JavaScript客戶端網頁程式設計

授課講師 董淑蕙

教材編寫 董淑蕙

課程綱要

- 課程所需知識及使用環境
- Web Application 簡介
- Programming
- Browser Object Model
- Document Object Model
- Event-driven programming
- Core JavaScript Built-in Objects
- Form validation

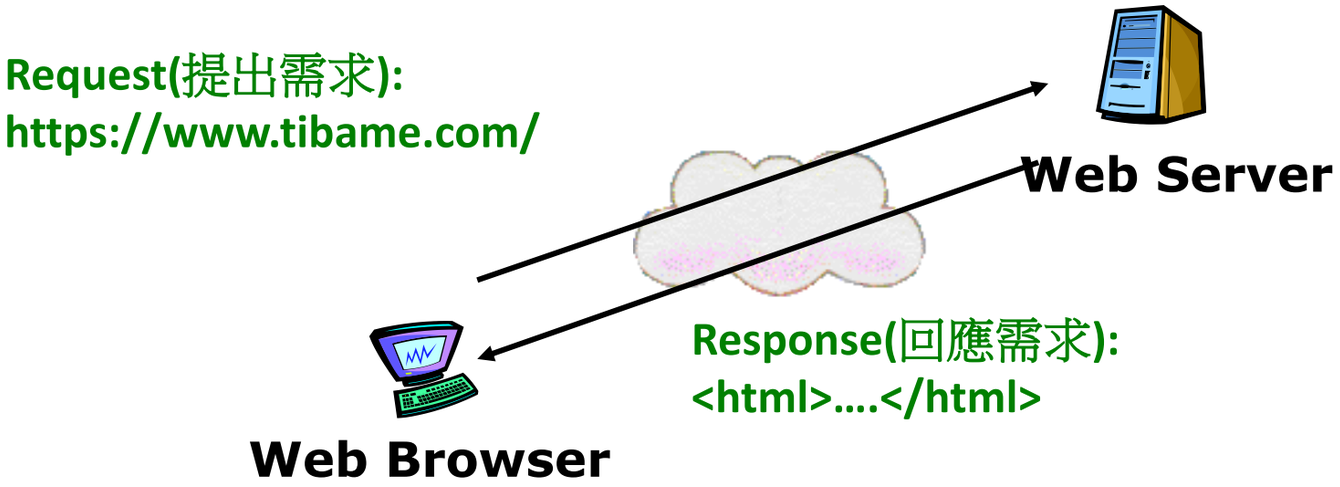
課程所需知識及使用環境

- 課程所需知識
  - HTML網頁設計能力
  - CSS技術基本能力
- 課程使用環境
  - Windows 作業系統
  - Sublime Text編輯軟體
  - 瀏覽器

WWW Application簡介

WWW Application簡介

- World Wide Web全球資訊網
- 伺服器(Server)上置放許多的資訊供客戶端(Client)來擷取



Web application客戶端(client)相關的網頁技術

- 主要有
  - HTML：描述網頁的內容與結構
  - CSS：美化網頁的外觀
  - JavaScript：可以動態的改變頁面中的內容及外觀，並讓使用者能與網頁有更好的互動性
- 衍生的相關技術有
  - Bootstrap工具
  - jQuery函式庫
  - 框架
  - More...

客戶端動態網頁設計的主要功能

- 動態改變頁面中元素的內容
- 動態改變頁面中元素的外觀
- 提昇網頁的趣味性
- 提供使用者與網頁間有更好的互動性
- 分擔伺服端的運算工作
- 減少網路傳輸所需的負荷
- 做一些基本的表單驗證
- More...

