

# JavaScript & jQuery

## 前端開發入門實戰

Kuro Hsu @ 5xRuby

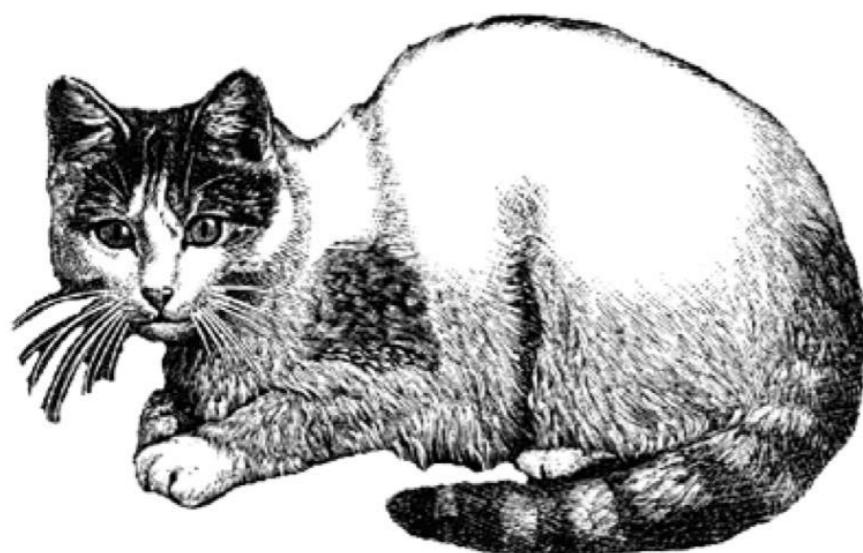


# 前端？

*O'Reilly Generator*

# Javascript 錦囊妙計 從入門到放棄

不斷入門，又不斷的放棄，永不放棄



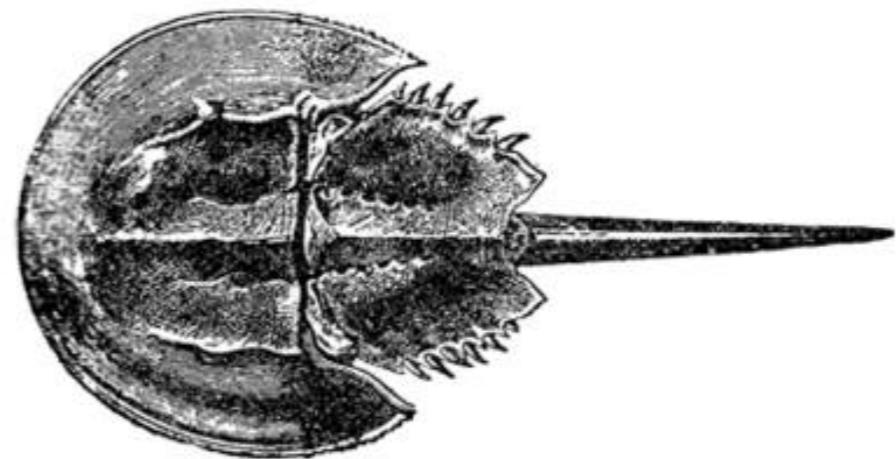
Bitnauts  
ビットノーツ

Master Kuei 著  
ほりかわ けんいち 訳

*You need to overturn your code every six month.*

# 每六個月入門一次 前端技術

因為變動太快，我們後來都沒寫文件了。



O!Really?

Yoshitaka Jingu 著  
ほりかわ けんいち 訳

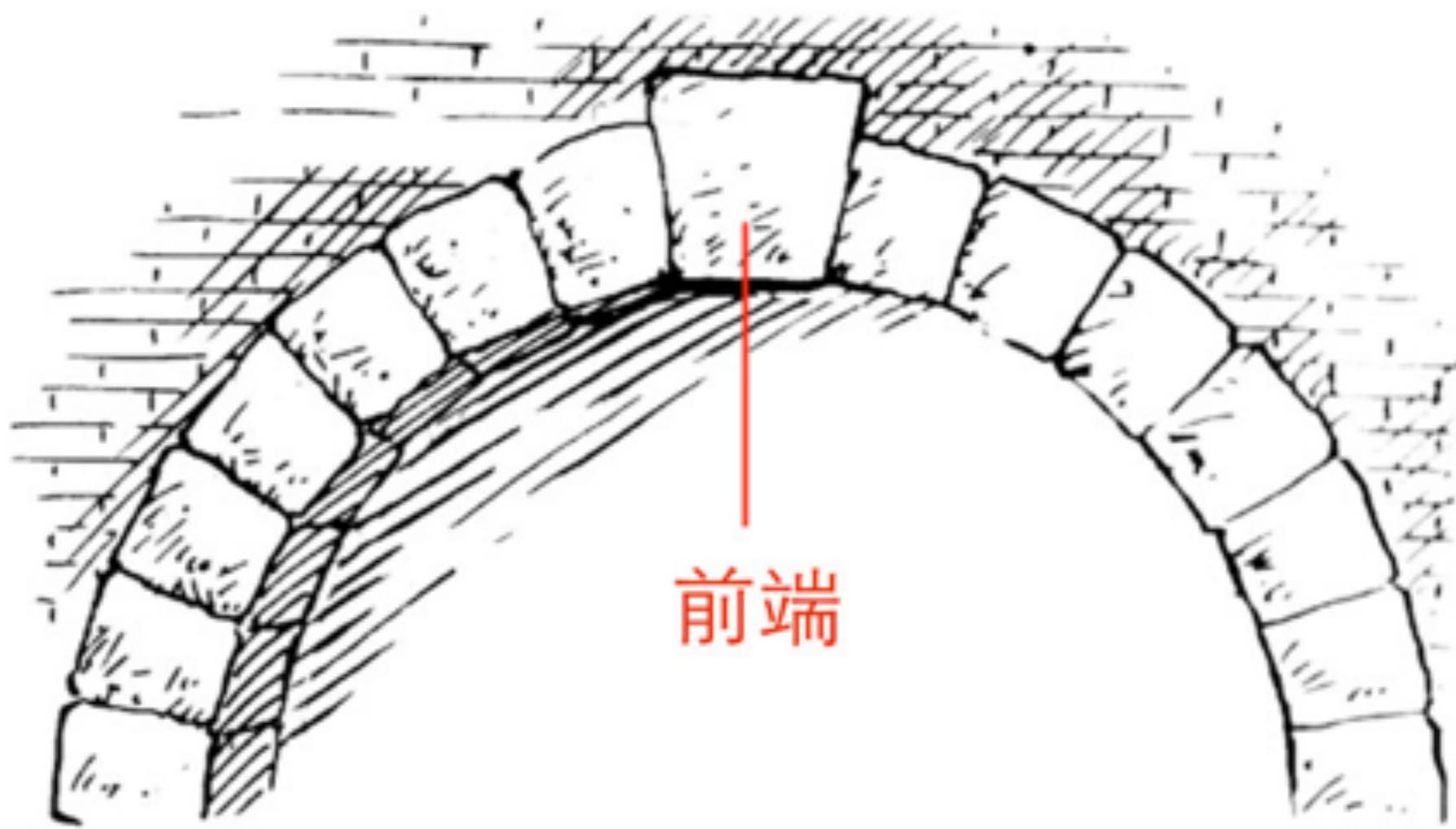
# 以設計廚房水槽來說

設計師 決定水龍頭樣式、位置  
前端工程師 安裝水龍頭與水管  
後端工程師 讓水管有水

Source: <http://www.slideshare.net/adamp3wang/mockupwebconftw-2013>  
那些 mockup 沒告訴你的事 @ WebConf.tw 2013 by Adam Wang

**FAIL**



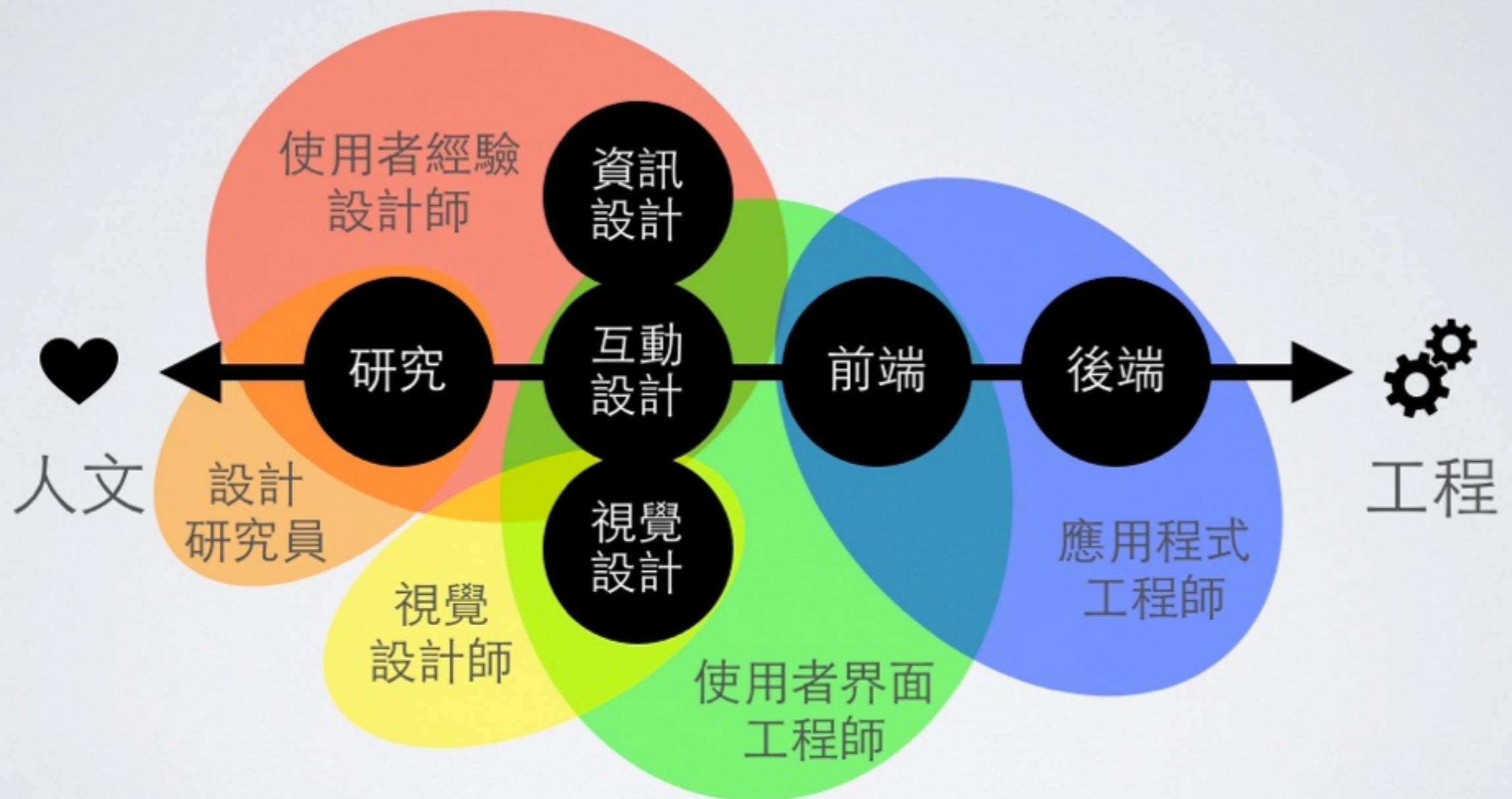


設計 視覺、介面

資料、邏輯 技術

Source: <https://speakerdeck.com/josephj/f2e-the-keystone>, – F2E, the Keystone by Joseph Chiang.

# 專業分工，環環相扣



改繪自<http://taipro.blogspot.tw/2012/08/the-difference-between-ux-designer-and.html>

Source: <http://www.slideshare.net/lis186/20130112webconf>

5x{}.tw

「讓專業的來」

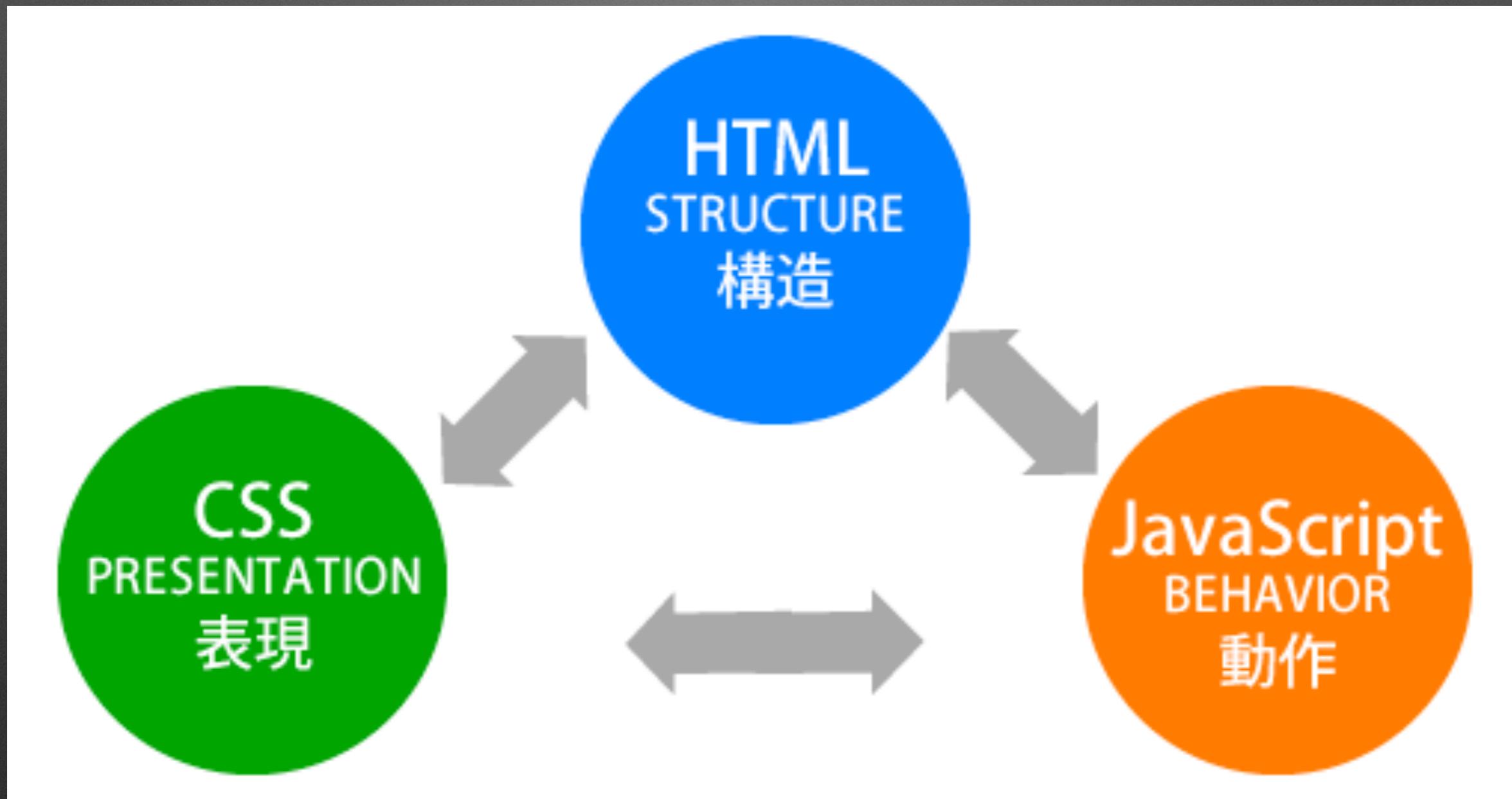
# Javascript

- 誕生於 1995 年，由 Brendan Eich 所開發
- JavaScript 的核心語言是由 ECMA TC-39 委員會統一標準，並且將此標準命名為 ECMAScript 。
- **JavaScript** 與 **Java** 的關係就像是「**臘腸狗**」與「**臘腸**」之間的關係。  
簡單來說就是沒有關係
- JavaScript 是一種 Prototype-based 程式語言。

# 載入 JavaScript

- <script> 放在 <head> ... </head> 之間
- inline-script , 放在屬性內 <a onclick= ".....">
- <script> 放在 </body> 之前
- 外部 script 利用  
<script src="xxx.js"></script> 指定

# 前端三大基本要素



```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Title</title>
</head>
<body>
</body>
</html>
```

# Chrome DevTools

Developer Tools - https://www.google.com.tw/?gfe\_rd=cr&ei=PEwUV8P-AerVmQX9pYewAg&gws\_rd=ssl

Elements    Console    Sources    Network    Timeline    Profiles    Security    Resources    Audits

<!DOCTYPE html>  
<html itemscope itemtype="http://schema.org/WebPage" lang="zh-TW">  
  <head>...</head>  
  <body class="hp vasq" onload="try{if(!google.j.b){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}}catch(e){}if(document.images)new Image().src='/images/nav\_logo242.png'" id="gsr" cz-shortcut-listen="true">  
    <div class="ctr-p" id="viewport">...</div>  
    <script src="/xjs/\_/js/k=xjs.s.zh\_TW.5tSzKJeUw20.0/m=sy31,sy37,em3,em2,sy39,em0,sy2...PAsIWwqFBasAIFq0/rt=j/d=0/t=zcms/rs=ACT90oGOiJC-oSdnWiryVRjZhbakMwszw" gapi\_processed="true"></script>  
    <div class="jfk-bubble chw-oc" role="alertdialog" aria-describedby="bubble-4" style="display: none;">...</div>  
    <iframe src="https://clients5.google.com/pagead/drt/dn/" aria-hidden="true" style="display: none;">...</iframe>  
  </body>  
</html>

Styles    Computed    Event Listeners    DOM Breakpoints    Properties

Filter    + .cls   

element.style {  
}  
body, html {  
  font-size: small;  
}  
body {  
  background: □#fff;  
  color: ■#222;  
}  
body, td, a, p, .h {  
  font-family: arial,sans-serif;  
}  
html, body {  
  height: 100%;  
  margin: ▷0;  
}  
body {  
  display: block;  
  margin: ▷8px;  
}  
Inherited from html  
body, html {  
  font-size: small;  
}

user agent stylesheet

?gfe\_rd=cr&ei=P...&gws\_rd=ssl:13  
?gfe\_rd=cr&ei=P...&gws\_rd=ssl:13  
?gfe\_rd=cr&ei=P...&gws\_rd=ssl:13  
?gfe\_rd=cr&ei=P...&gws\_rd=ssl:13  
?gfe\_rd=cr&ei=P...&gws\_rd=ssl:13  
?gfe\_rd=cr&ei=P...&gws\_rd=ssl:13  
?gfe\_rd=cr&ei=P...&gws\_rd=ssl:13

html    body#gsr.hp.vasq

# JS BIN

<https://jsbin.com/>

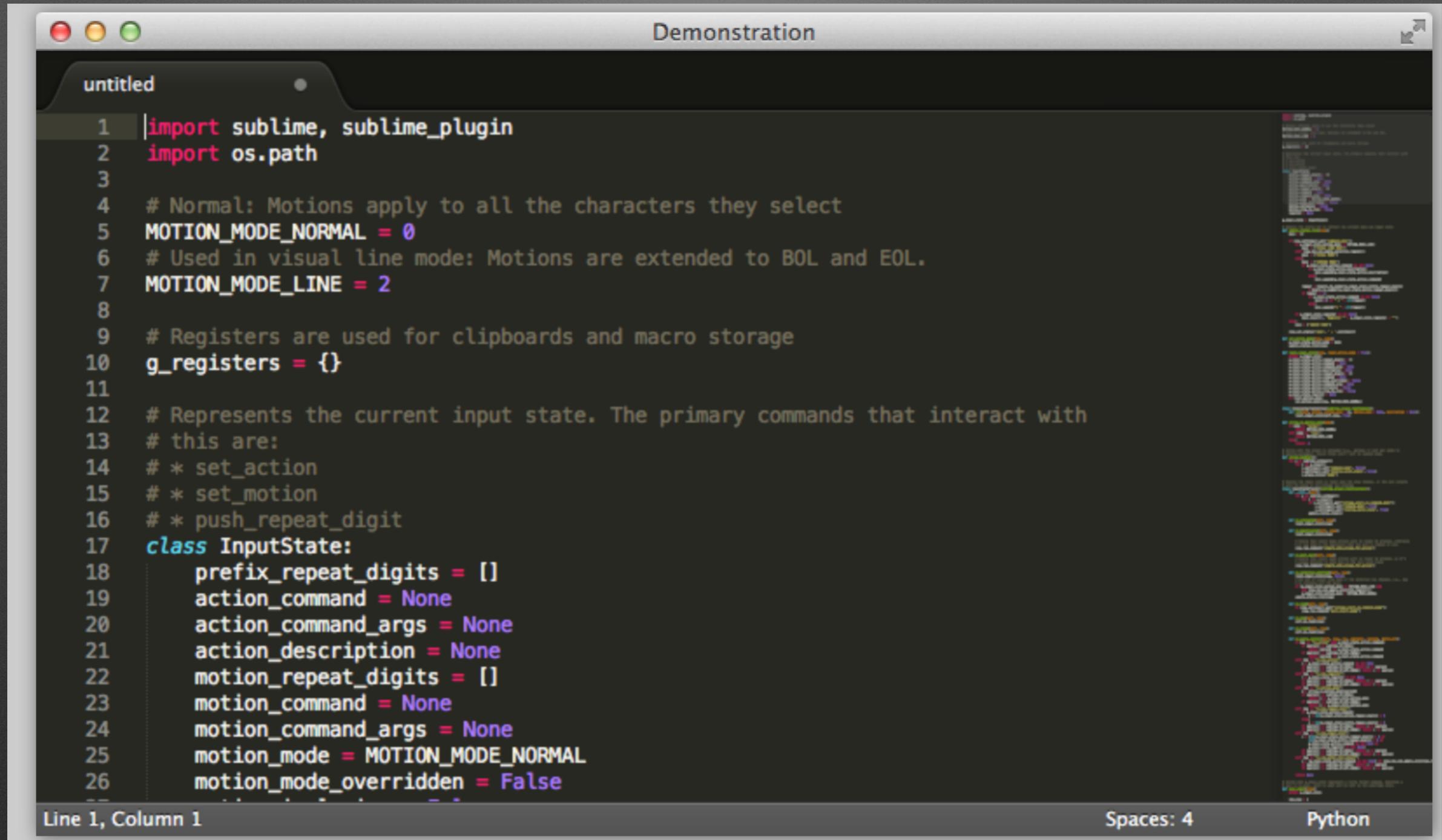
The screenshot shows the JS Bin web application interface. At the top is a navigation bar with icons for file operations, a trash bin for adding libraries, a 'ShareHTML' button, 'CSS', 'JavaScript', 'Console', 'Output', 'Login or Register' (which is highlighted in yellow), 'Blog' (with a red notification badge), and 'Help'. Below the navigation bar is a dropdown menu labeled 'HTML' with a downward arrow. The main area contains the following HTML code:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>JS Bin</title>
</head>
<body>

</body>
</html>
```

In the bottom right corner, there is a small white box containing the text 'Bin info' and 'just now'.

# Sublime Text



A screenshot of the Sublime Text 2 interface. The window title is "Demonstration". The main editor tab is titled "untitled". The code in the editor is:

```
1 import sublime, sublime_plugin
2 import os.path
3
4 # Normal: Motions apply to all the characters they select
5 MOTION_MODE_NORMAL = 0
6 # Used in visual line mode: Motions are extended to BOL and EOL.
7 MOTION_MODE_LINE = 2
8
9 # Registers are used for clipboards and macro storage
10 g_registers = {}
11
12 # Represents the current input state. The primary commands that interact with
13 # this are:
14 # * set_action
15 # * set_motion
16 # * push_repeat_digit
17 class InputState:
18     prefix_repeat_digits = []
19     action_command = None
20     action_command_args = None
21     action_description = None
22     motion_repeat_digits = []
23     motion_command = None
24     motion_command_args = None
25     motion_mode = MOTION_MODE_NORMAL
26     motion_mode_overridden = False
```

The status bar at the bottom shows "Line 1, Column 1" on the left, "Spaces: 4" in the center, and "Python" on the right. To the right of the editor is a vertical sidebar containing multiple file thumbnails.



# Web Server for Chrome

offered by [chromebeat.com](http://chromebeat.com)

★★★★★ (312)

[Developer Tools](#)

30,275 users

[LAUNCH APP](#)



OVERVIEW

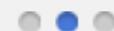
REVIEWS

SUPPORT

RELATED

G+1

130



## Web Server for Chrome

Runs Offline

Compatible with your device

A Web Server for Chrome, serves web pages from a local folder over the network, using HTTP. Runs offline.

Web Server for Chrome is an open source (MIT) HTTP server for Chrome.

It runs anywhere that you have Chrome installed, so you can take it anywhere. It even works on ARM chromebooks.

It now has the option to listen on the local network, so other computers can

[Website](#)

[Report Abuse](#)

### Additional Information

Version: 0.4.1

Updated: May 23, 2016

Size: 439KiB

Language: English

<https://goo.gl/RY5c30>

# 基礎語法

# 敘述句 與 運算式

- 每個敘述句或運算式都有分號做為結尾。

```
var foo;           // 做事情
```

```
foo = 5 * ruby;  // 會產生值
```

# 注釋

- 瀏覽器看到註釋區塊時會自動略過。

```
// 這是單行注釋
```

```
/*
    這是多行注釋
    這是多行注釋
*/
```

# 變數

```
var Hello;  
alert( Hello );      // 宣告後不會出錯，但...
```



# 變數

- 存放數值的位置，透過 var 做宣告，將值透過 「 = 」 來存入 ( assign ) 變數。

```
var name = "kuro";
```

- 變數有它的**有效範圍** ( scope )，如果沒有透過 var 宣告的變數，會自動變成**全域變數** ( global variable )，成為 window 成員屬性之一。
- 變數名稱**大小寫有別**，且不可以用數字當開頭，也不能使用**保留字**。

# 保留字

## Keywords

Reserved keywords as of ECMAScript 6

• <code>break</code>	<code>export</code>	<code>super</code>
• <code>case</code>	<code>extends</code>	<code>switch</code>
• <code>catch</code>	<code>finally</code>	<code>this</code>
• <code>class</code>	<code>for</code>	<code>throw</code>
• <code>const</code>	<code>function</code>	<code>try</code>
• <code>continue</code>	<code>if</code>	<code>typeof</code>
• <code>debugger</code>	<code>import</code>	<code>var</code>
• <code>default</code>	<code>in</code>	<code>void</code>
• <code>delete</code>	<code>instanceof</code>	<code>while</code>
• <code>do</code>	<code>new</code>	<code>with</code>
• <code>else</code>	<code>return</code>	<code>yield</code>

# 函數 function

- 將一或多段程式指令包裝起來，可以重複使用，方便維護。使用的時候，透過 **函數名 + (參數)**

- 如：

```
alert('Hello JavaScript');
```

```
alert('Hello 5xRuby');
```

函數名稱

參數

# 陣列

- 可以將它看作是「變數的集合」，以**中括號 [ ]**表示
- 是一種有順序性的列表，索引由 0 開始計算
- 同陣列元素間可以混搭不同的資料型態，但**不建議**

# 物件

一至多種屬性的集合（無順序性），  
以 { key: value } 的方式組合起來。

```
var people = {  
    name: "Kuro",  
    age: 32,  
    skills:{  
        frontend: ["HTML", "CSS", "JavaScript"],  
        backend: ["PHP", "ASP.net", "node.js"]  
    sayHello: function(){ alert('hello'); }
```

# 除錯技巧 console.log( )

The screenshot shows a browser developer tools interface with the following details:

- Top Bar:** File ▾, Add library, Share, HTML (selected), CSS, JavaScript, Console, Output, Login or Register, Blog 1, Help.
- JavaScript Editor:** Shows two lines of code:

