style.css

Index.html

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
/* 第四章 */
var title = “JavaScript”
```

style.css

index.html

{ JavaScript基本介紹 }

一種程式語言，通常做為網頁程式設計之用，可以暫存資料、數學運算、條件判斷、字串處理等等，增加網頁互動性，也是現今網站不可或缺的一部分。主要在客戶端瀏覽器執行，伺服器僅負責提供檔案。優點是不會耗用伺服器資源，但是使用者可以查看完整的程式碼，沒有辦法隱蔽資料。

style.css index.html

{ 基本概念：變數(variable) }

- 變數作為暫存資料之用
- 可以把一個變數想像成一個空白的字卡
- 我們將要儲存的資料寫上字卡，並且可以任意修改內容
- 我們會把每張字卡命名，方便辨識以及取用內容

有些程式語言會將變數分類，限制可以寫入的內容型態

例如：C語言中int只能存入整數，不能存取字串

但JavaScript沒有此限制，型態可以互相轉換儲存

style.css

index.html

{ JavaScript中的變數 }

Step 1 - 建立變數：

語法：`var 變數名稱;` » 範例：`var first;`

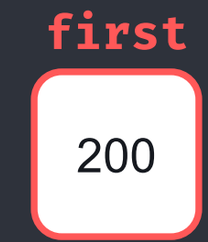
💡 拿出一張空白字卡，並且命名為「 first 」

style.css

index.html

{ JavaScript中的變數 }

Step 2 – 寫入、儲存： `first = 200;`



💡 在剛剛拿出的字卡寫上「 200 」

⚠️ 注意「 = 」符號與平時數學上的意義不同

在程式中稱為「指派」(Assign)，意思是把數值或文字寫入變數

style.css

index.html

{ JavaScript中的變數 }

變數可以互相複製內容

```
var first = 200;
var second = first;
```

style.css

index.html

{ JavaScript中的變數 }

變數可以互相複製內容

```
var first = 200;
var second = first;
```

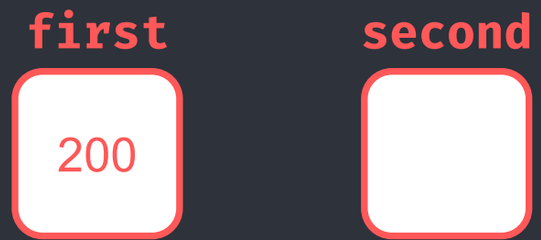
style.css

index.html

{ JavaScript中的變數 }

變數可以互相複製內容

```
var first = 200;
var second = first;
```

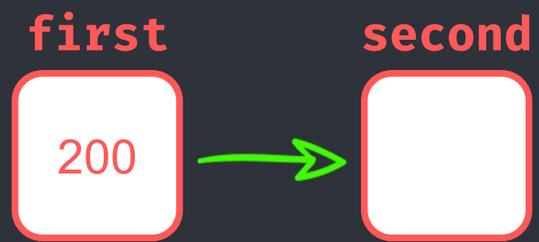


style.css

index.html

{ JavaScript中的變數 }

變數可以互相複製內容




```
var first = 200;
var second = first;
```

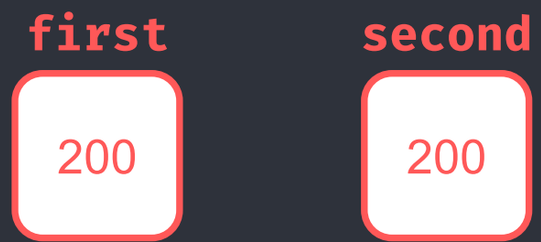
style.css

index.html

{ JavaScript中的變數 }

變數可以互相複製內容