Web application伺服端(Server)的動態網頁技術

- 客戶端動態網頁程式設計
  - 雖可做各項動態的圖文整合與展現
  - 却無法讓使用者和資料庫中的資料做互動
    - 如線上查詢、網路購物、群組討論及訪客留言等等，這些工作必須透過伺服端的動態網頁程式設計方可完成。

- 伺服端常使用的語言有
  - PHP
  - Asp.net，Asp
  - Jsp，Java Servlet
  - Python
  - Node.js
  - More…

Programming

程式設計

程式語言簡介_1

- 是一種提供程式設計師用來指示電腦幫忙解決問題的語言。
- 語言的演變
  - 第一代：機器語言(machine language)
    - 是由0與1組成一長串的數字，不容易撰寫及閱讀。
  - 第二代：組合語言(assembly language)
    - 使用一些英文簡寫來代表各種基本運算及功能。
    - 如：ADD、SUB
  - 第三代：高階語言(high level language)
    - 屬於程序性的(procedural)語言，使用一些更容易為人類所接受且簡易的寫法。如：c=a+b;
  - 第四代：查詢語言
    - 語法較接近人類語言。如SQL指令:
    - select name, phone, address from student
  - 第五代：自然語言
    - 比較接近人類使用的自然語言，可用多種不同的敘述方式來陳述。

程式語言簡介_2

- 程式設計就是將解題的方法以程式語言來撰寫，指示電腦完成我們所要的功能。
- 除了機器語言之外，所有的程式語言都需透過翻譯程式將其轉譯為機器語言才能讓電腦來執行。
- 翻譯程式常見的有
  - 組譯程式, 也稱之為組譯器(assembler)
  - 編譯程式, 也稱之為編譯器(compiler)
  - 直譯程式, 也稱之為直譯器(interpreter)

程式語言的構成元素

- 主要包含
  - 基本語法
  - 資料型別與變數
  - 運算式
  - 流程控制
  - 陣列
  - 函數
  - 物件
  - More…

JavaScript簡介_1

- 由網景公司(Netscape Communication Corporation)的Brendan Eich發明
- 最先是在Netscape Nevigator中使用
- 1996年提交給ECMA進行標準化
  - European Computer Manufacturer's Association
  - 歐洲電腦製造商協會

JavaScript簡介_2

- 1997年通過ECMA-262標準，當中記載了ECMAScript的規格
- JavaScript是ECMAScript的實作者
- ECMAScript 版本
- JavaScript也包含了一些不是ECMAScript 規格中的特性
- JavaScript的設計是用來可以放入到不同的環境中來運行
- JavaScript 若寄宿在其它的應用程式環境中(如瀏覽器)，便可以使用程式控制在宿主中(如瀏覽器)的物件完成所要做的事
- 目前主流瀏覽器中都內建JavaScript 直譯程式

課程涵蓋內容_1

- Core JavaScript
  - 變數
  - 資料型別
  - 常數
  - 運算式與運算子
  - 型別轉換
  - 流程控制結構
  - 陣列
  - 函式
  - Built-in objects
    - Object、String、Number、Boolean、Array、Function、Date、RegExp、Math

課程涵蓋內容_2

- Client-side JavaScript
  - BOM objects
    - window
    - location
    - history
    - navigator
    - Console
    - document
  - DOM objects
  - Event-driven programming
    - Event
    - Event handler
  - Form validation
  - 互動特效

JavaScript程式結構

```
// 註解1 ：此為單行註解

/* 註解2：
   此為多行註解
*/
```

變數的宣告

函數的定義

敘述

- 註：
  - JavaScript大小寫有別
  - 敘述以分號(；)代表敘述結束

內嵌在html檔中的JavaScript_1

```
<script type="text/javascript">
   //  註解1 ：此為單行註解
   /*  註解2：
       此為多行註解
   */

   變數的宣告

   函數的定義

   敘述
</script>
```

內嵌在html檔中的JavaScript_2

- 可寫在<head>元素中
- 可寫在<body>元素中
- 可以內嵌多份的<script>…</script>區塊
- 以往通常會將函數或事件處理程式寫在<head>標籤中
- 可以將script寫在<body>元素的最後面
  - 可以提昇使用者瀏覽到網頁內容的時間點
  - 避免有些JS碼要取用的網頁元素放在JS碼之後

引用外部的JavaScript檔

- 使用<script src="js檔名"></script>引用外部js檔案
  - 外部檔案附檔名為.js。
  - <script src ="js檔名"></script>此元素中不要再放入其它的JavaScript程式碼。

