JavaScript

王維駿

JavaScript 介紹

- 是一種程式語言
- C-like語法
- 腳本語言、弱型別、免編譯
- 與Java沒有關係

基本資料型別

- Number 數字
- Boolean 布林值
- String 字串

基本資料型別 - Number數字

- 沒有整數、浮點數、精度的區別
  - 0
  - 1
  - -500
  - 3.1415926

基本資料型別 – Boolean布林值

- `true`
- `false`

基本資料型別 – String字串

- "這個是字串"
- '這也是字串'

特殊資料型別

- Null 沒有值
- Undefined 未定義
- NaN 不該被運算

特殊資料型別 – Undefined未定義

- 宣告變數沒有給初始值
- 讀取不存在的屬性
- 讀取陣列維度以外的元素

特殊資料型別 – NaN不該被運算

- 代表 Not A Number
- 例如:1 + undefoned

變數

- 必須以字母、$、 _ 為起始字元
- 必須以字母、數字、$、 _ 等字元組合
- 區分大小寫
- 不得為保留字、關鍵字

變數宣告

- 變數用來存放資料型別、也可以存放物件
- 用 var 來進行宣告

變數名稱

- 有效的名稱

```
var money = 0;
var _moreMoney = 10000;
var $lover = 999999.99999;
```

- 無效名稱

```
var 0money = 0;
var var = 1000;
```

基礎語法

- 程式碼區塊
- 指派運算子
- 關系運算子
- 物件
- 條件判斷式
- 迴圈

基礎語法 – 程式碼區塊

- {} 內為程式碼的區塊
- 與C#相同

基礎語法 – 指派運算子

- 用 = 指派值變數或屬性
  - `var a = 1;`
  - `you.tomorrow = 99999999999;`

基礎語法 – 關係運算子

- 比大小
  - A > B
  - A < B
  - A >= B
  - A <= B

基礎語法 – 關係運算子

- 比內容
  - A == B
  - A != B
- 比內容還要比型別(嚴格)
  - A === B
  - A !== B

基礎語法 – 物件

- JavaScript的物件非常靈活，任意建立不必依賴型別

基礎語法 – 建立物件

- 建立物件的方式

```
var object1 = {};
var object2 = new Object();
```

基礎語法 – 建立物件

- 建立JavaScript物件，給予初始值，指派給變數

```
var guessRecord = {
    "userAnswer":"1234",
    "result":"1A2B"
}
```

屬性名稱　　屬性值

透過 guessRecord.result 或 guessRecord["result"] 都可以取得 1A2B

基礎語法 – 條件判斷式

- `if else`
- `switch case`
- 與 C# 相同

基礎語法 – 迴圈

- for loop
- for in
- while

基礎語法 - 迴圈 for loop

- for (起始;執行條件;迭代) {}
  - 起始:迴圈開始的變數設定
  - 執行條件:不符合時跳離迴圈
  - 迭代:每次執行完，進行的變數調整

基礎語法 - 迴圈 for in

- for (子 in 母) {}
  - 逐一取出母體物件內的成員的名稱

基礎語法 – 迴圈 while

- while (執行條件) {}
  - 條件成立時，執行程式碼區塊
  - 完成區塊執行後，重新進行評估

陣列

- 儲存多個變數內容，只需要一個變數名稱
- 可以有一個維度以上的儲存空間
- 可以任意修改，沒有型別約束
- 索引從 `0` 開始

陣列宣告

```
var array1 = [];                          // 一個空陣列
var array2 = new Array();                 // 也是一個空陣列
var array3 = [1, 2, 3];                   // 一個初始內容為 1, 2, 3 的陣列
var array4 = new Array(1, 2, 3);          // 同上
var array5 = [ [1, 2, 3], [4, 5, 6] ]     // 一個二維陣列
var array6 = new Array(20);               // 陣列長度為 20, 全部都是預設值的陣列
```

JavaScript in Browser

- 可以執行運算
- 沒辦法獨立I/O，需要透過瀏覽器
- 透過操作Dom物件與HTML互動

Dom

- 文件對象模型 (Document Object Model)
- 提供給程式控制網頁的介面

Dom

- HTML本質上是一份靜態文件，使用瀏覽器開啟時，瀏覽器會解釋這份文件，在記憶體中產生相對應的物件(HTML標籤->物件)，這些物件是動態的，可以對它的外觀、內容、行為進行改變