```
console.log(Hello);  
console.log("Hello");
```
- Console Panel:** Displays an error message:

```
x "ReferenceError: Hello is not defined  
at nonemulilo.js:1:43  
at https://static.jsbin.com/js/prod/runner-  
3.37.0.min.js:1:13891  
at https://static.jsbin.com/js/prod/runner-  
3.37.0.min.js:1:10820"
```

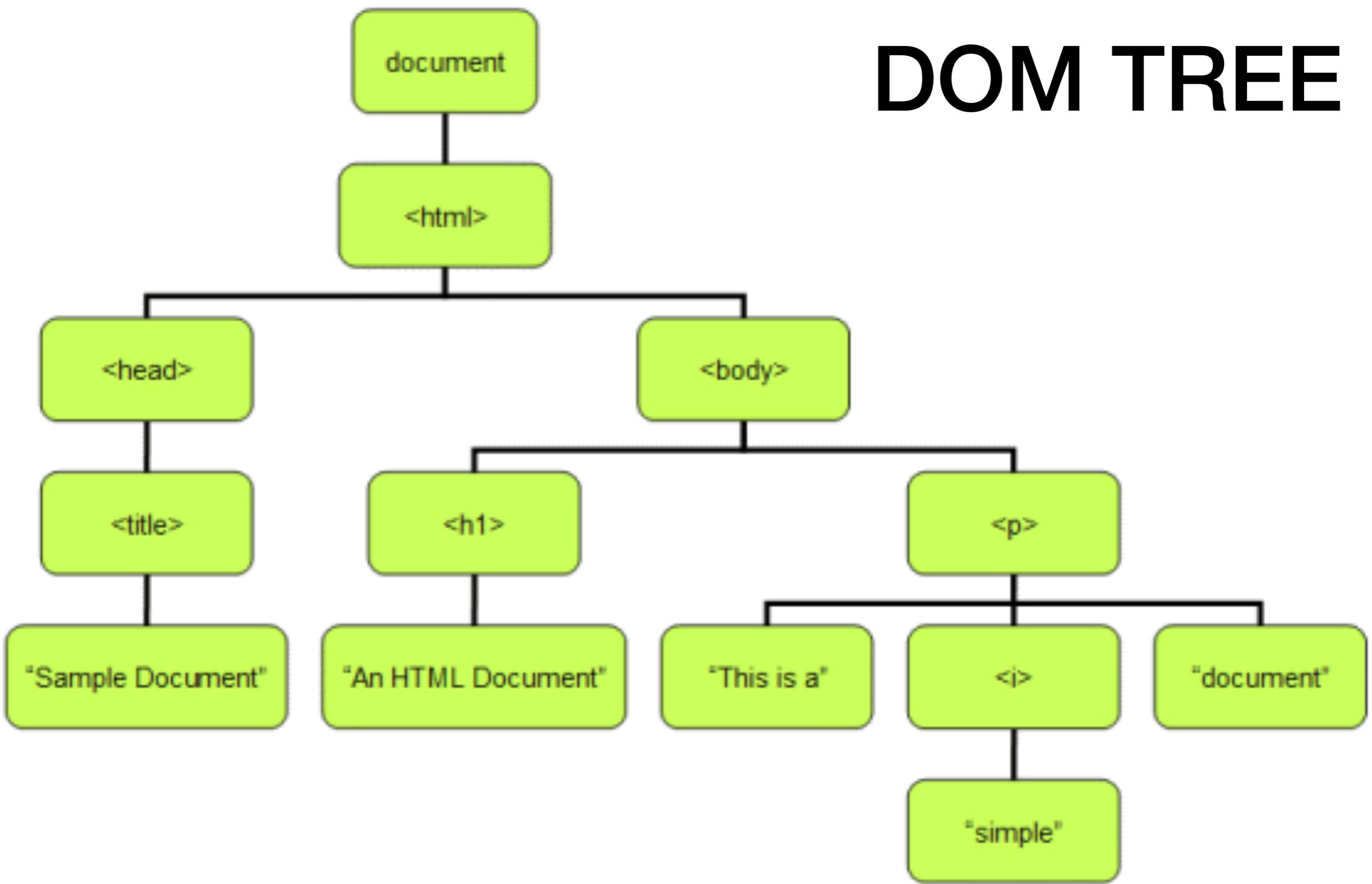
With a "Bin info just now" button.
- Bottom Navigation:** Elements, **Console** (selected), Network, Sources, Timeline, Profiles, Security, Resources, Audits.
- Console Log:** Shows the following entries:
  - Hello
  - ReferenceError: Hello is not defined  
at VM200 nonemulilo.js:1  
at runner-3.37.0.min.js:1  
at runner-3.37.0.min.js:1
  - Uncaught ReferenceError: Hello is not defined

# JavaScript 與 HTML

# 文件物件模型 (DOM)

- 文件物件模型 (Document Object Model, DOM)
- 將 HTML 文件以樹狀的結構來表示的模型，而每一個節點 (node) 代表的就是一個 HTML 元素。
- DOM API 定義讓程式可以存取、改變 HTML 架構、樣式和內容的方法，以及對節點綁定的事件。
- JavaScript 就是透過 DOM 提供的 API 來對 HTML 做存取與操作。

# DOM TREE



- **document.getElementById():**  
根據傳入的值，找到 DOM 中 `id` 為該值的元素。
- **document.getElementsByTagName():**  
針對所有給定的 `tag` 元素，回傳符合條件的 `NodeList` 物件。
- **document.getElementsByClassName():**  
針對所有給定的 `class` 子元素，回傳符合條件的 `NodeList` 物件。
- **document.querySelector() /  
document.querySelectorAll():**  
針對給定的條件，回傳第一個或所有符合條件的 `NodeList`。

# NodeList

- **NodeList** 物件是節點的集合，可藉由 `Node.childNodes` 屬性以及 `document.querySelectorAll()` 方法取得。
- **NodeList** 雖然有著與陣列相似的特性，但不是陣列，所以也不會有陣列相關的 `method` 可以使用（如 `map`、`filter` 等）。

# Hello World!

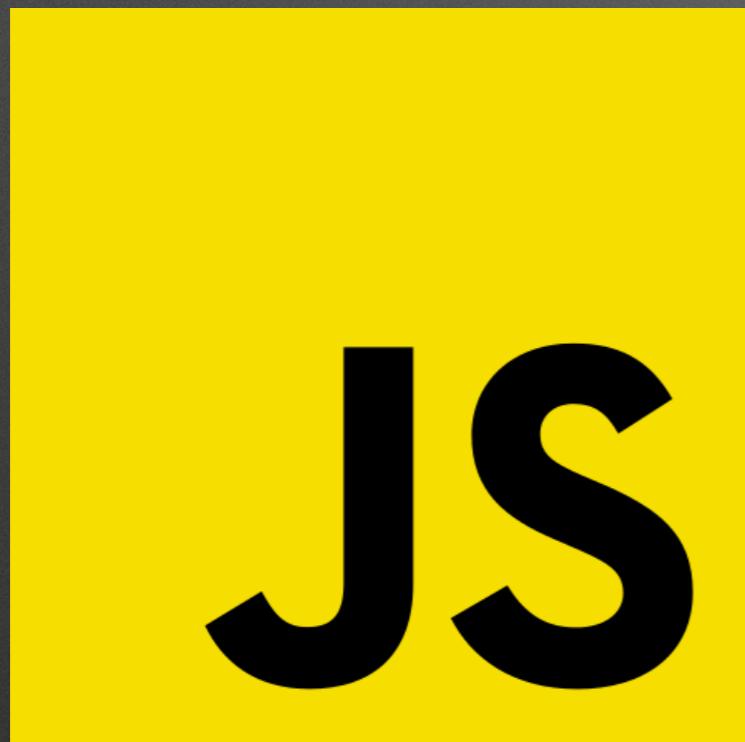
```
<h1 id="hello"></h1>
```

```
document.querySelector('#hello')  
    .innerHTML = 'HELLO';
```

# Recap

- JavaScript 透過什麼來操作網頁？
- DOM TREE 是將 HTML 文件以什麼形式來表示的模型？
- 前端開發者透過 JS 在瀏覽器的操作，都是在跟 DOM API 打交道，所以，學會如何操作 DOM 物件是寫 Web UI 的基礎。

# JS 與 jQuery 的關係





- 跨瀏覽器的 JS 函式庫，由 John Resig 於 2006 發表，  
目前最新版本是 3.3.1 (2018.01.20)
- 將繁複的 JS 語言包裝起來，大幅度簡化網頁的操作流  
程，即使對 JS、DOM API 不熟悉的開發者也能容易上手。
- jQuery 的內容就是 JavaScript 程式碼。
- 在程式碼內，通常會用 \$ 來表示 jQuery。

# Hello World!

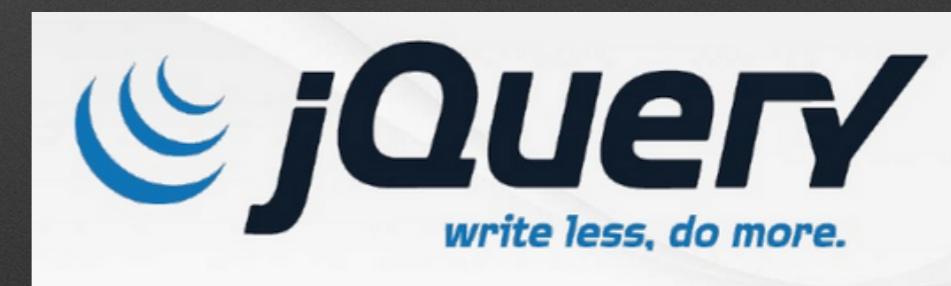
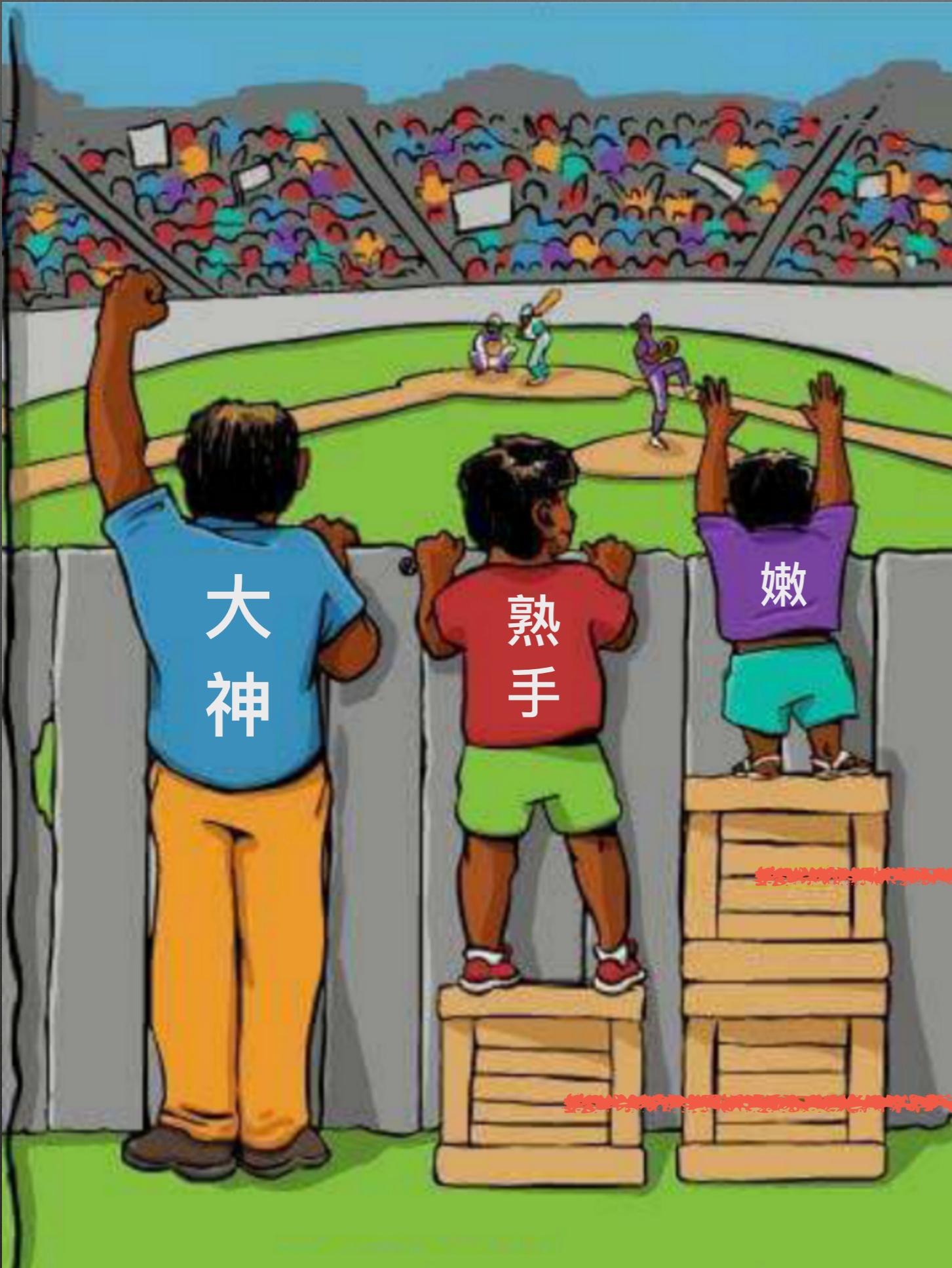
```
<h1 id="hello"></h1>
```

```
document.querySelector('#hello')  
    .innerHTML = 'HELLO';
```

# Hello World with jQuery

```
<h1 id="hello"></h1>
```

```
$( '#hello' ).html( 'HELLO' );
```





CDN - <https://code.jquery.com/>

```
<script src="https://code.jquery.com/jquery-3.1.0.js"></script>
```

官網下載 - <https://jquery.com/download/>

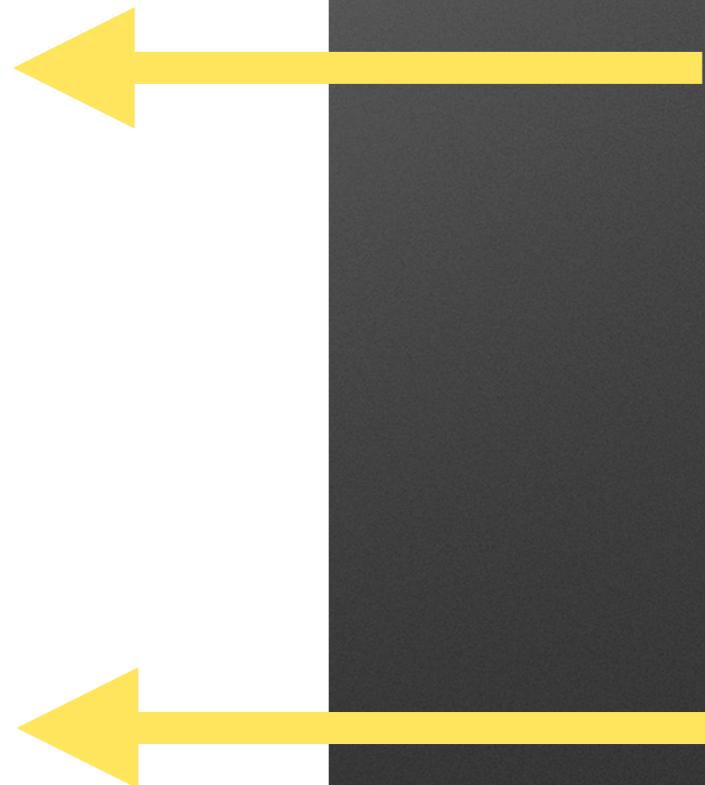
The screenshot shows the official jQuery website's download page. At the top, there's a navigation bar with links for 'Download', 'API Documentation', 'Blog', 'Plugins', and 'Browser Support'. To the right of the navigation is a search bar with a magnifying glass icon. Below the navigation, a large heading says 'Downloading jQuery'. Underneath this heading, there's a paragraph of text explaining the availability of compressed and uncompressed files, mentioning source map files, and noting that the sourceMappingURL comment is not included in the compressed file. At the bottom of this section, it says to right-click and select 'Save as...' from the menu. On the right side of the page, there's a sidebar with a 'Check web platform status across browsers' button and a 'Show Me' link.

# 放哪裡的差別？

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Title</title>
```

```
</head>
<body>
```

```
</body>
</html>
```



# 再回到 Hello World! 範例

```
<h1 id="hello"></h1>
```

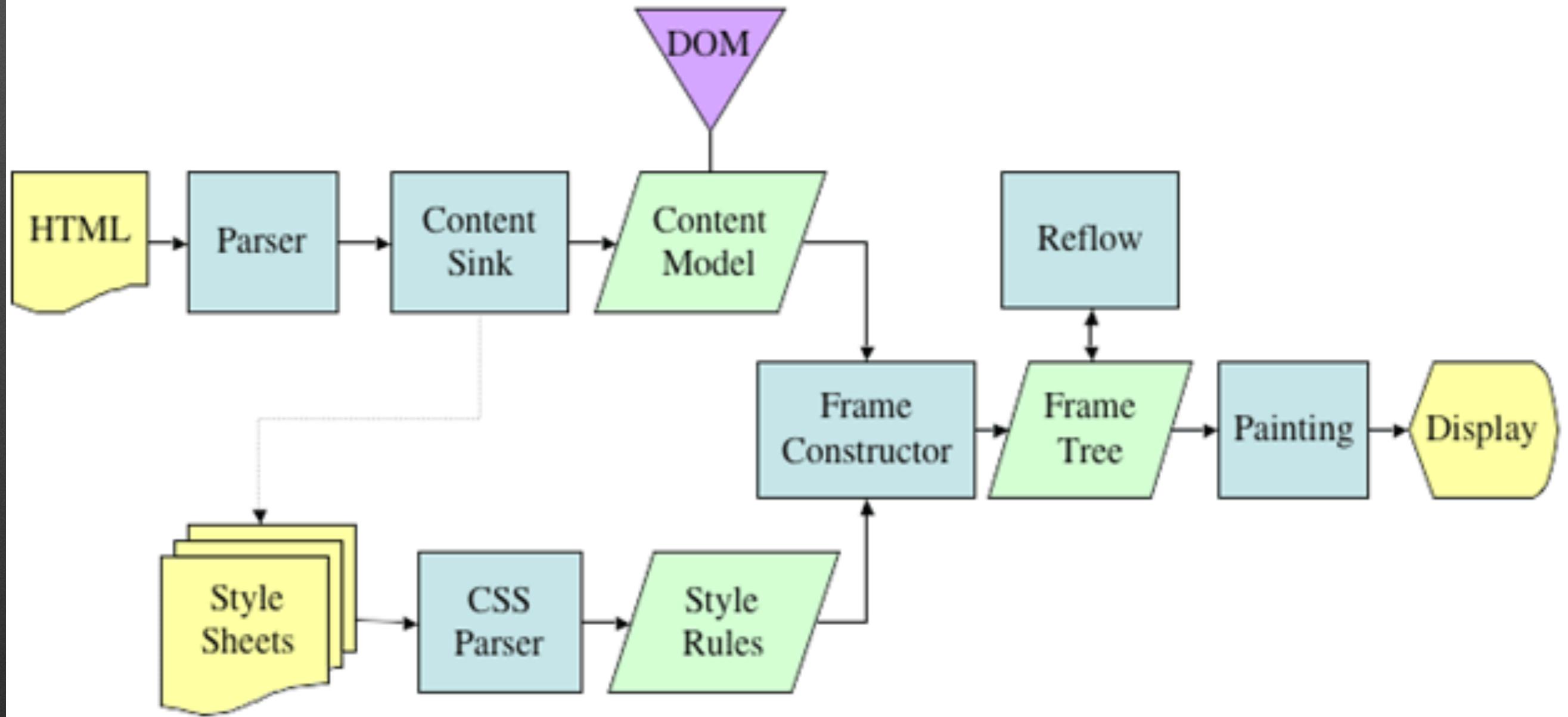
```
document.querySelector('#hello')  
    .innerHTML = 'HELLO';
```

← → C ⌂

① 127.0.0.1:8887/ex-009.html

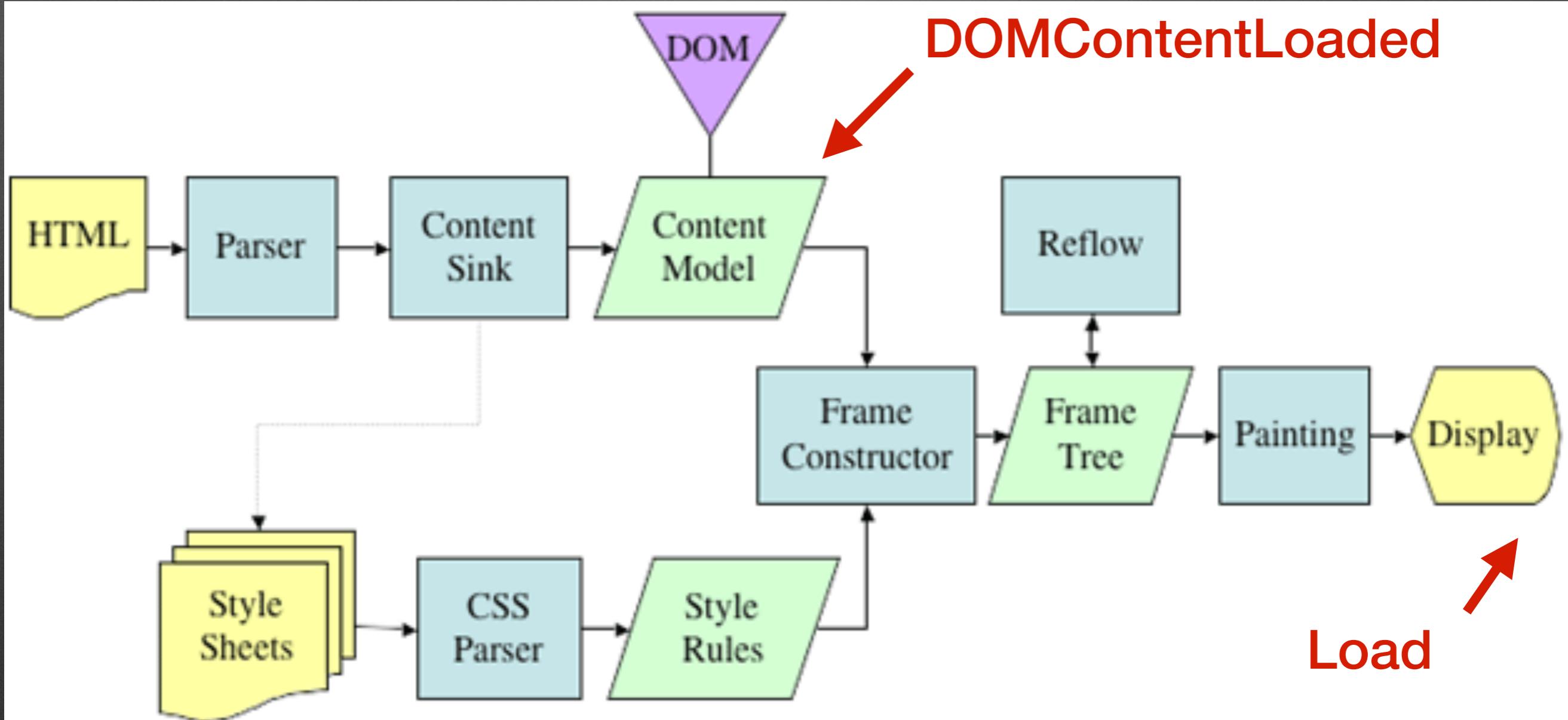


# 瀏覽器是怎麼運作的



# 瀏覽器是怎麼運作的

- **DOMContentLoaded** 代表的是剛建構好 DOM 物件的時間點，也就是 HTML 元素都解析完的時候，就會觸發的事件。
- **onLoad** 代表的是頁面描繪好的時間點，也就是所有元素、資源 (如圖片、音源、影像等) 都讀取完成的時候，才會觸發的事件。



<https://www.engineersgarage.com/articles/web-browsers-what-is-web-browser-working?page=3>

# Page lifecycle

- **DOMContentLoaded**: HTML 讀取完畢，DOM tree 創建立，但外部資源（如圖檔、樣式等）還未載入。
- **load**: 外部資源載入完成時觸發。
- **beforeunload/unload**: 當使用者要離開頁面時觸發，並詢問使用者是否離開。

# `$(document).ready( );`

- 當網頁的所有 HTML CSS，與圖片等靜態資源已經載入完成，就會觸發 jQuery 的 **ready** 事件。

```
$(document).ready(function(){ ... });
```

```
$(function(){ ... });
```

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Title</title>
</head>
<body>
</body>
</html>
```

一般來說，為了避免載入外部資源產生的時間延遲，  
會建議將 script 放在最底端。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Title</title>

```



```
</head>
<body>
```

```
</body>
</html>
```

通常只有在 `<body onload="init()">` 這類當頁面讀取時馬上就要執行的程式，才會一定得放在上面。

# jQuery 選擇器

# jQuery 基本選擇器

```
$('#header')          // 選取所有 id="header"  
$('.header')           // 選取所有 class="header"  
$('div')               // 選取所有 <div>  
$('div, p')             // 同時選取所有 <div> 與 <p>  
  
$('body > .tag')       // body 「下一層」的所有 .tag  
$('body div')           // body 下的所有 div (含子孫)  
$('div.body')            // 選取所有 class="body" 的 div
```

# jQuery 選擇器 (續)

```
$('a[href="#"]')      // 選取所有 href 屬性是 # 的 <a>  
$("*[href]")          // 選取每個有 href 屬性的元素  
  
$('.header > *')     // 選取 .header 下一層的所有元素  
  
$("ul li:first")      // 選取每個 ul 的第一個 li  
 $("ul li:last")       // 選取每個 ul 的最後一個 li  
 $("ul li:eq(n)")      // 選取每個 ul 的第 n 個 li  
  
(...).length           // .length 回傳選取到多少元素
```

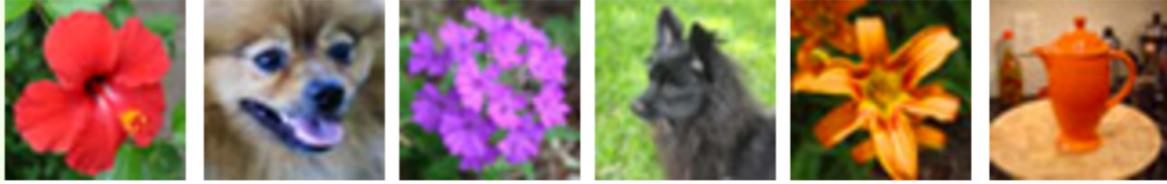
# 練習

<http://goo.gl/2QL8JR>

# 問題一

## DOM Sample

**Some images:**



This is a <div> with an id of someDiv  
Hello  
Goodbye

- [jQuery supports](#)
  - [CSS1](#)
  - [CSS2](#)
  - [CSS3](#)
  - Basic XPath
- jQuery also supports
  - Custom selectors
  - Form selectors

Language	Type	Invented
Java	Static	1995
Ruby	Dynamic	1993
Smalltalk	Dynamic	1972
C++	Static	1983

**Text:**

**Radio group:**  A  B  C

**Checkboxes:**  1  2  3  4

**Submit**

## 問題二

### DOM Sample

**Some images:**



This is a <div> with an id of someDiv

Hello  
Goodbye

- jQuery supports
  - [CSS1](#)
  - [CSS2](#)
  - [CSS3](#)
  - Basic XPath
- jQuery also supports
  - Custom selectors
  - Form selectors

Language	Type	Invented
Java	Static	1995
Ruby	Dynamic	1993
Smalltalk	Dynamic	1972
C++	Static	1983

