    left: 30%;
  }
  .rocket-3{
    left: 50%;
  }
</style>
</head>
<body>
  <div class="rocket rocket-1">✈</div>
  <div class="rocket rocket-2">✈</div>
  <div class="rocket rocket-3">✈</div>
<script>
  var body = document.body;
  function goRocket(e){
    var key = e.keyCode;
    switch(key){
      case 49: //隨機設置 讓style 裡的 bottom距離遠離螢幕
        document.querySelector('.rocket-1').style.bottom = '2000px';
        break;
      case 50:
        document.querySelector('.rocket-2').style.bottom = '2000px';
        break;
      case 51:
        document.querySelector('.rocket-3').style.bottom = '2000px';
        break;
    }
  }
  body.addEventListener('keydown', goRocket);
</script>
```



10-95 blur – 離開焦點時進行事件觸發

案例：當屬標在框框內顯示後又未輸入值，而離開即會觸發 blur

```
<body>
  <input type="text" id="textId">
</script>
var text = document.getElementById('textId');
text.addEventListener('blur', checkTextContent);

function checkTextContent(e){
  var str = e.target.value;
  if(str == ""){ //如果target的value是空值就印出提示訊息
    alert("此欄位不可為空");
  }
}
</script>
</body>
```



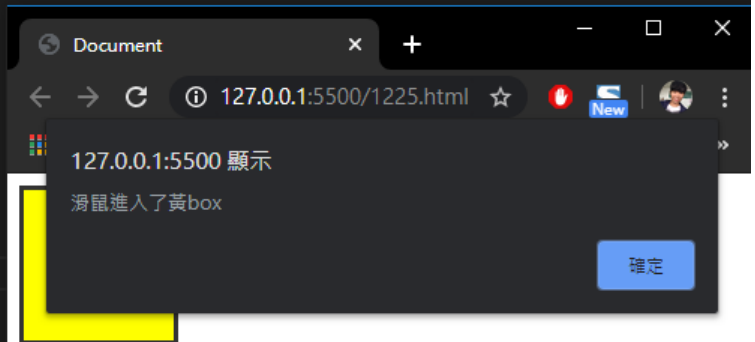
錯誤案例：因此 blur 事件 是 for addEventListener 這款監聽事件的事件，因此底下作法無法實施!

```
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <title>Document</title>
8 </head>
9 <body>
10  <input type="text" id="textId">
11 </script>
12
13  var text = document.getElementById('textId');
14  text.addEventListener('blur', checkTextContent);
15
16  function checkTextContent(){
17    if(text.value == ""){
18      alert("此欄位不可為空");
19    }
20  }
21 </script>
```



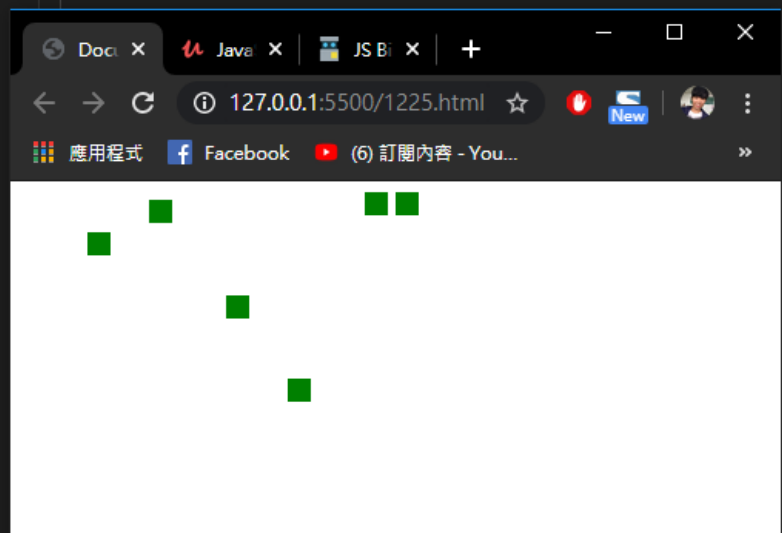
10-96 mouse – 當滑鼠滑入指定內容時觸發

```
    }  
  </style>  
</head>  
<body>  
  <div class="box"></div>  
  
<script>  
  var box = document.querySelector('.box');  
  function inMouse(){  
    alert("滑鼠進入了黃box");  
  }  
  box.addEventListener('mousemove', inMouse);  
</script>
```



```
01 > # 1225.css > @keyframes fontbulger  
1 .box{  
2   background: green;  
3   height: 15px;  
4   width: 15px;  
5   position: absolute;  
6 }  
7 .box1{  
8   top: 0px;  
9   left: 50px;  
10  animation: fontbulger 2s infinite;  
11 }  
12 .box2{  
13   top: 0px;  
14   left: 90px;  
15   animation: fontbulger 3s infinite;  
16 }  
17 .box3{  
18   top: 0px;  
19   left: 140px;  
20   animation: fontbulger 2.5s infinite;  
21 }  
22 .box4{  
23   top: 0px;  
24   left: 180px;  
25   animation: fontbulger 1.2s infinite;  
26 }  
27 .box5{  
28   top: 0px;  
29   left: 230px;  
30   animation: fontbulger 4s infinite;  
31 }  
32 .box6{  
33   top: 0px;  
34   left: 250px;  
35   animation: fontbulger 4s infinite;  
36 }  
37 @keyframes fontbulger {  
38   0% {  
39     top: 0px;  
40   }  
41   30% {  
42     top: 150px;  
43   }  
44   100% {  
45     top: 0px;  
46   }  
47 }
```

```
01 > 1225.html > html > body > script  
1 <!DOCTYPE html>  
2 <html lang="en">  
3 <head>  
4   <meta charset="UTF-8">  
5   <meta name="viewport" content="width=device-width,&br/>6   <meta http-equiv="X-UA-Compatible" content="ie=edge  
7   <title>Document</title>  
8   <link rel="stylesheet" href="1225.css">  
9 </head>  
10 <body>  
11   <div class="box box1"></div>  
12   <div class="box box2"></div>  
13   <div class="box box3"></div>  
14   <div class="box box4"></div>  
15   <div class="box box5"></div>  
16   <div class="box box6"></div>  
17 <script>  
18   var el = document.querySelectorAll('.box');  
19  
20   var Len = el.length;  
21   //因數量多以下以陣列配上for來監聽  
22   for(var i = 0;i<Len;i++){  
23     el[i].addEventListener('mousemove', function(e){  
24       alert('你輸了!');  
25     });  
26   }  
27 </script>  
28 </body>  
29 </html>
```



10-97 網頁座標 - 了解 screen、page、client 箇中差異

Page：範圍： 自製、客製的整個網頁

Client：範圍： 你開啟來的視窗大小，判斷你在哪裡

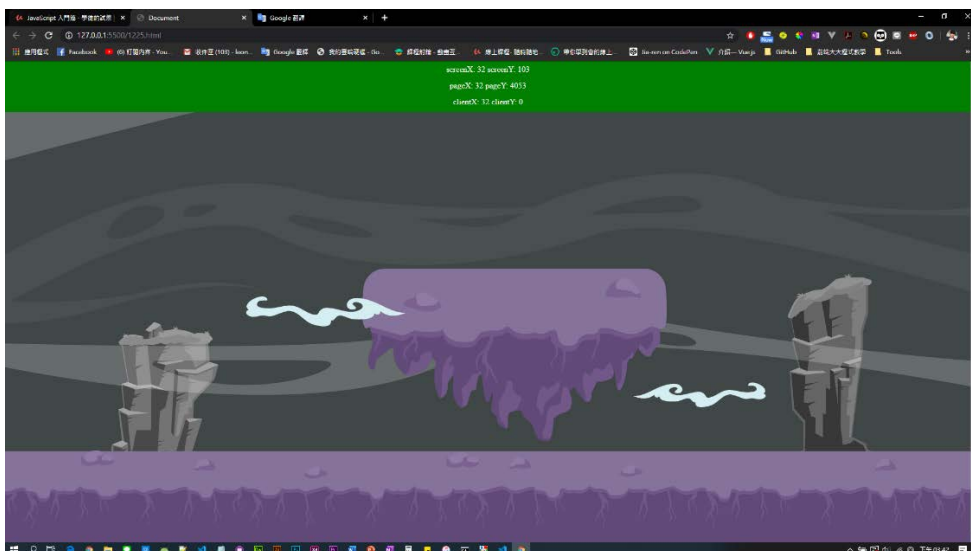
Screen：範圍： 並不是看瀏覽器，而是看你整個桌面螢幕大小(解析度)來判定，偵測出整個螢幕的解析度後，再看你滑鼠目前位於整個螢幕的哪裡。