- <script async defer src="js檔名">
  - 在script標籤中加async屬性，可以讓js檔以非同步的方式下載，此時
    - 網頁結構可以繼續剖析
    - 待js檔下載完畢之後暫停網頁剖析，先執行JS程式
    - 再繼續剖析網頁上的元素
  - 在script標籤中加defer屬性，可以讓js檔以非同步的方式下載，此時
    - 網頁結構可以繼續剖析
    - 待js檔下載完畢之時仍繼續剖析網頁
    - 直到網頁剖析完畢之後再執行JS程式

類別與物件

- 類別（Class）與物件（Object）
  - 類別：是物件的藍圖，定義出物件應有的特性及行為。
    - 如車子、電腦、 狗狗、 經理、 員工、 客戶、 文字盒、 按鈕...等。
  - 物件：根據類別的定義所做出來的一份實體，又稱為執行個體、執行實體、Instance。
    - 如PI-1689車、Snoopy、Ann...等。

物件與物件的操作

- 物件.屬性
  - document.title="Welcome to Tibame";
- 物件["屬性"]
  - document["title"] ="Welcome to Tibame";
- 物件.方法(參數值1,參數值2,...)
  - document.write("Hello world");
- 物件.on事件屬性
  - btnSend.onclick
- 註：document是文件物件，它提供了許多功能讓我們可以操作文件上的元素。

一個簡單的JavaScript範例

```
<script type="text/javascript" >
  //這是一個簡單的JavaScript範例
  document.write("Hello world!");
</script>
```

寫在其它標籤中的JavaScript程式

- 大部分的JS code
  - 另外存成js檔
  - 寫在<script>標籤中

- 比較特殊的情形
  - 寫在<a>標籤中的JavaScript程式
    - 例：

    ```
    <a href="javascript:window.alert('Hello world!') "> 問候</a>
    ```

  - 寫在一般標籤中的JavaScript程式
    - 參見事件與事件處理程序(在後面的章節)

變數

- 在記憶體中宣告一個儲存空間，用來存放可以變動的資料
- 在JavaScript中宣告變數時
  - 變數是弱型別的模式，宣告時不需指定變數內所要存放的資料之資料型別
  - 變數中的資料，其資料型別可以動態改變
- 語法
  - var 變數名稱;
  - let 變數名稱; (let是ES6新增的功能，有區塊作用域的特性，後面介紹)
- 變數名稱大小寫視為不同
  - 變數名稱取有意義的名稱(a,b,c ? name,price?)
  - 命名的方式可以採用
    - 駝峰命名方式(camelCase), 例: productName
    - Pascal 命名方式(又稱Upper Camel Case) , 例: ProductName
    - 其它, 例: product_name, PRODUCT_NAME…

資料型別與字面值(literals)表示法_1

- 原始型別(primitive), 又稱為純量(scalar)：
  - String資料型別
    - 字面值表示法, 有:"abc"或'abc', 或`abc`(反引號是ES6新增的功能)
  - Number資料型別
    - 字面值表示法, 有
      - 帶小數的資料 : 3.14159
      - 整數 :
        - 十進位: 17 (其十進位的值為17)
        - 八進位: 0o17, 017(其十進位的值為15)
        - 十六進位: 0x17(其十進位的值為23)
        - 二進位:0b1000(其十進位的值為8)

資料型別與字面值(literals)表示法_2

- Boolean資料型別
  - 只有兩個值, 字面值為 : true, false
  - 使用的時機
    - 資料本身的特性是二元的情形
    - 可用來做為決策的依據，根據這個值來決定哪些事要做，哪些事不做
- undefined資料型別
  - 字面值表示法只有一個undefined
  - 表示變數有宣告，但尚未給值
- Null資料型別
  - 字面值表示法只有一個null
  - 表示變數沒有值或沒有物件
  - 通常用在物件參考型別(後面介紹)