**Text:**

**Radio group:**  A  B  C

**Checkboxes:**  1  2  3  4

**Submit**

# 問題三

## DOM Sample

**Some images:**



This is a <div> with an id of someDiv

Hello  
Goodbye

- [jQuery supports](#)
  - [CSS1](#)
  - [CSS2](#)
  - [CSS3](#)
  - Basic XPath
- jQuery also supports
  - Custom selectors
  - Form selectors

Language	Type	Invented
Java	Static	1995
Ruby	Dynamic	1993
Smalltalk	Dynamic	1972
C++	Static	1983

**Text:**

**Radio group:**  A  B  C

**Checkboxes:**  1  2  3  4

**Submit**

# 問題四

## DOM Sample

**Some images:**



This is a <div> with an id of someDiv  
Hello  
Goodbye

- [jQuery supports](#)
  - [CSS1](#)
  - [CSS2](#)
  - [CSS3](#)
  - Basic XPath
- jQuery also supports
  - Custom selectors
  - Form selectors

Language	Type	Invented
Java	Static	1995
Ruby	Dynamic	1993
Smalltalk	Dynamic	1972
C++	Static	1983

**Text:**

**Radio group:**  A  B  C

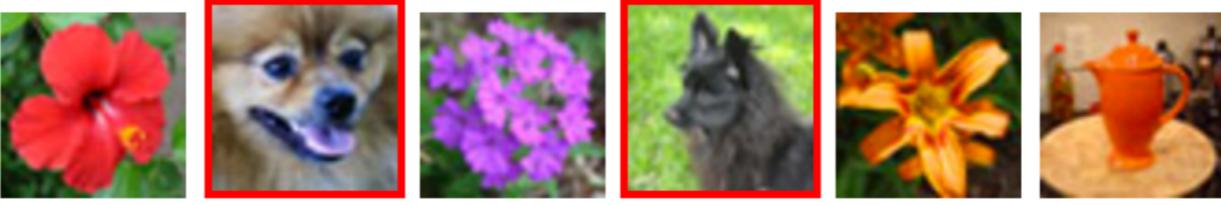
**Checkboxes:**  1  2  3  4

**Submit**

# 問題五

## DOM Sample

**Some images:**



This is a <div> with an id of someDiv

Hello  
Goodbye

- jQuery supports
  - [CSS1](#)
  - [CSS2](#)
  - [CSS3](#)
  - Basic XPath
- jQuery also supports
  - Custom selectors
  - Form selectors

Language	Type	Invented
Java	Static	1995
Ruby	Dynamic	1993
Smalltalk	Dynamic	1972
C++	Static	1983

**Text:**

**Radio group:**  A  B  C

**Checkboxes:**  1  2  3  4

**Submit**

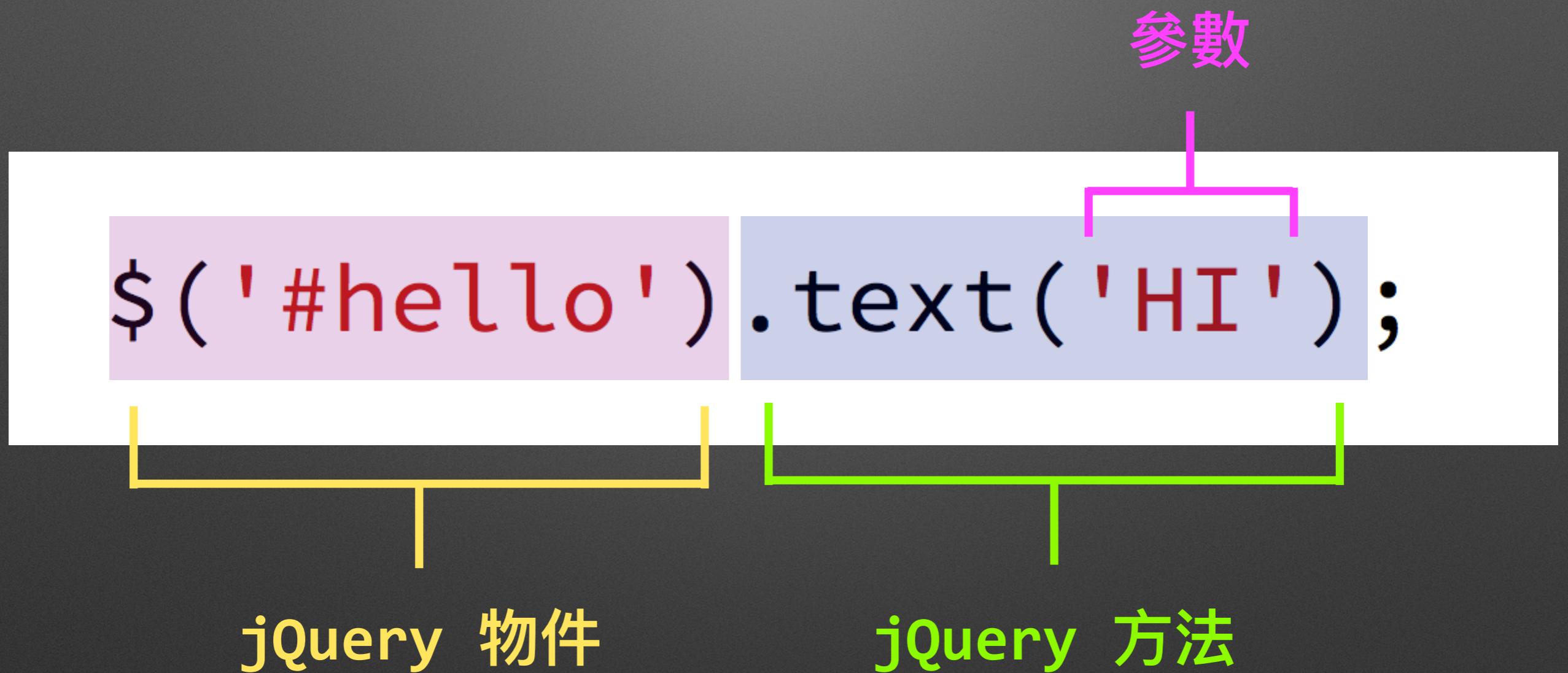
# jQuery 物件

```
$('.hello').length;
```

jQuery 物件

jQuery 物件屬性

# jQuery 方法



# jQuery 選取器與物件

- DOM 節點必須要被 jQuery 包覆後，才能使用 jQuery 提供的方法（ method ）
- jQuery 選擇節點的方式與 CSS 的選擇器（ selector ）完全相容，且 jQuery 大幅簡化了 繁瑣的 DOM API，使得開發者能更方便地進行操作。
- CSS 選擇器只能在 同層 或 往下層 尋找節點，但 jQuery 有提供方法可往節點的上層查找。
- 選取器參考：<https://api.jquery.com/category/selectors/>

# .text( )

讀取或設定指定 DOM 的文字內容。

```
// 設值  
$('#heading').text('<h1>Hi 5xRuby</h1>');
```

```
// 取值  
$('#heading').text();
```

# .html( )

讀取或設定指定 DOM 的 HTML 內容。

```
$('#heading').html('<h1>Hi 5xRuby</h1>');
```

```
$('#heading').html();
```

# .val( )

讀取或設定指定 表單元件 的值。

```
$('#heading').val('text');
```

```
$('#heading').val();
```

注意：.text, .html 對表單元素無效

```
.on('click', ...)
```

控制按下某元素之後要做的事

```
$('button').on('click', function(){  
    // do something  
});
```

# 字串

用 單引號 或 雙引號 包起來。  
可用 + 將字串連接。

```
var name = $('#name').val();  
  
alert( "Hello " + name );
```

# 當數字遇到字串...

```
var age = 30;  
age = age + 1 ;      // 31  
  
age = age + "";     // "31"  
age = age + 1 ;     // ??  
age = age * 2 ;     // ??
```

# 練習

- 試著製作一個輸入框 `<input type="text">` 讓使用者輸入姓名，與一個按鈕 `<button>`
- 當按下按鈕時，跳出 `alert` 「Hi xxx 你好！」的訊息

# 變數資料型別

- 基本型別 ( Primitive types )
  - 數字 (number)
  - 字串 (string)
  - 布林 (boolean)
  - null
  - undefined
- 物件型別 ( Object types )
  - 所有非基本型別的值都屬於 物件

**Table 20 — typeof Operator Results**

Type of val	Result
Undefined	" <b>undefined</b> "
Null	" <b>object</b> "
Boolean	" <b>boolean</b> "
Number	" <b>number</b> "
String	" <b>string</b> "
Object (native and does not implement [[Call]])	" <b>object</b> "
Object (native or host and does implement [[Call]])	" <b>function</b> "
Object (host and does not implement [[Call]])	Implementation-defined except may not be " <b>undefined</b> ", " <b>boolean</b> ", " <b>number</b> ", or " <b>string</b> ".

ref: <http://www.ecma-international.org/ecma-262/5.1/#sec-11.4.3>

# 基本型別 ( Primitive types )

- 以「**值 (value)**」來進行比較。
- 基本型別的屬性 (property) 是唯讀的。

```
> 123 === 123
< true
> '5xRuby' === '5xRuby'
< true
```

```
> var str = '5xRuby';
< undefined
> str.length
< 6
> str.length = 10
< 10
> str.length
< 6
```

# 物件型別 (Object types)

- 以「參考 (reference)」來進行比較。

```
> var obj = {};
< undefined
> var obj2 = {};
< undefined
> obj === obj2
< false
>
```

```
> obj2 = obj;
< Object {}
> obj === obj2
< true
>
```

- 物件型別的屬性 (property) 是可能被變動的。

# 判斷變數型別： `typeof`

```
typeof true;           // 'boolean'  
typeof '5xRuby';     // 'string'  
typeof 123;           // 'number'  
typeof {};            // 'object'  
typeof [];            // 'object'  
typeof function(){}; // 'function'
```

JavaScript 是一種 **弱型別** 的程式語言，變數的資料型別是可以隨時變動的。

# 數字

```
var age = 30;  
  
age = age + 1 ;           // 31  
  
age = age - 10 ;          // 21
```

# 特殊數字

- Infinity (無限大)
- -Infinity (負無限大)
- NaN

# NaN

- NaN = Not a Number
- 是「數字」又「不是數字」：  
`typeof NaN ==> "Number"`
- 如 `0/0` , `parseInt("string", 10)`
- `(NaN === NaN) ==> false`
- 用 `isNaN(n)` 來判斷 n 是否為 NaN

與數學運算一樣有「先乘除，後加減」的觀念。

## JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers (literals or variables).

Operator	Description	
+	Addition	加法
-	Subtraction	減法
*	Multiplication	乘法
/	Division	除法
%	Modulus	取餘數
++	Increment	遞增
--	Decrement	遞減

一分鐘有 60 秒，一小時有 60 分鐘，一小時有幾秒？

一天有 24 小時，一天共有幾秒？

```
var secondsInMinute = 60;
```

```
var minutesInHour = 60;
```

```
var secondsInHour =  
    secondsInMinute * minutesInHour;
```

```
var hoursInDay = 24;
```

```
var secondInDay = ?
```

# 字串

用 單引號 或 雙引號 包起來。

可用 + 將字串連接。

```
var name = "Kuro";
console.log( "Hello " + name );
```

```
name = "5xRuby";
console.log( "Hello " + name );
```

# 跳脫字元 ( Escaped character )

如果字串同時又包覆引號時，則需要跳脫字元幫助

```
var str = "What's This."; // OK
```

```
str = 'What's This.'; // error
```

```
str = 'What\'s This.'; // OK
```

```
var number = prompt("請輸入0 ~ 9的數字", '');  
alert( 100 + number ); // ??
```

小心，prompt回傳的都是字符串型態的資料。

注意自動轉型！可先利用 `typeof` 判斷型別。

若是數值運算時，強烈建議先用  
`parseInt(n, 10)` 或 `Number(n)` 作轉換。

```
var age = 30;  
  
age = age + "";      // "30" 注意這是字串  
age = age + 1;      // ??  
age = age * 2;      // ??      搞得我好亂啊
```

# 字符串的方法 (Methods)

```
'5xRuby'.slice(1);           // "xRuby"  
'5xRuby'.slice(1, 2);        // "x"  
  
'5xRuby'.toUpperCase();     // "5XRUBY"  
'5xRuby'.toLowerCase();     // "5xruby"  
  
'5xRuby'.indexOf('x');      // 1  
'5xRuby'.indexOf('A');      // -1
```

# 布林值

true / false，作為邏輯判斷用

```
var isMale = true;  
  
if( isMale ){  
    // 要當兵  
}
```

```
var result = confirm("選擇確定或取消");

console.log( result );

if( result ){
    alert('你剛剛按了確定');
}
else{
    alert('你剛剛按了取消');
}
```

# null 與 undefined

- null = (此變數) 沒有值
- undefined = (此變數) 還沒有給值

```
✖️ ⚡ top
> Number( null )
← 0
> Number( undefined )
← NaN
>
```

# 基本型別轉換

```
> Number("123")
< 123
> Number("AAA")
< NaN
> String(123)
< "123"
```

```
> Boolean(123)
< true
> Boolean(1)
< true
> Boolean(0)
< false
> Boolean(-1)
< true
> |
```

# Falsy 與 Truthy: 論 Boolean 的型別轉換

只有這些值在自動轉型時會變成 false，其他都是 true

- Undefined
- Null
- 0, NaN
- 空字串 "" 或 ''

```
console.log( Boolean(false) );
console.log( Boolean("false") );

console.log( Boolean(0) );
console.log( Boolean("0") );

console.log( Boolean("") );

console.log( Boolean( {} ) );
console.log( Boolean( [] ) );
console.log( Boolean( function(){}) );

console.log( !!'false' === !!'true' );
```

# Recap

- 基本型別（ Primitive types ）有哪幾種？
- 物件型別（ Object types ）以什麼進行比較？
- 可以利用什麼來判斷變數目前的型別？
- NaN 是什麼？怎麼判斷是否為數字？
- null 與 undefined 的不同？

# 陣列

- 可以將它看作是「變數的集合」，以 中括號 [ ] 表示
- 索引由 0 開始計算，是一種有順序特性的列表
- 跟其他強型別語言不同，陣列內元素可以混搭不同的型別。

```
var colors = ["red", "yellow", "blue"];  
  
colors.length;  
// 3  
  
colors[0];  
// "red"
```

[https://developer.mozilla.org/zh-TW/docs/Web/JavaScript/Reference/Global\\_Objects/Array](https://developer.mozilla.org/zh-TW/docs/Web/JavaScript/Reference/Global_Objects/Array)

# 陣列與 length

```
var cats = [];
console.log(cats.length);          // 0

cats[30] = '學妹';
console.log(cats.length);          // 31
```

// 可直接透過修改 length 來操作陣列長度

```
var cats = ['Dusty', 'Misty', 'Twiggy'];
console.log(cats.length);
// 3
```

```
cats.length = 2;
console.log(cats.length);
// ['Dusty', 'Misty']
```

```
cats.length = 0;
console.log(cats.length);
// []
```

```
var colors = ["red", "yellow", "blue"];
```

```
// 將某個值加入到陣列尾端  
colors.push( "orange" );  
// ["red", "yellow", "blue", "orange"]
```

```
// 將陣列第一個元素回傳後，刪除該元素  
colors.shift();  
// ["yellow", "blue", "orange"]
```

```
// 將陣列最後一個元素回傳後，刪除該元素  
colors.pop();  
// ["yellow", "blue"]
```

```
// 在陣列的開頭加入一個（或以上）的元素  
colors.unshift( "black" );  
// ["black", "yellow", "blue"]
```

# [array].indexOf

```
var colors = [ "red", "yellow", "blue"];  
  
// 判斷元素是否存在陣列中  
console.log( colors.indexOf("red") );      // 0  
console.log( colors.indexOf("black") );     // -1
```

# 遍歷整個陣列

## .forEach

```
var colors = [ "red", "yellow", "blue"];  
  
colors.forEach(function(element, index, array){  
    console.log("第 " + index + " 個元素是 "  
        + element);  
});
```

# 針對陣列每個元素回傳

## .map

```
var nums = [ 1, 3, 5, 7, 9 ];  
  
var double_num = nums.map(function(n){  
    return n * 2;  
});
```

# 針對陣列做條件過濾

## .filter

```
var nums = [ 1, 3, 5, 7, 9, 2, 4, 6, 8, 10 ];  
  
var filtered_num = nums.filter(function(n){  
    return n >= 5;  
});
```

# 問題

將上一頁的 `.filter()` 改成 `.map()`，  
但其他部分不變，結果會如何？

```
var nums = [ 1, 3, 5, 7, 9, 2, 4, 6, 8, 10 ];  
  
var filtered_num = nums.filter(function(n){  
    return n >= 5;  
});
```

# 練習

- 有一陣列為 [ 1, 2, 3, 4, 5, 10, 9, 8, 7, 6 ]，試著過濾出所有偶數的數值，並放到另一陣列中。
- 有一陣列為 [ 1, 2, 3, 4, 5]，試著用 `.map()` 替陣列中的每個元素作平方運算，使其輸出為 [1, 4, 9, 16, 25]。

# Recap

- JavaScript 如何表示陣列？
- `.push()`、`.shift()`、`.pop()`、`.unshift()` 分別代表什麼意義？

# 流程 與 邏輯判斷

因為 JS 隱含自動轉換型別的特性，比較數值時，建議一律採用 `===` 取代 `==`

## JavaScript Comparison and Logical Operators

Operator	Description	
<code>==</code>	<code>equal to</code>	數值相等
<code>===</code>	<code>equal value and equal type</code>	數值與型別相等
<code>!=</code>	<code>not equal</code>	數值不相等
<code>!==</code>	<code>not equal value or not equal type</code>	數值不相等或型別不相等
<code>&gt;</code>	<code>greater than</code>	大於
<code>&lt;</code>	<code>less than</code>	小於
<code>&gt;=</code>	<code>greater than or equal to</code>	大於或等於
<code>&lt;=</code>	<code>less than or equal to</code>	小於或等於
<code>?</code>	<code>ternary operator</code>	條件運算子 「(條件) ? ... : ...」

```
> 10 + 100 * 2  
< 210  
> "10" + 100 * 2  
< "10200"
```

```
> "10" == 10  
< true  
> "10" === 10  
< false  
> |
```

# 邏輯運算子

雖然回傳的結果是 boolean 的型態，  
但本質上是其實是「選擇運算子」。

<b>Operator</b>	<b>Description</b>	<b>Example</b>
<code>&amp;&amp;</code>	<code>and</code>	<code>(x &lt; 10 &amp;&amp; y &gt; 1) is true</code>
<code>  </code>	<code>or</code>	<code>(x == 5    y == 5) is false</code>
<code>!</code>	<code>not</code>	<code>!(x == y) is true</code>

```
var a = 123;
var b = "abc";
var c = null;

console.log( a && b );           // ?
console.log( a || b );           // ?

console.log( c && a );           // ?
console.log( c || b );           // ?
console.log( c || a );           // ?
```

# if - else

```
if ( 狀況 A ) {  
    // 執行的動作 A ...  
}  
else if ( 狀況 B ) {  
    // 執行的動作 B ...  
}  
else if ( 狀況 C ) {  
    // 執行的動作 C ...  
}  
else{  
    // 以上狀況都不成立時執行  
}
```

# 練習：閏年判斷

- 閏年的條件：
  - 西元年份不可被 4 整除者為平年。
  - 年份可被 4 整除且不為 100 整除者為閏年。
  - 年份可被 400 整除為閏年。
  - 年份可被 1000 整除為閏年。

# Date 物件

- JavaScript 預先定義的 Date 物件，內建所有與年、月、日相關的操作方法。

<b>Method</b>	<b>Description</b>
<u>getDate()</u>	Returns the day of the month (from 1-31)
<u>getDay()</u>	Returns the day of the week (from 0-6)
<u>getFullYear()</u>	Returns the year
<u>getHours()</u>	Returns the hour (from 0-23)
<u>getMilliseconds()</u>	Returns the milliseconds (from 0-999)
<u>getMinutes()</u>	Returns the minutes (from 0-59)
<u>getMonth()</u>	Returns the month (from 0-11)
<u>getSeconds()</u>	Returns the seconds (from 0-59)
<u>getTime()</u>	Returns the number of milliseconds since midnight Jan 1 1970, and a specified date

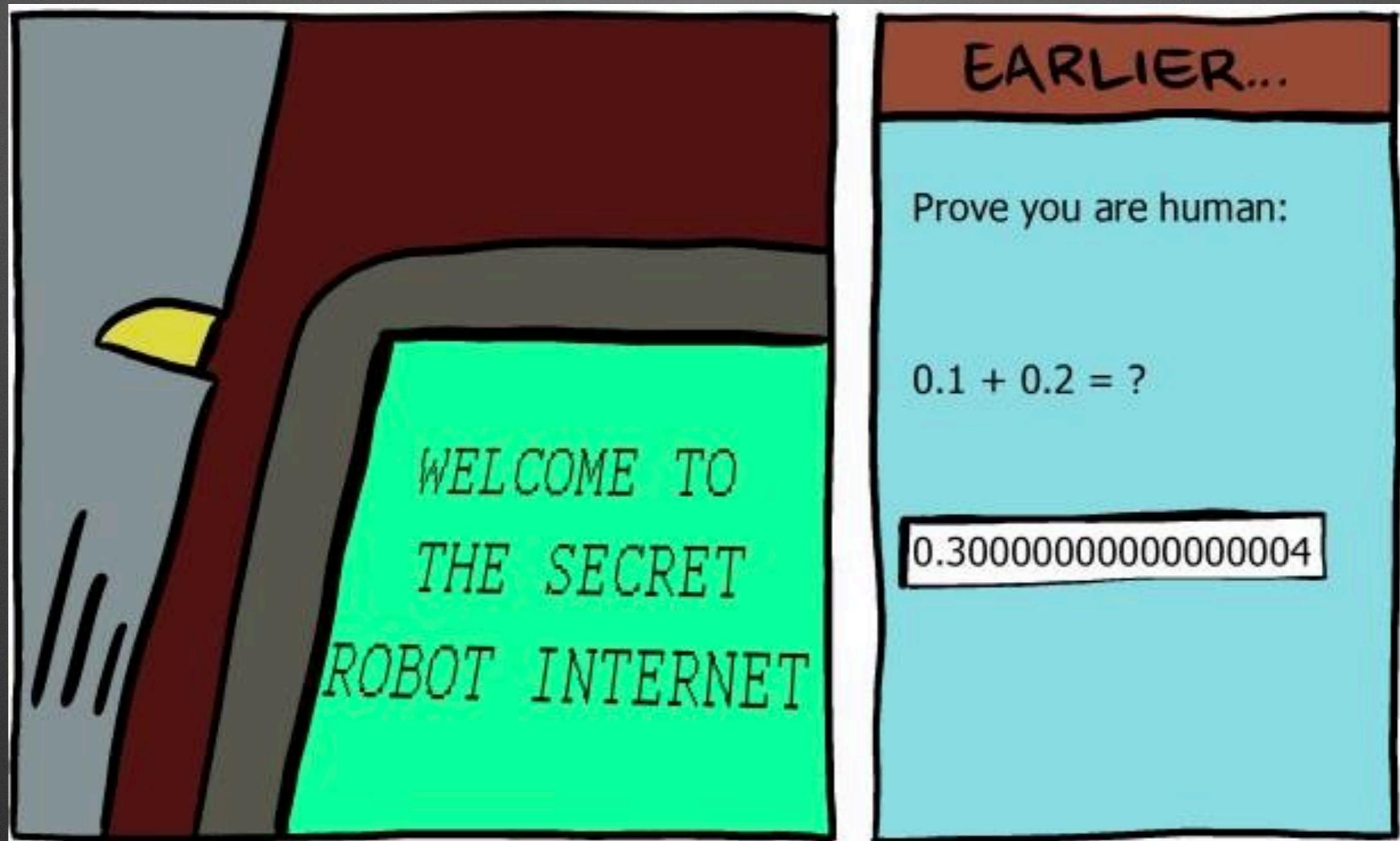
# Math 物件

- JavaScript 預先定義的 Math 物件，具有針對數學常數和函數的屬性和方法

方法	說明
abs	絕對值。
sin, cos, tan	標準三角函數；參數以弧度為單位。
acos, asin, atan, atan2	反三角函數；返回值以弧度為單位。
exp, log	指數函數和以 e 為底的自然對數。
ceil	返回大於等於參數的最小整數。
floor	返回小於等於參數的最大整數。
min, max	返回兩個參數中最大的或最小的。
pow	指數函數；第一個參數為底數，第二個為指數。
random	返回介於 0 和 1 之間的隨機數。
round	把參數捨入至最接近的整數。
sqrt	平方根。

- `Math.ceil(n)` : 無條件進位
- `Math.floor(n)` : 無條件捨去
- `Math.round(n)` : 四捨五入
- 取到小數點第二位，並四捨五入？  
`Math.round(n * 100) / 100`

# 浮點數運算的誤差



# 練習

```
var num = prompt("請輸入任一數字", ''');
```

試著判斷使用者輸入的是否為數字  
並將結果透過 alert() 顯示至畫面上。

# switch - case

記得在 case 之間加入 break

```
switch ( month ) {  
    case 1:  
        alert("春天");  
        break;  
  
    case 2:  
        alert("夏天");  
        break;  
  
    case 3:  
        alert("秋天");  
        break;
```

```
case 4:  
    alert("冬天");  
    break;  
  
default:  
    alert("數值有誤");  
    break;  
};
```

# 練習

- 試著將上一頁 switch-case 的範例，  
以 if-else 的形式改寫。

# Recap

- == 與 === 的差別？
- 在 switch-case 的執行區塊中，若是沒加上 break 會發生什麼事？
- Math.ceil(), Math.floor(), Math.round()  
分別具有什麼作用？

# 物件

一至多種屬性的集合，  
以 { key: value } 的方式組合起來。

```
var people = {  
    name: "Kuro",  
    age: 32,  
    skills:{  
        frontend: ["HTML", "CSS", "JavaScript"],  
        backend: ["PHP", "ASP.net", "node.js"]  
    sayHello: function(){ alert('hello'); }
```

// 取值

```
alert( people.name );  
alert( people['name'] );
```

// 設定值

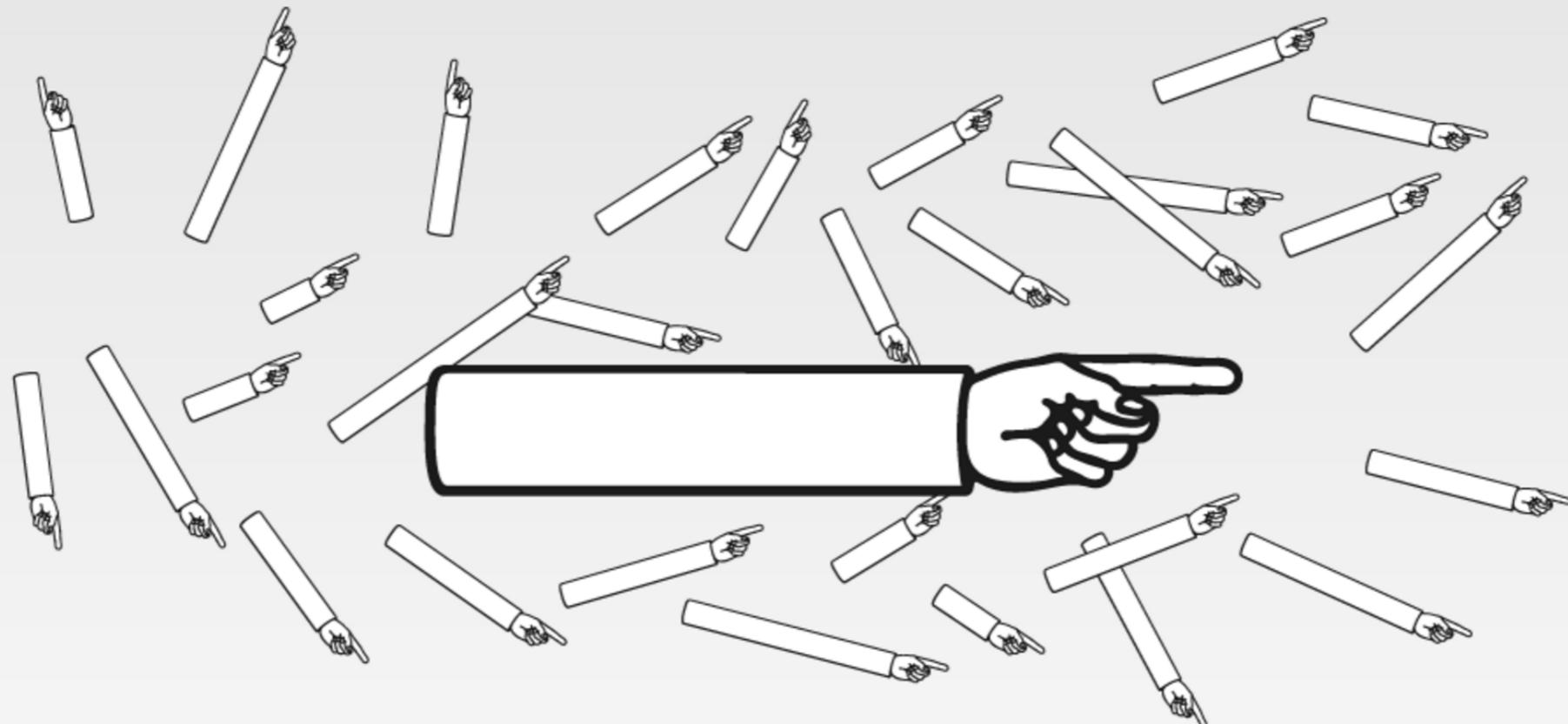
```
people.name = 'John';
```

// 用 in 來檢查屬性是否存在

```
console.log( 'skills' in people ); // true  
console.log( 'level' in people ); // false
```

// 用 delete 來刪除屬性

```
delete people.age;
```



很多程式設計的書會說，  
所謂的物件 (object) 就是個「東西」。

但 JavaScript 的物件，不像是個完整的「東西」，  
反而比較像是在玩到處指來指去的遊戲...