```
<body id="body">
  <div class="wrap">
    <div class="header">
      <p>
        screenX: <span class="screenX"></span>
        screenY: <span class="screenY"></span>
      </p>
      <p>
        pageX: <span class="pageX"></span>
        pageY: <span class="pageY"></span>
      </p>
      <p>
        clientX: <span class="clientX"></span>
        clientY: <span class="clientY"></span>
      </p>
    </div>
  </div>
  <script>
    var sX = document.querySelector('.screenX');
    var sY = document.querySelector('.screenY');
    var pX = document.querySelector('.pageX');
    var pY = document.querySelector('.pageY');
    var cX = document.querySelector('.clientX');
    var cY = document.querySelector('.clientY');

    function getPosition(e) {
      sX.textContent = e.screenX;
      sY.textContent = e.screenY;
      pX.textContent = e.pageX;
      pY.textContent = e.pageY;
      cX.textContent = e.clientX;
      cY.textContent = e.clientY;
    }

    var el = document.body;
    el.addEventListener('mousemove', getPosition, false);
  </script>
</body>
```

```
10 article, aside, canvas, details, embed,
11 figure, figcaption, footer, header, hgroup,
12 menu, nav, output, ruby, section, summary,
13 time, mark, audio, video {
14   margin: 0;
15   padding: 0;
16   border: 0;
17   font-size: 100%;
18   font: inherit;
19   vertical-align: baseline;
20 }
21 /* HTML5 display-role reset for older browsers */
22 article, aside, details, figcaption, figure,
23 footer, header, hgroup, menu, nav, section {
24   display: block;
25 }
26 body {
27   line-height: 1;
28 }
29 ol, ul {
30   list-style: none;
31 }
32 blockquote, q {
33   quotes: none;
34 }
35 blockquote:before, blockquote:after,
36 q:before, q:after {
37   content: '';
38   content: none;
39 }
40 table {
41   border-collapse: collapse;
42   border-spacing: 0;
43 }
44
45 html{
46   height: 100%;
47 }
48 body{
49   background-image: url('../img/map.png');
50   background-repeat: no-repeat;
51   background-size: 100% 100%;
52   background-position: bottom ;
53 }
54 .wrap{
55   max-width: 1920px;
56   height: 5000px;
57   background: url('../img/dark.png') bottom #404747 no-repeat;
58 }
59 .header{
60   height: 100px;
61   position: fixed;
62   top: 0;
63   width: 100%;
64   color: #fff;
65   text-align: center;
66   background: green;
67 }
68 .header p{
69   padding: .5em;
70 }
```



10-98 網頁座標 - 應用篇

範例：添加自製滑鼠游標

```
41 border-collapse: collapse;
42 border-spacing: 0;
43 }
44
45 html{
46 height: 100%;
47 }
48 body{
49 background-image: url('/img/d...');
50 background-repeat: no-repeat;
51 background-size: 100%;
52 background-position: center;
53 }
54
55 .wrap{
56 max-width: 1920px;
57 height: 500px;
58 background: url('/img/d...');
59 }
60 .header{
61 height: 100px;
62 position: fixed;
63 top: 0;
64 width: 100%;
65 color: #fff;
66 text-align: center;
67 background: green;
68 }
69 .header p{
70 padding: .5em;
71 }
72
73 body{
74 cursor: none;
75 }
76 .mouseImg{
77 position: fixed;
78 }
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2
```

10-99 事件監聽優化篇 – 從父元素來監聽子元素內容

1. 範例：用父元素來選取較大範圍，再用 `target.nodeName` 來去做過濾，解決以往要對每個元素做監聽綁定的差效率

Step1: (如果擴大變成父層 `.list` 這個 `` 的話 這時雖然點擊查理或卡斯柏可以正常顯示，但如果我點集到空白之處-`` 的部分就會顯示整個 `` 出來)

```
<body>
  <ul class="list">
    <li>查理</li>
    <li>卡斯柏</li>
  </ul>

  <script>
    var list = document.querySelector('.list');
    list.addEventListener('click', checkName);

    function checkName(e) {
      console.log(e.target);
    }
  </script>
</body>
```

153 x 241

Console

top

- 查理 1225.html:24
- 卡斯柏 1225.html:24
- <ul class="list"> 1225.html:24
 - 查理
 - 卡斯柏

Step2: 我想要的是他很剛好的點擊到 `li` 才去觸發事件 因此便要寫 `if` 判別式。

以下便是點集 `` 之外地方的動作都不會進行 `console.log` 印出東西的範例。

*** **return 特性**： 1. 回傳資料 2. 還傳空值，做為一個中斷點 ***

```
<body>
  <ul class="list">
    <li>查理</li>
    <li>卡斯柏</li>
  </ul>

  <script>
    var list = document.querySelector('.list');
    list.addEventListener('click', checkName);

    function checkName(e) {
      if (e.target.nodeName !== "LI") { // 這裡的寫法便是如果nodeName不等於LI
        return // 就用return做回傳空值的動作(回傳一個空值，使其不往下執行其他動作)
      }; // return 特性： 1. 回傳資料 2. 還傳空值，做為一個中斷點
      console.log(e.target);
    }
  </script>
</body>
```

一般的做法 (2.範例有誤) ↓ ↓ ↓

2. 範例：因為 `querySelector` 的特性(只撈第一筆資料)就算用滑鼠點擊卡斯柏也一樣只會出現查理

```
<body>
  <ul class="list">
    <li>查理</li>
    <li>卡斯柏</li>
  </ul>

  <script>
    var list = document.querySelector('.list li');
    list.addEventListener('click', checkName);

    function checkName(e){
      console.log(e.target);
    }
  </script>
</body>
```

Browser Console Log:

Log Entry	File	Line
查理	1225.html	21
查理	1225.html	21
查理	1225.html	21
查理	1225.html	21

3. 範例：解決以上方法可用 `querySelectorAll` 解決

```
</head>
<body>
  <ul class="list">
    <li>查理</li>
    <li>卡斯柏</li>
  </ul>

  <script>
    var list = document.querySelectorAll('.list li');

    for(var i=0;i<list.length;i++){
      list[i].addEventListener('click', checkName);
    }

    function checkName(e){
      console.log(e.target);
    }
  </script>
</body>
```

Browser Console Log:

Log Entry	File	Line
卡斯柏	1225.html	24
卡斯柏	1225.html	24
查理	1225.html	24
卡斯柏	1225.html	24

11 – 101 什麼是 localStorage

瀏覽器的資料庫

```
<script>
  // 設置 (Key值, Value)
  localStorage.setItem("myName", "陳嘉仁");
  // 透過Key 取得value
  var show = localStorage.getItem("myName");
  console.log(show);
</script>
```

Browser Application Tab - Storage:

Key	Value
myName	陳嘉仁

11-102 setItem、getItem 基本操作

範例：利用監聽達到輸入值存進 localStorage 再取出來的效果。

The screenshot shows a web application with a text input field, a 'click' button, and a 'show' button. The DevTools Application panel is open, showing the 'Storage' tab. The 'Local Storage' section for 'http://127.0.0.1:5500' contains one item with the key 'myName' and the value '莊憶萱'.

```
</head>
<body>
  <input type="text" class="textName">
  <input type="button" class="btnName" value="click">
  <input type="button" class="showbtnName" value="show">
  <script>
    var btn = document.querySelector('.btnName');
    var show = document.querySelector('.showbtnName');

    btn.addEventListener("click",enterLoc);
    show.addEventListener("click",showLoc);

    function enterLoc(e){
      var text = document.querySelector('.textName').value;//抓出輸入框值
      localStorage.setItem("myName",text); //寫入localStorage
    }
    function showLoc(e){
      alert(localStorage.getItem('myName'));//用getItem取值並alert出來
    }
  </script>
</body>
</html>
```

11-103 JSON.parse、JSON.stringify 來編譯資料

1. localStorage 只吃 string 的資料行別，諸如 array 等格式則不吃。因此透過 JSON.stringify 來將 array 的資料字串化。
2. 當我們要把資料存到 localStorage 時不論是 json 或是 object、array 格式時，都必須先轉成 string 格式。事後如又需要從 localStorage 中撈出資料，或者從其他資料庫中撈出資料，都必須再把 string 轉成原本的格式，因此就需要用到 parse(解析)，將資料再次還原成原資料了。

範例：將一陣列資料存進 localStorage 中，再取出。

The screenshot shows a web application with a script that stores an array of farmer data in localStorage and then retrieves it. The DevTools Application panel is open, showing the 'Storage' tab. The 'Local Storage' section for 'http://127.0.0.1:5500' contains one item with the key '岡山' and the value '[[{"farmer": "卡斯柏"}, {"farmer": "陳嘉仁"}]]'. The Console panel shows the output of the script: '卡斯柏'.

