

A dark blue rounded square containing the white text 'JS'.

Javascript Basic

Lecturer: MING ZHEN CAI

Today's goals

認識Javascript

如何編寫Javascript：

資料型態、變數

運算式、陳述

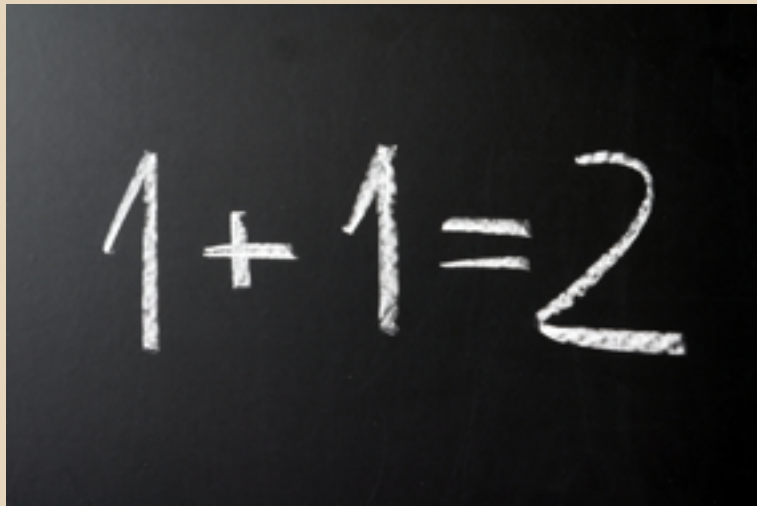
選擇

迴圈

函數

一些小練習

- 寫在script標籤內
`<script type="text/javascript"></script>`
- 寫在*.js檔內，
`<script type="text/javascript" src="路徑與檔名"></script>`
- `<script>` 寫在 `<head>` 或 `<body>` 裡都可以。
瀏覽器一讀到就會立即執行（Load-and-go）。
建議寫在 `</body>` 之前

A black rectangular area representing a chalkboard, with the equation 1+1=2 written in white chalk. The numbers and symbols are hand-drawn and slightly irregular.
$$1+1=2$$

Back to the Basics

程式註解

- HTML : `<!-- 多行註解 -->`
- CSS : `/* 多行註解 */`
- Javascript : `/* 多行 */` 與 `// 單行`

變數的宣告

- var 敘述：建立新的抽屜
- 例如：
var count; //建立名為 count 的空抽屜
var count = 0; //建立 count 並把 0 放進去
- = ：設定 (Assign) 。（不是「等於」喔！）
- 把「=」右邊的值（資料），放進等號左邊的變數（抽屜）。
- 規則：
保留字：Javascript內建具有功能的字彙（reserved word）
大小寫有別、開頭、_符號

常見保留字

break	delete	function	return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	finally	instanceof	throw	while
default	for	new	try	with

等號是「設定」

- `count = count + 1;`
遞增 `count` 值，放回 `count`抽屜中。
上述是將`count`設定為`count`現在的「值」+ 1
- `count = count - 1;`
上述是將`count`設定為`count`現在的「值」- 1

運算子

- 負數：- (如 -35)
執行結果：新數字
- 算數：* / % + - ++ --
執行結果：數字運算結果
- 比大小：=== !== < > <= >=
執行結果：true 或 false

小練習

- `var a = 3;`
`var b = a++; var c= ++a; b,c各為多少 ?`
- `var i = true;`
`var j=!i;`
`var k=!j;`
- `var a = 15;`
`var b = a + 11;`
`var c = b - 20;`
`var d = c * 9;`
`var e = d %10;`
`b,c,d,e各為多少 ?`
- `(5=== "5");`

運算子

- 邏輯運算：&& || !
執行結果：true 或 false
- 三元運算子：A ? B : C
B、C 的執行結果擇一
- 設定：= += -= *= /= %=
執行結果：等號左邊的變數值改變

運算子 練習

試著判斷以下條件結果為true or false

`((500%3>5) && (44/11===8));`

`((9%6>=2) || (26*2%8<7));`

`((99*7/4===0)?28:32)!==104?true:false;`

資料型態

- 抽屜裡可以放不同種類的東西。
變數值 (value) 有數種資料型態 (Data type) 。
- `var count = 0;`
`count` 抽屜裡放數字；
`count` 屬於 `Number` 這個 Javascript 資料型態。