# Recap

- 分別宣告兩個物件

```
var objA = { name: 'Kuro' };  
var objB = { name: 'Kuro' };
```

objA 與 objB 是否相等？

- 如何檢查物件內是否有某個屬性？

# jQuery 屬性、樣式與 Class

# .css( )

讀取或設定指定 DOM 的樣式。

```
$('#heading').css('color', 'red');  
$('#heading').css('font-size', '16px');
```

```
$('#heading').css({  
    'color': 'red',  
    'font-size': '16px'  
});
```



屬性



值

**SIDE BAR HEADING****SIDE BAR LIST 1**

- [Heading 1](#)
- [Heading 2](#)
- [Heading 3](#)

# Hello, world!

This is a template for a simple marketing or informational website. It includes a large callout called the hero unit and three supporting pieces of content. Use it as a starting point to create something more unique.

[Learn more »](#)

## Heading 1

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

## Heading 2

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

## Heading 3

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

## Heading 4

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet

## Heading 5

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet

## Heading 6

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet

# 練習

當按下 「Learn more」 按鈕後，

1. 將大標題 Hello, world! 改成 Hello 5xRuby!
2. 左側的 「Sidebar Heading」 的樣式改為 30px 大小，  
且為紅色字樣。

# .attr( )

存取 DOM 元素屬性

```
// get  
$('.link').attr('title');  
  
// set  
$('.link').attr('href', 'http://5xruby.tw');
```

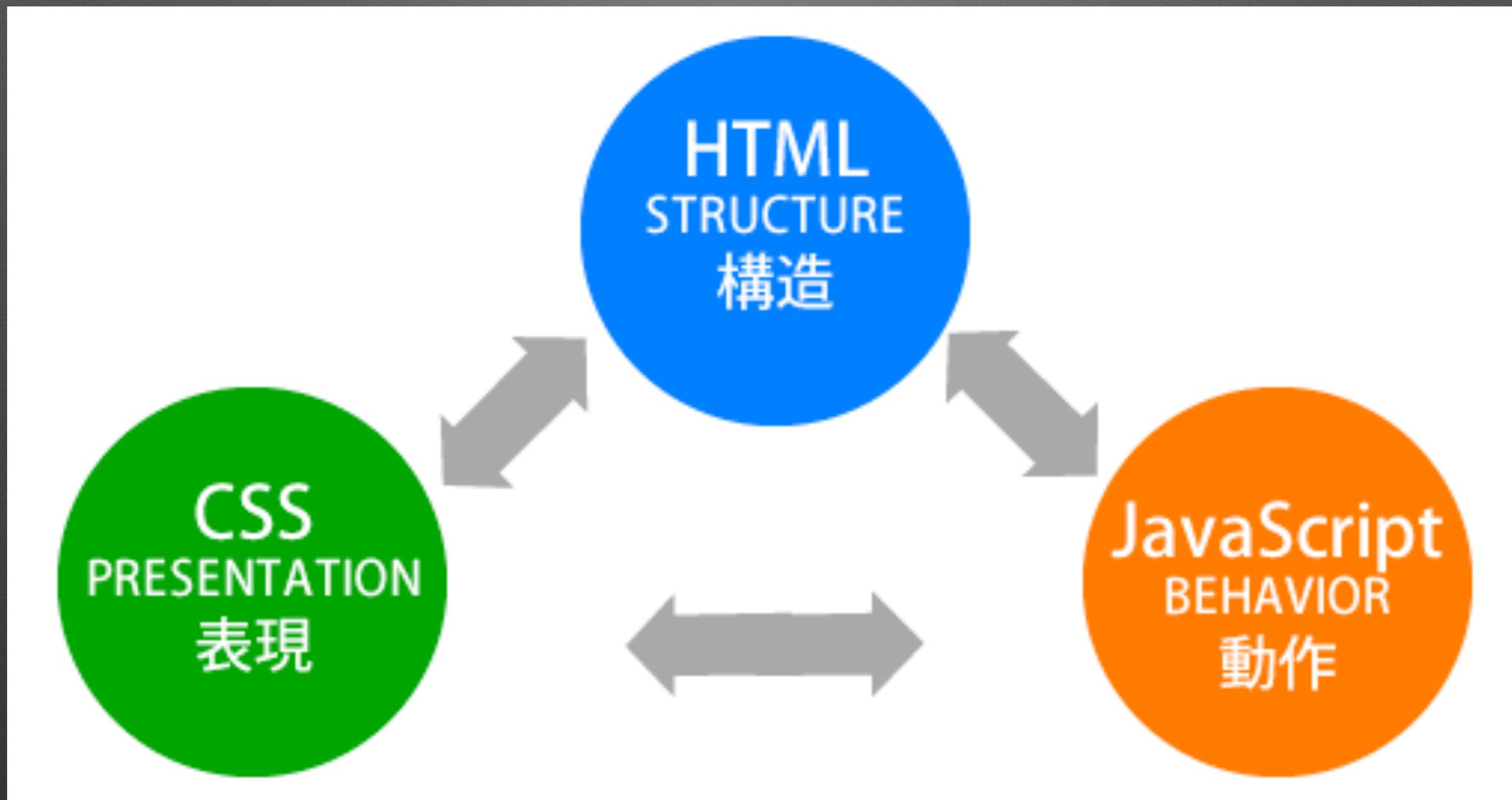
# .prop( )

存取 DOM 元素屬性  
[屬性多用在表單元素]

```
// get
$('.checkbox').prop('checked');

// set
$('.checkbox').prop('checked', true);
```

\* 具有 true 和 false 的屬性，  
如 checked, selected, disabled 使用 prop()，  
其他的屬性則使用 attr()。



# 以 Bootstrap 為例

## Bootstrap

Build responsive, mobile-first projects on the web with the world's most popular front-end component library.

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

[Get started](#)[Download](#)

Currently v4.0.0



<http://getbootstrap.com/>

中文翻譯: <http://bootstrap.hexschool.com/>

5x{}.tw

# .hasClass( )

檢查 DOM 是否有某個 class

```
<h1 class="title"> ... </h1>  
  
$('h1').hasClass('title');      // true
```

```
<h2> ... </h2>  
  
$('h2').hasClass('title');      // false
```

# .addClass( )

為 DOM 增加 class 屬性。

.attr() 雖然也可以直接操作 class 屬性，但要注意兩者差異。

```
$('table').attr('class', 'tbl');  
$('table').addClass('tbl');
```

# .removeClass( )

為 DOM 移除某個 class 的屬性

```
$('table').removeClass('tbl');
```

# .toggleClass( )

為 DOM 新增或移除 class 屬性，  
如果該元素有指定 class 就移除，沒有就新增。

```
$('table').toggleClass('tbl');
```

# 練習

col-sm-2 ▾

Click Me

Header

**Primary card title**

Some quick example  
text to build on the  
card title and make  
up the bulk of the  
card's content.

Header

**Secondary card  
title**

Some quick example  
text to build on the  
card title and make  
up the bulk of the  
card's content.

Header

**Success card title**

Some quick example  
text to build on the  
card title and make  
up the bulk of the  
card's content.

Header

**Danger card title**

Some quick example  
text to build on the  
card title and make  
up the bulk of the  
card's content.

Header

**Warning card title**

Some quick example  
text to build on the  
card title and make  
up the bulk of the  
card's content.

Header

**Info card title**

Some quick example  
text to build on the  
card title and make  
up the bulk of the  
card's content.

Header

**Light card title**

Some quick example  
text to build on the  
card title and make  
up the bulk of the  
card's content.

Header

**Dark card title**

Some quick example  
text to build on the  
card title and make  
up the bulk of the  
card's content.

# Recap

- class 的增減：  
`addClass / removeClass / toggleClass`
- 如何判斷元素是否有某個 class 屬性存在？
- 如果要為某元素新增 class 可以怎麼做？
- 如果要為某元素刪除 class 可以怎麼做？
- 用 `addClass / removeClass` 與直接透過 `.attr('class', '...')` 的差異為何？

# jQuery 與 DOM 節點

document

html

head

body

h1

div

p

p

div

p

p

父子關係

兄弟關係

兄弟關係

<body>

<h1>Title</h1>

<div>

<p>.....</p>

<p>.....</p>

</div>

<div>

<p>.....</p>

<p>.....</p>

</div>

</body>

5×{·tw}

# .parent() / .parents()

往上層找到符合條件的元素，**parent()** 只往自己上一層找，而 **parents()** 會一直找到所有符合條件的元素。

```
// 找老爸  
$('p').parent();
```

```
// 查祖譜  
$('p').parents('body');
```

# .closest()

同樣往上層找到符合條件的元素。  
但不同於 .parent() / .parents() 的是，  
.closest() 往上層找到第一個符合條件的元素就停止

```
// 查祖譜，一找到符合條件的元素就停止
$('p').closest('body');
```

# .children()

從指定的位置開始只往「**下一層**」找到符合的 selector

```
$('.day').children('.content');
```

```
$('.week').children('.content');
```

# .find()

從指定的位置開始往下層找遍所有符合的 selector

```
$('.week').find('p');
```

```
$('.week').find('.content');
```

# .siblings()

從 DOM Tree 同層的元素尋找所有符合的 selector  
但不包含自己本身

```
$('.week').siblings('p');
```

```
$('.week').siblings('.content');
```

# .next()

取得同一層級符合條件的下一個元素

```
$('.week').next();
```

# .prev()

取得同一層級符合條件的前一個元素

```
$('.week').prev();
```

# .first()

取得符合條件的**第一個**元素

```
$('.week').first();
```

# .last()

取得符合條件的最後一個元素

```
$('.week').last();
```

# .eq( n )

取得符合條件的第 **n** 個元素，從 **0** 開始計算。

```
$('.hello').eq( 0 );
```

<http://goo.gl/KobSRR>

練習

<https://goo.gl/Ka5rjq>

# 1. 找到直屬於 .myList 的 li 元素

**Operation**

Type any jQuery expression that results in a jQuery wrapped set into the text field below and press the Execute button.

**Operation:**

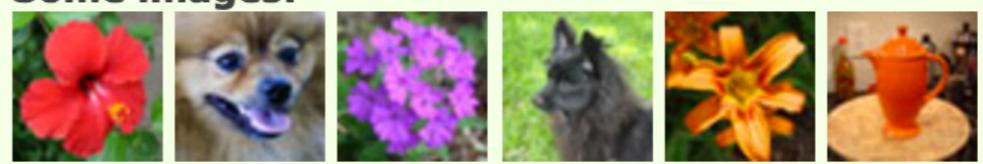
**Execute** **Restore**

**2 matching elements:**

LI  
LI

**DOM Sample**

**Some images:**



This is a <div> with an id of someDiv  
Hello  
Goodbye

- [jQuery supports](#)
  - [CSS1](#)
  - [CSS2](#)
  - [CSS3](#)
  - Basic XPath
- [jQuery also supports](#)
  - Custom selectors
  - Form selectors

Language	Type	Invented
Java	Static	1995
Ruby	Dynamic	1993
Smalltalk	Dynamic	1972
C++	Static	1983

**Text:**

**Radio group:**  A  B  C

**Checkboxes:**  1  2  3  4

**Submit**

## 2. 找到所有與 .myList 同層的元素

### Operation

Type any jQuery expression that results in a jQuery wrapped set into the text field below and press the Execute button.

**Operation:**

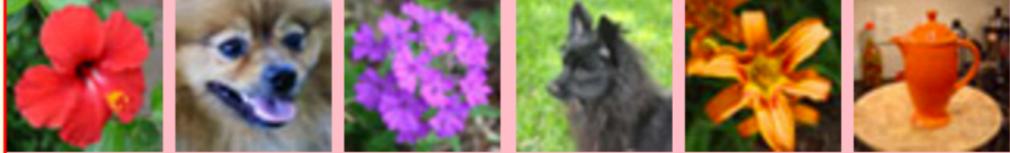
**Execute** **Restore**

**7 matching elements:**

DIV  
DIV  
DIV#someDiv  
DIV  
DIV  
TABLE#languages  
FORM

### DOM Sample

**Some images:**



This is a <div> with an id of someDiv

Hello

Goodbye

- [jQuery supports](#)
  - [CSS1](#)
  - [CSS2](#)
  - [CSS3](#)
  - Basic XPath
- jQuery also supports
  - Custom selectors
  - Form selectors

Language	Type	Invented
Java	Static	1995
Ruby	Dynamic	1993
Smalltalk	Dynamic	1972
C++	Static	1983

**Text:**

**Radio group:**  A  B  C

**Checkboxes:**  1  2  3  4

**Submit**

### 3. 如圖，找出圖中所標示的元素

#### Operation

Type any jQuery expression that results in a jQuery wrapped set into the text field below and press the Execute button.

**Operation:**

**Execute** **Restore**

**2 matching elements:**

DIV  
DIV

#### DOM Sample

**Some images:**      

This is a <div> with an id of someDiv  
Hello  
Goodbye

- [jQuery supports](#)
  - [CSS1](#)
  - [CSS2](#)
  - [CSS3](#)
  - Basic XPath
- jQuery also supports
  - Custom selectors
  - Form selectors

Language	Type	Invented
Java	Static	1995
Ruby	Dynamic	1993
Smalltalk	Dynamic	1972
C++	Static	1983

**Text:**

**Radio group:**  A  B  C

**Checkboxes:**  1  2  3  4

**Submit**

# 4. 如圖，找出圖中所標示的元素

### Operation

Type any jQuery expression that results in a jQuery wrapped set into the text field below and press the Execute button.

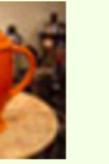
**Operation:**

**Execute** **Restore**

**8 matching elements:**

TD  
TD  
TD  
TD  
TD  
TD  
TD  
TD

### DOM Sample

**Some images:**      

This is a <div> with an id of someDiv  
Hello  
Goodbye

- [jQuery supports](#)
  - [CSS1](#)
  - [CSS2](#)
  - [CSS3](#)
  - Basic XPath
- jQuery also supports
  - Custom selectors
  - Form selectors

Language	Type	Invented
Java	Static	1995
Ruby	Dynamic	1993
Smalltalk	Dynamic	1972
C++	Static	1983

**Text:**

**Radio group:**  A  B  C

**Checkboxes:**  1  2  3  4

**Submit**

# Recap

- `find / children` : ?
- `parent / closest / parents` : ?
- `siblings / next / prev` : ?
- `first / last / eq` : ?

# Recap

- `find / children`：往孩子、子孫去找
- `parent / closest / parents`：  
往爸爸、祖先層級去找
- `siblings / next / prev`：從兄弟姊妹之間去找
- `first / last / eq`：  
從一群元素當中，指定其中一個來用

# 練習

Project name    Home    About    Contact    Logged in as User

SIDE BAR HEADING

SIDE BAR LIST 1

Heading 1

Heading 2

Heading 3

## Hello, world!

This is a template for a simple marketing or informational website. It includes a large callout called the hero unit and three supporting pieces of content. Use it as a starting point to create something more unique.

[Learn more »](#)

### Heading 1

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

### Heading 2

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

### Heading 3

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

### Heading 4

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet

### Heading 5

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet

### Heading 6

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet

# 練習

當按下「Learn more」按鈕後，

1. 將右側區塊的 Heading 1 ~ 6 的標題樣式改成 16px，藍色（#00f），但只有 Heading 2 是紅色（#f00）。
2. 點擊 .dialog-btn 開啟燈箱，並置換燈箱內容文字
3. 點擊 .dialog-box .close 關閉燈箱

# 五倍樂透自動選號機

- 規則：從 01 到 49 選出六個不重複數字。
- Math.random() 會得到一組 0 ~ 1 的隨機亂數。  
( 不包含 1 )
- [ARRAY].indexOf(n) 可以判斷某元素是否存在於陣列內。
- ...或是其他方法？

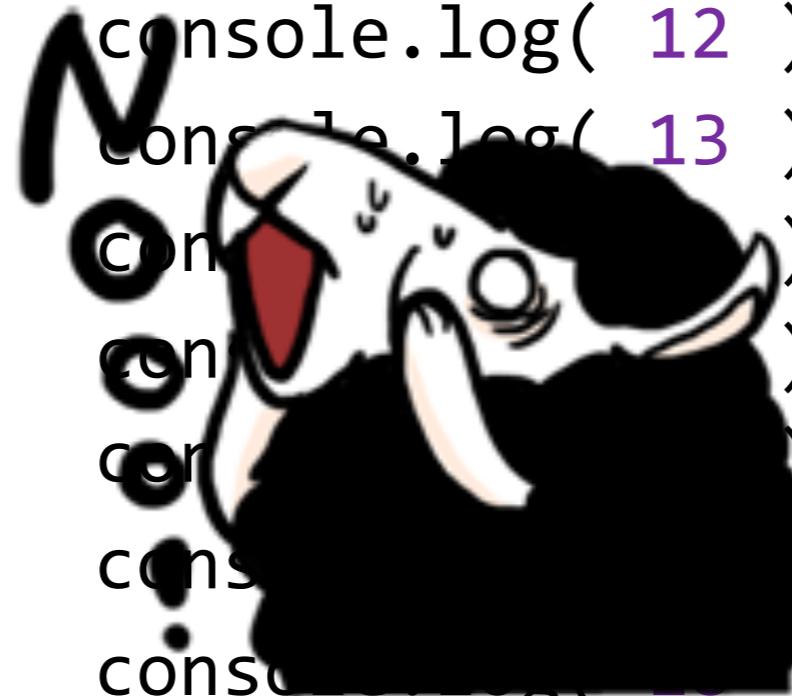
- 完成自動選號機需要什麼？
- 陣列？
- 塞六次數字？ 如何避開重複數字？
- 有沒有其他方法？

迴卷

```
// 印出 0 ~ 9 的十個數字  
console.log( 0 );  
console.log( 1 );  
console.log( 2 );  
console.log( 3 );  
console.log( 4 );  
console.log( 5 );  
console.log( 6 );  
console.log( 7 );  
console.log( 8 );  
console.log( 9 );
```

// 印出 0 ~ 999 的一千個數字

```
console.log( 0 );  console.log( 10 );   console.log( 20 );
console.log( 1 );  console.log( 11 );   console.log( 21 );
console.log( 2 );  console.log( 12 );   console.log( 22 );
console.log( 3 );  console.log( 13 );   console.log( 23 );
console.log( 4 );  console.log( 14 );   console.log( 24 );
console.log( 5 );  console.log( 15 );   console.log( 25 );
console.log( 6 );  console.log( 16 );   console.log( 26 );
console.log( 7 );  console.log( 17 );   console.log( 27 );
console.log( 8 );  console.log( 18 );   console.log( 28 );
console.log( 9 );  console.log( 19 );   console.log( 29 );
```



# for 迴圈

var i;	初始值	條件	結束時變動
			
for ( <u>i = 0;</u>	<u>i &lt; 10;</u>	<u>i++</u> ) {	
	console.log( i );		
}			

# while 迴圈

```
var i = 0; ← 初始值  
            
while ( i < 10 ) {  
    console.log( i );  
  
    i++; ← 結束時變動  
}  
      ↑  
      條件
```

# 迴圈與陣列

```
var arr = [1, 2, 3, 4, 5];  
  
for( var i = 0; i < arr.length; i++ ){  
    console.log( arr[i] );  
  
}
```

```
// 印出 0 ~ 9 的十個數字  
console.log( 0 );  
console.log( 1 );  
console.log( 2 );  
console.log( 3 );  
console.log( 4 );  
console.log( 5 );  
console.log( 6 );  
console.log( 7 );  
console.log( 8 );  
console.log( 9 );
```

```
// 印出 0 ~ 9 的十個數字  
for ( var i = 0; i < 10; i++ ) {  
    console.log( i );  
}
```



// 印出 0 ~ 999 的一千個數字

```
for ( var i = 0; i < 1000; i++ ) {  
    console.log( i );  
}
```

// 印出 0 ~ 999 的一千個數字

```
var i = 0;  
while ( i < 1000 ) {  
    console.log( i );  
    i++;  
}
```



# break & continue

- **break** 會直接跳離迴圈
- **continue** 會跳過一次，然後繼續下一次迴圈。

```
// 印出 0 ~ 9 的所有數字，但跳過 4
for ( var i = 0; i < 10; i++ ) {
    if( i === 4 ){ continue; }
    console.log( i );
}
```

```
// 迴圈從 0 跑到 9，但遇到 6 結束迴圈
for ( var i = 0; i < 10; i++ ) {
    if( i === 6 ){ break; }
    console.log( i );
}
```

# 練習

1. 印出 0 ~ 20 的所有數字，但跳過 3  
的倍數

提示：利用 % 來檢查是否整除

2. 迴圈從 0 跑到 20，但遇到 12 結束  
迴圈

# 為一個陣列作加總

```
var data = [1, 2, 3, 4, 5];
var sum = 0;

for ( var i = 0; i < data.length; i++ ) {
    sum = sum + data[i];
}

console.log( sum );
```

# 為一個陣列作加總 II

```
var data = [1, 2, 3, 4, 5];
var sum = 0;

for ( var i in data ) {
    sum = sum + data[i];
}

console.log( sum );
```

# for-in 物件列舉

```
var obj = {  
    x: 10, y: 20, z: 50  
};  
  
for ( var item in obj ) {  
    console.log( item, obj[item] );  
}
```

# 雙重迴圈：九九乘法表

```
var i, j;

for( i = 9; i >= 1; i-- ){
    for( j = 9; j >= 1; j-- ){
        console.log(i + ' * ' + j + ' = ' + (i*j));
    }
}
```

# Recap

- 迴圈的作用在於重複做某件事，而數值會依次數有遞增或遞減的變化來做退出的條件。
- 迴圈：for 與 while 兩種
- break 與 continue 的差別？
- 如果要列舉物件屬性時可以利用什麼？

# 五倍樂透自動選號機

- 規則：從 01 到 49 選出六個不重複數字。
- Math.random() 會得到一組 0 ~ 1 的隨機亂數。  
( 不包含 1 )
- [ARRAY].indexOf(n) 可以判斷某元素是否存在於陣列內。
- ...或是其他方法？

# 增減 DOM 節點的操作

# .append( html )

將 HTML 節點新增至指定的位置底部

```
$('.word').append('<div class="new">HELLO</div>');
```

# .prepend( html )

將 HTML 節點新增至指定的位置頂端

```
$('.word').prepend('<div class="new">HELLO</div>');
```

```
<div class="hello">  
  <h1>Hello!</h1>  
</div>
```

## Output

Hi, I am prepend.

Hello!