常數

- 可以使用const來宣告一個常數。如：
  - const PI = 3.141596;
- 常數在宣告時必需給定一個值
- 常數一經宣告之後就不可再改變其值
- 常數大都使用大寫字母
  - 大部分的程式語言約定成俗
  - 如:Math.PI, Math.SQRT2, PHP_VERSION, DOCUMENT_NODE
  - 註: ES6之後很多js撰寫者常使用一般的變數命名方式來定義常數

樣板字面值Template literals_1

- 以往我們要產生一長串複雜的資料時，經常需要使用較多的程式碼，將所要輸出的資料串接起來。

- 如:

```
str = "<table border='1' cellspacing='0'>";
str += "<tr><td>data</td><td>data</td></tr>";
str += "<tr><td>data</td><td>data</td></tr>";
str += "</table>";
document.write(str);
```

Template literals_2

- ES6提供template literal(樣版字面值)功能，可以簡化產生一長串複雜資料時所需的程式碼
  - 可將要串接的資料放在一對反引號中``
  - 可以將一個字串可以寫成多行
  - 可在反引號字串中使用${ }套入變數及運算
- 例:

```
let name = "Ann";
document.write( `姓名: ${name}<br>` );
```

運算式簡介

- 「運算式」(Expressions)是由「運算子」(Operators)和「運算元」(Operands)組成
  - 例如:
    - 22 * 33 + 55
    - 10 > 5
      - 其中數值 22、33、55 、10、5都是運算元
      - 「+」、「*」、「>」則為運算子

運算式種類

- 算術運算
- 遞增/遞減運算
- 字串運算
- 比較運算
- 邏輯運算
- 位元運算
- 指定運算
- 三元運算
- typeof運算
- 型別轉換

算術運算

- 運算的結果得到一個數值性的資料
  - 假設y的內容值為5

運算子	說明	範例	結果
+	加	x=y+2	x=7
-	減	x=y-2	x=3
*	乘	x=y*2	x=10
/	除	x=y/2	x=2.5
%	取餘數	x=y%2	x=1
++	遞增	++y	y=6
--	遞減	--y	y=4

遞增/遞減運算_1 (Incrementing/Decrementing Operators)

- 遞增/遞減運算子，意指將變數的內容值增加1或減少1。
- 可以置於變數之前或之後
  - 如果在前面，變數值先遞增/遞減，再取值做其它運算
  - 如果在後面，則先取值(以便執行後續運算)，再遞增/遞減變數值

遞增/遞減運算_2 (Incrementing/Decrementing Operators)

- 範例一：

```
x = 10;
y = ++x + 5;
document.write( "x : ", x, "<br>"); ; //11
document.write( "y : ", y, "<br>"); ; //16

x = 10;
y = x++ + 5;
document.write( "x : ", x, "<br>"); ; //11
document.write( "y : ", y, "<br>"); ; //15
```

遞增/遞減運算_3 (Incrementing/Decrementing Operators)

- 範例二：

```
a = 10;
b = ++a + ++a;
document.write( "a : ", a, "<br>"); //12
document.write( "b : ", b, "<br>"); //23

a = 10;
b = a++ + a++;
document.write( "a : ", a, "<br>"); //12
document.write( "b : ", b, "<br>"); //21
```

字串運算

- 運算的結果得到一個字串性的資料
- 運算子： + :字串結合運算子
- 範例 :

  str = "Name : " + "Sara"; (str內容: "Name : Sara")

比較運算

- 運算結果得到一個布林值(可做為決策的依據)

運算子	說明	範例	結果
>	大於	5 > 10	false
>=	大於等於	5 >= 5	true
<	小於	5 < 10	true
<=	小於等於	5 <= 5	true
==	等於	0 == false	true
===	型值相等	0===false	false
!=	不等於	5 != 5	false
!==	型值不等於	5 !==5	false

嚴格的相等比較 ===

- 此種比較，必須型別、值都相等才是true。
- 範例：

```
document.write(0 === 0); // true
document.write("0" === "0"); // true
document.write( 0 === 0.0);   // true
document.write( NaN === NaN); //false
document.write(0 === "0"); // false
document.write(null === undefined); // false
```