Dom

- Dom是一套介面，擁有基本的標準
  - https://www.w3.org/DOM/
  - https://dom.spec.whatwg.org/
  - https://www.w3schools.com/js/js_htmldom.asp
- 也有各自瀏覽器加強的版本

Bom

- 瀏覽器對象模型 (Browser Object Model)
- 存取Dom，要透過Bom提供的方法

Bom

```
document.getElementById("result").innerHtml = text;
```

Bom物件　　Bom物件提供的方法　　寫入Dom物件的innerHTML屬性

透過Bom物件提供的方法取的Dom物件

常用的取的Dom物件方法

- document.getElementById
  - 依ID取得DOM物件，由於id必須唯一，取回是單一物件或null
- document.getElementsByTagName
  - 依標籤取得DOM物件，回傳陣列或空陣列
- document.getElementsByClassName
  - 依Class取得DOM物件，回傳陣列或空陣列

Dom事件

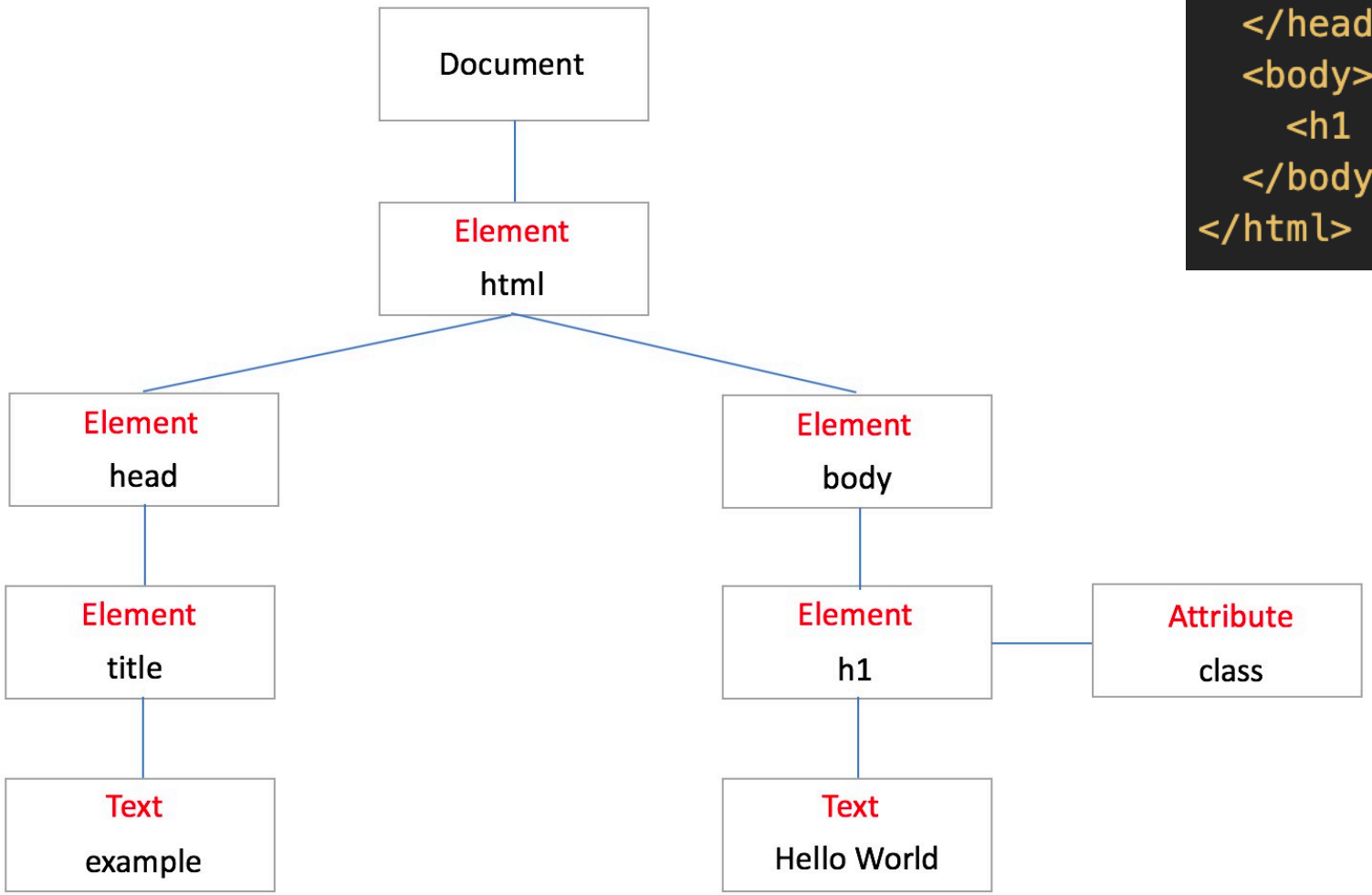
- 程式碼主動執行
  - 打開HTML文件後立即執行
- 傾聽Dom事件

Dom事件

- 只要是Dom提供的事件，都可以攔截處理
- 如何知道Dom提供了什麼事件
  - https://www.w3schools.com/jsref/dom_obj_event.asp
  - https://www.w3schools.com/jsref/dom_obj_event.asp

Dom Tree

```
<html>
  <head>
    <title>example</title>
  </head>
  <body>
    <h1 class="txt">Hello World</h1>
  </body>
</html>
```



Build School

取得Dom節點

- 透過瀏覽器的document取得
  - `getElementById`
  - `getElementsByTagName`
  - `getElementsByClassName`
- 以上為Method，傳入字串參數，根據方法比對取得Dom

創建Dom節點

- 有兩種方法可以創造新的節點
- 須先取得標的父節點