```
<body>
  <script>
    var farms = [
      {
        farmer: "卡斯柏"
      },
      {
        farmer: "陳嘉仁"
      }
    ];

    //將上方農場的資料轉乘字串，並存入localStorage
    var farmsString = JSON.stringify(farms); //array to string
    localStorage.setItem("岡山",farmsString);

    //從localStorage中取出資料(字串狀態)，在將其解析還原成物件(陣列)的資料型態
    var getData = localStorage.getItem("岡山");
    var getDataArry = JSON.parse(getData); //string to array

    console.log(getDataArry[0].farmer); //此時已恢復成原型態
  </script>
</body>
```

11-104 - data-* - 透過 dataset 讀取資料

1. 當今天遇到資料過於龐大時，可以透過自訂義標籤裡的屬性，從而判斷當屬性值為多少時，再去做新增、修改、刪除 等等動作 此為較新的做法

2. 自定義屬性進去標籤，用於想要埋一些自己的資料去做綁定動作時(注意:命名中不可摻雜"大寫"、"-" 但是底線"_"可以) 否則撈資料出來時會 **undefined**

3. 透過 `.dataset` 可以撈出自訂義進去標籤裡的屬性出來

The screenshot illustrates the first example. On the left, the HTML code defines a list with a single item: `<li class="listli" data-num="2" data-jiaren="24old">1234`. The `data-num` and `data-jiaren` attributes are highlighted with red boxes. Below the HTML, a JavaScript script uses `document.querySelector('.listli').dataset` to access these attributes and logs them to the console. On the right, the browser's developer tools show the DOM element with the text "1234". The console output shows a `DOMStringMap` object with properties `num: "2"` and `jiaren: "24old"`. The `24old` value is highlighted in the console.

範例：撈出自訂義的學號、年齡的屬性

The screenshot illustrates the second example. On the left, the HTML code defines a list item: `<li class="listli" data-school_num="2" data-jiare_age="24old">陳嘉仁學號、年齡`. The `data-school_num` and `data-jiare_age` attributes are highlighted with red boxes. Below the HTML, a JavaScript script defines a `checkData` function that uses `e.target.dataset` to access these attributes and logs them to the console. On the right, the browser's developer tools show the DOM element with the text "陳嘉仁學號、年齡". The console output shows the values `2` and `24old` being logged, which are highlighted with orange arrows pointing to the corresponding attributes in the HTML code.

15 – 122 透過 XMLHttpRequest 物件跨瀏覽器撈資料

readyState (撈資料的狀態)：

- readyState : 0 - 已產生一個 XMLHttpRequest，但尚未連結你要的資料。
- readyState : 1 - 用了 open() 讀取資料，但還沒將資料串送過去。
- readyState : 2 - 偵測到你已經有使用 .send() 了
- readyState : 3 - 資料大，還正在 loading
- readyState : 4 - 你撈到資料了，數據已完全接收

```
<body>
<script>
// 起手式：用XMLHttpRequest去跟其他伺服器要資料
// 執行以下這行的步驟會跑出 readyState:0
var xhr = new XMLHttpRequest();

// 準備撈資料(並非可撈到)：
// 執行以下這行的步驟會使 readyState變為1(用了open() 讀取資料，但還沒將資料串送過去。) ， 另外此時的responseText仍是空值
/* .open( 格式，要讀取的網址，同步與非同步)
| | | | 以上格式又分： get(讀取網址裡的資料) / post傳送資料到伺服器(常用於帳號驗證，例如傳資料過去確認資料是否重複等等) */
xhr.open('get','https://hexschool.github.io/ajaxHomework/data.json',true);

// 傳送null用意是，因為我上面只用get，我並沒有真的要傳送資料，只打算讀取資料，所以可以傳送空值
// 因為今天只是單純get資料而已並不是真的要傳資料過去，也沒有使用post去做驗證，所以傳送null空值過去是可以的
// 除非有需要跟後端或伺服器確認否些狀態等等，才會用到post做驗證
// 執行以下這行的步驟會使 readyState變為4(你撈到資料了，數據已完全接收) ， 但此時的responseText便因為.send而帶入了值
xhr.send(null);

// 使用open 'post' 時機：會讓我 send('check') 這筆資料過去後端，讓後端再回傳 'check' 此資料的檢查結果給我

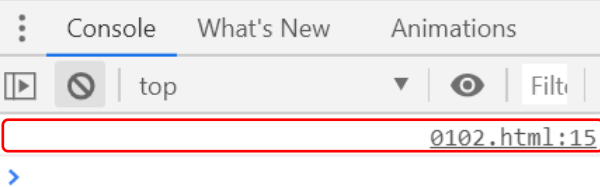
</script>
</body>
```

15-123 AJAX 非同步觀念(上)

非同步(true)：不等資料傳回來，就讓程式繼續往下跑，等到回傳才自動回傳

範例：要在第 15 行 console.log 出值卻沒撈到值，原因是因為非同步的概念是當 14 行.send 過去之後(readyState : 4 就已形成)，所以她不會等到資料還傳回來就直接往 15 行執行了，因此撈不到資料。

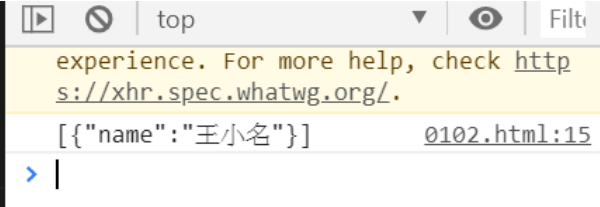
```
10 <script>
11
12 var xhr = new XMLHttpRequest();
13 xhr.open('get', 'https://hexschool.github.io/ajaxHomework/data.json', true);
14 xhr.send(null);
15 console.log(xhr.responseText);
16
17 </script>
```



同步(false)：他等資料傳回來，才讓程式往下跑

範例：因此上述改成 false 便可行。

```
10 <script>
11
12 var xhr = new XMLHttpRequest();
13 xhr.open('get', 'https://hexschool.github.io/ajaxHomework/data.json', false);
14 xhr.send(null);
15 console.log(xhr.responseText);
16
17 </script>
```



15-124 AJAX 非同步觀念(下)

如需同上單元使用非同步作法來使程式正常撈資料則可以用 onload 狀態

範例：當.send 以後，我執行並確認 onload (readyState : 4) 以後才去觸發 function 內的事件。

```
10 <script>
11
12 var xhr = new XMLHttpRequest();
13 xhr.open('get', 'https://hexschool.github.io/ajaxHomework/data.json', true);
14 xhr.send(null);
15 xhr.onload = function(){ //當onload確認資料有回傳回來時
16   console.log(xhr.responseText);
17 }
18
19 </script>
```



範例 2：直接印出來

```
10 <script>
11
12 var xhr = new XMLHttpRequest();
13 xhr.open('get', 'https://hexschool.github.io/ajaxHomework/data.json', true);
14 xhr.send(null);
15 xhr.onload = function(){ //當onload確認資料有回傳回來時
16   var str = JSON.parse(xhr.responseText); //string to array
17   document.write(str[0].name); //撈出json格式裡的名字
18 }
19
20 </script>
```



15-125 HTTP 狀態碼

Status 200：資料有正確回傳，有撈到。

Status 404：資料讀取錯誤，沒撈到。

```
var xhr = new XMLHttpRequest();
xhr.open('get', 'https://hexschool.github.io/ajaxHomework/data1.json', true);
xhr.send(null);
xhr.onload = function(){ //當onload確認資料有回傳回來時
  if(xhr.status == 200){
    var str = JSON.parse(xhr.responseText); //string to array
    document.write(str[0].name); //撈出json格式裡的name
  }else{
    document.write("資料錯誤");
  }
}
```

正確網址是 data.json

資料錯誤

Name	Status
0102.html	200
data1.json	404
ws	101
3 / 9 requests 8.0 KB / 2	

15-126 Cross-Origin Resource Sharing (CORS)

CORS (跨來源資源共用)：是否可以跨網域撈取資料

有些網站會鎖住 CORS 不做共享的動作，只給予同網域的網站做撈取，以免資安上有所風險，因此如果只是使用自己本地端本機或是不同網域要做撈取的動作，那將無法成功撈取未開啟 CORS 網域的資料。

案例：以下去 get 一個未提供 CORS 服務的 opendata 網域，因此會跑出提示訊息：已被 CORS 策略阻止：所請求的資源上沒有 “Access-Control-Allow-Origin” 標頭。

```
var xhr = new XMLHttpRequest();
xhr.open('get', 'http://opendata.epa.gov.tw/webapi/Data/REWIQA/?$orderby=SiteName&$skip=0&$top=1000&format=json');
xhr.send(null);
xhr.onload = function(){
  if(xhr.status == 200){
    var str = JSON.parse(xhr.responseText);
    document.write(str[0].name);
  }else{
    document.write("資料錯誤");
  }
}
</script>
```

DevTools - 127.0.0.1:5501/0102.html

Access to XMLHttpRequest at 'http://opendata.epa.gov.tw/webapi/Data/REWIQA/?\$orderby=SiteName&\$skip=0&\$top=1000&format=json' from origin 'http://127.0.0.1:5501' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

以上網址也可透過 test-cors.org 來做檢測

Client

HTTP Method: GET

With credentials? ☐

Request Headers

Request Content

Send Request

Server

Remote: Local

Remote URL: http://opendata.epa.gov.tw

Results

Code

Link to this test

Sending GET request to

http://opendata.epa.gov.tw/webapi/Data/REWIQA/?\$orderby=SiteName&\$skip=0&\$top=1000&format=json

Fired XHR event: loadstart

Fired XHR event:readystatechange

Fired XHR event: error

XHR status: 0

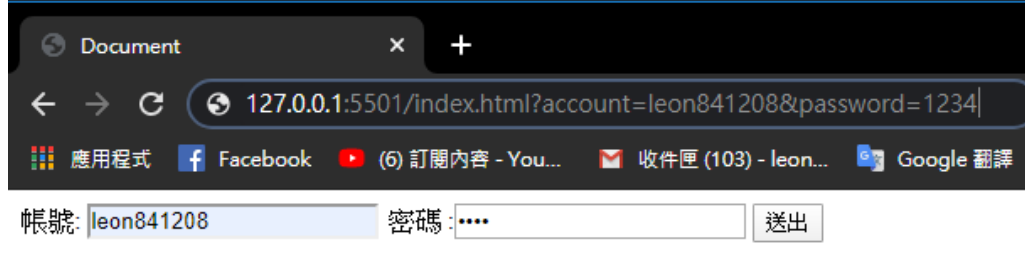
XHR status text:

Fired XHR event: loadend

15-127 傳統表單輸入介紹

送出後以下輸入的帳號密碼將會以此格式顯示在網址上：“?”後接 “name=” ” & “name” =” ”
index.html?account=leon841208&password=1234