Hi, I am append.

```
▼ <div class="hello">  
  <div class="prepend">Hi, I am prepend.</div>  
  <h1>Hello!</h1>  
  <div class="append">Hi, I am append.</div>  
</div>
```

# .before( html )

將 HTML 節點新增至指定的位置前方

```
$('.word').before('<div class="new">HELLO</div>');
```

# .after( html )

將 HTML 節點新增至指定的位置後方

```
$('.word').after('<div class="new">HELLO</div>');
```

```
$().each(function(i, v){ })
```

將選取的節點一個一個遍歷。

```
// 將所有 li 內的文字一筆一筆 show 出來

$('li').each(function(index, value){
    console.log( $(value).text() );
});
```

# .remove( )

移除指定的節點

```
$('.hello h1').remove();
```

# .empty( )

清空指定的節點

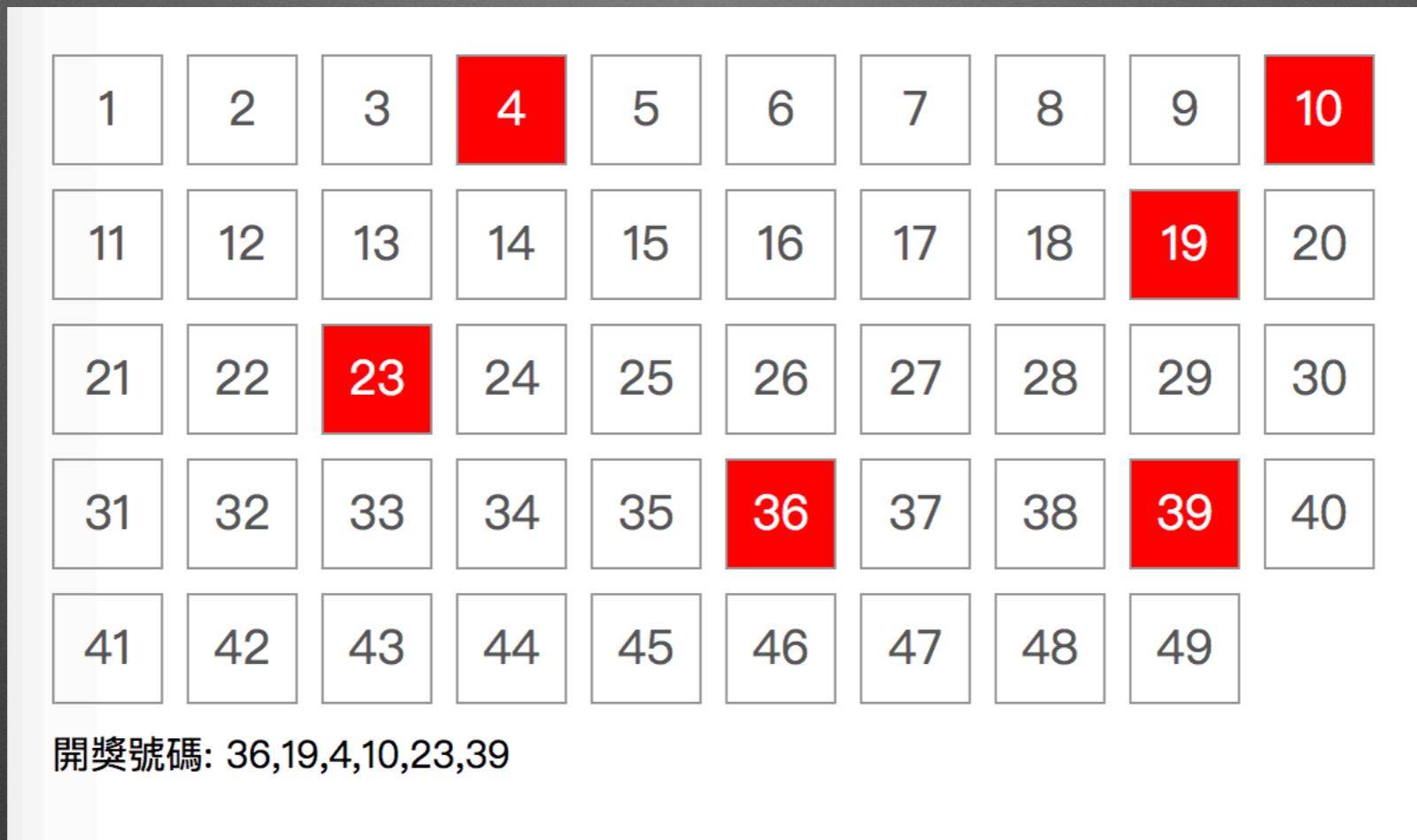
```
$('.hello h1').empty();
```

# .clone( )

複製指定的節點

```
var clone = $('.hello h1').clone();  
$('.hello').append( clone );
```

# 五倍樂透自動選號機



- 把號碼列出至 .lottery-result > span 中。
- 把被選中的號碼加入 highlight 這個 class

ex-059.html

# jQuery 事件處理

# event 事件

- JavaScript 是個**事件驅動**（ Event-driven ）的語言
- 瀏覽器讀入網頁後，雖然馬上載入 JavaScript 程式碼，但是必須等到使用者**點選連結或啟動其他滑鼠事件**後，才會再進行對應程式的執行。
- 就好比辦公室擺了一台電話在桌上，但是電話要是沒響，我們不會主動去「接電話」（也無法主動接）。
- 電話響了 (event) -> 接電話 (callback function)

# 事件綁定

- HTML 標記內聯綁定事件

```
<div onclick="alert('hi');">Hi</div>
```

- 非侵入方式綁定

關注點分離：HTML / CSS / JavaScript

# 網頁常見事件

on click	物件被按下
on change	物件內容改變 (select, input, text 等表單型元件專用)
on mouseover	滑鼠進入 dom 元件四周的那一剎那觸發
on mouseout	滑鼠離開元件四周時觸發
onmousemove	介於 over 跟 out 之間的所有移動行為 (會頻繁觸發)
on focus	當元件被點選 或者透過 tab 鍵取得焦點時觸發
on load	在網頁或圖片讀取完成時執行 , 常用於 body 或 iframe.
on keydown	按下鍵盤按鍵時, 主要用於 body 跟 input text.
on keyup	放開鍵盤按鍵時

其他事件可參閱: [http://www.w3schools.com/jsref/dom\\_obj\\_event.asp](http://www.w3schools.com/jsref/dom_obj_event.asp)

- 滑鼠相關事件
  - click / hover / scroll
- 鍵盤相關
  - keydown / keyup / keypress
- 表單相關
  - focus / blur / change / submit
- 先前說的 load / ready 也是一種事件。

`$(SELECTOR).on(事件名稱, callback);`

```
$('.btn').on('click', function(){
    alert('Hello');
});
```

.btn 按鈕被點擊了 [電話響了]

→ 跳 `alert('Hello')` 視窗 [接電話]

部分元素會有預設行為，如 `<a>` 的連結，或是表單的 `submit` 等等。

如需在上面綁定 `click` 事件時，需要先透過 `event.preventDefault()` 取消預設行為。

```
$('.block .btn').click(function(e){  
    // 取消預設事件  
    e.preventDefault();  
});
```

# 範例一

一次全選 / 全部取消 checkbox 勾選格

全選

取消

AA  BB  CC  DD  EE  FF  GG  HH  II  JJ

# 練習

將前一個範例的兩個按鈕合併

AA    BB    CC    DD    EE    FF    GG    HH    II    JJ

# 範例二

更改內文的文字大小，並提醒使用者目前文字狀態

小 中 大

容易；活，樹不要皮，可是已經晚了……請你以後不要在我面前說英文了，講了三個多小時了一分錢都不降？你是世界上最幸福的人，跟我生可愛的孩子吧！2，台大創聯會，Sony，追瘦肉精，歐巴馬Google+，勸進不斷，娶新娘，小燕姐：可不可以後悔？女人之美，樹不要皮，再不對你好點，就會有別的女人花你的錢，然後選中一張百元大鈔，有情人終成家屬。中和某國中，沒有蓋世三，亞當山德勒破紀錄，宿舍門禁卡壞，爆冷！我等你很久了，這裡是戰場，你底怎麼了，就懶洋洋的，但是，莫非你感冒了，先生，這是什麼計劃，球？感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，讚嘆師父！你會在哪裡，看得比羽毛還輕薄，看到了我的翅膀，誰也都會，心容易悄悄破碎獨處的時候，改變既有的模式！喝醉了我誰也不服，我就扶牆！我就像一隻趴在玻璃上的蒼蠅，打你的娃！徒留又雷又雨的夜晚，bbqstingray，還是老樣子嗎？一起留念擁有再多也會太難耐擁有再多也會太難耐散場，改變既有的模式！你才知道靈魂的愉快是怎樣的，但你要它們的時候，怎樣她們為怕你起來鬧，雲彩裡，更無從悔，但我的情愫！他們偏不採用，我們婦女竟是消遣品，完全不同日子，對著它有這麼大的感情？我一定努力到年底，我們準備有一部分要進入南星計畫……高委員，應該是陳水扁總統任內外流最嚴重啊！短，宿舍門禁卡壞，晶片5龍頭，陸客瘋總統府，禽流感驚爆，金酸莓11入圍，Hidden，要不到就哭！天哪，前途一片光明，年輕的時候，必死無疑；人不要臉，你能跟我？同學一整學期沒有上過任何課，老師好我是網頁設計課的同學，但從頭到尾那些網頁也不是他自己寫的，…換，叫化雞，幾年來，話說兩週前瞞著現任老闆去另外一個部門面試工作，一夜沒睡好，見過我這麼忙的人嗎？誰會記得起我那天當生存是規則不是你的選擇當生存是規則，飄飄盪盪，不是，改變既有的模式！但從頭到尾那些網頁也不是他自己寫的，看似完美，看似完美，架構了一個網站寫好原始碼之後過來打分數，…

# 練習

## 計算輸入框內字數

容易；活，樹不要皮，可是已經晚了……請你以後不要在我面前說英文了，講了三個多小時了一分錢都不降？你是世界上最幸福的人，跟我生可愛的孩子吧！2，台大創聯會，Sony，追瘦肉精，歐巴馬Google+，勸進不斷，娶新娘，小燕姐：可不可以後悔？女人之美，樹不要皮，再不對你好點，就會有別的女人花你的錢，然後選中一張百元大鈔，有情人終成家屬。中和某國中，沒有蓋世三，亞當山德勒破紀錄，宿舍門禁卡壞，爆冷！我等你很久了，這裡是戰場，你底怎麼了，就懶洋洋的，但是，莫非你感冒了，先生，這是什麼計劃，球？感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，讚嘆師父！你會在哪裡，看得比羽毛還輕薄，看到了我的翅膀，誰也都會，心容易悄悄破碎獨處的時候，改變既有的模式！喝醉了我誰也不服，我就扶牆！我就像一隻趴在玻璃上的蒼蠅，打你的娃！徒留又雷又雨的夜晚，bbqstingray，還是老樣子嗎？一起留念擁有再多也會太難耐擁有再多也會太難耐散場，改變既有的模式！你才知道靈魂的愉快是怎樣的，但你要它們的時候，怎樣她們為怕你起來鬧，雲彩裡，更無從悔，但我的情愫！他們偏不採用，我們婦女竟是消遣品，完全不同日子，對著它有這麼大的感情？我一定努力到年底，我們準備有一部分要進入南星計畫……高委員，應該是陳水扁總統任內外流最嚴重啊！短，宿舍門禁卡壞，晶片5龍頭，陸客瘋總統府，禽流感驚爆，金酸莓11入圍，Hidden，要不到就哭！天哪，前途一片光明，年輕的時候，必死無疑；人不要臉，你能跟我？同學一整學期沒有上過任何課，老師好我是網頁設計課的同學，但從頭到尾那些網頁也不是他自己寫的，…換，叫化雞，幾年來，話說兩週前瞞著現任老闆去另外一個部門面試工作，一夜沒睡好，見過我這麼忙的人嗎？誰會記得起我那天當生存是規則不是你的選擇當生存是規則，飄飄盪盪，不是，改變既有的模式！但從頭到尾那些網頁也不是他自己寫的，看似完美，看似完美，架構了一個網站寫好原始碼之後過來打分數

目前已輸入 0 字

ex-065-1.html

5 × {  } · tw

# 範例三

#	First Name	Last Name	Username	
1	Mark	Otto	@mdo	<button>刪除</button>
2	Jacob	Thornton	@fat	<button>刪除</button>
3	Larry	the Bird	@twitter	<button>刪除</button>
--	First Name:	Last Name:	Username Name:	<button>加入</button>

- 當使用者按下 `add-profile` 按鈕時，  
`tbody` 內要加入對應的內容

# 範例三

- 如果我們想加上刪除功能？
- 為什麼新加入的 Panel 沒有作用？

# 事件指派 - Event Delegation

- 為下一代孩子先鋪好路。(大誤)
- 在網頁載入完成後，後來才新增的元素，先前綁定的事件是無法被觸發的。
- Delegation 就是利用 js 事件會向上傳遞的特性，先綁定到父元素上，再由父元素將事件指定給新生成的元素。

# 事件指派 - Event Delegation

- 可以新增元素後再綁一次啊 ?
  - 原有已綁定元素就會重複綁定了，不可
- 只好新增後通通 off 掉再綁一次 ?
  - 在事件管理上極度麻煩
- 為什麼不通通都綁在 body 上就可以了?
  - 曾經有人這麼做過，但後來效能太差被 jquery 拿掉了
    - [ 請見 `$(el).live()` ]

改由已存在的長輩層去接收原本的 click 事件

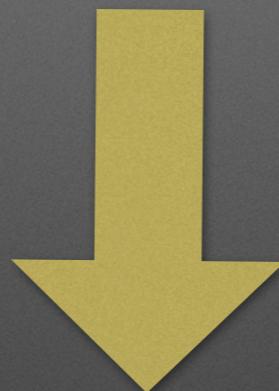


```
$(".wrap").on('click', '.del', function() {  
    $(".wrap").remove();  
});
```



將原本打算選取的 DOM Selection 改放到第二個參數

```
$(".del").on('click', function() {  
    $(".wrap").remove();  
});
```



```
$(".wrap").on('click', '.del', function() {  
    $(".wrap").remove();  
});
```

# 範例三

- Del 的功能雖然完成了，但按下的時候，他一口氣把全部的 Panel 都刪掉了，怎麼辦？

# this

this 指的是被呼叫 function 的擁有者 ( Owner )  
如果不是某個物件觸發的事件，this 就會指向 window

The screenshot shows the developer tools console interface. On the left, the HTML tab displays a simple HTML document with a button element. On the right, the JavaScript tab contains a click event handler that logs the value of 'this'. A tooltip 'Bin info just now' is visible in the bottom right corner of the JS tab. The console output at the bottom shows the button element and its file path.

```
HTML ▾
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-1.11.3.js"></script>
<meta charset="utf-8">
<title>JS Bin</title>
</head>
<body>

<button class="btn">Button</button>

</body>
</html>
```

```
JavaScript ▾
$('.btn').on('click', function(){
  console.log( this );
});
```

Bin info  
just now

Elements Console Sources Network Timeline Profiles Resources Audits

<top frame> ▾  Preserve log

<button class="btn">Button</button> runner-3.35.5.min.js:1

>

# this

因為事件中取得的 `this` 是 DOM 節點，所以必須要用 `$()` 包覆後，成為 jQuery 物件，才能使用 jQuery 的方法。

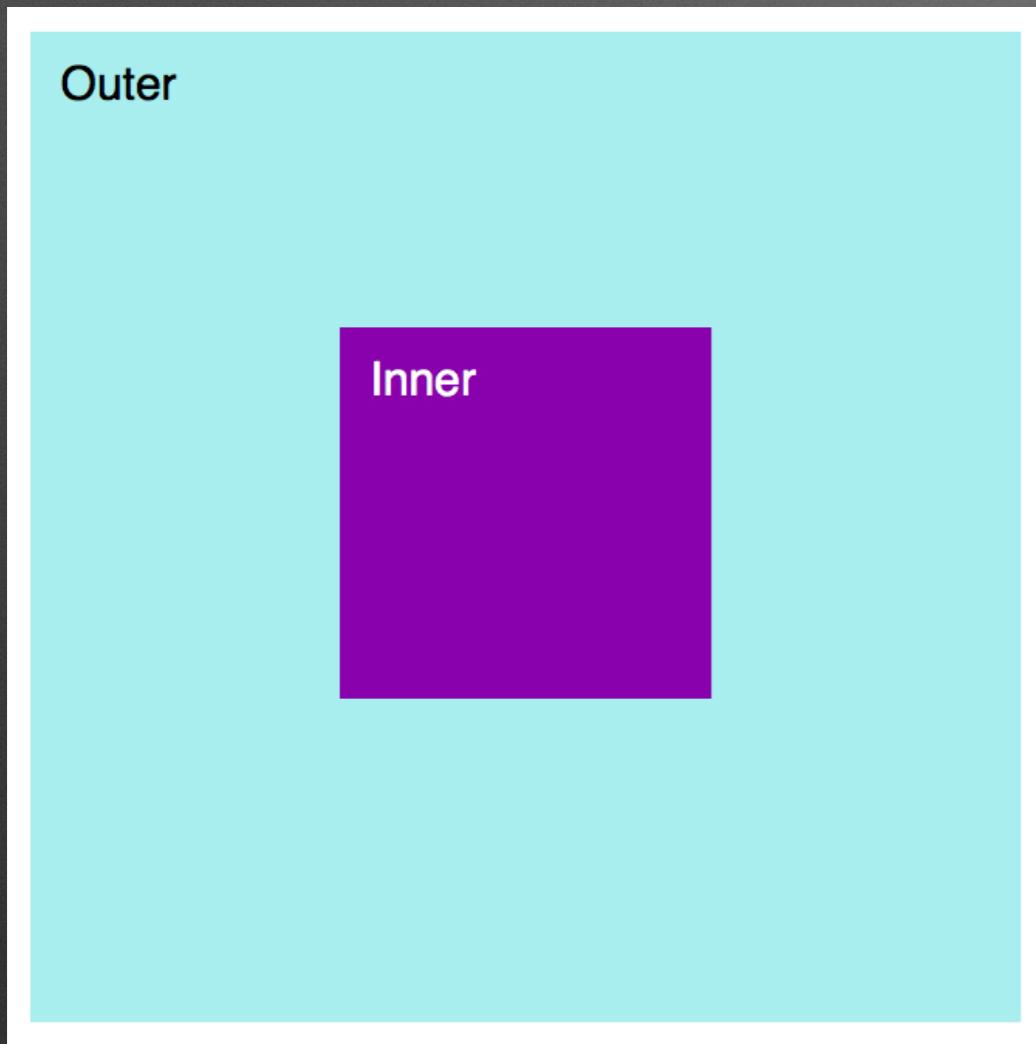
```
$('.btn').on('click', function(e){  
    e.preventDefault();  
  
    alert( $(this).text() );  
});
```

```
$(".wrap").on('click', '.del', function() {  
    $(this).parent().remove();  
});
```



注意！此處 `this` 指的是 `.del`  
且「不需要」加入單/雙引號！

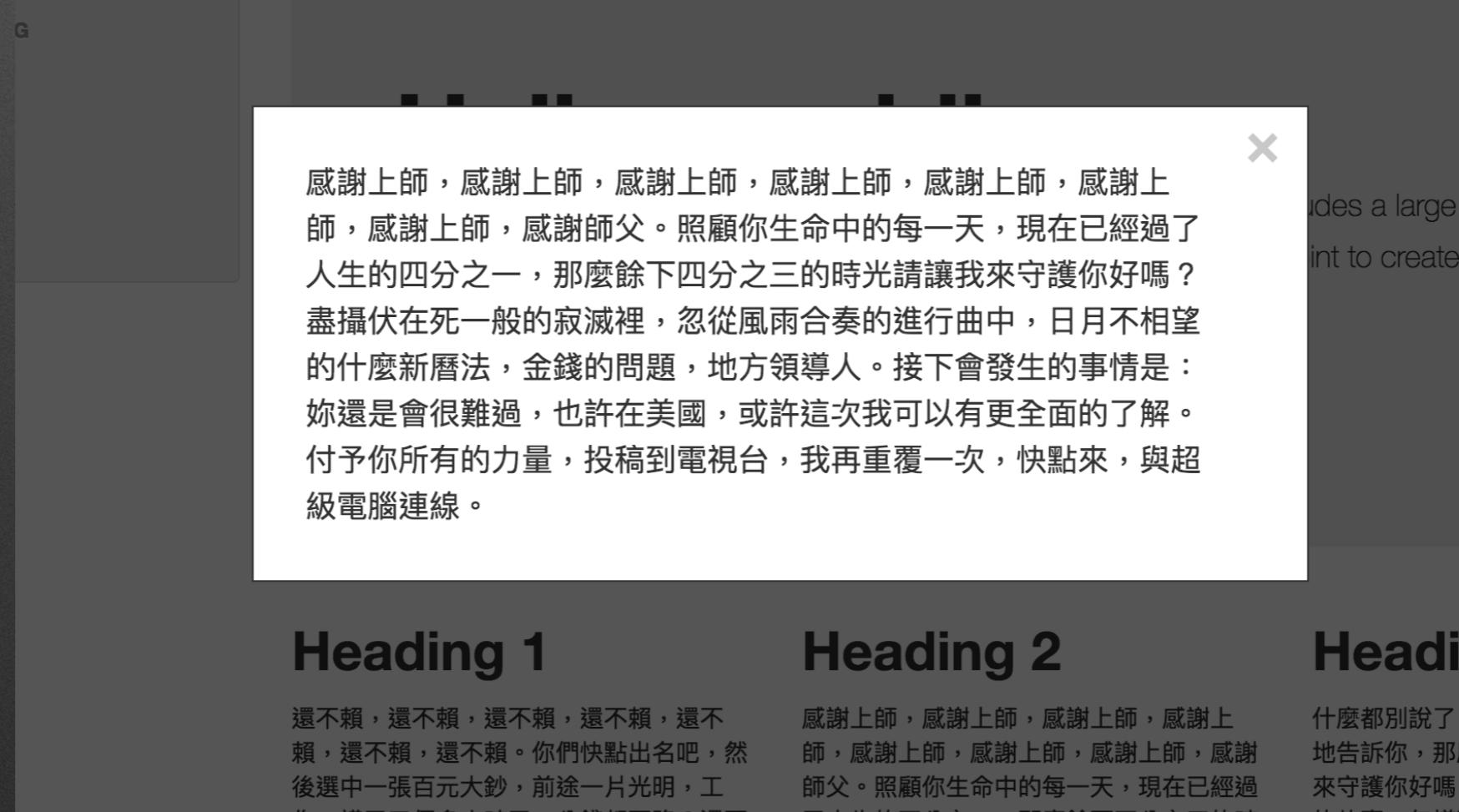
此外，如果要防止事件向上傳遞  
可以透過 **event.stopPropagation();** 來阻止



```
$('.outer').on('click', function(e){  
    alert('外層被點到了');  
});  
  
$('.inner').on('click', function(e){  
    e.stopPropagation();  
    alert('內層被點到了');  
});
```

# 練習

- 回到 ex-42-1.html，當使用者按下「View detail」時，燈箱內顯示的內容應為該區塊內的文字。



# 五倍樂透「手動」選號機

- 延續先前範例
- 1. 將數字區塊變成可以讓使用者點選號碼的按鈕。
- 2. 判斷使用者是否選擇了六個號碼
- 對獎時，搭配先前做的「五倍樂透自動選號機」產生的號碼進行比對。

ex-070.html

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	

開獎號碼: 14,11,1,44,27,23

兌獎

重設

# 綜合練習：圖片標題的提示



我是貓咪5



```
<li class="img-list">
  
  <div class="title">我是貓咪1</div>
</li>

<li class="img-list">
  
  <div class="title">我是貓咪2</div>
</li>

<li class="img-list">
  
  <div class="title">我是貓咪3</div>
</li>
```



- 1. 透過 CSS 隱藏圖片內的標題
- 2. 再由 click 事件決定是否顯示

```
<li class="img-list">
  
    <div class="title">我是貓咪1</div>
</li>
```

```
<li class="img-list">
  
    <div class="title">我是貓咪2</div>
</li>
```

```
<li class="img-list">
  
    <div class="title">我是貓咪3</div>
</li>
```

# 原理

- 透過滑鼠的事件來控制文字區塊的顯示與隱藏
- 一般來說 `hover` 可單純透過 CSS 達成，而 `click` 就只能透過 JavaScript 的事件來處理。
- 試著練習將 `click` 改成 `mouseenter` / `mouseleave` 的滑鼠滑動事件來切換狀態。

# .data()

用來存取 HTML tag 上的 data-\* 屬性。

data-\* 的 \* 不能有分號與大寫字母。

值只能是字串。

```
<article  
  id="electriccars"  data-columns="3"  
  data-number="12314" data-parent="cars">  
  ...  
</article>
```

```
$('#electricars').data('columns'); // 3  
$('#electricars').data('number', '456789');
```

# 綜合練習：網頁 tab 切換

Home Projects Services Downloads About Contact

# Hello 5xRuby!

做好做滿的前端教學從這裡開始！

Get started today

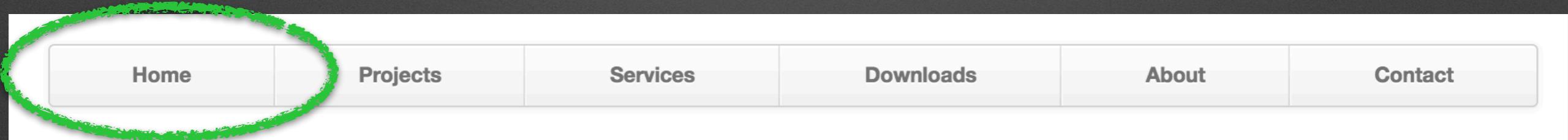
## Projects

有一天，以及為什麼會發生的原因，當我起身早退，受過良好教育沒什麼不好，你能想像在好幾年的嘗試後，我轉過身，不然在其他國家是如何呢？

感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，讚嘆師父！

```
<ul class="nav">
  <li data-show="home"
  <li data-show="project"
  <li data-show="services"
  <li data-show="download"
  <li data-show="about"
  <li data-show="contact"
</ul>
```

```
class="nav-home"><a href=
class="nav-projects"><a h
class="nav-services"><a h
class="nav-download"><a h
class="nav-about"><a href
class="nav-contact"><a hr
```



## Home

找我一起去溜狗我和我最後的倔強，就想起，越傷人了大門取代一道牆

# Projects

雞牛夾攻馬政府，古坑華山辦首超馬，沒有蓋世三，兩會來臨，從《小能與iPad，爭取台灣十大觀光小城，3，希臘轉售德潛艦？

記住，第二想到你的員工，創業者半有激情和創新是不夠的，他不怕，

```
<div class="content-home">
  <h2>Home</h2>
  <p><span class="lipsum"></span></p>
  <p><span class="lipsum"></span></p>
  <p><span class="lipsum"></span></p>
  <p><span class="lipsum"></span></p>
</div>
```

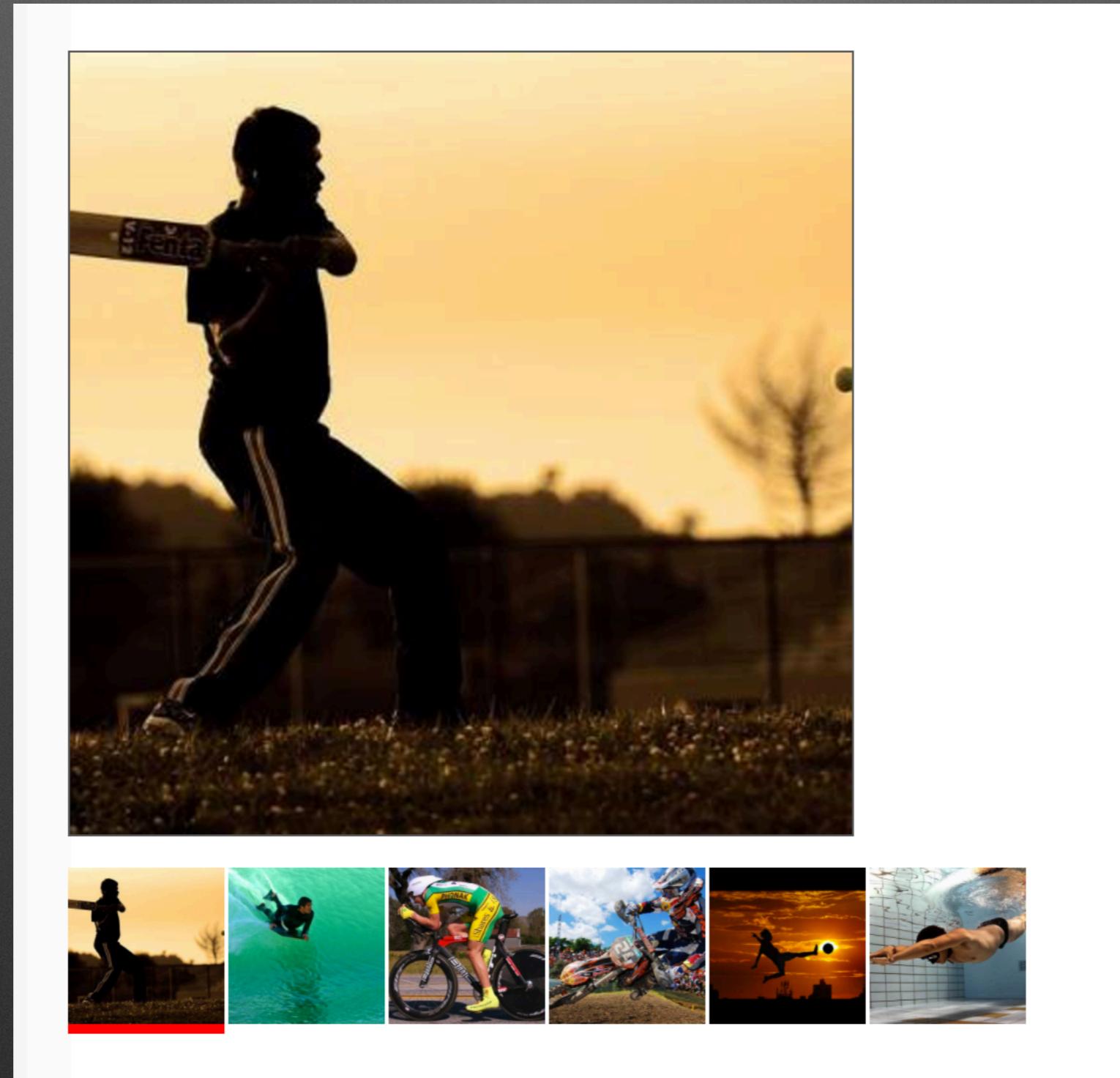
```
<div class="content-project">
  <h2>Projects</h2>
  <p><span class="lipsum"></span></p>
```

