

JavaScript

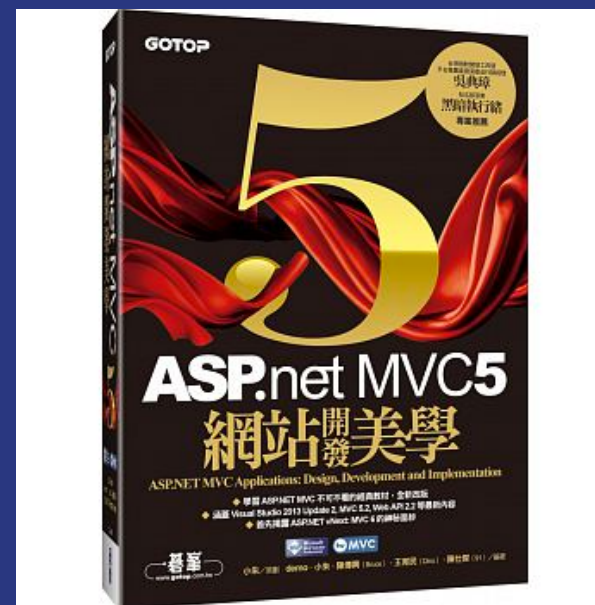
Dino Wang
Build School 講師

請尊重講師的著作權及智慧財產權!

Build School 課程之教材、程式碼等、僅供課程中學習用、請不要任意自行散佈、重製、分享，謝謝

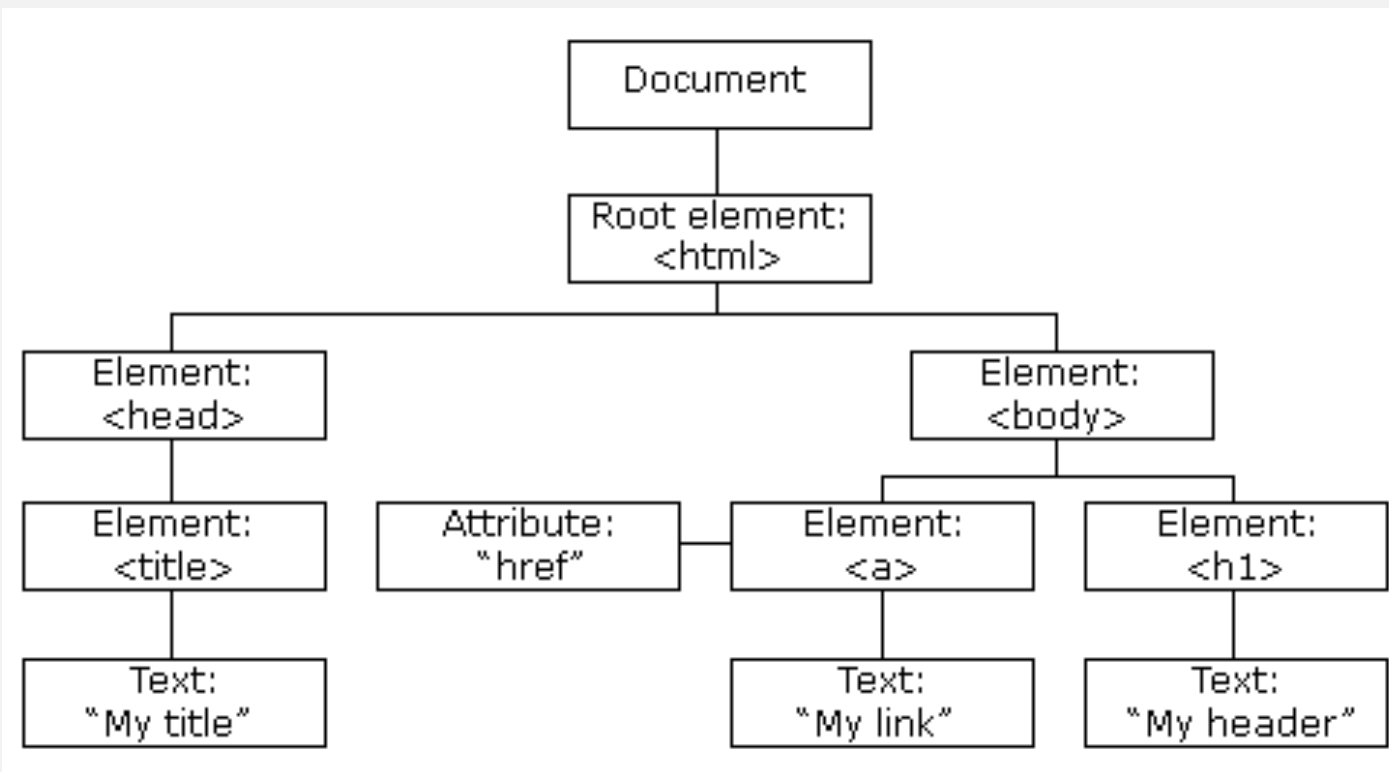
講師介紹 – Dino Wang

- 微軟最有價值專家 – Azure
- twMVC Community Co-funder
- ASP.NET MVC 5 開發美學共同作者
- hexdigits
 - dino@hexdigits.com