```
<body>
  <form action="index.html"> <!-- 表單送至原page-->
    帳號:<input type="text" name="account">
    密碼:<input type="password" name="password">
      <input type="submit" value="送出">
  </form>
</body>
```



The screenshot shows a web browser window with a single tab titled 'Document'. The address bar displays the URL '127.0.0.1:5501/index.html?account=leon841208&password=1234'. Below the address bar, there are several icons for applications: '應用程式', 'Facebook', '(6) 訂閱內容 - You...', '收件匣 (103) - leon...', and 'Google 翻譯'. The main content area of the browser shows the rendered HTML form. It consists of a label '帳號:' followed by a text input field containing 'leon841208', a label '密碼:' followed by a password input field with masked characters '....', and a '送出' (Submit) button.

15-128 AJAX POST 寫法

註冊 (表單輸入 格式)

這裡提供一 api 做練習，原理是 假如我要與後端工程師做配合，得傳遞帳密過去以下網只做確認的動作看看帳密是否有被註冊走 Api 使用講解 - URL: 做驗證內容的網址 / Data: 資料格式 / Success Response: 如沒有問題(無人使用此組帳密)便回傳物件做通知 / Error Response: 失敗(已有人註冊過)

setRequestHeader : (header , value) : 設定我要傳送過去的格式

以下是表單輸入的格式：

```
setRequestHeader("Content-type","application/x-www-form-urlencoded");
```

注意，此範例僅供練習，並不會儲存用戶資料至資料庫(僅緩存)。

註冊

新增一個帳號。

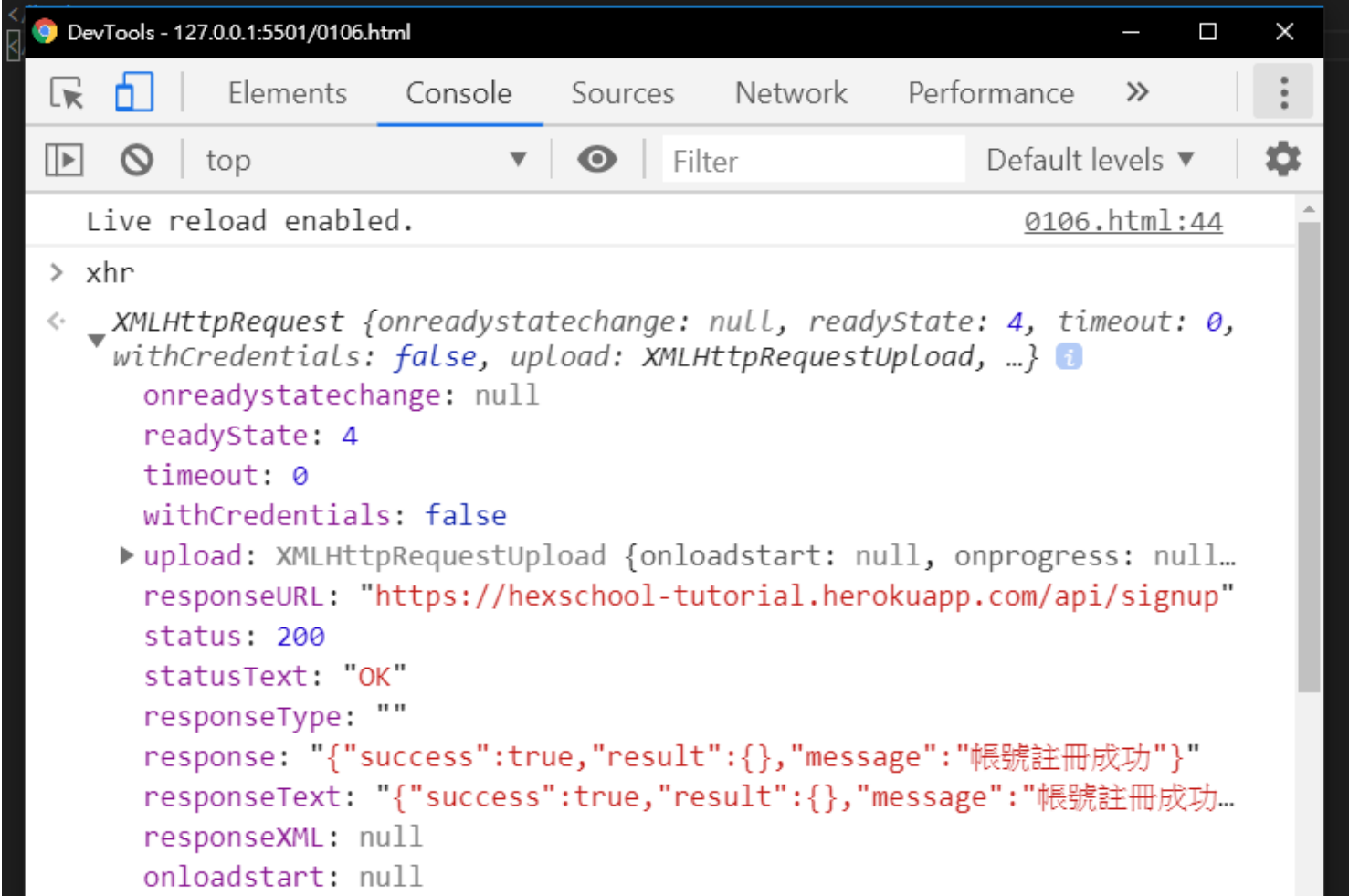
- Method: POST
- URL: `https://hexschool-tutorial.herokuapp.com/api/signup`
- Data:

```
{  email: 'lovef2e@hexschool.com',  password: '12345678'}
```
- Success Response:

```
{  "success": true,  "result": {},  "message": "帳號註冊成功"}
```
- Error Response:

```
{  "success": false,  "result": {},  "message": "此帳號已被使用"}
```

```
<body>
<script>
var xhr = new XMLHttpRequest(); //新增xhr
xhr.open('post', 'https://hexschool-tutorial.herokuapp.com/api/signup', true); //開啟要傳過去驗證值的網址
xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded"); //設定我要傳過去驗證的是甚麼格式
xhr.send('email=sde@gmail.com&password=0000'); //傳送的值(相對應內容)
//此格式將模擬一般form表單輸入的狀況 也是 post API 最常見的格式之一
</script>
```



15-130 AJAX JSON 傳遞 (POST 寫法)

註冊 (json 格式)

以下是 JSON 的格式：

```
.setRequestHeader("Content-type","application/json");
```

```
<body>
  <script>
    var account = {
      email: 'leon8324@gmail.com',
      password: '0000',
    }
    var xhr = new XMLHttpRequest(); //新增xhr
    xhr.open('post', 'https://hexschool-tutorial.herokuapp.com/api/signup', true); //開啟要傳過去驗證值的網址
    xhr.setRequestHeader("Content-type","application/json");//設定我要傳過去驗證的是甚麼格式
    var data = JSON.stringify(account); //object to string 將以上json物件格式的帳密轉成字串
    xhr.send(data); //傳送的值(相對應內容)
  </script>
</body>
```

DevTools - 127.0.0.1:5501/0102.html

Elements Console Sources Network Performance Memory >>

top Filter Default levels

Live reload enabled. 0102.htm

> xhr

< XMLHttpRequest {onreadystatechange: null, readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, ...}

onreadystatechange: null

readyState: 4

timeout: 0

withCredentials: false

▶ upload: XMLHttpRequestUpload {onloadstart: null, onprogress: null, onabort: null, responseURL: "https://hexschool-tutorial.herokuapp.com/api/signup", status: 200, statusText: "OK", responseType: "", response: '{"success":true,"result":{},"message":"帳號註冊成功"}', responseText: '{"success":true,"result":{},"message":"帳號註冊成功"}'}

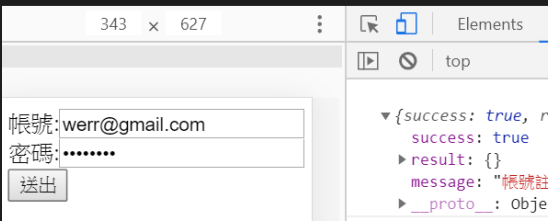
15-131 實務範例設計