- 常見資料型態
 - Boolean 真假值 (true 或 false)
 - Number 數字 (小數 / 浮點數)
 - String 字串
 - Object 物件。
Javascript 有一些特殊的物件，如陣列 (Array) 以及日期物件 (Date)。
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures

String 字串

- `count = count + 1;` 是給電腦執行的程式碼。
- 給人來讀的「句子」要用引號 `"` 或 `'` 括起來，告訴瀏覽器「這段句子不是程式碼，是給人看的文字」。
- 例如：
`var count = 3;`
`console.log("我買" + count + "個橘子去，你就在此處不要走動。");`
`// 我買3個橘子去，你就在此處不要走動。`

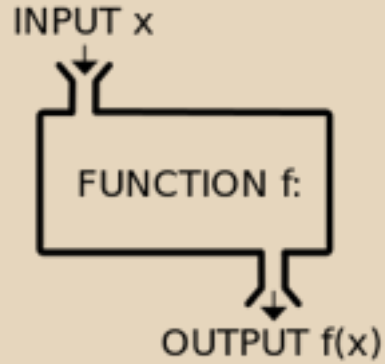
文字與數字

- 數字 + 數字 是加法。
- 任何東西 + 字串、字串 + 任何東西：字串串接
- 轉字串時會嘗試呼叫其 `toString` 方法

型別轉換

- 想將字串 "3" 轉成數字 3，可以使用下面的方式
 - 將文字轉整數：`parseInt(str, base)`
 - 將文字轉浮點數：`parseFloat(str)`
- 對於物件：
 - 撰寫`toString()`方法

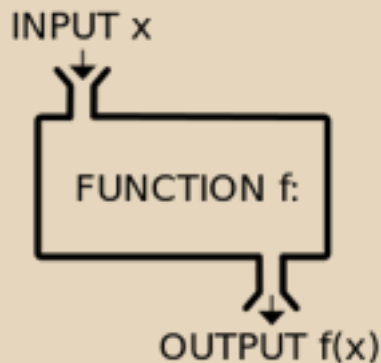
- 若有個字串變數名叫 `str`.....
 - `str[位置]`：取得此位置的字（位置從 0 起算）
 - `str.length`：屬性或長度
 - `str.indexOf(字)`：字在字串的位置（從左邊數來）
 - `str.slice(位置)`：切字串
 - 含前不含後



function

function

- 將重複的程式碼與步驟包裝起來
- function可以代表：1. 數學函式 2. 功能
- 把功能包裝成一個 Function，以後只要呼叫此 Function 就能作一樣的事情。



function的input與output

- input 輸入
用 (a,b,...) 代表有a,b,...等input的值
- Output 輸出
 - return 敘述：把運算結果傳出去。
 - return 後面的值，會變成 functionXXXX(...) 的值。
- 語法
var 函數名 = function(a,b){....
};
- 例如
var add = function(a,b){
 return a+b;
};

小練習

已知圓周率pi為3.14，請寫出一個function可以輸入半徑r，快速計算出圓的面積 area:

```
var pi = 3.14;  
function area(){  
    ???  
}
```



Conditions

if...else (如果...對的話就...,不然就...)

```
if (條件){
```

條件是true的話，要執行的程式碼寫這邊

```
console.log("條件對了");
```

```
}
```

```
else{
```

條件是false的話，要執行的程式碼寫這邊

```
console.log("條件不成立");
```

```
}
```

程式碼如果只有一行，可以不寫{ }

if...else if else if else

if(條件1)

條件1是true的話，要執行的程式碼寫這邊

console.log("條件1對了");

else if(條件2)

條件2是true的話，要執行的程式碼寫這邊

console.log("條件2對了");

else if(條件3)

條件3是true的話，要執行的程式碼寫這邊

console.log("條件3對了");

else

都不對，所以就...

某生想要設計一個將成績轉換成等地的程式，
對照如下表，請幫幫他！

成績範圍	等地
$\text{score} \geq 80$	A
$80 > \text{score} \geq 70$	B
$70 > \text{score} \geq 60$	C
$\text{score} < 60$	D

switch case

//把上面的條件變成情況來列舉的一種選擇方式

```
switch (情境變數) {
```

```
  case 情境1:
```

```
    //是情境1要做的事情
```

```
    break; //做完離開
```

```
  case 情境2:
```

