

Rails Girls Weekly

# JavaScript Basic

**Week 3**

2015-10-06 @ 五倍紅寶石出礦坑

Speaker: Win Wu

# Agenda

- **Function** 函式
- **Event** 事件

# Function 函式

一段執行特定功能的程式碼集合起來。

簡化, 重複利用, 回收再利用!

# Function 函式

函式分為指名函式及匿名函式

函式的宣告方式有兩種：

- ✖ 直接對函式命名。
- ✖ 將匿名函式指定給一個變數。

# Function 指名函式

指名函式代表這個函式有一個名字。  
可以在宣告之前呼叫，也可在宣告之後呼叫。

```
function 函式名稱 () {  
  // 函式要做的事情  
}
```

01.js

# Function 指名函式

```
// 指名函式
```

```
// 在宣告之前呼叫  
sayHello();
```

```
function sayHello()  
{  
    // do something  
    console.info('welcome!');  
}
```

```
// 也可在宣告之後呼叫  
sayHello();
```

# Function 匿名函式指定給變數

另外你也可以把匿名函式指定給一個變數，  
函式也可以當作一個變數儲存。

```
var 變數名稱 = function () {  
    // do something...  
}  
  
// 呼叫時要記得 ()，否則不會執行  
變數名稱();
```

# Function 匿名函式指定給變數

02.js

```
// 匿名函式指定給變數  
// Uncaught TypeError: sayHello is not a function  
// sayHello();
```

```
var sayHello = function () {  
  console.log('hihi!');  
}
```

```
sayHello();
```



# 注意事項

- JavaScript 大小寫有別。關鍵詞 **function** 必須是**小寫**
- 使用時必須以與**函數名稱**相同的大小寫來調用函數

# 參數 Argument

在調用函式時，您可以向函式傳遞值，這些值被稱為參數。

```
function funName(arg1) {  
  console.log(arg1);  
}
```

函式可以帶入多個參數。

```
function funName(arg1, arg2, arg3) {  
  console.log(arg1);  
  console.log(arg2);  
  console.log(arg3);  
}
```

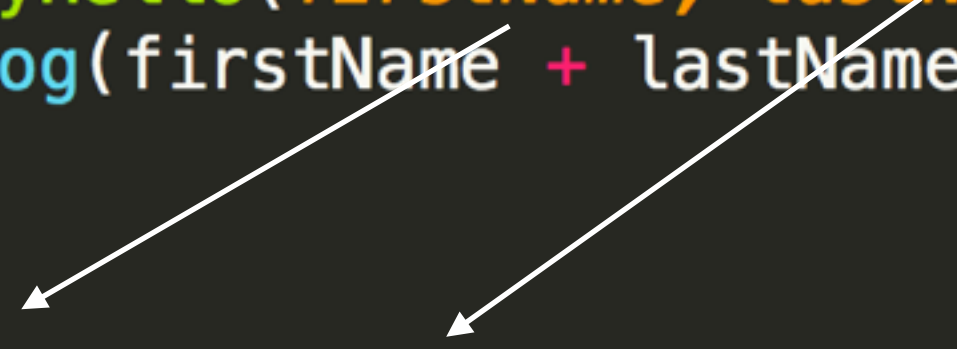
# 參數 Argument

帶入參數到函式時，帶入參數的順序必須一致。

03.js

```
function sayHello(firstName, lastName) {  
  console.log(firstName + lastName + '你好!');  
}
```

```
sayHello('wu', 'yiying');
```

Two white arrows originate from the function call. The first arrow starts at the string 'wu' and points to the parameter 'firstName' in the function definition. The second arrow starts at the string 'yiying' and points to the parameter 'lastName' in the function definition.