```
<body>
  帳號:<input type="text" class="account">
  <br>
  密碼:<input type="password" class="password">
  <br>
  <input type="button" class="send" value="送出">

<script>
  var send = document.querySelector('.send');
  send.addEventListener('click',signup,false);

  function signup(){
    var emailStr = document.querySelector('.account').value;
    var passwordStr = document.querySelector('.password').value;
    var account = {};
    account.email = emailStr;
    account.password = passwordStr;

    var xhr = new XMLHttpRequest();
    xhr.open('post','https://hexschool-tutorial.herokuapp.com/api/signup',true)
    xhr.setRequestHeader("Content-type","application/json");
    var data = JSON.stringify(account);
    xhr.send(data); //此時在console中無法透過xhr查詢response的資料，因為程式碼在函式之中(區域變數)
    //因此在以下在設一onload後會執行的函式，使其console.log出xhr
    xhr.onload = function(){
      var callbackData = JSON.parse(xhr.responseText); //string to array
      console.log(callbackData);
    }
  }
}
</script>
</body>
```




```
<body>
  帳號:<input type="text" class="account">
  <br>
  密碼:<input type="password" class="password">
  <br>
  <input type="button" class="send" value="送出">

<script>
  var send = document.querySelector('.send');
  send.addEventListener('click',signup,false);

  function signup(){
    var emailStr = document.querySelector('.account').value;
    var passwordStr = document.querySelector('.password').value;
    var account = {};
    account.email = emailStr;
    account.password = passwordStr;

    var xhr = new XMLHttpRequest();
    xhr.open('post','https://hexschool-tutorial.herokuapp.com/api/signup',true)
    xhr.setRequestHeader("Content-type","application/json");
    var data = JSON.stringify(account);
    xhr.send(data); //此時在console中無法透過xhr查詢response的資料，因為程式碼在函式之中(區域變數)
    //因此在以下在設一onload後會執行的函式，使其console.log出xhr
    xhr.onload = function(){
      var callbackData = JSON.parse(xhr.responseText); //string to array
      var veriStr = callbackData.message; //取回傳object格式裡的message字串做判斷
      if(veriStr == "帳號註冊成功"){
        alert('帳號註冊成功!!');
      }else{
        alert("帳號註冊失敗!");
      }
    }
  }
}
</script>
</body>
```

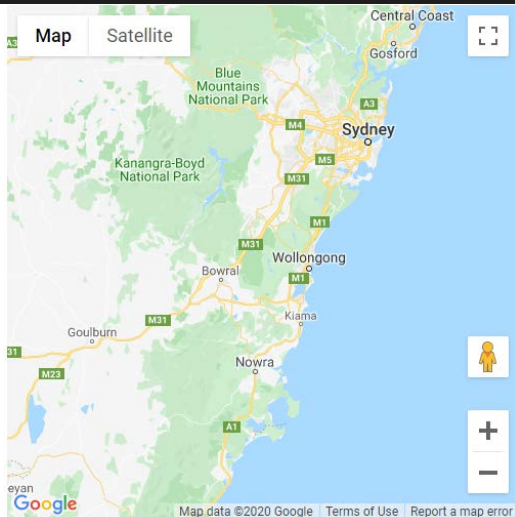


16-137 Google Map API 原理

原理：

1. 先取得 金鑰：(如不取得金鑰將無法顯示地圖，因 google 不知道使用者是誰也不知道是要 for 哪個專案)
2. 當以 google map api 載入完成後，會 callback initMap 函式
3. 編輯 initMap() 此函式

```
<body>
<div id="map"></div>
<script>
  var map; //設一全域變數的map以便在其他地方也使用的到
  // 以下函式將載google api 載入後 透過callback 呼叫出來執行
  // 然後針對map這個變數賦予一個新的地圖：new ，後面接一個google裏頭maps在裏頭Map的一個函式(此函式給予兩個值:指定的div、地圖資訊) ，
  // ==> new 出物件的概念
  function initMap() { //Map( <div id="map"> , {object 資料} )
    map = new google.maps.Map(document.getElementById('map'), {
      center: {lat: -34.397, lng: 150.644}, // 中心點的經緯度
      zoom: 8 // 視角遠鏡
    });
  }
</script>
<!-- 金鑰 AIzaSyDy7niYo3hKVLoxDPCbZqRZWG_x7jml4k & callback initMap這個函示 -->
<script async defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDy7niYo3hKVLoxDPCbZqRZWG_x7jml4k&callback=initMap"></script>
</body>
```

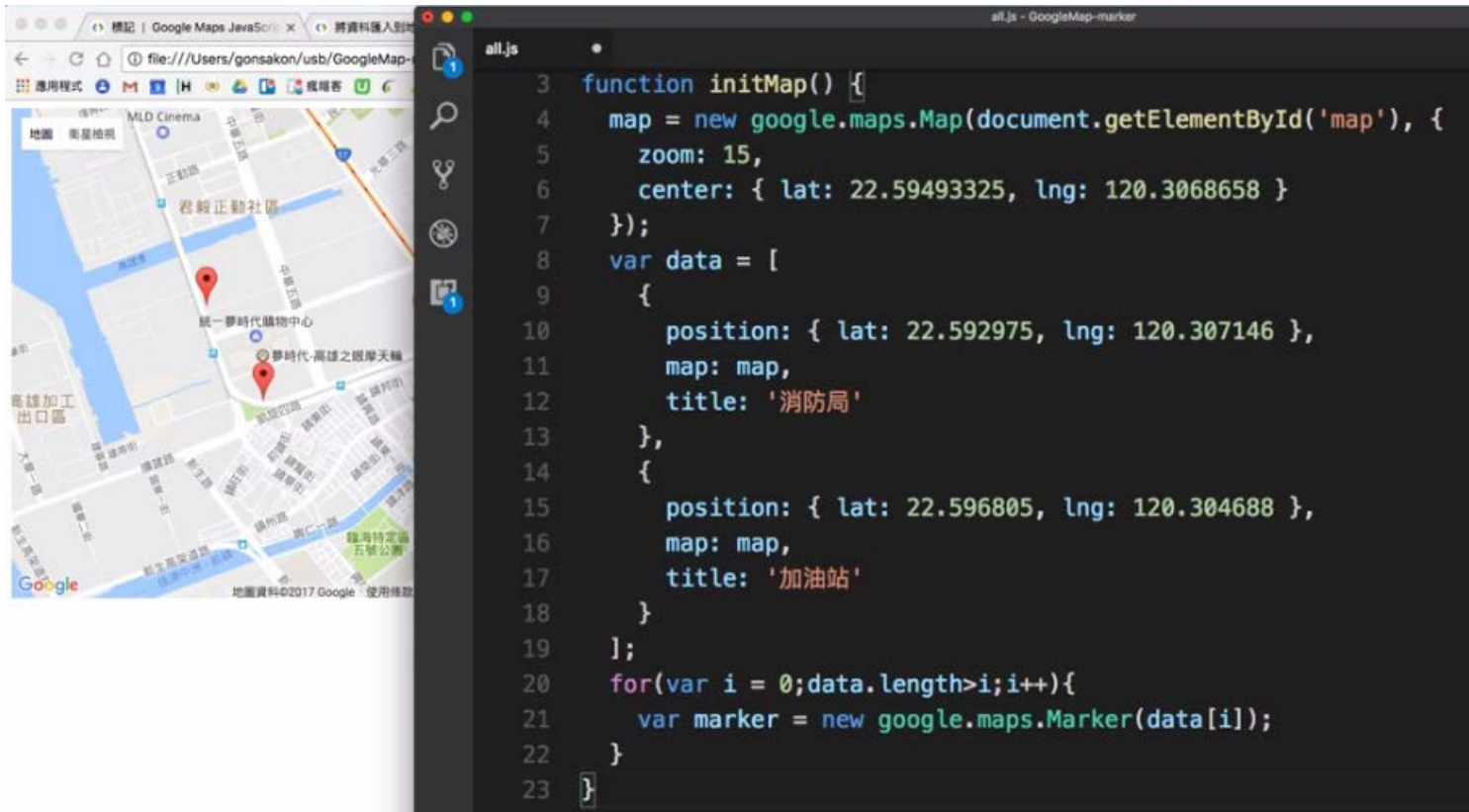


16-138 增加座標