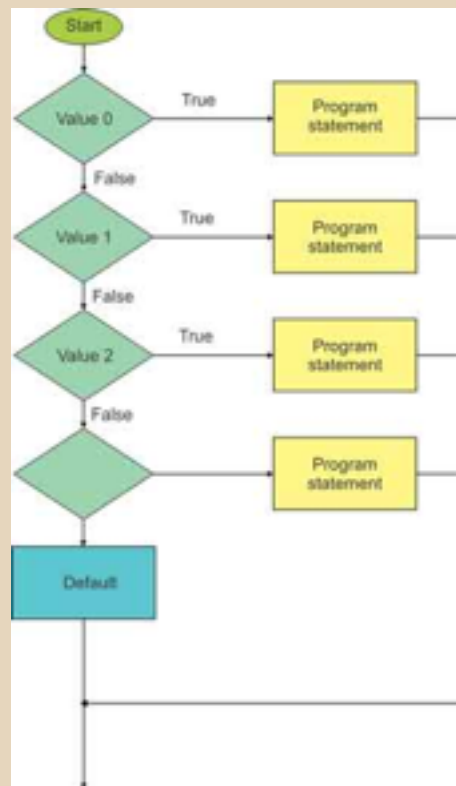
```
    //是情境2要做的事情
```

```
    break;
```

```
  default: //(預設情境) 預設情境要做的事情
```

```
    break;
```

```
}
```



小練習

```
<script>  
abc = 1;  
switch(abc){  
    case 0: abc = 3;  
    case 1: abc--;  
    case 2: abc += 2;  
} </script>
```

請問上述程式碼執行後，abc 的值為何？

Example

```
switch (fruittype) {  
  case "橘子":  
    document.write("橘子一公斤 $50元 < br >");  
    break;  
  case "蘋果":  
    document.write("蘋果一公斤 $100元 < br >");  
    break;  
  case "香蕉":  
  case "鳳梨":  
    document.write("香蕉跟鳳梨大特賣 < br >");  
    break;  
  default:  
    document.write("不好意思， 沒有你要的 " + fruittype + ". < br >");  
}
```