- 通過.innerHTML或.appendChild()方法增加子節點

創建Dom節點 - innerHTML

```
var target = document.getElementById("標的Id");
target.innerHTML = "<h1>Hellp</h1>";
```

創建Dom節點 - appendChild()

```
var target = document.getElementById("標的id");
var newNode = document.createElement("h1");
newNode.innerHTML = "Hello";
target.appendChild(newNode);
```

該用哪一個新增？

- innerHTML很快，但是會將父節點下原有的節點清除
- appendChild 寫起來麻煩，但是可以遞增的加入新增元素，不會清除掉節點上的狀態

Function

- 以function為起始
- 然後是函式名稱，命名規則如同變數，習慣小寫開頭
- 參數使用()括起來，不需要加上var宣告，以逗點隔開
- 最後加上程式碼區塊

Function 匿名函式

- 將function指派給變數

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

Function 物件

- 可以用組字串的方式，產生程式

```
var add = new Function("a", "b", "return a + b;");
add(1,4); //得到5
```

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";
```

Function

- 函式可以沒有回傳值，如果沒有指定回傳值，卻嘗試取值則，會得到 undefined

```
function doSomething() {
    //沒有return
}

var x = doSomething(); // x = undefined
```

Function 與 Dom事件處理

- 首先撰寫自定義事件處理函式
- 然後與Dom綁定事件
  - 透過屬性標籤
  - 透過Dom屬性

Function 與 Dom事件處理

```
<button id="btn1" onclick="doClick();">按我</button>
function doClick() {
    // 事件處理
};
```

Function 與 Dom事件處理