```
<body>
<div id="map"></div>
<script>
  var map;
  function initMap() {
    map = new google.maps.Map(document.getElementById('map'), {
      center: {lat: -34.397, lng: 150.644},
      zoom: 8
    });
    var marker = new google.maps.Marker({ // 設定API裡Marker函數的資料
      position: { lat: 22.592975, lng: 120.307146}, // 指定座標要放在哪
      map: map // 指定座標要放在哪張地圖上
    });
  }
</script>
<script async defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDy7niYo3hKVLoxDPCbZqRZWG_x7jml4k&callback=initMap"></script>
</body>
```


16-139 增加多個標記

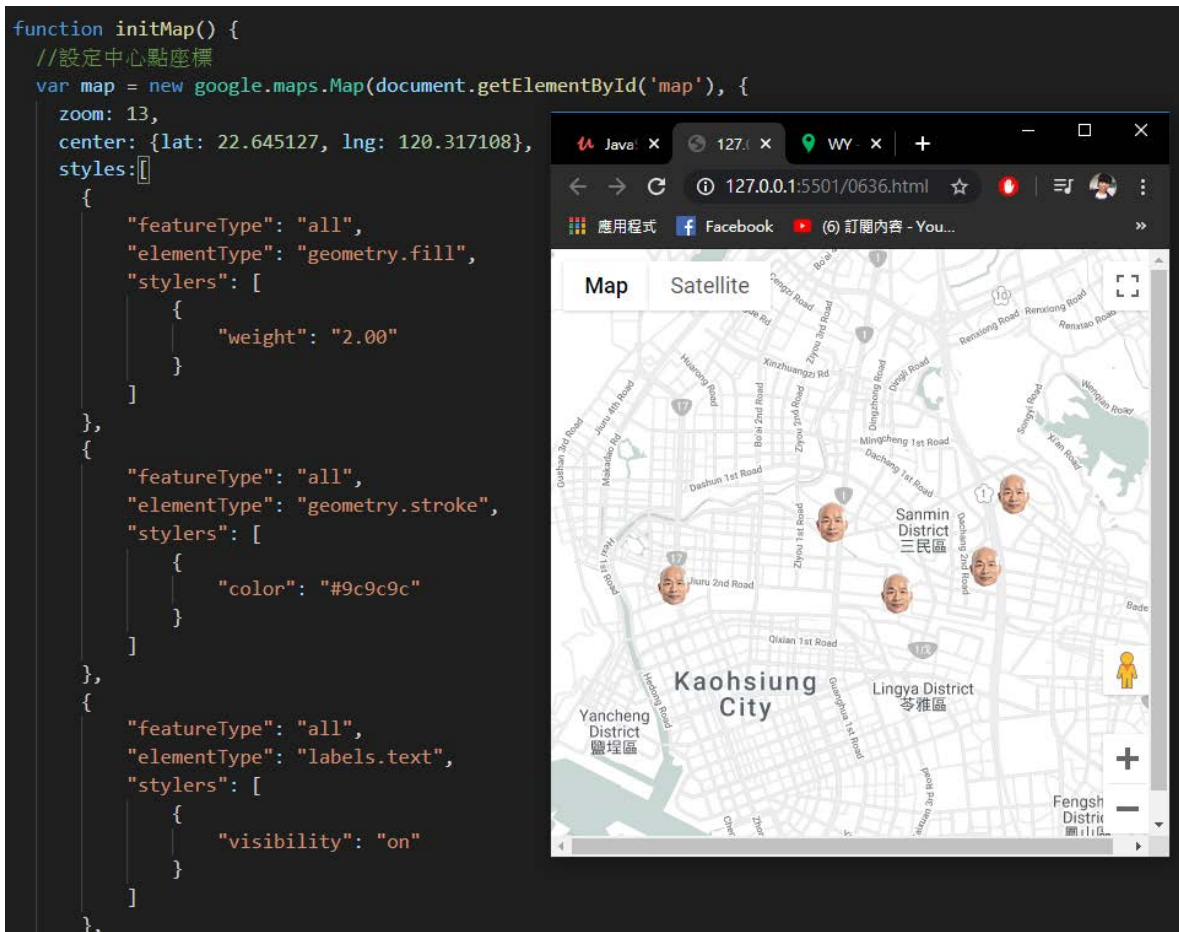
以下這個範例式寫死的 並無使用 JSON 靈活性地帶入多筆資料。



16-141 自製客製化 google map 樣式

可以到 <https://snazzymaps.com/> 選擇大神設計的 map 風格並做套用。

將大神寫好的樣式 (array object) 塞入 物件之中，並給予屬性 “styles”



16-140 標記 + 第三方 JSON 資料

以下範例未用 AJAX 原因是高市政府並未開放 CORS 因此直接用複製 JSON 物件來自己的 JS 裡。

```
01 > JS all.js > [?] data > [?] "result" > [?] "records" > [?] "臨時停車處所"
1  var data={
2      "help": "http://data.kcg.gov.tw/api/3/action/help_show?name=datastore_search",
3      "success": true,
4      "result": {
5          "records": [
6              {
7                  "編號": "1",
8                  "本府負責督導機關": "教育局",
9                  "行政區": "苓雅區",
10                 "備註": "",
11                 "下班後值勤電話": "2235940*136",
12                 "地址": "憲政路233巷2號",
13                 "執行單位": "總務處",
14                 "Lat": "22.6353841",
15                 "可提供小型車停車位": "20",
16                 "_id": 1,
17                 "Lng": "120.3225235",
18                 "臨時停車處所": "高雄啟智學校"
19             },
20             {
21                 "編號": "97",
22                 "本府負責督導機關": "教育局",
23                 "行政區": "三民區",
24                 "備註": "",
25                 "下班後值勤電話": "3839350*18",
26                 "地址": "黃興路206號",
27                 "執行單位": "總務處",
28                 "Lat": "22.6482837",
29                 "可提供小型車停車位": "20",
30                 "_id": 97,
31                 "Lng": "120.3389731",
32                 "臨時停車處所": "東光國小"
33             },
34         ]
35     }
36 }
```

```
80 > function initMap() {
81     //設定中心點座標
82     var map = new google.maps.Map(document.getElementById('map'), {
83         zoom: 13,
84         center: {lat: 22.645127, lng: 120.317108}
85     });
86
87     //跑迴圈依序將值塞入到 marker
88     for(i=0;i<data.result.records.length;i++){
89         var str ={};    // 創空物件來存要塞入的資料
90         var place = {}; // 創空物件來存要塞入的資料
91
92         place.lat = parseFloat(data.result.records[i].Lat);
93         place.lng = parseFloat(data.result.records[i].Lng);
94
95         str.map = map;
96         str.icon='fish1.png'; //自訂 logo
97         str.title= data.result.records[i]['臨時停車處所']
98         str.position = place;
99         new google.maps.Marker(str);
100     }
101 }
```

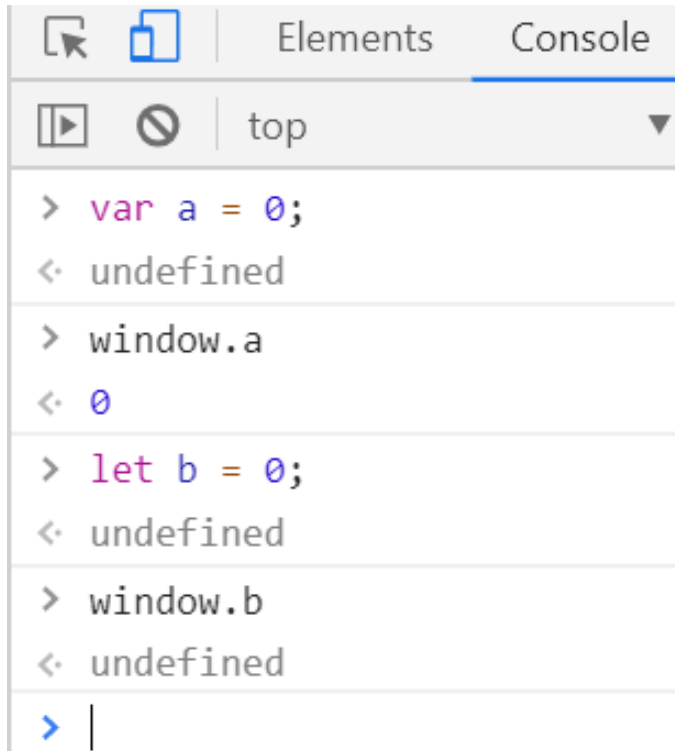

17-145 window、var 特性

let：重新綁定變數

const：唯讀，綁定後不可修改

ES6- let、const 可以避免汙染全域變數 (一般就連 for 迴圈裡的宣告 var=i 都會影響到全域變數)

以下案例：用 var 宣告一變數 a 之後 使用 window 來查詢，發現確實全域變數已遭汙染，但使用 let 宣告 b 則不會，會顯示 undefined

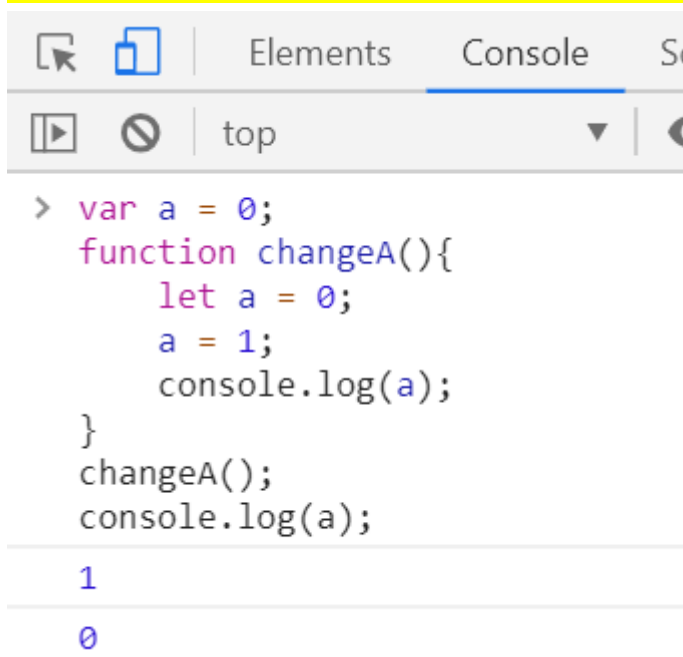