表示 香蕉和鳳梨是一組
沒有
case "香蕉", "鳳梨": 這種寫法

for example

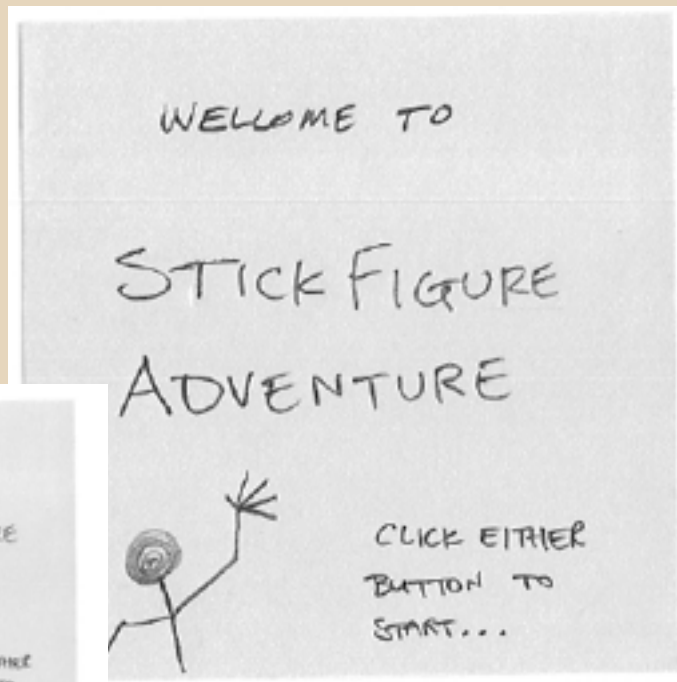




Practice

火柴人大冒險

- 火柴人即將展開他的旅程
- 在旅程的路上，總會有一個命運的路口，火柴人必須選向左走或者向右走
- 畫面中將有兩個按鈕讓玩家點選向左或向右走



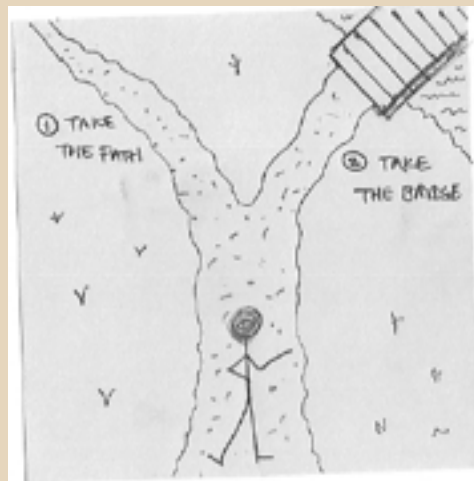
遊戲路徑

場景0
介紹一下這個冒險故事



場景1：火柴人上路囉

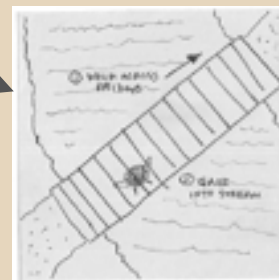
1. 走過小徑
2. 走過小橋



場景2：林間小屋



場景3：在橋上展望



場景2：林間小屋

1. 繞過小屋
2. 跟女巫打招呼



①

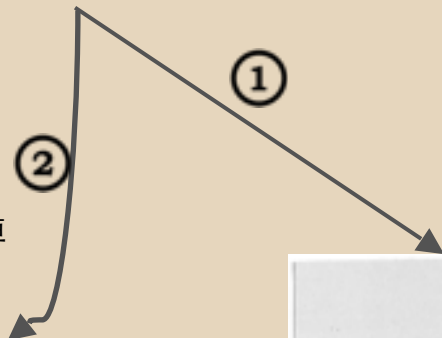


②

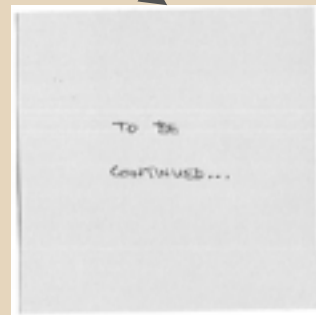


場景4：女巫站在窗邊

1. 悄悄的走過去
2. 跟女巫打招呼



場景5：被女巫吃掉 遊戲結束



場景6：被巨人吃掉 遊戲結束

場景3：在橋上展望

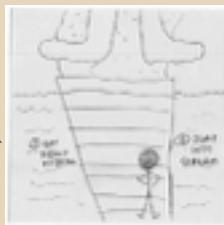
1. 走過小橋
2. 看看小溪



①

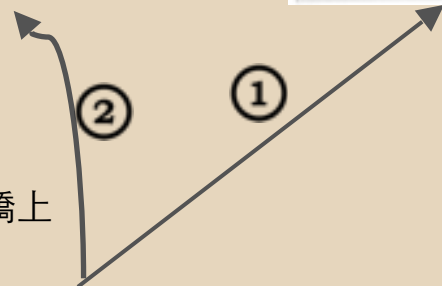


②



場景7：巨人來到橋上

1. 跟巨人打聲招呼
2. 快逃！



Tips

- 要知道自己走到哪了，需要一個變數儲存現在在哪
- 透過if else或者switch case選擇下一個場景是什麼

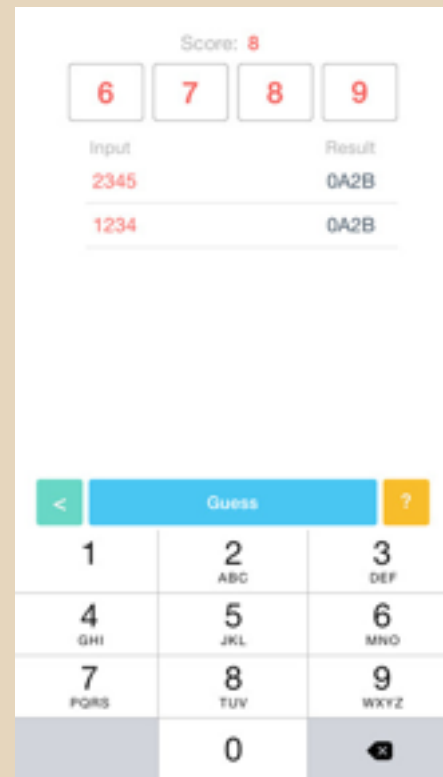
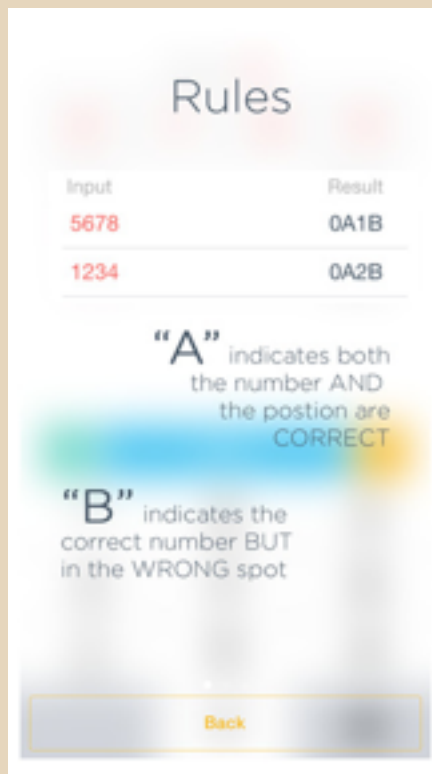