```
var first = 200;
var second = first;
```



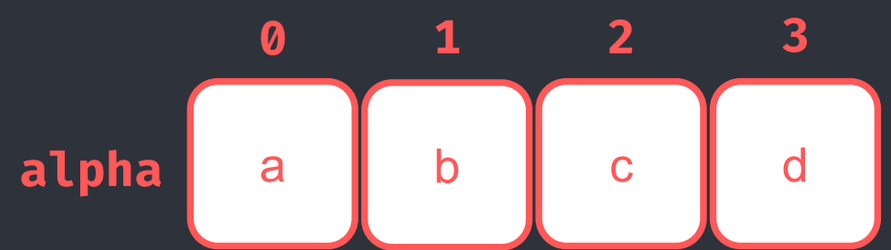
style.css

index.html

{ JavaScript中的變數 }

⭐ **Array**：一組變數的集合，可以想成一排相連的字卡（或格子）

範例：`var alpha = ['a', 'b', 'c', 'd'];`



style.css

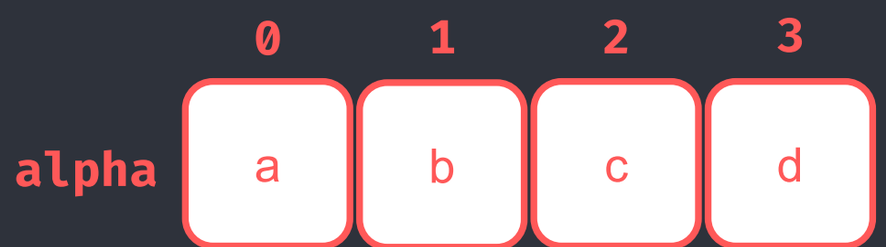
index.html

{ JavaScript中的變數 }

⭐ **Array**：一組變數的集合，可以想成一排相連的字卡（或格子）

💡 字卡從**0**開始依序編號，取用時必須指定想要取用第幾格

想要取用第三格時並更改為‘e’時，寫成：**alpha[3]=‘e’;**



style.css

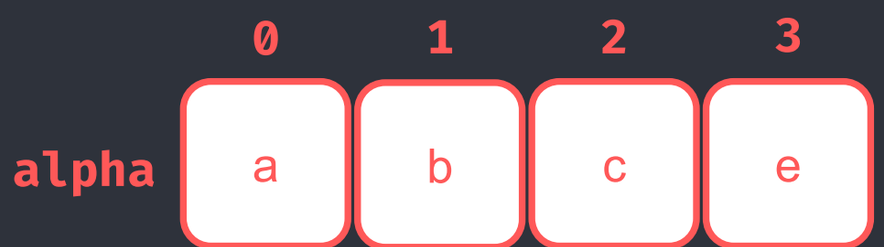
index.html

{ JavaScript中的變數 }

⭐ **Array**：一組變數的集合，可以想成一排相連的字卡（或格子）

💡 字卡從**0**開始依序編號，取用時必須指定想要取用第幾格

想要取用第三格時並更改為‘e’時，寫成：**alpha[3]=‘e’;**



style.css

index.html

{ JavaScript中的變數 }

⭐ **Array**：一組變數的集合，可以想成一排相連的字卡（或格子）

💡 也可以建立空白的**Array**，再依照需求放入字卡

style.css

index.html

{ JavaScript中的變數 }

⭐ **Array**：一組變數的集合，可以想成一排相連的字卡（或格子）

💡 也可以建立空白的**Array**，再依照需求放入字卡

範例：`var alpha = [];`

alpha

style.css

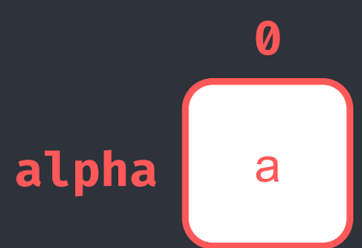
index.html

{ JavaScript中的變數 }

⭐ **Array**：一組變數的集合，可以想成一排相連的字卡（或格子）

💡 也可以建立空白的**Array**，再依照需求放入字卡

```
alpha.push('a');
```



style.css

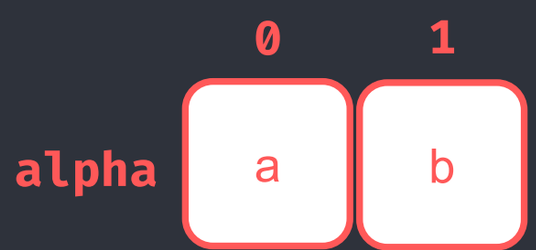
index.html

{ JavaScript中的變數 }

⭐ **Array**：一組變數的集合，可以想成一排相連的字卡（或格子）

💡 也可以建立空白的**Array**，再依照需求放入字卡