```
> var a = 0;
< undefined
> window.a
< 0
> let b = 0;
< undefined
> window.b
< undefined
> |
```

17-146 let-if、function 用法

let、const 用來宣告區塊裡的變數，區塊“{}”

以下範例：var 與 let 宣告出來的東西互不影響。



```
> var a = 0;
    function changeA(){
      let a = 0;
      a = 1;
      console.log(a);
    }
    changeA();
    console.log(a);

1
0
```

17-147 let-for 用法

由以下範例也可以看到 a 用 var 宣告 會汙染到之後 a 的全域變數 b 使用 let 宣告，則不會，所以在後期呼叫 b 時會找不到 b

```
> for(var a=0 ; a<3 ; a++){
  console.log(a);
}
0
1
2
< undefined
> a
< 3
> for(let b=0 ; b<3 ; b++){
  console.log(b);
}
0
1
2
< undefined
> b
✖ ▶ Uncaught ReferenceError: b is not defined
   at <anonymous>:1:1
```

用 var 宣告變數導致錯誤 - 範例：

此範例因為 for 迴圈裡採 var 宣告 i 導致 因為全域變數遭到汙染，所以不管點擊哪個都會顯示 i 的值 3。

The screenshot shows a code editor on the left and a web browser on the right. The code editor contains the following HTML and JavaScript:

```
html>
ml lang="en">
<head>
</head>
<body>
  <ul class="list">
    <li>0</li>
    <li>1</li>
    <li>2</li>
  </ul>
  <script>
    const listLen = document.querySelectorAll('.list li').length;
    for(var i=0 ; i<listLen ; i++){
      document.querySelectorAll('.list li')[i].addEventListener('click',function(){
        alert(i);
      })
    }
  </script>
</body>
```

The browser window shows the URL 127.0.0.1:5501/0636.html. A list with three items (0, 1, 2) is displayed. An alert dialog box is open, showing the value 3, which is the final value of the variable i after the loop completes. This demonstrates that using var for the loop counter i causes all click events to trigger the alert with the value 3, instead of the value of i at the time the event was triggered.

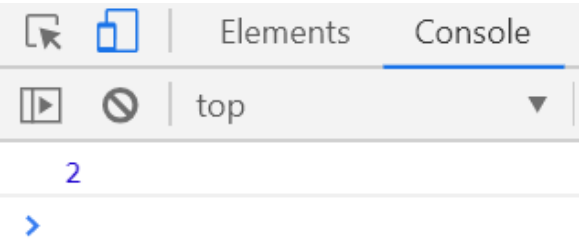
更改成 `let` 後便不會有汙染全域變數的問題了

```
<body>
  <ul class="list">
    <li>0</li>
    <li>1</li>
    <li>2</li>
  </ul>
  <script>
    const listLen = document.querySelectorAll('.list li').length;
    for(let i=0 ; i<listLen ; i++){
      document.querySelectorAll('.list li')[i].addEventListener('click',function (){
        alert(i);
      })
    }
  </script>
</body>
```

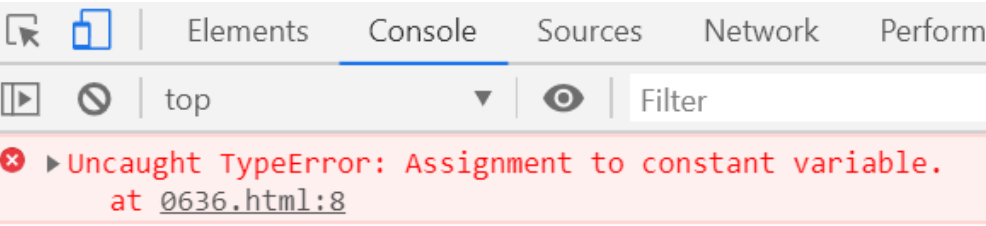
17-148 `const` 特性

`const` (comstant-不變的) 宣告變數，只唯讀，不可修改。用於一些固定的變數 例如:圓周率。
範例：一般 `var` 宣告可以重新給予變數值，`const` 無法。

```
<head>
</head>
<body>
  <script>
    var a = 1;
    a = 2 ;
    console.log(a);
  </script>
</body>
```

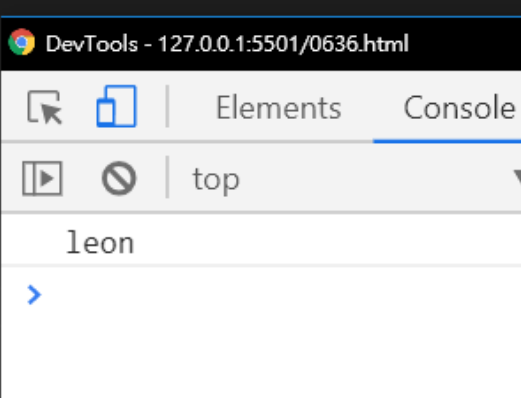


```
<head>
</head>
<body>
  <script>
    const a = 1;
    a = 2 ;
    console.log(a);
  </script>
</body>
```



但如果是 物件、陣列 則可以修改：

```
</head>
<body>
  <script>
    const obj = {
      name: 'jiaren'
    }
    obj.name = 'leon';
    console.log(obj.name);
  </script>
</body>
```



The screenshot shows the Chrome DevTools interface. The left pane displays the HTML code with a script block that creates a constant object 'obj' with a 'name' property set to 'jiaren', then changes it to 'leon' and logs it. The right pane shows the 'Console' tab with a single log entry 'leon'.

如果要解決以上問題，可以使用一語法 凍結裡面的任何內容：**freeze**。

Freeze 好處在於某些東西往想下預設的值，不想再讓其他共同開發者將其物件污染或搞亂

```
<body>
  <script>
    const obj = {
      name: 'jiaren'
    }
    // 凍結 obj 這個物件使其無法被更改
    Object.freeze(obj);