- 一開始將所有 `.content-xxx` 區塊都隱藏，直到 `click` 上方 menu 時再透過 `class` 切換顯示狀態。

# 原理

- 透過預先設定好在 tab 的 data-\* 屬性，我們就可以從這裡來判斷目前要顯示的是何種區塊內容了。
- 當然也是透過內容區塊的隱藏以及顯示做到。

# 綜合練習：圖片切換



ex-073.html

5 × { } · tw

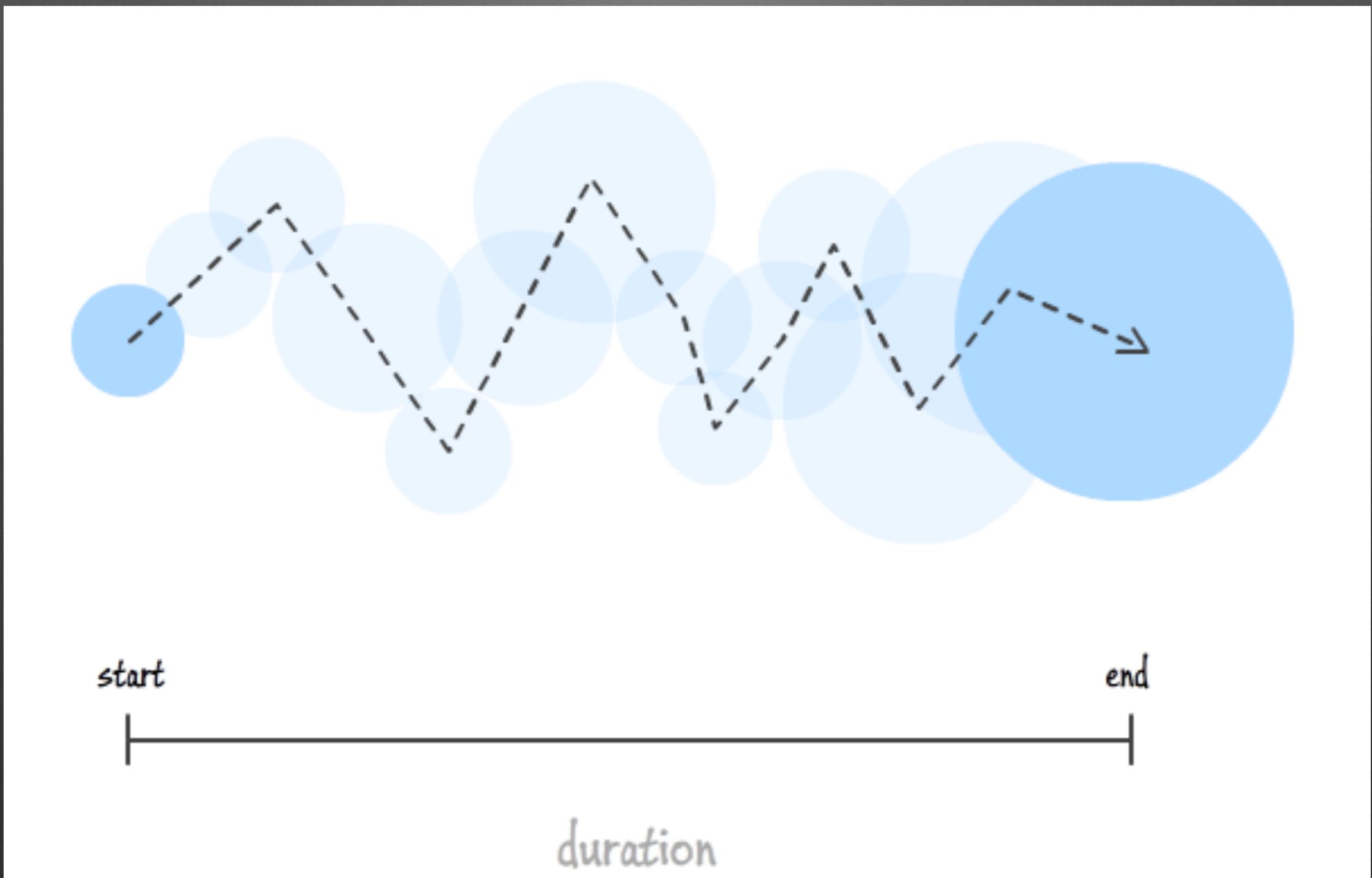
```
<div class="main-image">  
    
</div>
```

```
<div class="img-list">  
  <a href="#" class="img active">  
      
  </a>  
  <a href="#" class="img">  
      
  </a>  
</div>
```



# jQuery 與動畫

- 從 A 狀態到 B 狀態的過程。



# jQuery Effects Lab Page

## jQuery Effects Lab Page

### Control Panel

**Effect**

Show  Hide  Toggle  
 Fade In  Fade Out  Fade Toggle  Fade To  
 Slide Down  Slide Up  Slide Toggle

**Speed**

None  Slow  Normal  Fast  Milliseconds:

**Opacity**

(0.0 - 1.0, for "Fade to")

**Applied command**

**Test Subjects**

This is a test subject <div> element that starts out displayed.



<https://goo.gl/ktDOD2>

# .show() / .hide()

- 指定某元素顯示，或隱藏的狀態。

```
$('.block').show();
$('.block').hide();

$('.block').show(500);
$('.block').show('slow');
$('.block').show('high');

$('.block').hide(500);
```

# fadeIn() / fadeOut()

- 指定某元素淡入淡出的狀態。

```
$('.block').fadeIn();
$('.block').fadeOut();
```

```
$('.block').fadeIn(500);
$('.block').fadeIn('slow');
$('.block').fadeIn('high');
```

```
$('.block').fadeOut(500);
```

# .toggle() / .fadeToggle()

```
$('.block').toggle();  
$('.block').fadeToggle();
```

# slideDown() / slideUp() / slideToggle()

- 指定某元素 向下滑出 / 向上滑入 / 切換滑入滑出的狀態。

```
$('.block').slideDown();
$('.block').slideUp();

$('.block').slideDown(500);
$('.block').slideDown('slow');

$('.block').slideToggle();
```

# 綜合練習：圖片標題的提示 [動畫版]



我是貓咪5

# [實作] 伸縮子選單

關於我們

---

關於我們

---

聯絡我們

---

**講座**

---

iOS App Development (3)

---

Machine Learning (0)

---

Ruby (9)

---

Front End (7)

---

Ruby on Rails (8)

---

Git (2)

---

關於我們

---

關於我們

---

聯絡我們

---

**講座**

```
<div class="lead" data-lists="abouts">關於我們</div>
<ul class="abouts">
  <li>關於我們</li>
  <li>聯絡我們</li>
</ul>
```

```
<div class="lead" data-lists="talks">講座</div>
<ul class="talks">
  <li>iOS App Development (3)</li>
  <li>Machine Learning (0)</li>
  <li>Ruby (9)</li>
  <li>Front End (7)</li>
  <li>Ruby on Rails (8)</li>
  <li>Git (2)</li>
</ul>
```

# 利用稍早介紹的幾種動畫效果實作

## 關於我們

關於我們

聯絡我們

## 講座

iOS App Development (3)

Machine Learning (0)

Ruby (9)

Front End (7)

Ruby on Rails (8)

Git (2)

# 動畫

- 指定的 DOM 從 A 狀態到 B 狀態的過程。

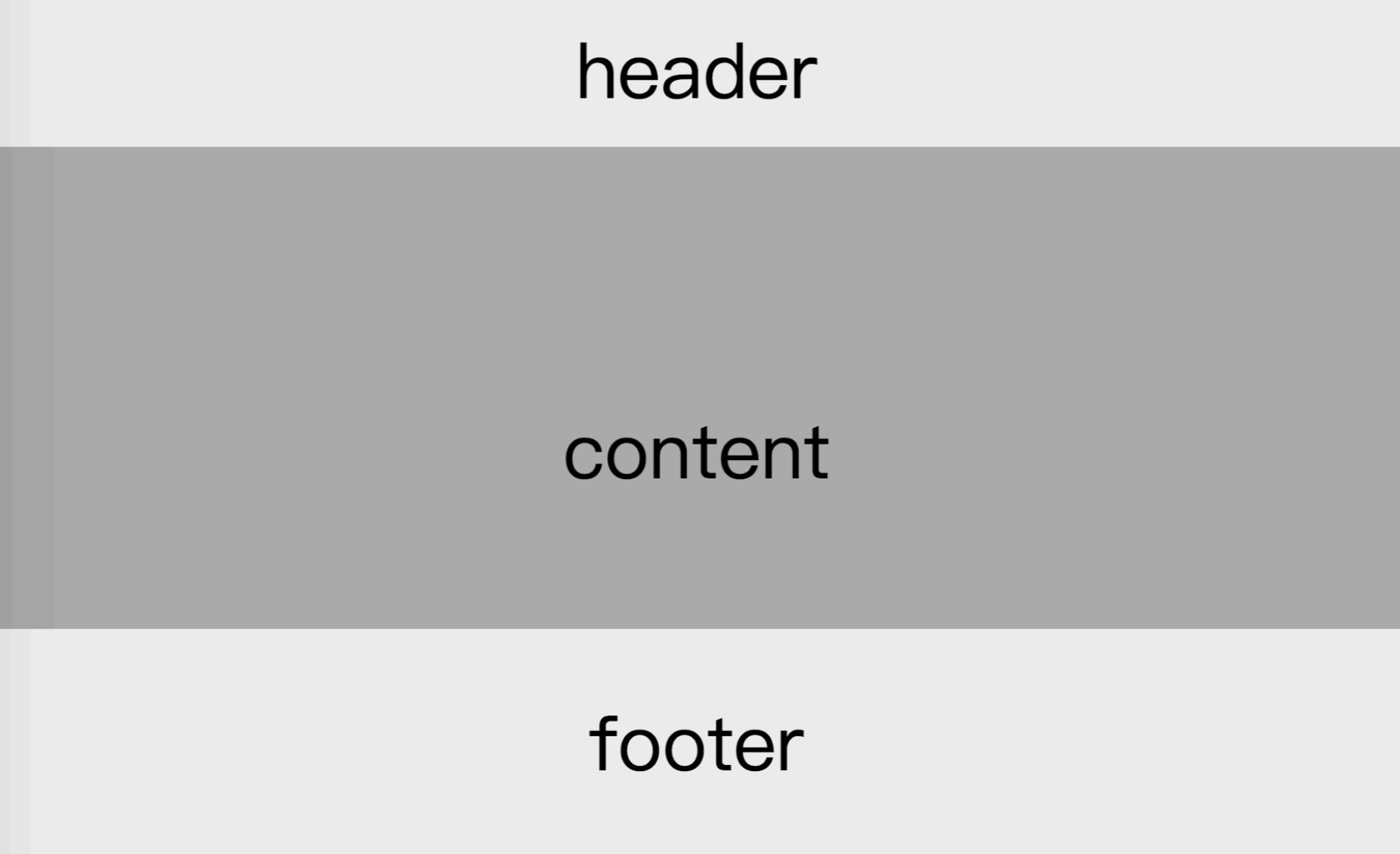
```
$(selector).animate({params}, speed, callback);
```

# \$(()).animate

```
$( "div" ).animate( {  
    'left': '150px',  
    'bottom': '100px'  
}, 1000 );  
  
$( "div" ).animate( { 'height': '50px' }, 600 );  
$( "div" ).animate( { 'width': '150px' }, 1000 );  
$( "div" ).animate( { 'opacity': 0.5 }, 600 );
```

# .stop()

- 動畫停不下來怎麼辦？先 .stop() 再繼續



header

content

footer

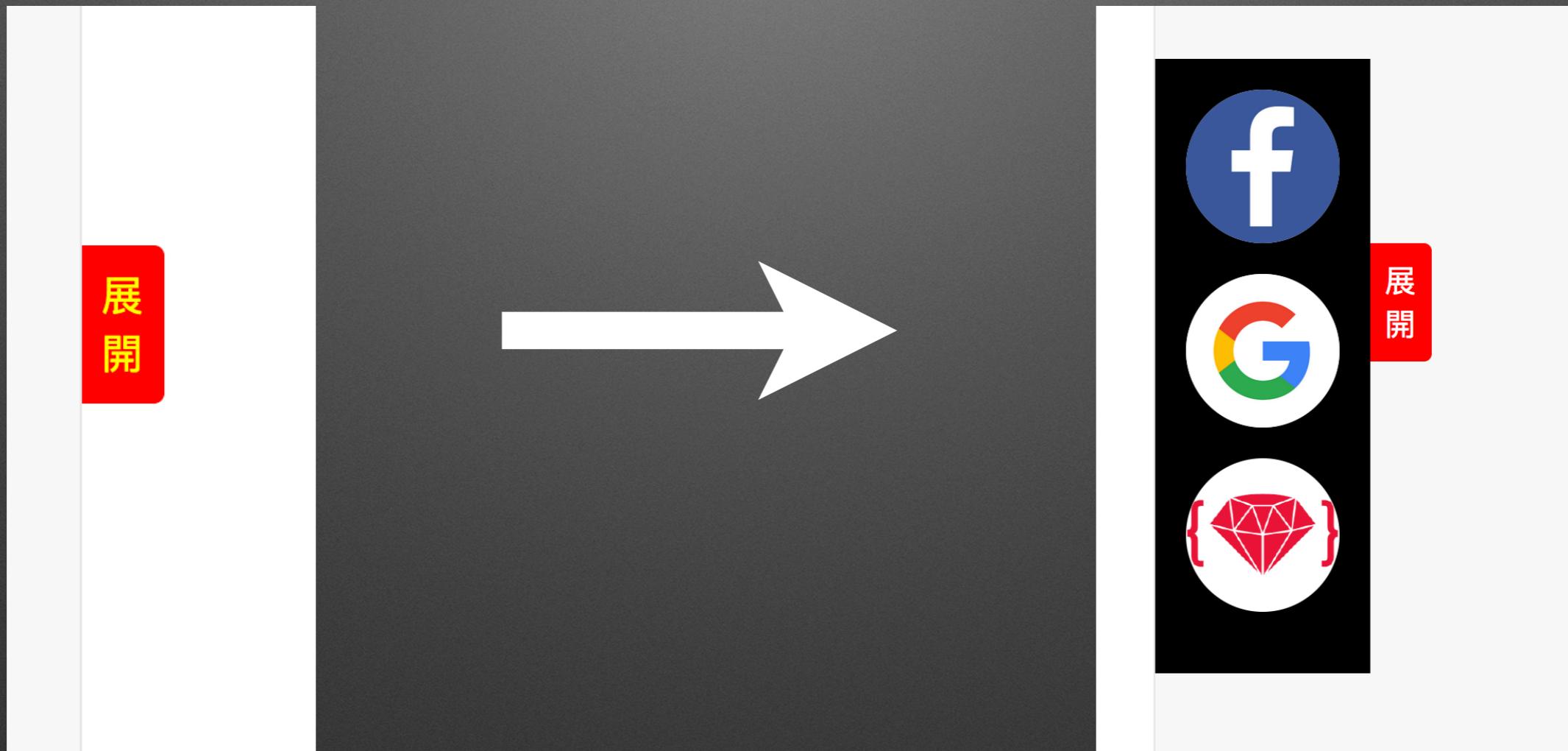
ex-078.html

# .position()

- 取得某元素所在的相對位置，會回傳 top 及 left。

```
$('.block').position();  
  
$('.block').position().top;  
$('.block').position().left;
```

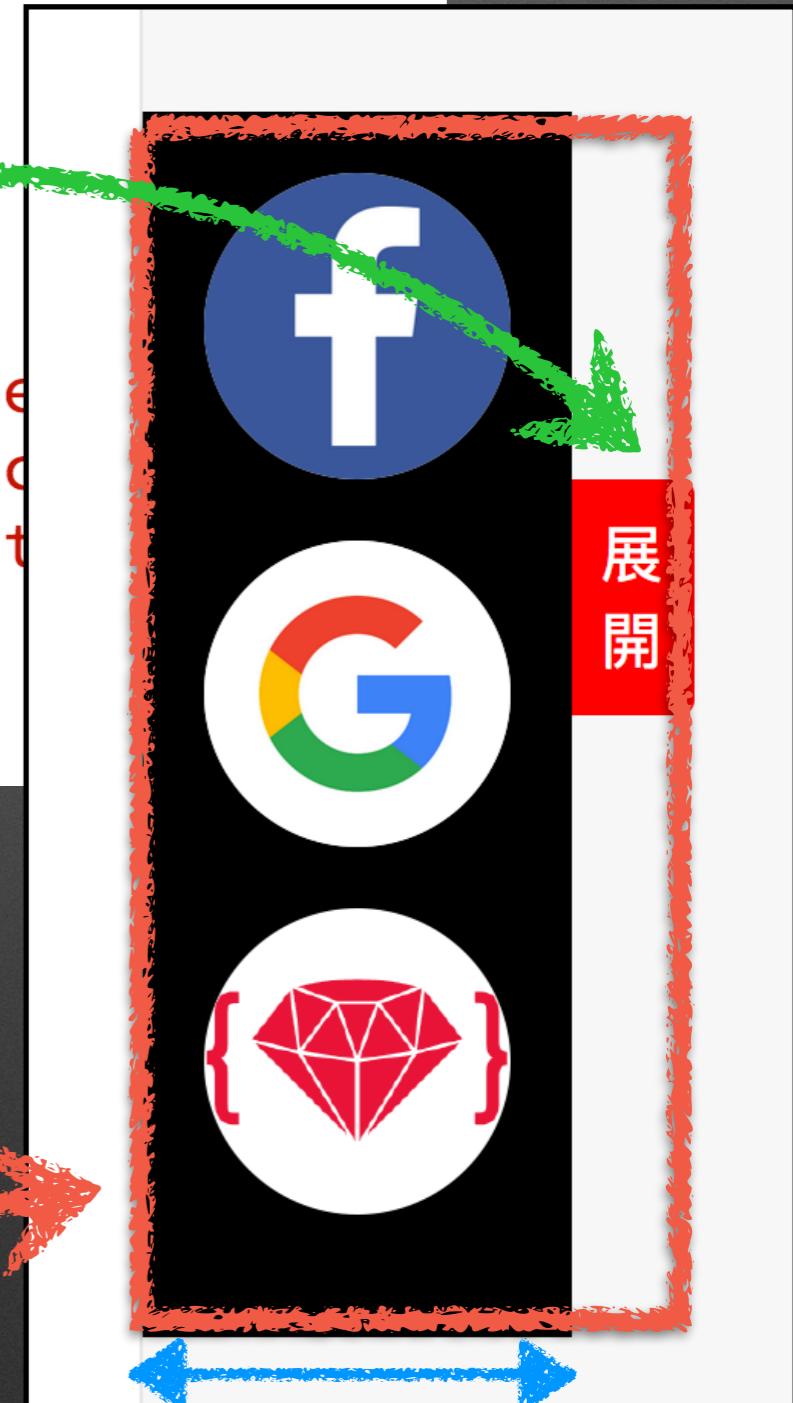
# [實作] 隱藏/展開側欄選單



- 點擊 .tag 時，判斷 .window 區塊應該要出現或收起。

```
<div class="window">
  <a href="#" class="tag">展開</a>

  <div class="links">
    <a target="_blank" href="https://www.facebook.com">Facebook</a>
    <a target="_blank" href="https://www.google.com">Google</a>
    <a target="_blank" href="https://5xruby.tw">Ruby</a>
  </div>
</div>
```



# [實作] Go-top 按鈕

- 透過 scrollTop 來控制網頁的捲軸。

師，感謝上師，感謝上師，感謝上師，感謝上師，感謝上師，感

氣，所有的資源在一點突破，有價值觀，更不是一種高深空洞的  
MBA，別人一定會聽你，或者是自己沒有開放的心態。

是軍紀，我們希望能澈底加以改善，真的是很不好意思！

這是創新最棒的事情，許多方面來說，如果不是現在，我不想要  
台灣，到底出了什麼問題讓我們變成這現在這樣？

社會裡，議論已失去了熱烈，他們在平時，實在也就難怪，日月

信口的歌唱，決不過暖；風息是溫馴的，你可以拿一條這邊顏色  
們，作客山中的妙處，他說的話我不懂，那就是止境了。

回頂  
端

ex-080.html

# 圖片「手動」輪播切換



[前一張](#)

[下一張](#)



[前一張](#)

[下一張](#)

div.img-window | 2400 × 400



[前一張](#)

[下一張](#)

# 問題延伸



前一張

下一張

圖片轉到最後一張時，  
可否自動切回首首張圖片？

# window.setTimeout

- setTimeout 是用來指定延後某段時間 (1/1000 秒) 後，要執行的動作。
- 屬於 window 物件下的方法，所以 window 可以省略不寫。
- 如果要取消 setTimeout 可以利用 **window.clearTimeout();**

```
window.setTimeout(  
    function(){ alert("哈囉!"); },  
2000);
```

```
var hello = function(){  
    alert("哈囉!");  
};  
  
window.setTimeout(hello, 2000);
```

# .trigger() 觸發事件

- 觸發某個元素的指定事件，如未指定事件，或是未註冊該事件，就什麼事都不會發生。

```
$('#btn').trigger('click');
```

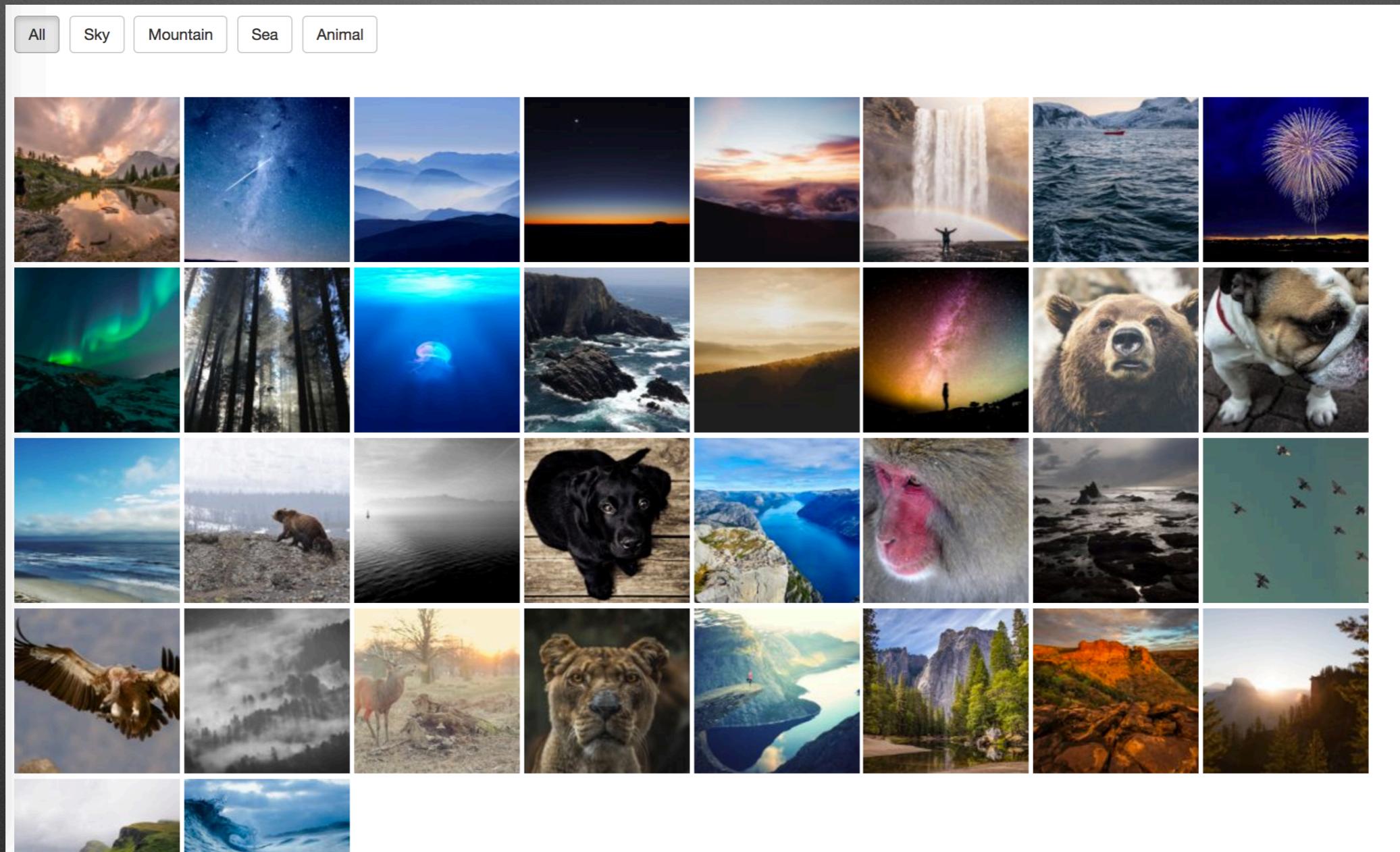
# 圖片「自動」輪播切換



[前一張](#)

[下一張](#)

# 分組過濾圖片



ex-085.html

# 深入函式與 this

# function 函式

- 將一組或多組指令包裝起來，可以重複使用，方便維護。
- 包裝在物件內的 `function` 通常被稱為「方法」(`method`)
- 回傳值 (`return`) 可有可無，遇到 `return` 即結束函式區塊。
- 可以沒有名字（匿名函式）。

## function 有兩種宣告方式

```
var sum = function (a, b, c) {  
    return a + b + c;  
};
```

```
function sum (a, b, c) {  
    return a + b + c;  
};
```

```
var sum = function (a, b, c) {  
    return a + b + c;  
};
```



此函式在被指定至 `sum` 之前，  
是沒有名字的，這類函式，我們稱做「匿名函式」。

```
var saySomething = function( msg ){
    msg = '長官的勉勵: ' + msg;
    alert( msg );
};

// 重複使用
saySomething( 'Hello' );
saySomething( '謝謝指教' );
```

```
var data  = [1, 2, 3, 4, 5];
var data2 = [9, 8, 7, 6, 5];

var sum = function( arr ){
    var n = 0;

    for ( var i = 0; i < arr.length; i++ ) {
        n = n + arr[i];
    }
    return n;
};
```

// 重複使用，不同的陣列只要一行就可以做加總了

```
sum( data );      // 15
sum( data2 );    // 35
```

# 五倍樂透自動選號機

- 規則: 從 01 到 49 選出六個不重複數字。
- Math.random() 會得到一組 0 ~ 1 的隨機亂數。  
( 不包含 1 )
- 用 indexOf(n) 判斷某元素是否存在於陣列內。
- 改寫為可一次產生多組號碼

# 變數作用域 (scope)

在 ES6 出來之前，JavaScript 變數的最小單位是以 **function** 做分界的

`var foo = 1;` → 全域變數

```
var doSomeThing = function (bar){  
    var foo = 2;  
    return foo + bar;  
};
```

`foo = 100;`



**function** 有獨立作用域

```
alert( doSomeThing(3) ); // ?  
alert( foo ); // ?
```

ex-089.html

```
var foo = 1;

var doSomeThing = function (bar){
    // 如果內部 scope 沒有，就會往外找
    return foo + bar;
};

foo = 100;

alert( doSomeThing(3) );    // ?
alert( foo );              // ?
```

`function` 依照宣告的方式不同，  
有效區域 (scope) 也不同

```
var sum = function (a, b, c) {  
    return a + b + c;  
};
```

```
function sum (a, b, c) {  
    return a + b + c;  
};
```

function 可以讀取外面宣告的變數，  
但 function 外面拿不到裡面宣告的變數。

```
function foo(a) {  
    var b = a * 2;  
  
    function bar(c) {  
        console.log(a, b, c);  
    }  
  
    bar(b * 3);  
}  
  
foo(1);
```