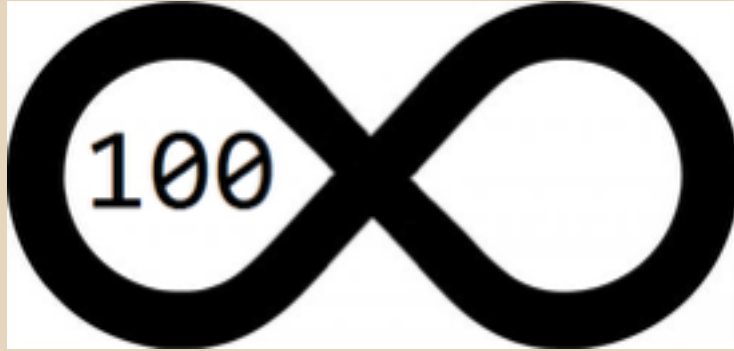


Homework

Number Guessing Game I

1. 使用JS寫出一個遊戲如右圖的APP
2. 程式隨機選出不重複四個數字
例如：8579
3. 使用者猜：
1258：0A2B
(兩個數字對，位置錯)
4. 繼續猜
3459：1A0B
(一個數字對且位置也對)





Loop

for loop

- 重複做一些事情，從編號(索引)=幾號開始，編號的範圍，如何叫號
例如 編號的變數是index，從10號開始，最多到50號，每次加1號
- 變成語法：

```
for(var index = 10; index <=50; index++){
```

數到號碼的時候，要做的事情
譬如 吃飯

```
}
```

for loop

- 正式的語法：

```
for(var 號碼的變數= 起始值; 號碼範圍; 叫號方式){  
    需要執行的程式碼，想重複做的事情  
}
```


Example: 把數字加總

- 將3,5,7,9,.....,103加起來怎麼加？
- 剛剛的叫號規則：
號碼從3號開始，上限到103，每次跳2號
- `for(var number = 3; number <=103; number+=2)`
- 怎麼加呢？
回到小學，拿兩個數先加起來，順便記好，用記好的數字加上下一個，加到結束

Example: 把數字加總

- 用一個變數充當大腦，存加好的數字
`var total = 0;`
//宣告，並指定一開始數字是0（因為還沒加阿）
- `for(var number = 3; number <=103; number+=2)`
{
 `total = total + number;`
 總和 = 之前的總和 + 現在叫到的號碼
 然後設定給總和;
}

while loop

- 當某個條件成立的時候，就重複做一些事情，直到條件不成立為止。
- 注意：如果條件永遠不會不成立，則會形成無窮迴圈，程式跑到天荒地老直到電腦資源被耗盡
- 例如：

```
while(沒中樂透且年紀小於65){  
    我就上班，  
    並且買樂透，  
    年紀++;  
}
```

Example: 加到多少超過1000

- 可怕的高中數學：
1, 3, 5, 7, 9, 11,....加起來，要加到第幾個才會超過1000
請將那時候的個數跟總和找出來



break

- 想在某種情況中，跳出迴圈，就加上break
 - 例如：剛剛的加法不想加到103，只想要到98
- ```
for(var number = 3; number <=103; number+=2)
{
 total = total + number;
 if(total == 98) break;
}
```

## continue

- 想在某個情況下，跳過一些迴圈內的程式碼，直接到下一步，可以用continue
- 例如：一樣的加法題，加到103，但是不想包含99

```
for(var number = 3; number <=103; number+=2)
{
 if(number == 99) continue;
 total = total + number;
}
```



# Homework

## Number Guessing Game II

1. 使用JS自動找出這個遊戲的答案
2. 先用紙筆找出你自己猜答案的邏輯，並且將此寫成程式
3. 試著平均10次內找到答案



Score: 8

|   |   |   |   |
|---|---|---|---|
| 6 | 7 | 8 | 9 |
|---|---|---|---|

|       |        |
|-------|--------|
| Input | Result |
| 2345  | 0A2B   |
| 1234  | 0A2B   |