    obj.name = 'leon'; // 嘗試更改 name屬性

    console.log(obj.name);
  </script>
</body>
```

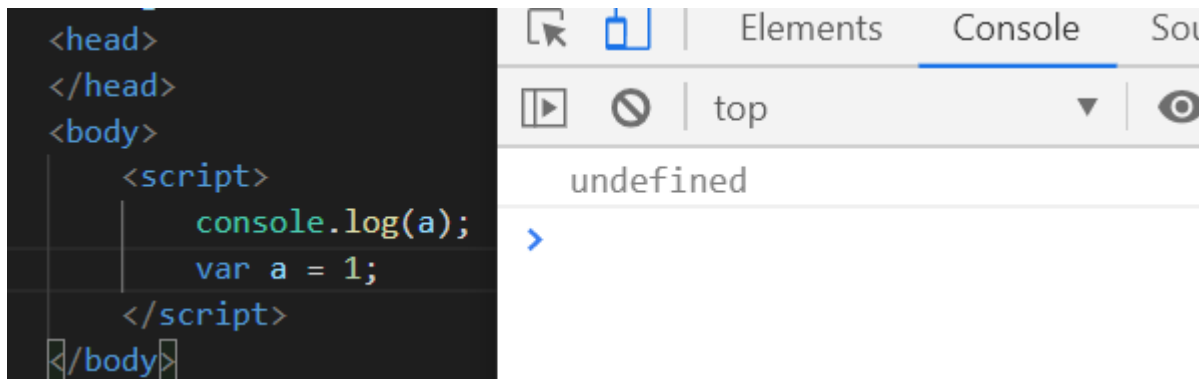


The screenshot shows the Chrome DevTools interface. The left pane displays the HTML code with a script block that creates a constant object 'obj' with a 'name' property set to 'jiaren', then uses 'Object.freeze(obj)' to freeze it. It then attempts to change 'obj.name' to 'leon' and logs it. The right pane shows the 'Console' tab with a single log entry 'jiaren', demonstrating that the frozen object's properties cannot be modified.

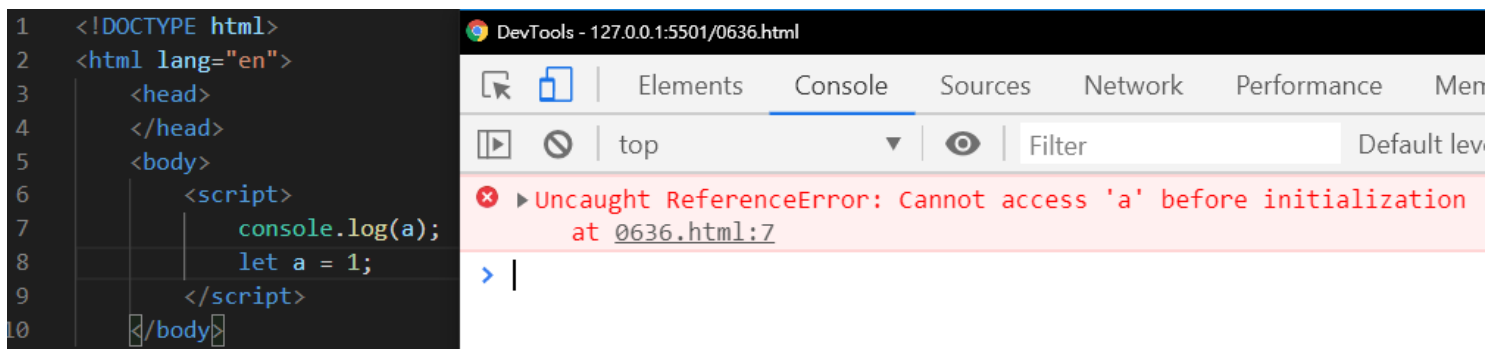
17-149 let、const 注意事項

1. 向上提升：

範例：因為 var 具有向上提升的特性，將後期 var 的宣告提升到最上方，以致 console.log 的結果是 undefined 而非 error，此特性將導致開發者如不小心犯此錯誤，便很難尋找到宣告變數的地方。

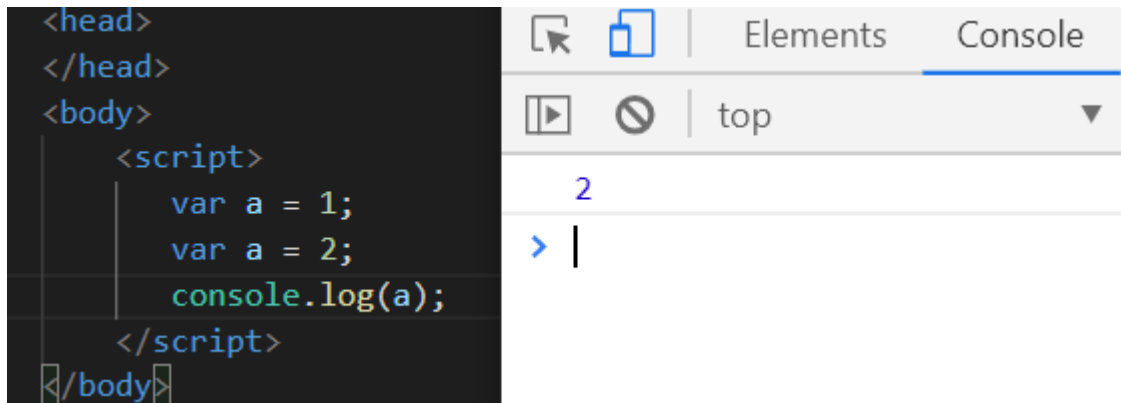


假如換成用 let 宣告則會有所錯問訊息來提示。

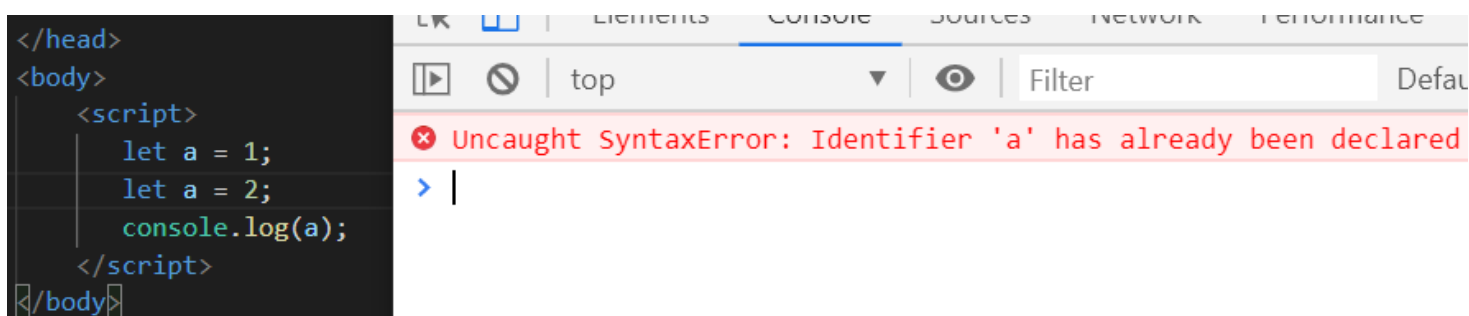


2. 同區塊重複命名

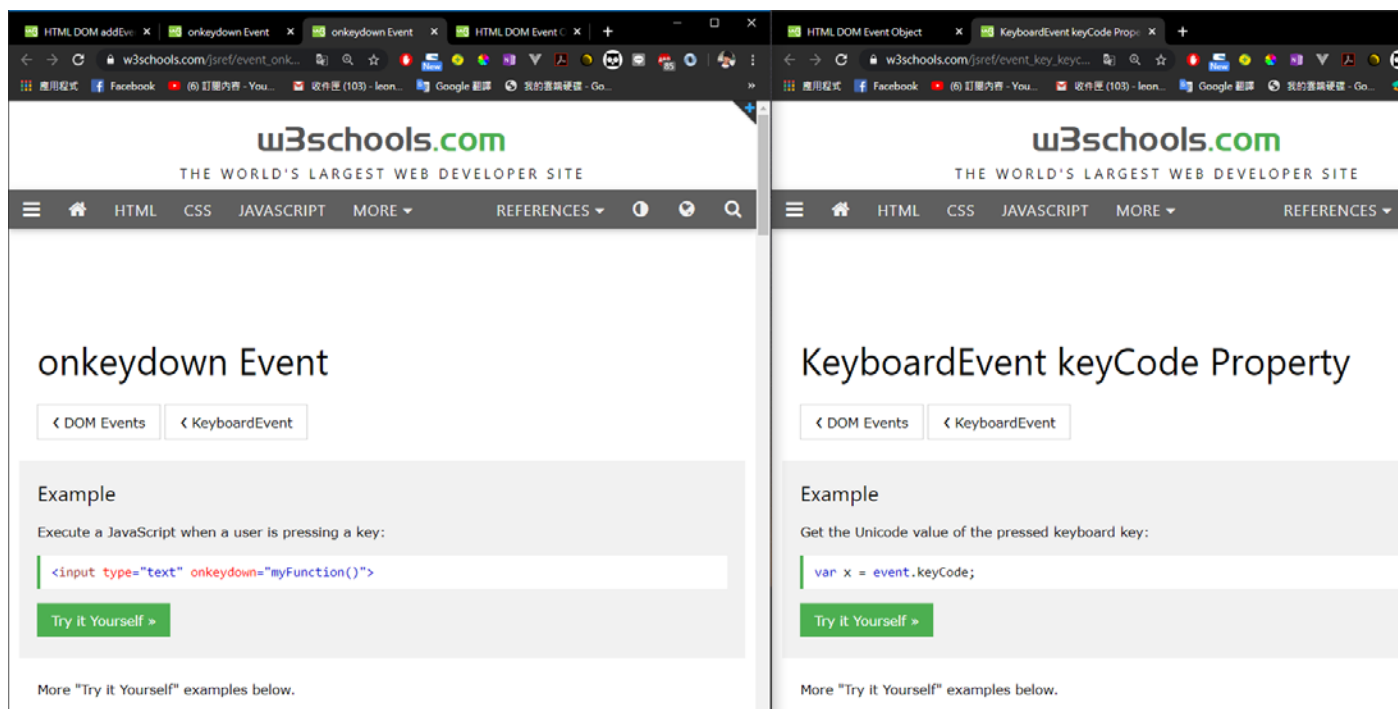
Var：可以重複命名



Let：錯誤訊息



W3c 監聽事件範例



HTML DOM事件

HTML DOM事件允許JavaScript在HTML文檔中的元素上註冊不同的事件處理程序。

事件通常與功能結合使用，並且在事件發生之前（例如，當用戶單擊按鈕時）不會執行該功能。

有關事件的教程，請閱讀我們的 [JavaScript事件教程](#)。

Event	Description	Belongs To
abort	The event occurs when the loading of a media is aborted	UiEvent , Event
afterprint	The event occurs when a page has started printing, or if the print dialogue box has been closed	Event
animationend	The event occurs when a CSS animation has completed	AnimationEvent
animationiteration	The event occurs when a CSS animation is repeated	AnimationEvent
animationstart	The event occurs when a CSS animation has started	AnimationEvent
beforeprint	The event occurs when a page is about to be printed	Event
beforeunload	The event occurs before the document is about to be unloaded	UiEvent , Event
blur	The event occurs when an element loses focus	FocusEvent
canplay	The event occurs when the browser can start playing the media (when it has buffered enough to begin)	Event
canplaythrough	The event occurs when the browser can play through the media without stopping for buffering	Event

addEventListener 的事件

點擊進入會有期可以對應的事件查詢資料