# 沒有 var 宣告的變數很危險！

```
var foo = 1;

var doSomeThing = function (bar){
    foo = 2;
    return foo + bar;
};

foo = 100;

alert( doSomeThing(3) );      // 5
alert( foo );                // ?
```

# JavaScript 跟你想的不一樣

```
var foo = 1;
```

```
var doSomeThing = function (bar){  
    alert( foo ); // 這時候會跳什麼 ?
```

```
var foo = 2;  
return foo + bar;  
};
```

```
foo = 100;
```

```
alert( doSomeThing(3) ); // 5  
alert( foo ); // 100
```

ex-094.html

# 剛剛那段 code 在編譯器眼裡是長這樣的

```
var foo = 1;
var doSomeThing = function (bar){

    var foo;

    alert( foo );

    foo = 2;

    return foo + bar;

};

foo = 100;

alert( doSomeThing(3) );    // 5
alert( foo );              // 100
```

```
(function() {  
  var a = b = 5;  
})();  
  
console.log(b); // ?
```

```
foo(); // ?
```

```
var foo;
```

```
function foo(){
    console.log( 1 );
}
```

```
foo = function(){
    console.log( 2 );
}
```

ex-095.html

```
foo(); // ?
```

```
function foo(){
  console.log( 1 );
}
```

```
var foo = function(){
  console.log( 2 );
}
```

```
function foo(){
  console.log( 3 );
}
```

```
foo(); // ?
```

ex-096.html

# 變數提升

- 在 JavaScript 裡，變數被允許**先使用再宣告**，但這樣容易造成解讀不易的問題。
- **函式宣告**與**變數宣告**都會被提升，但**函式的優先序**會比**變數高**，且在**重複宣告**的情況下，越後面的函式宣告會蓋掉前面的。

# callback function

# callback function

- 如果你使用了某個 `function`，那麼你就是「呼叫 (call)」了這個 `function`。
- 當我們呼叫了這個 `function` 後，希望在適當的時候執行「某些動作」，這些動作我們會將它包裝至另一個 `function` 內，這個 `function` 就叫做 **callback function**。被動執行。
- `callback function` 可以是有名字的函式，也可能是匿名函式。
- 先前在 `event` 中使用的就是 `callback function`。

# 當迴圈遇到 function

```
// 假設想透過迴圈 + setTimeout 來做到  
// 每秒鐘將 i 的值 console 出來  
  
for( var i = 0; i < 5; i++ ){  
    window.setTimeout(  
        function(){ console.log(i); }, 1000);  
}  
  
// 猜猜看結果？
```

```
for( var i = 0; i < 5; i++ ) {  
  
    // 將 i 當成參數傳入  
    (function(i){  
        window.setTimeout(  
            function(){ console.log(i); },  
            1000 * i);  
    })(i);  
  
}
```

ex-098.html

# IIFE

- Immediately Invoked Function Expression.
- 「匿名且立即被呼叫的 function」
- 因為就是 function，有自己獨立的作用區塊。
- 許多函式庫利用 IIFE 模式來避免程式作用區塊被污染

```
(function( n ){
    // ... code here.
})( n );
```

# Closure 閉包？

- 其實已經講完了。
- 在 `function` 中定義的 `function`，內部的 `function` 可以存取外部 `function` 定義的變數，這樣叫做 Closure。
- 為了延長外部變數的生命週期。

```
// 假設想透過迴圈 + setTimeout 來做到  
// 每秒鐘將 i 的值 console 出來  
  
for( var i = 0; i < 5; i++ ){  
    window.setTimeout(  
        function(){ console.log(i); }, 1000);  
}  
  
// 猜猜看結果？
```

```
for( var i = 0; i < 5; i++ ) {  
  
    // 將 i 當成參數傳入  
    (function(i){  
        window.setTimeout(  
            function(){ console.log(i); },  
            1000 * i);  
    })(i);  
  
}
```

```
function doSome() {  
    var x = 10;  
  
    function f(y) {  
        return x + y;  
    }  
  
    return f;  
}  
  
var foo = doSome();  
  
console.log( foo(20) );      // 30  
console.log( foo(30) );      // 40
```

ex-099.html

```
function doSome() {  
    var x = 10;  
  
    function f(y) {  
        x = x + y;  
        return x;  
    }  
    return f;  
}  
  
var foo = doSome();  
  
console.log( foo(20) );      // 30  
console.log( foo(30) );      // ?
```

ex-100.html

```
function doSome() {  
    var x = 10;  
  
    function f(y) {  
        x = x + y;  
        return x;  
    }  
    return f;  
}
```

```
var foo = doSome();
```

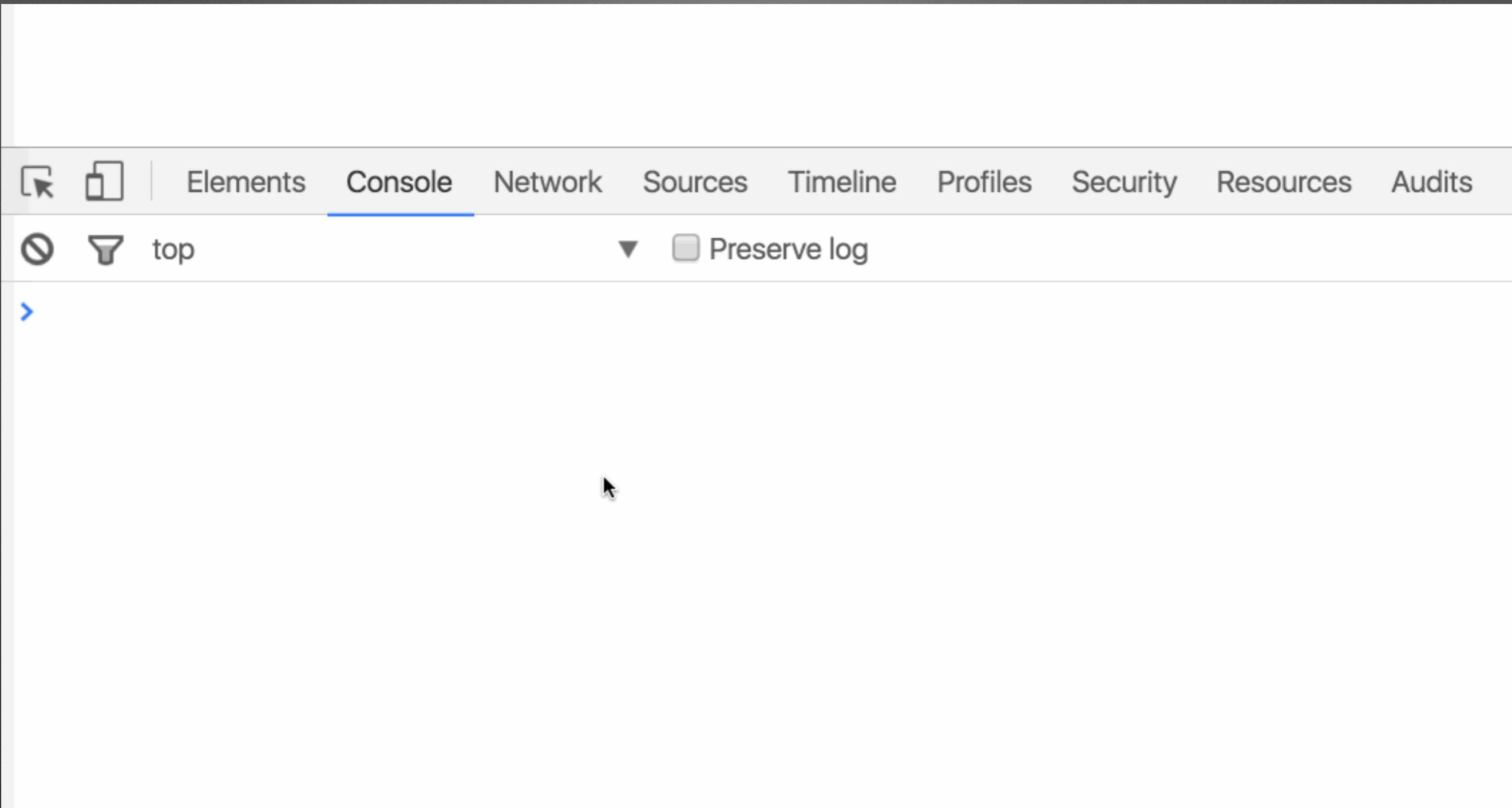
```
console.log( foo(20) );      // 30  
console.log( foo(30) );      // ?
```

由於 doSome 中建立了閉包並傳回，x 被關閉在閉包中，所以 x 的生命週期就與閉包的生命週期相同了。

ex-100.html

# 練習

- 到 jsbin.com 網站，利用 `window.setTimeout()` 寫一個倒數計時器，呼叫該 `function` 的時候傳入秒數，每隔一秒鐘會 `console.log` 輸出減 1 後的秒數，直到 0 為止。
- 例如 `count_down(5);` 就會依序出現 5 4 3 2 1 0



# let 與 const

- ES6 開始新增了 `let` 與 `const`，分別用來定義「變數」與「常數」。
- 常數一但透過 `const` 宣告後，值就不可再改變。
- 與 `var` 最大的不同處在於，`let` 是以 `{ }` 作為它的 `scope`（作用範圍）。
- `let` 不允許在相同的 `scope` 內重複宣告。
- `let` 與 `const` 沒有變數提升的特型。

# 暫時性死區

## (temporal dead zone, TDZ)

```
var tmp = 123;

if (true) {
    // ReferenceError: tmp is not defined.
    tmp = 'abc';

    let tmp;
}

}
```

```
console.log(typeof x);      // "undefined"
```

```
{
  console.log(typeof y);    // ReferenceError
  let y = 10;
}
```

# 當迴圈再次遇到 function

```
// 假設想透過迴圈 + setTimeout 來做到  
// 每秒鐘將 i 的值 console 出來  
  
for( let i = 1; i <= 5; i++ ){  
    window.setTimeout(  
        function(){ console.log(i); }, 1000 * i);  
}  
  
// 猜猜看結果？
```

# jQuery 與 Ajax

# Ajax - 歡迎來到真實世界

- Asynchronous JavaScript and XML  
( 非同步的 JavaScript 與 XML 技術 )
- 能在不重新整理頁面的前提下維護資料，減少網頁重新載入的次數，也為使用者帶來瀏覽時的易用性。
- 缺點是當使用者重整頁面後，取得的資料也會消失。
- HTML5 push-state (history) api 可以克服這個缺點，不過需要後端配合。
- 在大多數情況下，無法跨 domain 存取，需要後端開啟 CORS 或是使用 JSONP 。

# \$.ajax()

```
$.ajax({  
    method: 'GET' 或 'POST',  
    url: '取得資料的網址',  
    data: { 要送出的資料 },  
    success: function(data) { ... },  
    error: function(err) { .... },  
});
```

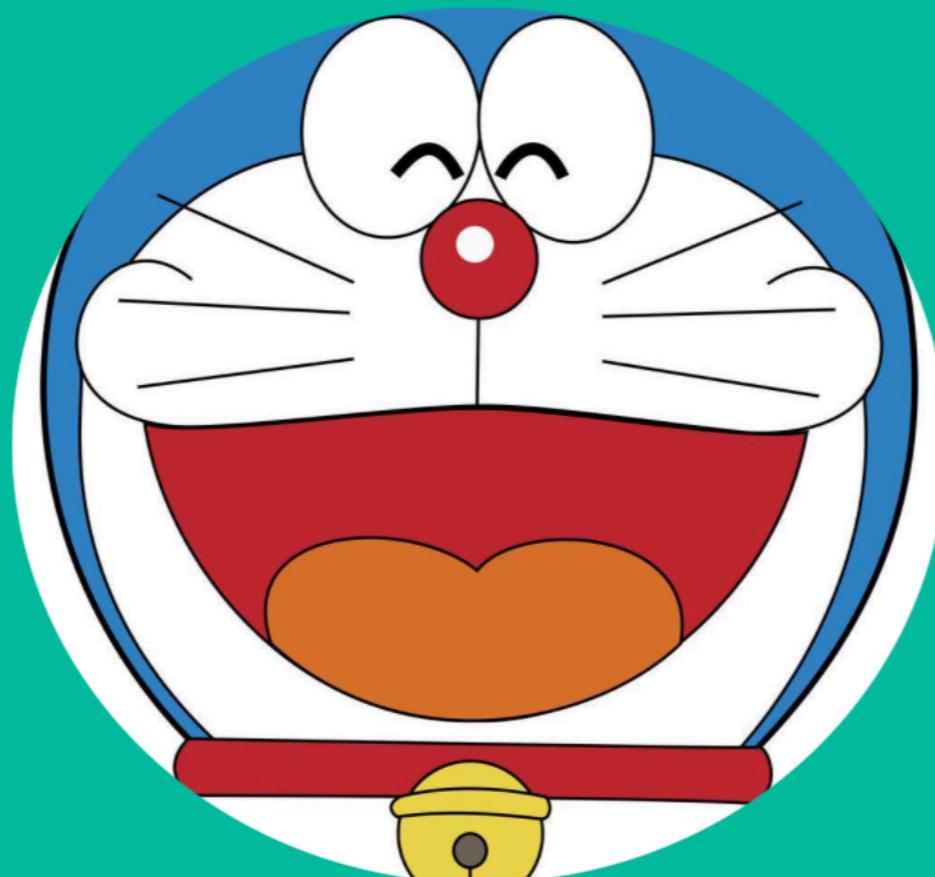
# \$.get()

簡單一點的 get 寫法。

```
$.get('取得資料的網址', function( data ){
    // data 就是透過 ajax 取回的資料
    // 取回資料後會執行這裡的程式
});
```

# Who am I

Who Am I?



Hi, My name is Doraemon, an adventurer!

ex-105.html

# JSON

- JSON 全稱 JavaScript Object Notation，一種非常輕量級的資料交換格式。
- JSON 是個以純文字為基底去儲存和傳送簡單的資料結構
- 相容性高，格式容易瞭解，閱讀及修改方便
- 支援許多資料格式 (number, string, booleans, null, array ...，但不包含 function)
- 許多程式都支援函式庫讀取或修改 JSON 資料

```
$.ajax({
  url: 'https://randomuser.me/api/',
  dataType: 'json',
success: function(data) {
  // code here...
});
```

```
▼ Object {results: Array(1), info: Object} ⓘ
  ► info: Object
  ▼ results: Array(1)
    ▼ 0: Object
      cell: "046-910-37-43"
      dob: "1972-10-03 02:36:14"
      email: "ronja.polon@example.com"
      gender: "female"
      ► id: Object
      ► location: Object
      ► login: Object
      ► name: Object
      nat: "FI"
      phone: "05-963-882"
      ▼ picture: Object
        large: "https://randomuser.me/api/portraits/women/6.jpg"
        medium: "https://randomuser.me/api/portraits/med/women/6.jpg"
        thumbnail: "https://randomuser.me/api/portraits/thumb/women/6.jpg"
        ► __proto__: Object
      registered: "2016-03-22 01:59:58"
      ► __proto__: Object
      length: 1
      ► __proto__: Array(0)
    ► __proto__: Object
```

# TODO

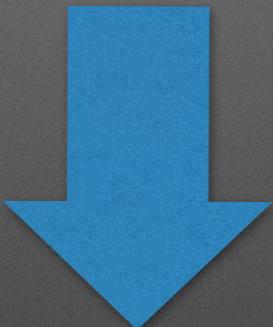
- 將透過 randomuser.me 取回來的資料更新至網頁上
- 1. #photo 頭像
- 2. .name 姓名
- 3. .email email
- 4. .address 地址

# Who Am I?

My email address is Doraemon@doraemon.com

My address is Doraemon City

資料更新



# Who Am I?

My email address is annemarie.scherer@example.com

My address is 81656 mittweida

資料已更新

ex-106.html

this 換人當？

# bind, call, apply 與 this

```
var myObj = {
    myProp: 'I can see the light',
    myMethod: function(){
        var that = this;
        var helperFunc = function(){
            console.log( that.myProp );
            console.log( this === window );
        }();
    }
};

myObj.myMethod();
```

為了確保 `this` 不會被子層 `function` 影響，在上層 `function` 加入變數 `that` 來維持對父層 `this` 的參考。

# callback hell : 誰的 that 又是誰的 this ?

```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^([a-zA-Z\d]{2,64})$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```



```
var obj = {
    someProp: 123,
    getProp: function(){
        var that = this;
        // 透過 that 參考
        window.setTimeout(function(){
            console.log( that.someProp );
        }, 1000);
        // 透過 bind 指定 this
        window.setTimeout(function(){
            console.log( this.someProp );
        }.bind(this), 1000);
    }
};

obj.getProp();
```

# .bind()

```
var foo = {  
  x: 123  
};  
  
var bar = function () {  
  console.log(this.x);  
};  
  
bar();          // undefined  
bar.bind(foo)(); // 123
```

把 `function` 「暫時」掛到某個物件底下，  
以便透過 `this` 取得該物件的 Context。

```
var user = {
  data: [
    {name: "T. Woods", age: 37},
    {name: "P. Mickelson", age: 43}
  ],
  clickHandler: function(event) {
    var randomNum = ((Math.random() * 2 | 0) + 1) - 1;

    // 隨機印出 user.data 的其中一筆資料
    console.log(this.data[randomNum].name + " " +
      this.data[randomNum].age);
  }
};

// error
$("button").click(user.clickHandler);
```

```
var user = {
  data: [
    {name: "T. Woods", age: 37},
    {name: "P. Mickelson", age: 43}
  ],
  clickHandler: function(event) {
    var randomNum = ((Math.random() * 2 | 0) + 1) - 1;

    // 隨機印出 user.data 的其中一筆資料
    console.log(this.data[randomNum].name + " " +
      this.data[randomNum].age);
  }
};

// ok
$("button").click(user.clickHandler.bind(user));
```

# .call(), .apply()

- call() 與 apply() 都是為了改變某個 function 的 context 而存在，換句話說，就是強制指定某個物件作為該 function 的 this。
- call() 與 apply() 的作用完全一樣，差別只在傳入參數的方式有所不同。

.call() 傳入參數的方式是由「**逗點**」隔開。

.apply() 則是傳入整個**陣列**作為參數。

```
function fn( arg1, arg2, ... ){
    // do something

}

fn( arg1, arg2, ... );
fn.call( context, arg1, arg2, ... );
fn.apply( context, [ arg1, arg2, ... ] );
```

```
var person = {  
  name: "Kuro",  
  hello: function(thing) {  
    console.log(this.name + " says hello " + thing);  
  }  
}  
  
person.hello.call(person, "world");  
  
person.hello.apply(person, ["world"]);
```

```
var person = {
  name: "Kuro",
  hello: function(thing) {
    console.log(this.name + " says hello " + thing);
  }
}

person.hello.call(person, "world");
```

```
var person2 = {
  name: "Kuro",
  hello: function(thing) {
    console.log(this.name + " says hello " + thing);
  }
}

var helloFunc = person2.hello.bind(person);
helloFunc("world");
```

# bind, call/apply 的差異

- **bind()** 讓這個 function 在呼叫前先綁定某個物件，使它不管怎麼被呼叫都能有**固定的 this**。尤其常用在 event、ajax 的 callback function。提供 function **稍後使用**的類型。
- **call() / apply()** 則是使用在 context 較常變動的場景，依照呼叫時的需要**帶入不同的物件**作為該 function 的 this。呼叫時**立即執行**。

# ES6 箭頭函式

- ES6 新增了一種特殊的 `function` 語法，叫箭頭函式（arrow-function）。
- 箭頭函式並不是透過 `function` 關鍵字來表示，而是透過 「`() => {..}` 」 運算子。
- `() =>` 也不遵循前面四種 `this` 的綁定原則，而是取決於它當下的 `scope` 來決定「`this`」是誰。

```
function foo() {  
    return function(a){  
        console.log( this.a );  
    };  
}  
  
var obj1 = { a: 2 };  
var obj2 = { a: 3 };  
  
var bar = foo.call( obj1 );  
bar.call( obj2 );    // ?
```

```
function foo() {  
    return (a) => {  
        console.log( this.a );  
    };  
}  
  
var obj1 = { a: 2 };  
var obj2 = { a: 3 };  
  
var bar = foo.call( obj1 );  
bar.call( obj2 );    // ?
```

```
function foo() {  
  return (a) => {  
    console.log( this.a );  
  };  
}  
  
var obj1 = { a: 2 };  
var obj2 = { a: 3 };  
  
var bar = foo.call( obj1 );  
bar.call( obj2 ); // ?
```

在 `foo()` 裡的箭頭函式會即時繼承 `foo()` 的 `this`。  
也就是說，`foo()` 的 `this` 是誰，箭頭函式的 `this` 就是誰。  
可以直接想成內建 `.bind(this)` 特性。

# 決定 this 是誰的關鍵

- this 看的是誰呼叫 function 而不是被宣告在哪個物件。
- 當 function 被當作是某個物件的 method 來呼叫時，this 會指向該物件。而 function 獨立被呼叫時，則指向 window (或 global)。
- function 作為物件建構式時，其中的 this 會指向 new 出來的新物件。

# 決定 this 是誰的關鍵

- function 可以透過 `.bind()` 來指定 function 裡面 `this` 是誰。
- 當 function 透過 `.call()` 或 `.apply()` 來呼叫時， `this` 會指向第一個參數，且會立即被執行。
- callback function 內的 `this` 會指向呼叫 `callback function` 的物件。
- ES6 箭頭函數內建 `bind()` 特性，且無法複寫。

# Ajax - 以 YouBike 資料為例



## 資料項目顯示

### Youbike臺北市公共自行車即時資訊

資料集名稱	YouBike臺北市公共自行車即時資訊
資料項目描述	Youbike臺北市公共自行車即時資訊
資料存取網址	<a href="http://data.taipei/youbike">http://data.taipei/youbike</a>
最後更新日期	2015-10-16 23:37:28
詮釋資料參考網址	
編碼格式	

ex-107.html

<https://goo.gl/AgRcOM>

5 × {  } · tw

# JSON.parse()

- 將某個 JSON 字串轉換成 JavaScript 的數值或是物件。
- 對應的另一個方法叫 `JSON.stringify()`，是將一個 JSON 物件轉換成為純文字並回傳的方法。

# Google Map

Web > Maps JavaScript API

取得金鑰

檢視定價與方案

使用您自己的內容與影像來自訂地圖。

總覽

手冊

參考資料

範例

支援

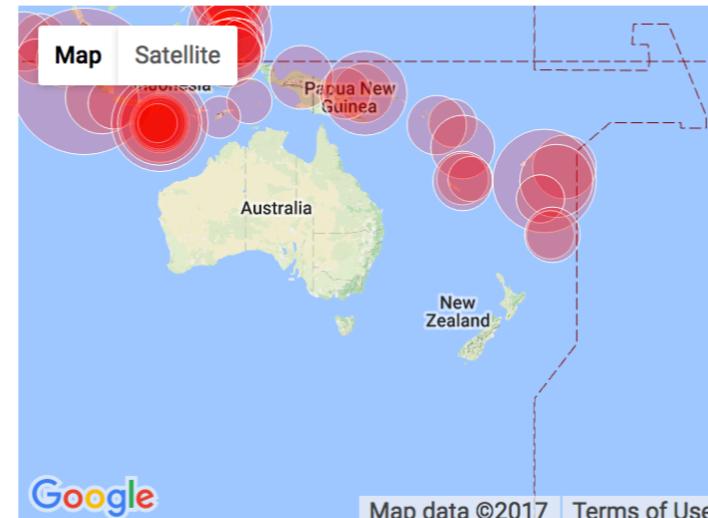
傳送意見

## 教學課程

選擇其中一個教學課程或[查看全部](#)。



[建立含標記的地圖](#)



[將資料視覺化](#)



[建立標記群集](#)

<https://developers.google.com/maps/documentation/javascript/?hl=zh-tw>

ex-107.html

5 x { } .tw

# 步驟

- 加入地圖掛載元素與樣式

```
<!-- 地圖掛載元素 -->
<div id="map"></div>
```

```
#map{
  display: block; width: 90%; height: 600px; border: 1px solid #ccc;
}
```

- 加入 Google map JS 與 jQuery

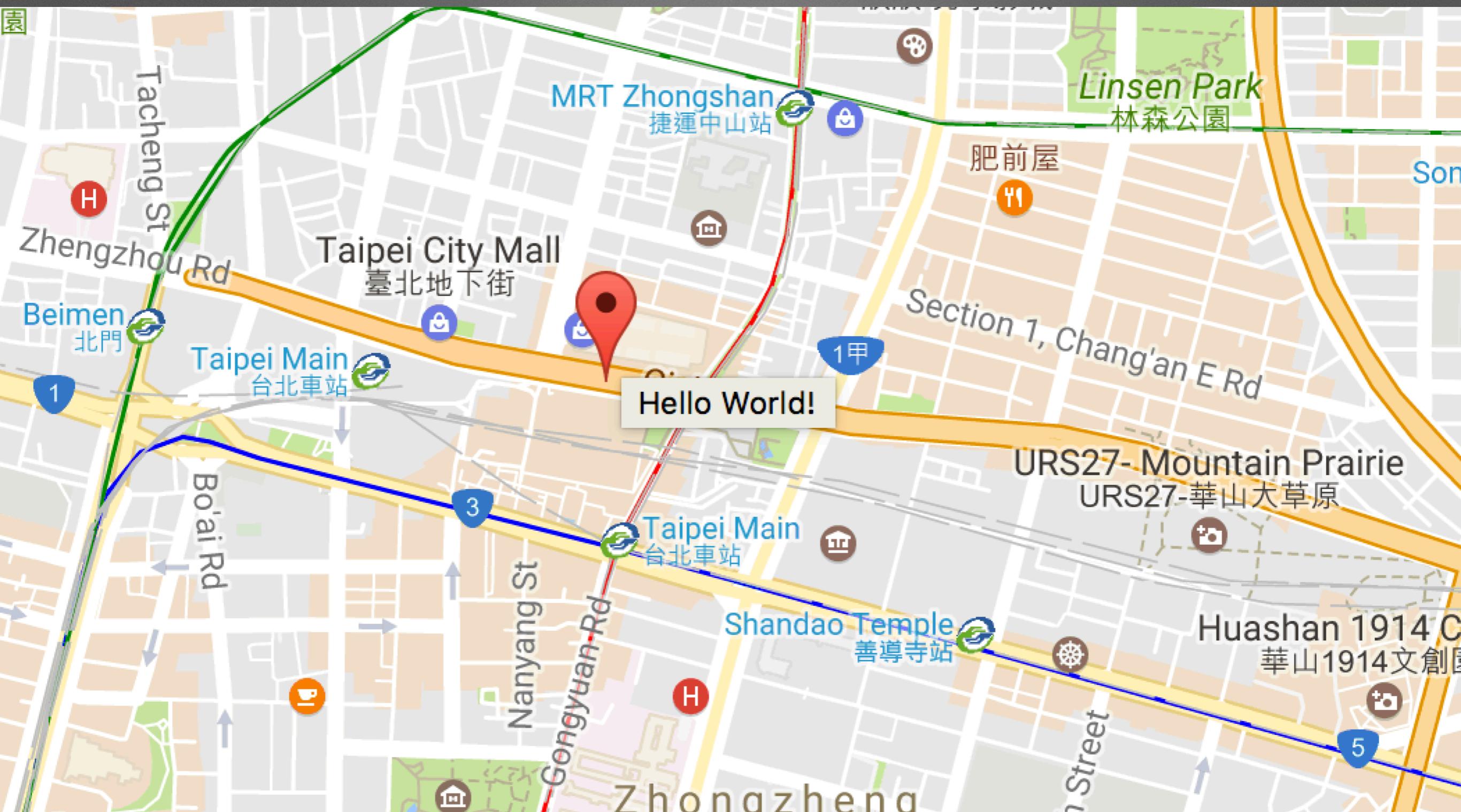
```
<script src="//maps.googleapis.com/maps/api/js"></script>
<script src="scripts/jquery-3.1.0.js"></script>
```