十分重要的 DOM Tree 觀念

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>My header</h1>
    <a href="#">My link</a>
  </body>
</html>
```



取得 DOM 節點

- 通過瀏覽器的 document 物件取得 DOM 節點
 - getElementById
 - getElementsByTagName
 - getElementsByClassName
- 以上都是 method，需要傳入一個字串參數，是一個名稱，根據不同的方法比對名稱後取得 DOM 物件

創建 DOM 節點的方法

- 共有兩種方法可以創建出新的節點
- 需先透過前述任一方法提取標的父節點
- 通過 `.innerHTML` 屬性，或是 `.appendChild()` 方法，增加子節點

innerHTML 屬性

```
var target = document.getElementById("標的id");
```

```
target.innerHTML = "<em>Hello</em>";
```

appendChild 方法

```
var target = document.getElementById("標的id");
```

```
var newNode = document.createElement("em");
```

```
newNode.innerHTML = "Hello";
```

```
target.appendChild(newNode);
```


JavaScript Function

- 以保留字 `function` 為起始
- 然後是函數名稱，命名規則同變數，習慣為小寫開始
- 所有參數以 `()` 括起，不用加上 `var` 宣告，以逗點隔開，
如果沒有任何參數，也要加上一組括弧
- 隨後加上程式碼區塊，函數本體在 `{ }` 中

該使用哪一種方法？

- `innerHTML` 很快，但會將父節點下原有的節點和狀態全部清除
- `appendChild` 寫起來很麻煩，但是可以遞增的加入新增元素，不會不小心清除掉節點上的狀態

JavaScript Function

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

```
function subtract(a, b)  
{  
    return a - b;  
}
```



左括弧不換行
JavaScript 的慣用風格

JavaScript Function 其它用法

- 匿名函式
- Function 物件
- 函式建構子 (JavaScript 1.4 的模擬物件導向寫法)

JavaScript Function – 匿名函式

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

// 將匿名函式指定給變數

JavaScript Function – 匿名函式

```
document.getElementById("button1") = function (a, b) {  
    return a + b;  
}
```

// 將匿名函式指定給變數

JavaScript Function – Function 物件

```
var add = new Function("a", "b", "return a + b;");
```

```
add(1, 4); // 得到 5
```

```
// 得以用組字串的方式產生程式
```

JavaScript Function – 函式建構子

```
function Person(firstName, lastName) {  
    this.firstName = firstName;  
    this.lastName = lastName;  
  
    return this;  
}
```

```
var x = new Person("Build", "School");  
// x.firstName = "Build";
```


JavaScript Function

- 函數可以沒有回傳值，若未指定回傳值，卻嘗試接回傳值，則會得到 `undefined`

```
function doSomething() {  
    // 寫了程式碼，但是沒有任何的 return 敘述  
}  
  
var x = doSomething();    // x = undefined
```

JavaScript 與 DOM 事件處理

- 什麼是「DOM 事件」
 - 點擊了網頁上的一個按鈕，引發 onclick 事件
 - 滑鼠移過 HTML 元素，引發 onmouseover 或 onmouseout 事件
 - 改變了表單上的一個欄位，引發 onchange 事件
 - 點擊了送出表單按鈕，引發表單的 onsubmit 事件

JavaScript 與 DOM 事件處理

- 如何知道有哪些事件可以處理？
 - 一般通用事件，所有 HTML 元素都有
例如：onclick, onmouseover
 - 特定元素事件，限定的 HTML 元素才有
例如：<body> 的 onload 事件、<form> 的 onsubmit 事件、
<video> 的 onplay 事件

JavaScript 與 DOM 事件處理

- JavaScript 與 DOM 事件概念

https://www.w3schools.com/js/js_htmlDOM_events.asp

- 可用的 DOM 事件查表

https://www.w3schools.com/jsref/dom_obj_event.asp

JavaScript 與 DOM 事件處理

- 首先撰寫 JavaScript function 自定義事件處理函式
- 然後與 DOM 連結綁定事件
 - 透過 屬性標籤
 - 透過 DOM 屬性

JavaScript Function 與 DOM 事件