寬鬆的相等比較 ==

- 此種比較會先將等號兩側的值轉型成同樣的型別再來做比較。
- 範例：

```
document.write(0 == 0); // true
document.write("0" == "0"); // true
document.write( 0 == 0.0);   // true
document.write( NaN == NaN); //false
document.write(0 == "0"); // true
document.write(null == undefined); // true
```

邏輯運算

- 運算結果得到一個布林值
- 假設x=20, y=30

運算子	說明	範例	結果
&&	and	(x < 30 && y > 1)	true
||	or	(x>5 || y>305)	true
!	not	!(x===y)	true

位元運算(Bitwise Operators)

- 用來執行二進位值的位元運算，通常用在某些較特定的領域
- 範例：

```
a = 10;
b = 6;
document.write( "a & b : ", a & b, "<br>"); //2
document.write( "a | b : ", a | b, "<br>");   //14
```

指定運算_1

- 指定運算
  - 將指定運算子右邊的運算結果指定給左邊的變數
  - 同時也是運算式的結果值
- 範例：
  - x = 2 * 3 + 5
  - x = x + 2
    - 上例可以簡化，改寫成 x += 2 ， += 為指定運算子的一種
  - x = x * 2
    - 上例可以簡化，改寫成 x *= 2 ， *= 為指定運算子的一種

指定運算_2

- 更多的例子...

指定運算子	範例	說明
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
&=	x &= y	x = x & y
|=	x |= y	x = x | y

三元運算(ternary expression)

- 根據條件運算式的結果為true或false來決定運算式的結果是值A或值B
- 語法: 條件運算式？ 值A：值B
  - 條件運算式的結果若為true，則運算式的結果為值A
  - 條件運算式的結果若為false，則運算式的結果為值B
- 範例:

```
avg = 86;
document.write( avg>=60 ？ "通過" ： "再努力!");
```

typeof運算

- 單元運算的一種
  - +, -
  - ++, --
  - !
  - typeof
- 用於取得運算元的資料型別
- 範例:
  - typeof 12 (number)
  - typeof "12" (string)
  - typeof true (boolean)
  - typeof undefined (undefined)

型別轉換_1

- 隱含轉換：又稱自動轉換
  - 不需要額外的程式碼來做轉換
  - 如：
    - 二元運算子 + 可做字串結合也可做數值相加
    - 當 + 的兩側有任一為字串，則會先將另一側也先轉型為字串

+ 運算	結果
數值和字串相加	字串
數值和布林相加	數值
字串和布林相加	字串

  - 單元運算子+
    - +price
  - 單元運算子!
    - !!errMsg

型別轉換_2

- 範例:

```
let str="abc";
let num=123;
let bool=true;

document.write(num + str , "<br>");
document.write(num + bool , "< br >");
document.write(str + bool , "< br >");
document.write(num + false , "< br >");
document.write(num + true  , "< br >");
```

結果:
123abc
124
abctrue
123
124

型別轉換_3

- 強制轉換：又稱外顯轉換
  - 需要額外的程式碼來做型別轉換
  - 如
    - 將資料轉成整數parseInt(資料[，基底])
    - 將資料轉成帶小數的數值parseFloat(資料)

- 範例:

```
str="321abc";
num=parseInt(str);
document.write(num, "<br>");
```

JS中的falsy成員

- 在自動轉型的過程中, 下列資料若轉成布林值會得到false的值
  - 數字零(0, -0)
  - 空字串('',"",``)
  - null
  - undefined
  - NaN
  - false

流程控制結構_1

- 結構化程式設計
- 程式碼大部分是一列指令接著一列指令循序的被執行，但是對於現實生活中的各項複雜工作，我們還需要使用其他的「流程控制結構」才能順利的完成我們所要的工作。

- 結構化程式設計含有三種結構
  - 循序結構：指令一列接著一列循序的被執行
  - 選擇結構：根據某些條件的判斷以決定執行哪一個區塊的程式碼
  - 重覆結構：根據某些條件以決定是否要重複執行某一區塊的程式碼，又稱為迴圈結構