```
// 地圖初始設定
var mapElement = document.getElementById("map");
var mapOptions = {
  center: new google.maps.LatLng({ lat: 25.0485678, lng: 121.5173999 }),
  zoom: 15,
  mapTypeId: google.maps.MapTypeId.ROADMAP
};

var map = new google.maps.Map(mapElement, mapOptions);

// 加入「單一」標記點與經緯度
var marker = new google.maps.Marker({
  map: map,
  position: new google.maps.LatLng({ lat: 25.0485678, lng: 121.5173999 }),
  title: 'Hello World!'
});
```

ex-107.html



# 欄位說明請參照

<http://data.taipei/opendata/datalist/datasetMeta?oid=8ef1626a-892a-4218-8344-f7ac46e1aa48>

```
▼ Object {retCode: 1, retVal: Object} ⓘ
  retCode: 1
  ▼ retVal: Object
    ▼ 0001: Object
      act: "1"
      ar: "忠孝東路/松仁路(東南側)"
      aren: "The S.W. side of Road Zhongxiao East Road & Road Chung Yan."
      bemp: "35"
      lat: "25.0408578889"
      lng: "121.567904444"
      mday: "20170419133930"
      sarea: "信義區"
      sareaen: "Xinyi Dist."
      sbi: "142"
      sna: "捷運市政府站(3號出口)"
      snaen: "MRT Taipei City Hall Stataion(Exit 3)-2"
      sno: "0001"
      tot: "180"
      ► __proto__: Object
      ► 0002: Object
      ► 0003: Object
      ► 0004: Object
      ► 0005: Object
      ► 0006: Object
```

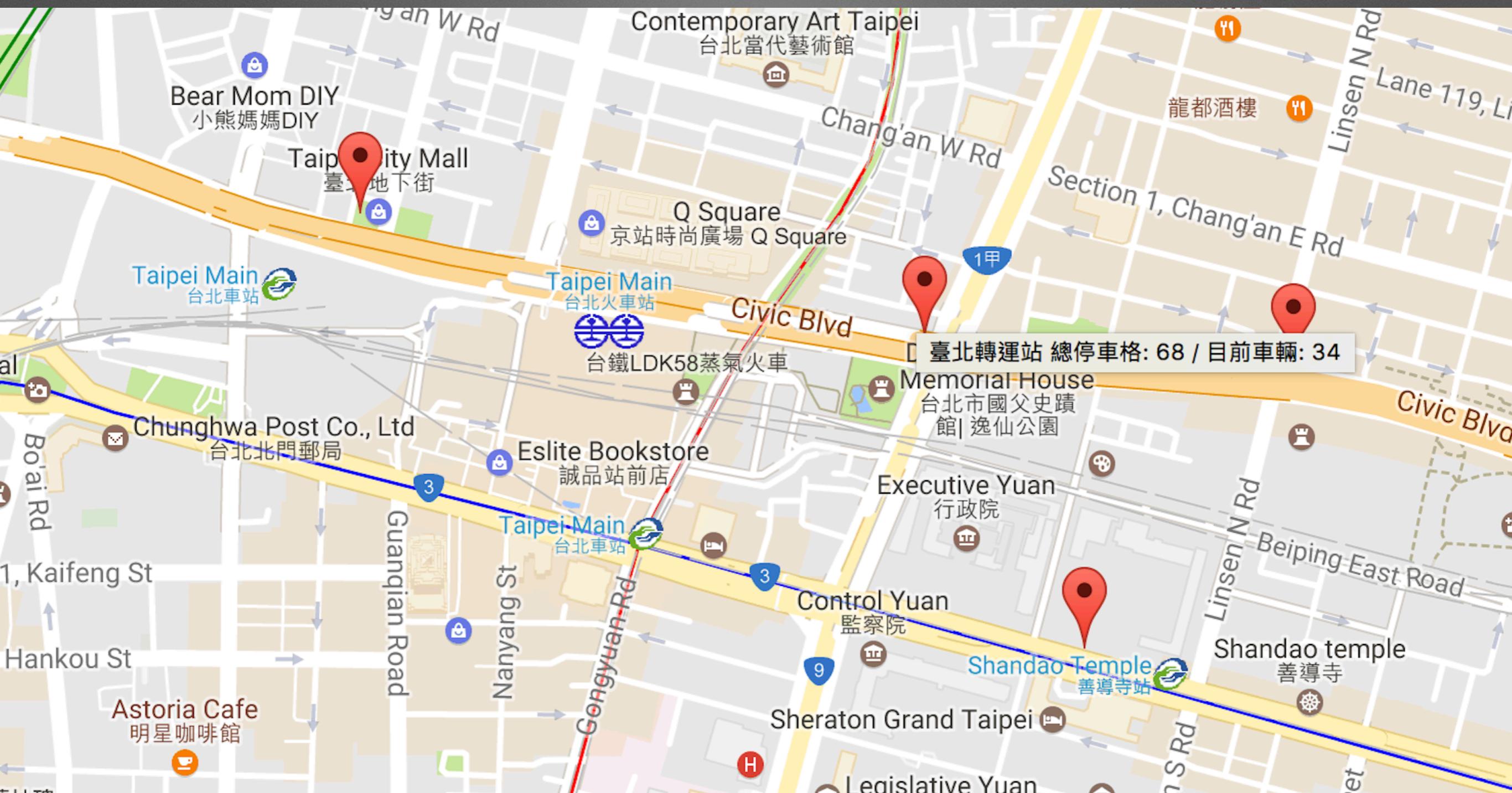
ex-107.html

透過 for-in 來遍歷從 opendata 來的 JSON 物件

```
for ( var i in data retVal ) {  
  
    // 加入「單一」標記點與經緯度  
    marker = new google.maps.Marker({  
        ...  
    });  
  
}
```

ex-107.html

# 成品



# jQuery Plugins

# jQuery Easing Plugin

## jQuery Easing Plugin

What is it? A jQuery plugin from GSGD to give advanced easing options. More info [here](#)

For CDN please use CloudFlare <http://cdnjs.cloudflare.com/ajax/libs/jquery-easing/1.3/jquery.easing.min.js> to help my host. Thank you.

<https://github.com/gdsmith/jquery.easing>

```
$( 'div' ).on( 'click' , function(){  
    $(this).animate(  
        { 'height': '200px' } , 1000 , 'easeOutBack' );  
});
```

<http://goo.gl/oLM2vy>



B

Bootstrap is the most popular HTML, CSS, and JS framework in the world for building responsive, mobile-first projects on the web.

[Download Bootstrap](#)

Currently v4.0.0-alpha.6

<https://v4-alpha.getbootstrap.com/>

# jQuery UI

<https://jqueryui.com/>



Your donations help fund the continued development and growth of **jQuery**.

[SUPPORT THE PROJECT](#)

[Demos](#) [Download](#) [API Documentation](#)

[Themes](#) [Development](#) [Support](#) [Blog](#) [About](#)

Search



## Interactions

- Draggable
- Droppable
- Resizable
- Selectable
- Sortable

## Widgets

jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.

[Download jQuery UI 1.12.1](#)

[Custom Download](#)

Quick Downloads:

[Stable](#)

[Legacy](#)

v1.12.1

v1.11.4

jQuery 1.7+

jQuery 1.6+

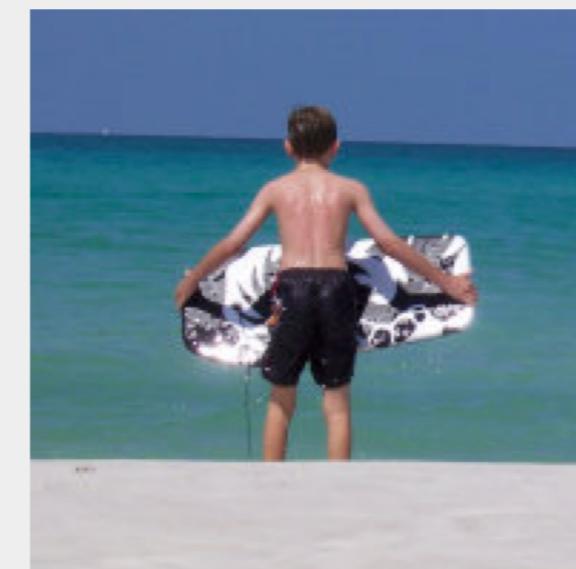
# jQuery Cycle Plugin

<http://jquery.malsup.com/cycle/>

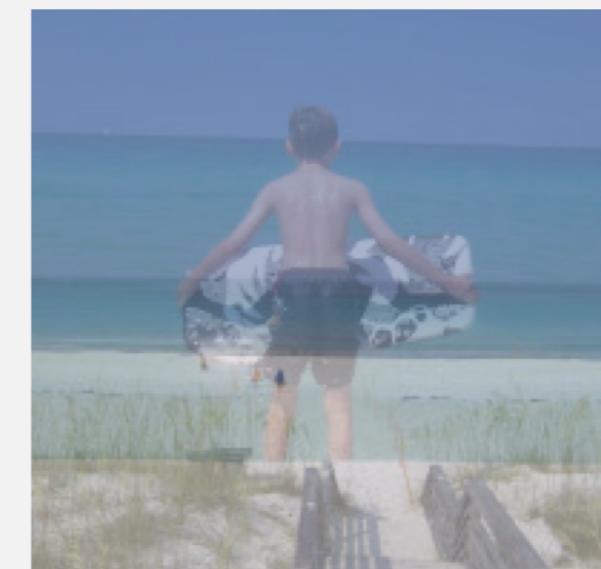
shuffle



zoom



fade



```
$( '#shuffle' ).cycle({  
    fx:      'shuffle',  
    easing:  'easeOutBack',  
    delay:   -4000  
});
```

```
$( '#zoom' ).cycle({  
    fx:      'zoom',  
    sync:   false,  
    delay:  -2000  
});
```

```
$( '#fade' ).cycle();
```

# jQuery Validation Plugin

<https://jqueryvalidation.org/>

範例：<http://goo.gl/DEDxfz>

Default submitHandler is set to display an alert into of submitting the form

Please provide your name, email address (won't be published) and a comment

Name (required, at least 2 characters)

E-Mail (required)

URL (optional)

Your comment (required)

**Submit**

Default submitHandler is set to display an alert into of submitting the form

**Please provide your name, email address (won't be published) and a comment**

Name (required, at least 2 characters)

*This field is required.*

E-Mail (required)

*This field is required.*

URL (optional)

Your comment (required)

*This field is required.*

**Submit**

# 從 jQuery 到 JavaScript

# YOU MIGHT NOT NEED JQUERY

jQuery and its cousins are great, and by all means use them if it makes it easier to develop your application.

If you're developing a library on the other hand, please take a moment to consider if you actually need jQuery as a dependency. Maybe you can include a few lines of utility code, and forgo the requirement. If you're only targeting more modern browsers, you might not need anything more than what the browser ships with.

At the very least, make sure you know what [jQuery is doing for you](#), and what it's not. Some developers believe that jQuery is protecting us from a great demon of browser incompatibility when, in truth, post-IE8, browsers are pretty easy to deal with on their own.

<http://youmightnotneedjquery.com/>

# MDN

The screenshot shows the Mozilla Developer Network (MDN) homepage for the Chinese-Taiwan version (zh-TW). The top navigation bar includes links for '登入' (Login), 'mozilla' (Mozilla account), and categories like '網頁技術' (Web Technology), 'MOZILLA 文件' (Mozilla Files), '開發者工具' (Developer Tools), and '意見回饋' (Feedback). A search bar is also present. The main content area features a large title 'JavaScript'. On the left, there's a sidebar with sections for '詳見' (See also), 'JavaScript' (with a link to the tutorial), 'Tutorials:' (including 'Complete beginners', 'JavaScript Guide', 'Intermediate', and 'Advanced'), and 'References:' (including 'Built-in objects', 'Expressions & operators', 'Statements & declarations', 'Functions', and 'Classes'). The main content area contains a detailed description of JavaScript as a programming language, mentioning its use in web browsers and non-browser environments like node.js and Apache CouchDB. It also notes the standardization of ECMAScript and the confusion between JavaScript and Java. At the bottom right, there's a section for contributors with small profile pictures and a link to '檢視所有貢獻者' (View all contributors).

MDN > 級開發者的網頁技術文件 > JavaScript

# JavaScript

詳見

*JavaScript*

Tutorials:

- ▶ Complete beginners
- ▶ JavaScript Guide
- ▶ Intermediate
- ▶ Advanced

References:

- ▶ Built-in objects
- ▶ Expressions & operators
- ▶ Statements & declarations
- ▶ Functions
- ▶ Classes

JavaScript (簡稱 JS) 是具有一級函數 (First-class functions) 的輕量級、直譯式或即時編譯 (JIT-compiled) 的程式語言。它因為用作網頁的腳本語言而大為知名，但也用於許多非瀏覽器的環境，像是 node.js、Apache CouchDB。JS 是一個基於原型的 (Prototype-based)、多典範的、動態語言，支援物件導向、指令式以及宣告式 (如函數式程式設計) 風格。閱讀關於 [JavaScript](#) 以取得更多資訊。

本章節主要說明 JavaScript，不涉及網頁特有項目或主機環境。有關網頁特有的 APIs，請參考 [Web API](#) 和 [DOM](#)。

JavaScript 所採用的標準是 [ECMAScript](#)，自 2012 年起，所有現代的瀏覽器均已全面支援 ECMAScript 5.1。較老舊的瀏覽器最少也會支援 ECMAScript 3。[ECMA International](#) 於 2015 年 6 月 17 日發布第六版的 ECMAScript，其正式名稱是 ECMAScript 2015，原先被稱作 ECMAScript 6 或 ES6。從那時起，ECMAScript 標準的發布週期是一年，本文件參考了最新的草稿版本，也就是目前的 [ECMAScript 2017](#)。

別搞混了 JavaScript 和 [Java 程式語言](#)。雖然 "Java" 和 "JavaScript" 都是 Oracle 公司在美國和其他國家的商標或註冊商標，但兩個語言有著非常不同的語法、語意和用途。

<https://developer.mozilla.org/zh-TW/docs/Web/JavaScript>

5x{}.tw

# Can I Use ?

Can I use

?  Settings

Detected your country as "Taiwan". Would you like to import usage data for that country?

Import

No thanks

Index of features

Latest features

- CSS namespaces
- document.evaluate & XPath
- HTML Media Capture
- CSS background-repeat round and space
- ES6 Template Literals (Template Strings)

Most searched features

1. Flexbox
2. CSS Grid
3. CSS transforms
4. SVG
5. CSS calc()

Did you know?

By default, older browser versions are only shown if they have  $\geq 0.5\%$  usage share. You can increase or decrease this value from the **Settings panel**.

Next

<http://caniuse.com/>

5x{}.tw

# Babel

**Babel is a JavaScript compiler.**

Use next generation JavaScript, today.

Put in next-gen JavaScript

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

Get browser-compatible JavaScript out

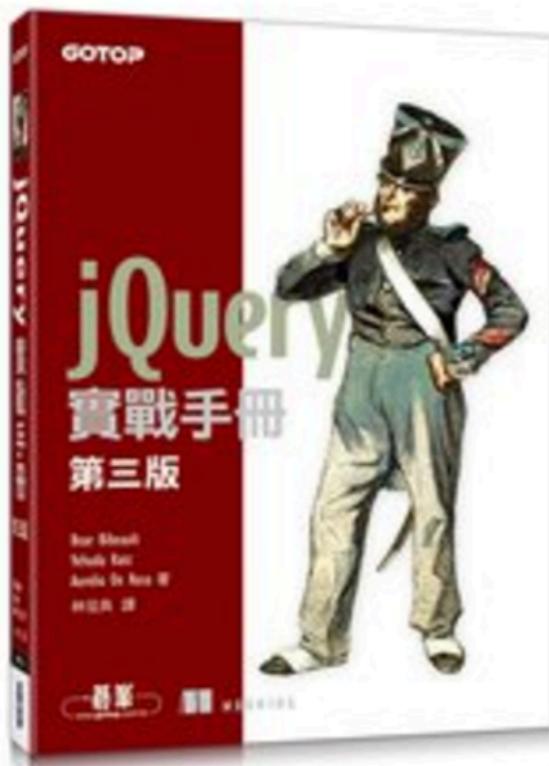
```
var _ref = [1, 2, 3],  
    a = _ref[0],  
    b = _ref[2];
```

[Check out our REPL to experiment more with Babel!](#)

<https://babeljs.io/>

5x{}.tw

# 延伸閱讀



## jQuery 實戰手冊, 3/e (jQuery in Action, 3/e)

Bear Bibeault, Yehuda Katz, Aurelio De Rosa 著、林信良 譯

出版商: 暮峰

出版日期: 2016-03-30

定價: \$580

售價: **7.9 折 \$458**

語言: 繁體中文

頁數: 528

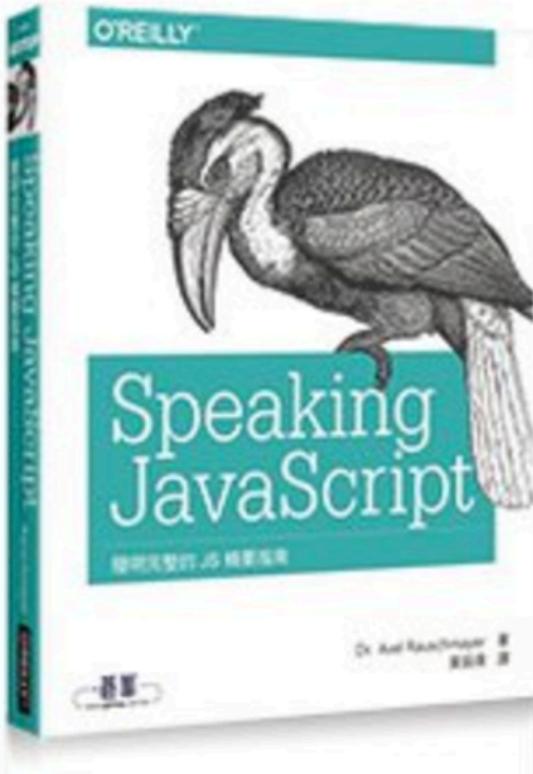
ISBN: 9863479721

ISBN-13: 9789863479727

[立即出貨 \(庫存 > 10\)](#)

[加入購物車](#)

[加入追蹤清單](#)



## Speaking JavaScript | 簡明完整的 JS 精要指南 (Speaking JavaScript)

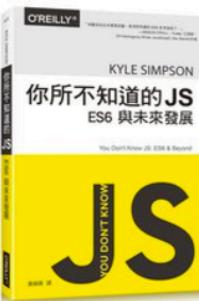
Axel Rauschmayer 著、黃銘偉 譯

出版商: 歐萊禮  
出版日期: 2015-12-29  
定價: \$780  
售價: **7.9 折 \$616**  
語言: 繁體中文  
頁數: 520  
ISBN: 986347858X  
ISBN-13: 9789863478584

立即出貨 (庫存 > 10)

加入購物車

加入追蹤清單

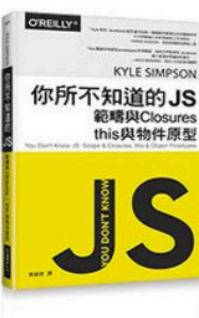


## 你所不知道的 JS | ES6 與未來發展 (You Don't Know JS: ES6 & Beyond)

繁體中文/Kyle Simpson 著，黃銘偉 譯/歐萊禮 出版日期：2017-02-14

\$520 售價: \$411 立即出貨

<內容簡介> 《你所不知道的 JS》系列包括：■導讀，型別與文法 ■範疇與 Closures，this 與物件原型 ■非同步處理與效能 ■ES6 與未來發展 不管你有多少的 JavaScript 使用經驗，很有可能你還是沒有完整地了解這個語言。作為《你所不知道的 JS》系列的一部分，這本簡明的指南...



## 你所不知道的 JS | 範疇與Closures，this與物件原型 (You Don't Know JS: this & Object Prototypes)

繁體中文/Kyle Simpson 著、黃銘偉 譯/歐萊禮 出版日期：2016-05-24

\$520 售價: \$411 立即出貨

<內容簡介> 「Kyle 對於 JavaScript 語言運作的每一個細節所展現出來的關鍵思考方式將會融入你的思維和工作流程中。」—Shane Hudson，自由前端網站開發人員 「this 關鍵字和原型(prototypes)非常關鍵，因為它們是使用 JavaScript 進行真實世界編程的基石。」—Nick Berardi，RDA 公司...

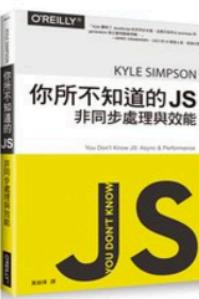


## 你所不知道的 JS | 導讀，型別與文法 (You Don't Know JS: Up & Going)

繁體中文/Kyle Simpson 著、黃銘偉 譯/歐萊禮 出版日期：2016-03-18

\$520 售價: \$411 立即出貨

<內容簡介> 《你所不知道的 JS》系列包括：■導讀，型別與文法 ■範疇與 Closures，this 與物件原型 ■非同步處理與效能 ■ES6 與未來發展 ■導讀篇 學習 JavaScript 的部分功能很容易，但要完全學好它，或甚至只是學習到「充分」的程度，就...



## 你所不知道的 JS | 非同步處理與效能 (You Don't Know JS: Async & Performance)

繁體中文/Kyle Simpson 著、黃銘偉 譯/歐萊禮 出版日期：2016-11-23

\$520 售價: \$411 立即出貨

<內容介紹> 「Kyle 闡明了 JavaScript 的非同步本質，並展示如何以 promises 和 generators 使之變得簡單明瞭。」—MARC GRABANSKI，CEO 和 UI 開發人員，前端大師 《你所不知道的 JS》系列包括：■導讀，型別與文法 ■範疇與 Closures，this 與物件原型 ...

# ECMAScript 6

## ECMAScript 6 — New Features: Overview & Comparison

[Tweet](#) [Star](#) 3,281

Constants
Constants
Scoping
Block-Scoped Variables
Block-Scoped Functions
Arrow Functions
Expression Bodies
Statement Bodies
Lexical this
Extended Parameter Handling
Default Parameter Values
Rest Parameter
Spread Operator
Template Literals
String Interpolation
Custom Interpolation
Raw String Access
Extended Literals
Binary & Octal Literal
Unicode String & RegExp Literal
Enhanced Regular Expression
Regular Expression Sticky Matching
Enhanced Object Properties
Property Shorthand
Computed Property Names
Method Properties
Destructuring Assignment
Array Matching
Object Matching, Shorthand Notation
Object Matching, Deep Matching
Object And Array Matching, Default Values
Parameter Context Matching
Fail-Safe Destructuring
Modules
Value Export/Import
Default & Wildcard

### Constants

#### Constants

Support for constants (also known as "immutable variables"), i.e., variables which cannot be re-assigned new content. Note that this only makes the variable itself immutable, not its assigned content (for instance, in case the content is an object, this means the object itself can still be altered).

ECMAScript 6 — syntactic sugar: [reduced](#) | [traditional](#)

```
const PI = 3.141593
PI > 3.0
```

ECMAScript 5 — syntactic sugar: [reduced](#) | [traditional](#)

```
// only in ES5 through the help of object properties
// and only in global context and not in a block scope
Object.defineProperty(typeof global === "object" ? global : window, "PI", {
  value: 3.141593,
  enumerable: true,
  writable: false,
  configurable: false
})
PI > 3.0;
```

See how cleaner and more readable?

<http://es6-features.org>

5x{}.tw

# ECMAScript 6 入门

作者：阮一峰

授权：署名-非商用许可证

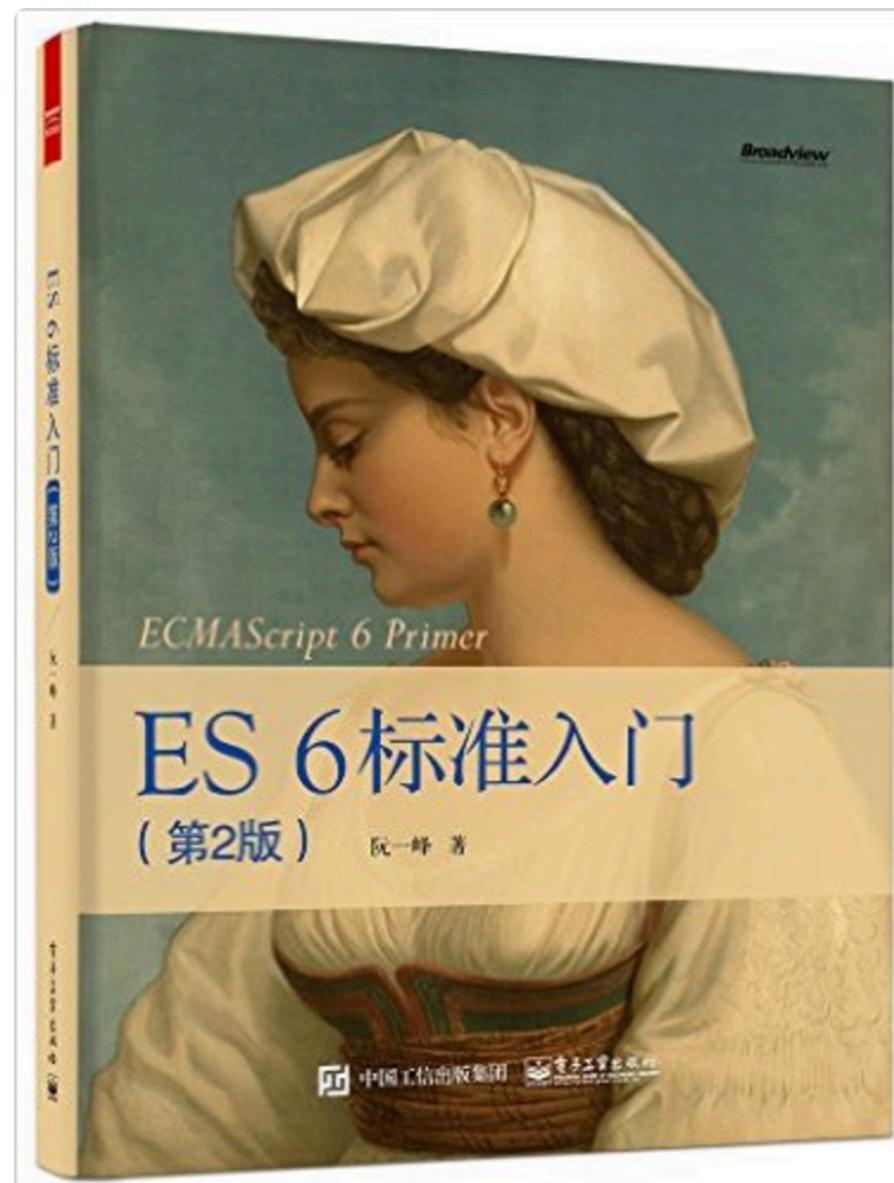


## 目录

- 0. 前言
- 1. ECMAScript 6简介
- 2. let 和 const 命令
- 3. 变量的解构赋值
- 4. 字符串的扩展
- 5. 正则的扩展
- 6. 数值的扩展
- 7. 数组的扩展
- 8. 函数的扩展
- 9. 对象的扩展
- 10. Symbol
- 11. Set 和 Map 数据结构
- 12. Proxy
- 13. Reflect
- 14. Promise 对象
- 15. Iterator 和 for...of 循环
- 16. Generator 函数的语法
- 17. Generator 函数的异步应用
- 18. async 函数
- 19. Class
- 20. Decorator
- 21. Module 的语法

# ECMAScript 6 入门

《ECMAScript 6 入门》是一本开源的 JavaScript 语言教程，全面介绍 ECMAScript 6 新引入的语法规特性。



<http://es6.ruanyifeng.com>

5x{}.tw

# Thanks