```
alpha.push('b');
```



style.css

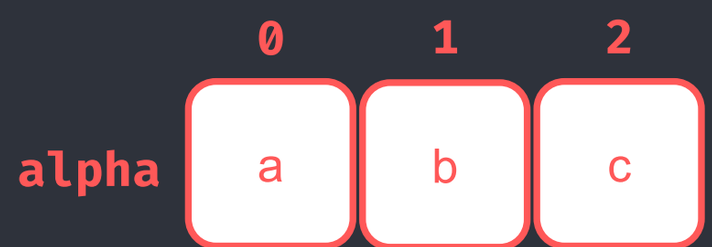
index.html

{ JavaScript中的變數 }

⭐ **Array**：一組變數的集合，可以想成一排相連的字卡（或格子）

💡 也可以建立空白的**Array**，再依照需求放入字卡

```
alpha.push('c');
```



style.css

index.html

{ JavaScript中的變數 }

練習：執行這些指令後，變數arr中的資料是什麼？

```
var arr = [];
var arr2 = ['1','2','3']
arr.push('a');
var a = 'k'
arr[1] = a;
var k = 2;
arr[k] = arr2[2];
arr[arr[2]] = 'd';
```

訣竅：拿紙畫圖模擬執行

提示：注意範例中‘’的使用時機

style.css

index.html

{ 程式基本概念：運算 }

- 「 + 」：數學加法或是字串串接
- 「 * 」：乘法、「 / 」：除法、「 % 」：取餘數
- 「 ++ 」：遞增（每執行一次該數字 +1）
- 「 -- 」：遞減（每執行一次該數字 -1）
- 「 += 」、「 *= 」、「 -= 」、「 /= 」：運算後指派

↳ 「 a = a + 3 」 同等於 「 a += 3 」

↳ 「 b = b / c 」 同等於 「 b /= c 」

style.css

index.html

{ JavaScript字串與非字串 }

⭐ 數字表示

```
var num = 1234
```

💡「 + 」運算

num += num 》 num變成2468

num += 111 》 num變成1345

⭐ 字串表示

```
var st1 = '12'
var st2 = "34"
```

💡「 + 」運算

st1 += st2 》 st1變成1234

st1 += '3' 》 st1變成123

st1 += 12 》 st1變成1212

style.css

index.html

{ JavaScript字串與非字串 }

練習：執行這些指令後，變數a1、a2、a3中的資料是什麼？

```
var a1 = 3;
var a2 = a1--;
++a1;
a1 * a2;
a1 /= a2;
a2 += 'a1'
var a3 = (a1 + 1) % 3;
```

訣竅：拿紙畫圖模擬執行

style.css

index.html

{ JavaScript字串與數字轉換 }

字串轉數字：parseInt()

```
var a = “12”; var b = parseInt(a) + 1; » b為13
```

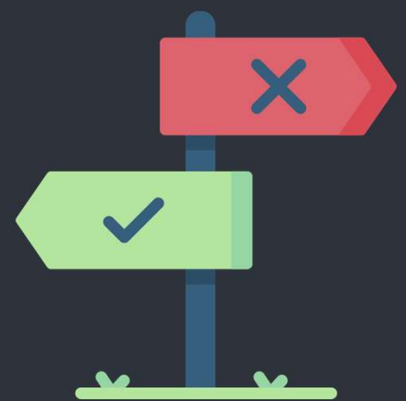
數字轉字串：toString()