```
<button id="btn1" onclick="doClick();" >按我</button>
```

```
function doClick() {  
    // 事件處理器  
};
```

JavaScript Function 與 DOM 事件

```
<button id="btn1">按我</button>
```

```
var button = document.getElementById("btn1");
```

```
button.onclick = function () {  
    // 事件處理器  
};
```

JavaScript Function 與 DOM 事件

```
<button id="btn1">按我</button>
```

```
document.getElementById("btn1").onclick = function () {  
    // 事件處理器  
};
```


完成溫度轉換 的未完部分

☒ 攝氏轉換為華氏

☐ 華氏轉換為攝氏

1

2

3

4

5

6

7

8

9

0

.

20

轉換

清除

結果

68

點擊「清除」

- 將輸入值清除
- 將前次計算結果清除

點擊「數字」或「.」輸入按鈕

- 將輸入值清除
- 將前次計算結果清除

Demo and exercise

運用屬性標籤加上事件


檔名 `temperature-convert-bs-event-by-attr.html`

點擊「清除」

```
<span class="input-group-btn">
  <a id="convert" type="button" class="btn btn-default" onclick="doCalc(); return false;">轉換</a>
  <a id="clear" type="button" class="btn btn-default" onclick="clearInput(); return false;">清除</a>
</span>
```

```
function clearInput() {
  var input = document.getElementById("input");
  var result = document.getElementById("result");

  input.value = "";
  result.value = "";
}
```



連結到事件處理

Demo and exercise

運用 DOM 加上事件

檔名 `temperature-convert-bs-event-by-dom.html`

點擊「數字」或「.»輸入按鈕

```
<div id="buttons">
  <div class="row">
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">1</a></div>
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">2</a></div>
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">3</a></div>
  </div>
  <div class="row">
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">4</a></div>
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">5</a></div>
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">6</a></div>
  </div>
  <div class="row">
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">7</a></div>
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">8</a></div>
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">9</a></div>
  </div>
  <div class="row">
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">0</a></div>
    <div class="col-md-4"><a class="btn btn-default btn-block input-pad" href="#" role="button">.</a></div>
  </div>
</div>
```

點擊「數字」或「.」輸入按鈕

```
// 「鍵盤」上的按鈕
function inputButton() {
    var input = document.getElementById("input");
    var text = this.innerHTML;
    input.value += text;
}

var inputButtons = document.getElementsByClassName("input-pad");

for (var i = 0; i < inputButtons.length; i++) {
    inputButtons[i].onclick = inputButton;
}
```


透過 屬性標籤 與 DOM 指定事件處理函式的差異

- 兩者的作用，是相同的
- 目前應該會感覺到「透過屬性標籤」似乎比較容易
 - 程式碼看起來好像比較少，也比較直觀
- 事實上，透過 屬性標籤 是有一些不方便性

非常重要 但 非常容易困惑人的 "this" 保留字

```
// 「鍵盤」上的按鈕
function inputButton() {
    var input = document.getElementById("input");
    var text = this.innerHTML;
    input.value += text;
}

var inputButtons = document.getElementsByClassName("input-pad");

for (var i = 0; i < inputButtons.length; i++) {
    inputButtons[i].onclick = inputButton;
}
```

非常重要但容易困惑人的 "this" 關鍵字

- this 代表目前執行函數所屬的物件

非常重要但容易困惑人的 "this" 關鍵字

- **this** 代表 目前執行函數 所屬的物件
- 預設值為 window 物件 (BOM, Browser Object Model)
 - 代表目前這個視窗
 - https://www.w3schools.com/jsref/obj_window.asp
- 而在事件處理函式中，this 代表現在這個物件

this 的觀念建立

```
// 「鍵盤」上的按鈕
```

```
function inputButton() {  
    var input = document.getElementById("input");  
    var text = this.innerHTML;  
    input.value += text;  
}
```

因為直接執行了 inputButton();
this == window 物件