```
<button id="btn1">按我</button>
var button = document.getElementById("btn1");
button.onclick = function () {
    // 事件處理
};
```

Function 與 Dom事件處理

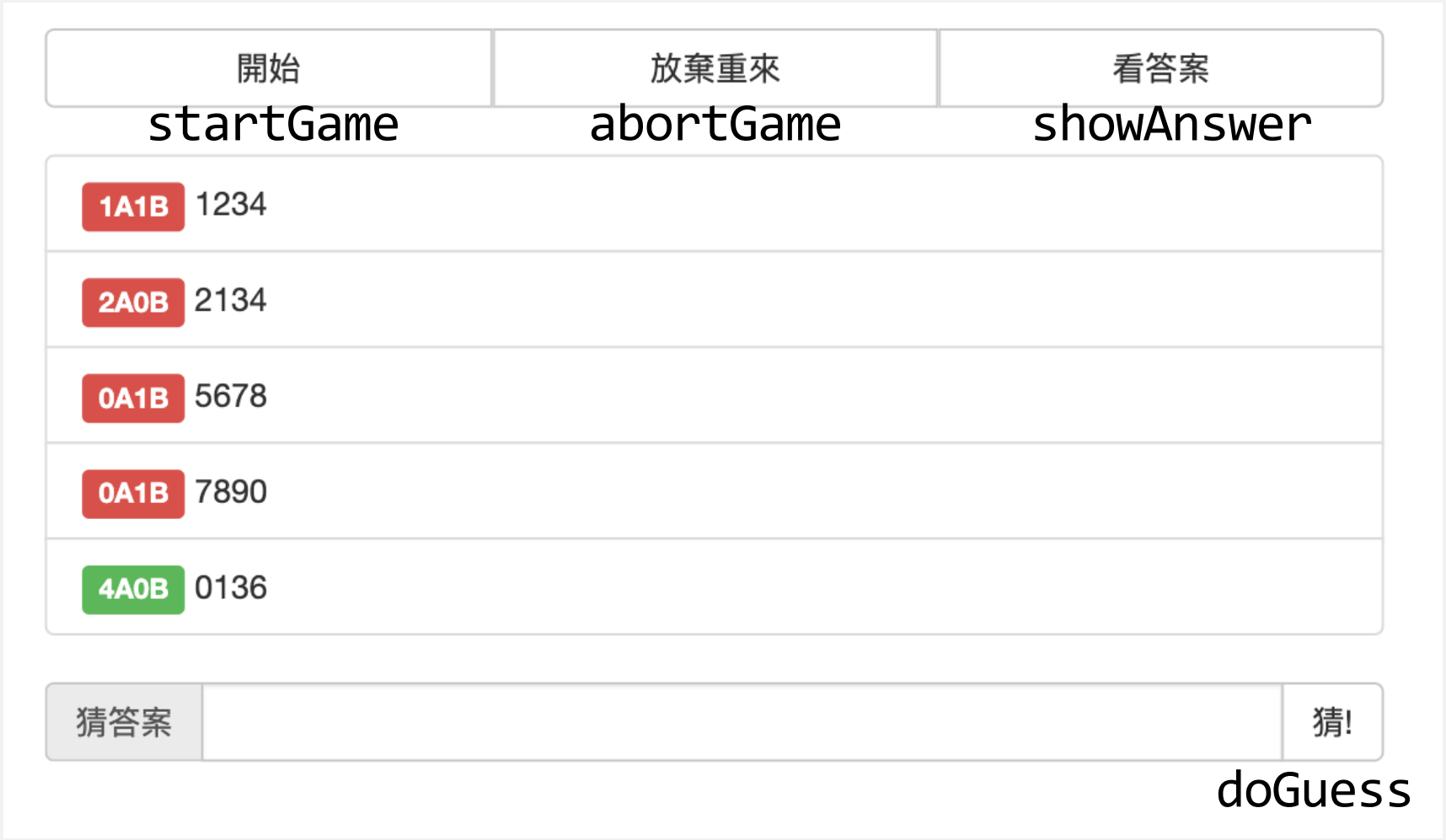
```
<button id="btn1">按我</button>
document.getElementById("btn1").onclick = function () {
    // 事件處理
};
```

1A2B

1A2B 規則

- 遊戲開始時，程式產生一組數字
  - 範圍0~9，不重複的4位數，例如:9487
- 玩家輸入數字，程式回應答對狀態
  - 猜1234，回答 0A1B
  - 猜6789，回答 1A2B

1A2B Function



Tip 如何取亂數?

- Math物件
  - Math.random()回傳0~1之間隨機亂數
- 取 0 ~ 9 整數值
  - Math.floor(Math.random() * 10)

改善使用者體驗

- 每一次輸入數字，都需要用滑鼠點擊猜，因為猜的動作很頻繁，不斷的切換鍵盤滑鼠，感覺很卡
- 能不能輸入完數字，按下Enter開始猜呢？

如何知道使用者在輸入框內按下Enter?

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

事件物件 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
- 事件物件存放事件名稱、何時發生、從何發生等資訊
- 特定的事件，另外包含了特定事件所屬資訊，例如:按下按鍵，就會有按下「哪一個按鍵」的資訊

事件物件 Event Object

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

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