```
var a = 12; var b = a.toString() + 1; » b為“121”
```

style.css

index.html

{ 基本概念：條件判斷 }



- 「 == 」、「 != 」：判斷左右是否相等
- 「 > 」、「 < 」：大於、小於
- 「 <= 」、「 >= 」：大於等於、小於等於
- 「 ! 」：**NOT**（反向）
- 「 && 」、「 || 」：**AND**（全部符合）、**OR**（只要一個符合）

style.css index.html

{ 基本概念：條件判斷 }

if else條件判斷

↳ `if( 條件01 ) { 符合條件01的動作 }`

`else if( 條件02 ){ 條件01❌ 條件02✔ 的動作 }`

`else { 其他所有狀況的動作 }`

while迴圈

↳ `while( 條件 ) { 符合條件時會重複執行這裡的指令 }`

style.css index.html

{ 基本概念：條件判斷 }

for迴圈（⭐通常作為計數之用）

for(1 初始動作; 2 條件判斷; 4 後綴)

{ 3 符合條件時會重複執行這裡的指令 }

執行步驟：1 → 2 → 3 → 4 → 2 → 3 → 4

→ 2 → 3 → 4 → 2 → 3 → 4 . . .

直到條件不符合為止

style.css

index.html

{ JavaScript條件判斷 }

練習：執行這些指令後，變數a、b、c、arr中的資料是什麼？

```
var a = 1;
var b = 5;
var c = 10;
var arr = [4,5,6];
while(a < 3)
{
    ++a; b+=a;
}
```

```
if(b <= c && !a < arr[2])
{
    for(var i = 0; i < arr.length; ++i)
    {
        if(arr[i]%2 == 0)
        {
            arr[i] -= c;
        }
        else
        {
            arr[i] += c;
        }
    }
}
```

style.css

index.html

{ 基本概念：函式(function) }

- 函式用來將一個程式切割成多個區塊和功能
- 當執行一個函式時，我們稱之為呼叫(call)
- 可以把函式想像成一台機器，依照我們定義的指令動作
- 我們可以把變數(字卡)傳進函式(機器)，經過運算處理後回傳
- 我們會將機器命名，方便辨識及使用

style.css

index.html

{ JavaScript中的函式 }

⭐ function 函式名稱(傳入變數){函式內容}

```
function start(a, b, c)
{
    var answer = (a+b)*c;
    return answer;
}
```

除了自定義函式之外，有許多內建實用函式，**parseInt**就是一個例子

style.css

index.html

{ 區域變數和全域變數 }

程式中的變數分成 **Local(區域)** 和 **Global(全域)**

- **區域變數**：在函式中宣告，只有該函式可以取用，執行完消失
- **全域變數**：不是在函式中宣告的，任何函式都可以取用

```
var a = 10; //a為全域變數
start(a); //呼叫start函式並傳入a
function start(input) //傳入值input為區域變數
{
    var answer = a * 2; //answer為區域變數
}
```

style.css

index.html

{ 呼叫函式觸發條件 }

我們通常會在網頁設定觸發條件，在事件發生時執行函式

- 常用事件一：**onload**

↳ 當某**html**元素完成讀取時 `<body onload="start()">`

- 常用事件二：**onclick**

↳ 當某**html**元素被按下時 `<div id="box" onclick="plus()"></div>`

style.css

index.html

{ JavaScript函式 }

```
<body onload="start(1, 5, 10)">
```

範例：執行這些指令後，陣列**answer**中的資料是什麼？

```
var answer = [0, 10];
function start(a, b, c)
{
    for(var i = a; i < b; ++i)
    { answer.push( fun1(a, b) ); }
    answer.push( fun2() + c );
}
function fun1(x, y)
{
    while(x < 3){ x *= answer[x] }
    return x + y + fun2();
}
```

```
function fun1(x, y)
{
    while(x < 3){ x *= answer[x] }
    return x + y + fun2();
}
function fun2()
{
    var sum = 0;
    for(var i = 0; i < answer.length; ++i)
    { sum += answer[i]; }
    return sum;
}
```

style.css

index.html

{ 在網頁使用JavaScript }

- **<script></script>**：**JavaScript**程式碼標籤區段
- **<script src=“檔名”></script>**：引入外部檔案

每個標籤只能擇一用途，不可在引入檔案的標籤內加上程式碼