```
inputButton();
```

this 的觀念建立

```
// 「鍵盤」上的按鈕
```

```
function inputButton() {
```

```
    var input = document.getElementById("input");
```

```
    var text = this.innerHTML;
```

```
    input.value += text;
```

```
}
```

因為掛載在每一個 input-pad 物件上
this == 發生 click 的 input-pad 物件

```
var inputButtons = document.getElementsByClassName("input-pad");
```

```
for (var i = 0; i < inputButtons.length; i++) {
```

```
    inputButtons[i].onclick = inputButton;
```

```
}
```

猜數字

開始	放棄重來	看答案
----	------	-----

0A1B 1234

0A3B 3456

0A2B 4567

2A0B 6089

3A0B 6345

4A0B 6385

猜答案

猜!

猜數字的各種細節

- 開始遊戲時，由程式自動產生一組數字
 - 範圍 0~9，不重複四位數，如：9487
- 使用者輸入猜測的數字組合，程式回應答對的狀態
 - 猜 1234，回答狀態為 0A1B
 - 猜 6789，回答狀態為 1A2B

猜數字的事件處理函數

開始	放棄重來	看答案
<div>0A1B 1234</div>		
<div>0A3B 3456</div>		
<div>0A2B 4567</div>		
<div>2A0B 6089</div>		
<div>3A0B 6345</div>		
<div>4A0B 6385</div>		
猜答案		猜!

猜數字的事件處理函數

開始	放棄重來	看答案
<div><div>0A1B</div>1234</div>		
<div><div>0A3B</div>3456</div>		
<div><div>0A2B</div>4567</div>		
<div><div>2A0B</div>6089</div>		
<div><div>3A0B</div>6345</div>		
<div><div>4A0B</div>6385</div>		
<div>猜答案</div> <div></div> <div>猜!</div>		

猜數字的事件處理函數

開始	放棄重來	看答案
0A1B 1234		
0A3B 3456		
0A2B 4567		
2A0B 6089		
3A0B 6345		
4A0B 6385		

猜答案		猜!
-----	--	----

abortGame

猜數字的事件處理函數

開始	放棄重來	看答案
0A1B	1234	
0A3B	3456	
0A2B	4567	
2A0B	6089	
3A0B	6345	
4A0B	6385	

猜答案

猜!

showAnswer


猜數字的事件處理函數

開始	放棄重來	看答案
0A1B	1234	
0A3B	3456	
0A2B	4567	
2A0B	6089	
3A0B	6345	
4A0B	6385	

猜答案

猜!

doGuess



值得提醒的部分

- 產生隨機答案，四個不重複的數字字元
- 解答過程中，勢必會有很多演算式針對字串物件
- 請查找 JavaScript 字串物件的存取方法
 - https://www.w3schools.com/jsref/jsref_obj_string.asp

Demo and exercise

檔名 guess-number.html

運用 DOM 加上事件，熟悉 JavaScript 字串操作、迴圈，邏輯條件等語法

你覺得「猜數字」操作起來方便嗎？滿意嗎？

- 每猜一次，就要用鍵盤輸入數字，用滑鼠點擊「猜」，沒辦法用鍵盤獨立完成，因為猜的動作很頻繁，使用起來會覺得超級卡的

你覺得「猜數字」操作起來方便嗎？滿意嗎？

- 能不能在輸入完數字後，按下 Enter 直接開始猜的動作

你需要知道使用者在輸入框內按下了 Enter

- 運用 onkeydown 或 onkeyup 事件
- 讀取「事件物件」中關於鍵盤輸入的資訊

Event Object

- 被指派的事件處理函式，在事件發生時，會額外得到一個事件物件
- 事件物件存放了事件名稱、從何發生？發生的時間點等資訊
- 特定的事件，又另外包含了特定事件所屬資訊，例如：
按下按鍵，就會包含了「按了哪一個按鍵」的資訊

Event Object

```
document.getElementById("userGuess").onkeyup = function (event) {  
    if (event.keyCode == 13) {  
        doGuess();  
    }  
};
```

每一個事件處理函式
都會接收到一個事件物件參數

- onkeyup event

- https://www.w3schools.com/jsref/event_key_keycode.asp

Demo and exercise

幫你的「猜數字」加上更好的使用體驗

生命靈數

- 使用 Bootstrap 完成本程式的簡單佈局
- 運用 JavaScript 的計算能力得到結果
- 結合 DOM 的操作，將結果輸出到畫面上

Exercise

檔名 numerology.html

在這裏你將學到

Learn How to Learn

- 學新東西、新技術的能力
- 尋找解答的能力
- 隨時吸取新知識的能力

跟著你一輩子的能力 ...

對學生的好處

不只是就業銜接，培養自我解決問題的能力及信心！



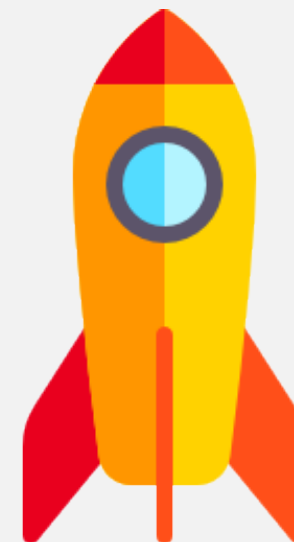
解決問題能力 (Problem-Solving)

- Learn how to learn
- Peer Learning
- Project-based



就業銜接 (Professional skills ready)

- Technical +
Soft Skill



擴散及影響 (Make impacts)

- Mentor
Networks

Q&A

課程資料 – <http://xxx.xxx>