流程控制結構_2

- 循序結構
- 選擇結構
  - if句型
  - switch句型
- 重覆(迴圈)結構
  - for句型
  - foreach句型
  - while句型
  - do.....while句型

選擇結構-if句型_1

- 句型

```
if(條件){
    敘述區塊一
}
[else{
    敘述區塊二
 }]
```

- 敘述區塊以 { } 框住
- 條件成立(true)執行述敘區塊一
- 條件不成立(false)執行述敘區塊二
- 若敘述區塊中的敘述只有一行，可以省略大括號
- 註:
  - 1 : 語法介紹中的**中括弧意指此段結構不一定要在句型中出現**

    [else {
    程式區塊N
    }]

  - 2 : 範例

選擇結構-if句型_2

- 範例：

```
if(avg>=60){
   document.write( "成功!<br>");
   document.write("恭喜<br>“);
}
else{
   document.write("再努力!<br>");
   document.write("加加油!!!<br>");
}
```

選擇結構-if句型_3

- 當我們的條件無法以簡單的二分法決定流程時，可以使用下列句型

```
if (條件式一){
    程式區塊一
} else if (條件式二){
    程式區塊二
} else if (條件式三){
    程式區塊三
    .....
}[else {
        程式區塊N
}]
```

選擇結構-if句型_6

- 範例

```
avg = 88;
if(avg >= 90) {
  grade = "A";
}else if(avg >= 80) {
      grade = "B";
}else if(avg >= 70) {
      grade = "C";
}else if(avg >= 60) {
      grade = "D";
}else{
   grade = "E";
}
document.write("等級 : ", grade);
```

插播 ~~

var 和 let 兩種變數宣告的差異

插播--var 和 let兩種變數宣告的差異

- 使用var宣告，有下列特性
  - 有hoist的現象
  - 可以重覆宣告
  - 没有區塊作用域，在區塊內宣告的變數在區塊外仍可以使用
- 使用let宣告，有下列特性(es6才提供)
  - 没有hoist的現象
  - 不可以重覆宣告
  - 有區塊作用域，在區塊內宣告的變數只能在區塊內使用
  - for句型的作用範圍(後面介紹)
- 在JavaScript中，變數未經事先宣告
  - 若要取值會引發錯誤
  - 若要設定值則是被允許的，如: amount=3;

選擇結構-switch句型_1

- 根據運算式的結果來選擇程式區塊中要執行的進入點在哪裡

```
switch(運算式){
    case 值1:
        敘述區塊1
        [break;]
    case 值2:
        敘述區塊2
        [break;]
    …
    case 值n:
        敘述區塊n
        [break;]
   [ default:
        敘述區塊n+1 ]
}
```


JavaScript
董淑蕙編著【版權所有，不得任意拷貝或引用】

選擇結構-switch句型_2

- 範例

```
let amount;
let  lowType="2";   //根據lowType來決定應執行的程式段落
 switch(lowType){
   case "1" :
          amout = 6000;
          break;
   case "2" :
          amout = 4500;
          break;
   case "3" :
          amout =  3000;
               break;
   default :
          amout =  0;
}
document.write("補助款  : ",  amount );
```

重覆結構-for句型_1

- 句型一

```
for(初始設定; 執行條件; 增或減運算式){
  敘述區塊      //重覆要執行的敘述有一個以上
  [break;]
  [continue;]
}
```

- 句型二

```
for(初始設定; 執行條件; 增或減運算式)
  敘述;      //重覆要執行的敘述只有一個
```

重覆結構-for句型_2

- 說明：
  - 步驟

1.變數初始設定只做一次
2.若測試條件成立才執行敍述，不成立則結束重覆結構
3.每次做完要重覆執行的敍述之後，加上遞增值或減去遞減值回到步驟2

  - break：可用來強制終止迴圈的執行

  - continue：迴圈內continue指令後的敍述不執行，直接執行下一迴圈

重覆結構-for句型_3

- 範例：

```
//使用for句型,顯示10列..I love JavaScript
let i;
for (i=1; i<=10; i++) {
    document.write("I  love JavaScript<br>");
}
```

- 練習...