# 參數 Argument

04, 05.js

```
function sayHello(firstName, lastName) {  
  if (typeof(lastName) == 'undefined')  
  {  
    // 如果沒有定義 lastName 的值  
    // 那我們就給他空值來當作預設值  
    lastName = '';  
  }  
  console.log(firstName + lastName + '你好!');  
}
```

```
// 本來應該要傳入 lastName 而沒有傳入 lastName  
// 你會得到 undefined，但是因為我們有判斷 lastName 是不是 undefined  
// 所以會得到正常的結果  
sayHello('wu');
```

# return 返回一個回傳值

有時候我們希望函式做的事情，  
可以把結果回傳給呼叫它的地方。

06.js

```
function addTwoNumber(x, y) {  
  return x + y;  
}
```

```
var result = addTwoNumber(1,2);  
console.log(result);
```

# return 返回一個回傳值

```
function addTwoNumber(x, y) {  
    return x + y;  
}
```

```
var result = addTwoNumber(1,2);  
console.log(result);
```



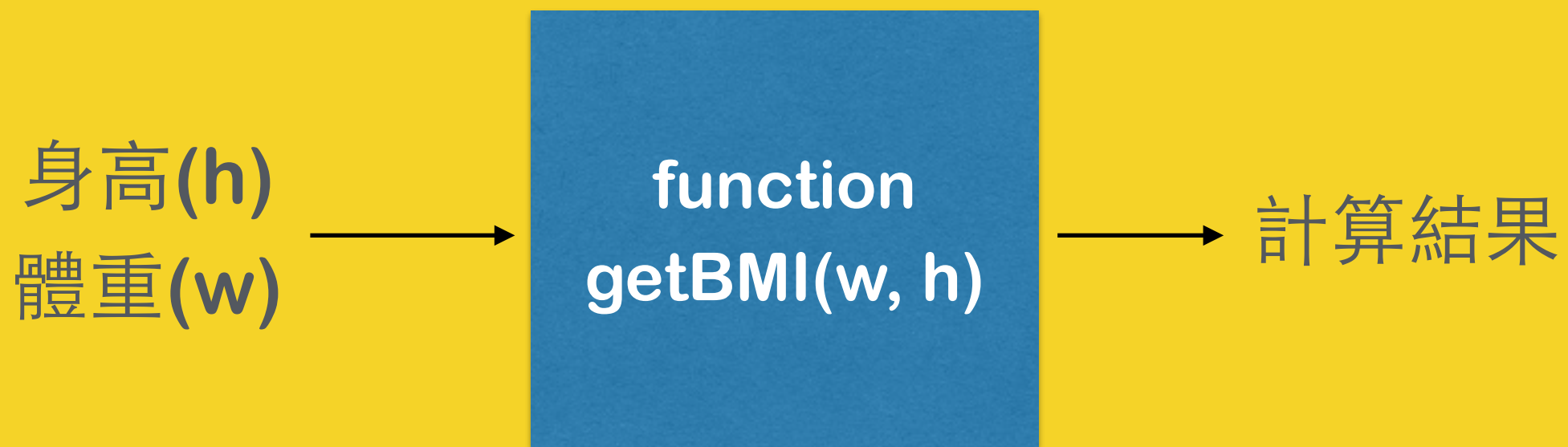
# Function 函式

函式就像是黑盒子一樣，  
資訊可以流入(參數)，可以流出(**return**)



# Function 函式

假如我們有個計算身體質量指數(BMI)的 **function**  
使用 **function** 的人可以不用知道到底該怎麼計算 BMI  
只需要提供 **身高** 以及 **體重** 就可以得到結果。



```
function getBMI(weight, height) {  
  // BMI = 體重(公斤) / 身高2(公尺2)  
  return weight/(height*height);  
}
```



# 注意事項

- 函數在執行過 return 後相當於立即停止後面的程式，因此，return 語句後的代碼都不會被執行。
- 如果函數無返回值，那麼它真正返回的值是undefined。  
(return is not necessary. But no return actually return the undefined)
- 參數的數量並沒有實際上的限制，但還是盡量控制在 2-3 個左右。