< Guess ?

|           |          |           |
|-----------|----------|-----------|
| 1         | 2<br>ABC | 3<br>DEF  |
| 4<br>GHI  | 5<br>JKL | 6<br>MNO  |
| 7<br>PQRS | 8<br>TUV | 9<br>WXYZ |
|           | 0        | ↩         |





# Array

## 陣列

- 陣列就像一個表格，把資料有組織的放在一起
- 陣列的宣告與初始化：

```
var colors = ["Red", "Green", "Blue"];
```

```
var colors = new Array("Red", "Green", "Blue");
```

```
var colors = Array("Red", "Green", "Blue");
```

- length屬性為陣列的長度：

colors變數有3個元素，因此可以透過.length取得長度

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

- 陣列的元素使用元素的索引來存取。

陣列是以 0 開始索引，因此陣列中的第一個元素是 0：

```
var red = colors[0];
```

## 將資料加入陣列

- javascript中的陣列是動態大小的：  
var colors = []; // 空陣列  
console.log(colors.length); // 0  
colors[0] = 'Red';  
colors[1] = 'Green';  
colors[2] = 'Blue';  
console.log(colors.length); // 3
- 常見的把資料丟到陣列最後一個的作法是：  
colors[colors.length] = 'Orange';

## 陣列的 push 與 pop

- 可以使用 `array.push()` 方法加元素到陣列尾端  
`colors.push('Purple');`
- 想拿出陣列中的最後一個元素，並把那個東西從陣列中刪除可以使用 `pop()`  
`var lastElement = colors.pop();`  
`/* colors 的最後一個元素也被移除 */`

走訪陣列所有元素

```
var colors = ['red', 'green', 'blue'];
for (var i = 0; i < colors.length; i++) {
 alert(colors[i]);
}
```

DOM元素的走訪

```
var divs = document.getElementsByTagName('div');
for (var i = 0, div; div = divs[i]; i++) {
 /* 以同樣方式處理 div */
}
```

## 陣列的基本方法

- `concat` 合併兩個陣列，並把新的陣列返回。  
`var a1 = [1, 2, 3];`  
`var a2 = a1.concat(['a', 'b', 'c']);`  
`print(a2); // 1,2,3,a,b,c`
- `reverse` 反轉陣列元素的順序至適當的位置。  
`var a = [1, 2, 3, 4];`  
`a.reverse();`  
`print(a); // 4,3,2,1`

## 陣列的基本方法

- shift 移除並返回陣列的第一個元素。

```
var a = [1, 2, 3];
```

```
var first = a.shift();
```

```
print(a); // 2,3
```

```
print(first);
```

## 陣列的排序

- `sort` 在適當的位置排序陣列的元素。  
`var a = ['Wind', 'Rain', 'Fire'];`  
`a.sort();`  
`print(a); // Fire,Rain,Wind`
- `sort` 也可以自定排序方法，只要你給一個函數會對兩個值做比較，並返回下列三個值其中之一：  
如果 `a` 在排序系統中小於 `b`，返回 `-1`（或任意的負數）  
如果 `a` 在排序系統中大於 `b`，返回 `1`（或任意的正數）  
如果 `a` 和 `b` 被認為是相等的，返回 `0`。



## 陣列的自訂排序

```
var a = ['Wind', 'Rain', 'Fire'];
function sortFn(a, b) {
 var lastA = a[a.length - 1];
 var lastB = b[b.length - 1];
 if (lastA < lastB) return -1;
 if (lastA > lastB) return 1;
 if (lastA == lastB) return 0;
}
a.sort(sortFn);
print(a); // Wind,Fire,Rain
```



**Practice**

## Online booking I

- 有兩位壯漢想要來訂個電影票，但他們比較大隻一點，所以想要一次訂三張票，而且座位要連在一起
- 寫一個系統，自動找出電影院中三個連在一起的座位，如果有座位，請讓他們決定是否要這個位置，要的話，將其更換顏色，並且結束程式



## Tips

- 使用一個陣列來儲存現在座位的狀態
- 使用迴圈找尋有三連號的座位
- 使用`document.getElementById("seat" + col).src = "xxx.png";`更換圖片





# Homework

## Online booking II

1. 將練習中的自動選位改為點選圖片後選位。
2. 可選擇人數例如兩人，預設點一個座位後，若可以就自動選兩個，無法坐一起時，自動隨機找另一個可以坐的位置