重覆結構-foreach句型_1

- 句型：

```
for(鍵值變數 in 物件變數){
    敘述區塊
}
```

- 說明：
  - 語意上是foreach，意指這個物件中有多少資料就執行幾次
  - 語法上只要寫for即可
  - 可使用這種句型操作物件中的資料
  - 迴圈每次執行時都會取得該資料的鍵值放在鍵值變數中, 可以是索引或屬性名稱

重覆結構-foreach句型_2

- 句型：

```
for(資料變數 of 物件變數){
    敘述區塊
}
```

- 說明：
  - 語意上是foreach，意指這個物件中有多少資料就執行幾次
  - 語法上只要寫for即可
  - 可使用這種句型操作物件中的資料
  - 迴圈每次執行時都會取得物件中的資料放在資料變數中

重覆結構-while句型_1

- 語法

```
while (執行條件){
  敘述區塊
    [break;]     [continue;]
}
```

- 說明
  1.先檢查執行條件是否為true，
  2.若為true則執行迴圈內的敘述，執行完畢之後回到while（執行條件）處重新檢查，以決定是否重覆執行
  3.若為false則不執行迴圈內的敘述，直接跳到while句型之後去執行

重覆結構-while句型_2

- 範例

```
//使用while句型,顯示10列..I love JavaScript
let i=1;
while(i<=10){
      document.write("I  love JavaScript~~<br>");
       i++;
}
```

- 練習...

重覆結構-do...while句型_1

- 語法

```
do{
  敘述區塊
    [break;]     [continue;]
} while (執行條件);
```

- 說明

1.無條件先執行一次迴圈內的敘述，執行完畢之後再檢查while後的執行條件是否為true，
2.若為true則繼續執行迴圈內的敘述
3.若為false則結束迴圈結構，直接跳到do...while句型之後去執行

重覆結構-do...while句型_2

- 範例

```
//使用while句型,顯示10列..I love JavaScript
let i=1;
do{
      document.write("I  love JavaScript<br>");
       i++;
} while(i<=10);
```

- 練習...

重覆結構-為什麼要提供三種重覆結構?

- 三種重覆結構都可以達到重覆執行的功能 (for, while, do...while)
- 但因為其字面上的**語意不同**，我們可以選擇較適合的句型，來讓程式具有自我說明的能力

重覆結構-巢狀迴圈_1

- 巢狀迴圈是在迴圈內擁有其它迴圈
- 例如：
  - 在for迴圈擁有for、while或do/while迴圈
  - while迴圈內有for、while和do/while迴圈
- 巢狀迴圈可以有很多層，二、三層...都可以。

重覆結構-巢狀迴圈_2

- 以九九乘法表為例，迴圈共有兩層，第一層迴圈執行9次，當i為1時，第二層迴圈要執行9次(j的值，從1到9)

第一層迴圈以 i 來控制執行次數	第二層迴圈以 j 來控制執行次數								
1	1	2	3	4	5	6	7	8	9
2	1	2	3	4	5	6	7	8	9
3	1	2	3	4	5	6	7	8	9
……									
9	1	2	3	4	5	6	7	8	9

Core JavaScript Built-in

- String
- Number
- Date
- Math
- Object
- Array
- Boolean
- Function
- RegExp

Math物件_1

屬性名稱	說明
Math.E	自然數 e=2.718281828459045
Math.LN2	自然對數2,ln2
Math.LN10	自然對數10,ln10
Math.LOG2E	Log_2 e
Math.LOG10E	Log e
Math.PI	圓周率π=3.141592653589793
Math.SQRT1_2	0.5的平方根
Math.SQRT2	2的平方根