# Event 事件

只要你跟瀏覽器有互動，就有事件。  
你只是需要知道有哪些事件可以監聽。

事件代表一種狀態，我們可以知道事件的狀態  
做出一些相對應的反應。

事件通常與函式結合使用，  
而這些函式不會在事件發生前被執行。

# Event 事件 - Event Handler

以下是一些常見的事件：

| 事件名稱               | 描述                     |
|--------------------|------------------------|
| <b>onchange</b>    | 內容被改變時。example: select |
| <b>onclick</b>     | 內容被點擊時                 |
| <b>onmouseover</b> | 滑鼠滑到某個物件上              |
| <b>onresize</b>    | 視窗重新被調整大小時             |
| <b>onsubmit</b>    | 表單被送出時                 |
| <b>onerror</b>     | 載入圖片或是文件錯誤時            |
| <b>onfocus</b>     | 元素獲得焦點的時候              |
| <b>onkeydown</b>   | 某個按鍵被按下的時候             |

# Event 事件 - 監聽

監聽事件有幾個方法：

(1) **HTML** 行內屬性（請避免使用）

(2) 對 **DOM** 屬性綁定

(3) 使用事件監聽函數 (**addEventListener**)

# Event 事件 - 對 DOM 屬性綁定

event1.html

監聽 **element** 節點的 **click** 事件。

此範例是一個針對一個超連結，做 **onclick** 的 **function**。

```
<a id="link" href="http://google.com" alt="這是一個測試連結">測試連結</a>

<script>
  var link = document.getElementById('link');
  link.onclick = function () {
    console.log(this.href); // 顯示 "http://google.com"
  }
</script>
```

備註：句點 (.) 後面接的是某個東西的屬性或方法

# Document 是什麼？

上一頁指的 **Document** 是 **Document Object Model** 物件檔案模型，提供修改或是取用網頁資料的一種機制。

**DOM** 把網頁視為節點構成的階層樹。

**Dom** 的節點，是依據節點的類型做分類。

# Event 事件 - 對 DOM 屬性綁定

- 上頁範例，**onclick** 表示 **HTML** 元素的按下事件。
- 可以接受一個匿名函式當做動作。
- 當按下按鍵時，就會觸發這個動作。
- 重複指定匿名函式的話，新的會蓋掉舊的，所以只會執行最新的。
- **onclick** 裡的匿名函式中的 **this** 代表 **onclick** 所屬的 **HTML** 元素。

# Event 事件 - addEventListener

可以對事件新增一個監聽的處理函式  
跟 **onXXXX** 不同的地方在於可以對同一事件註冊多個處理函式





# Event 事件 - addEventListener

event2.html

```
<a id="link" href="http://google.com" alt="這是一個測試連結">測試連結</a>
```

```
<script>
```

```
var link = document.querySelector('#link');
```

```
// 監聽 link 的 on click 事件
```

```
link.addEventListener('click', function () {  
    console.log(this.href); // 顯示 "http://google.com"  
});
```

```
// 監聽 link 的 mouseover 事件
```

```
link.addEventListener('mouseover', function() {  
    console.log('over');  
});
```

```
// 監聽 link 的 mouseout 事件
```

```
link.addEventListener('mouseout', function() {  
    console.log('out');  
});
```

```
</script>
```

# Event 事件 - removeEventListener

可以 **add** 當然也有 **remove** 嘍！  
移除已經被 **add** 某個物件上的事件。

```
document.removeEventListener(event, function)
```

解除事件綁定時要注意，使用的 **function**  
不能是匿名函數，必須要是指名函數。

## event3.html

```
<button id="testBtn">測試按鈕</button>
<script>
  var myBtn = document.getElementById('testBtn');

  var onlyOnce = function(){
    alert('這個按鈕只點一次，就被我解除 click 的綁定');
    myBtn.removeEventListener('click', onlyOnce);
  };

  myBtn.addEventListener('click', onlyOnce);
</script>
```

# Reference

- <https://github.com/jaceju/notes-javascript/blob/master/basic.md>
- [http://www.w3schools.com/js/js\\_htmlDOM\\_eventlistener.asp](http://www.w3schools.com/js/js_htmlDOM_eventlistener.asp)
- <http://blog.winwu.today/2013/08/eventpreventdefaultreturn-false.html>
- <http://yujiangshui.com/javascript-event/>
- <http://www.wibibi.com/info.php?tid=384>

**Thank you :D**