Math物件_2

方法	說明
Math.min(值1,值2,...)	傳回參數列中的最小值
Math.max(值1,值2,...)	傳回參數列中的最大值
Math.random()	傳回一個介於0~1間的亂數值
Math.round(x)	傳回x值四捨五入後的值
Math.ceil(x)	傳回等於或大於x的最小整數
Math.floor(x)	傳回等於或小於x的最大整數

Math物件_3

方法	說明
Math.exp(x)	e^x
Math.log(x)	$\log_e x$
Math.pow(x,y)	x的y次方
Math.sqrt(x)	x的平方根
Math.abs(x)	x的絕對值
sin,cos,tan, asin,acos,atan,..	三角函數
More…	

陣列

- 陣列簡介
- 一維陣列的建立
- 一維陣列的處理
- 整體陣列的操作
- 陣列的屬性和方法
- 二維陣列的建立
- 二維陣列的操作

陣列簡介_1

- 主要功能：有一群資料
  - 它們所代表的意義及被拿來處理的過程相似
  - 透過陣列的使用及迴圈技巧的應用，可以有效而簡便的處理

- 例如: 有一群資料，他們所代表的意義相同，被拿來處理的過程也相近，如班上40位同學的各科成績，其被用來計算學期成績的過程是一樣的

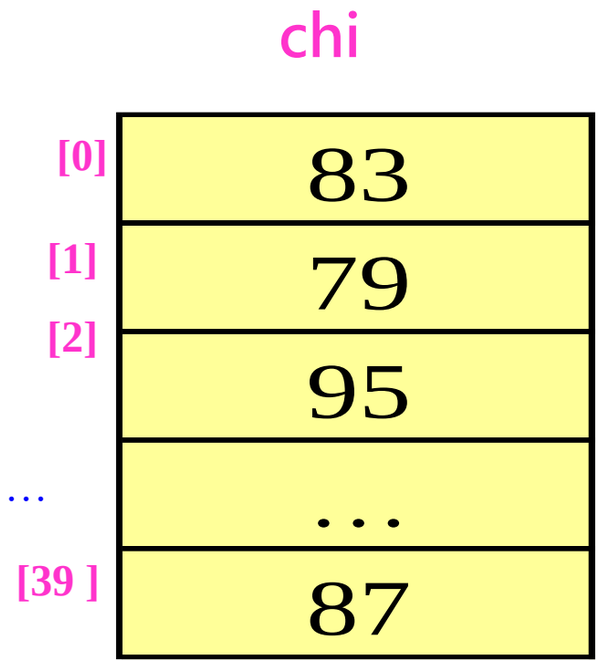
```
chi1=83
chi2=79
chi3=95
…
chi40=87
```

```
eng1=85
eng2=77
eng3=91
…
eng40=85
```

```
avg1= (chi1 + eng1)/2
avg2=( chi2 + eng2)/2
avg3= (chi3 + eng3)/2
…
avg40 = (chi40 + eng40)/2
```

陣列簡介_2

- 以宣告陣列的方式，替我們所要處理的資料在記憶體中要出一塊區域
- 改以陣列存放，如下所示：



chi[0]=83;

document.write(chi[0]);

一維陣列的建立_1

- JS中的陣列以物件的型式存在
- 語法:
  - let陣列變數 = new Array(元素個數)
  - let陣列變數= new Array(陣列元素1, 陣列元素2,...)
- 使用陣列變數 [索引]來取得陣列中的資料
- 陣列在JavaScript中可以動態的配置
- 在JavaScript中，陣列中的資料們，不需具備有相同的資料型別

一維陣列的建立_2

- 例:
  - let arr = new Array(5);
    - 共有五個資料: arr[0],arr[1],arr[2],ary[3],arr[4]
    - 內容值均為undefined
  - let arr = new Array(11,22,33,44,55);
    - 共有五個資料: arr[0],arr[1],arr[2],arr[3],arr[4]
    - 內容值分別為11,22,33,44,55

一維陣列的操作_1

- 使用陣列變數 [索引]來取得陣列中的資料
  - 例:

```
document.write(arr[0] ,  "<br>");
document.write(arr[1] ,  "<br>");
```

- 可以透過迴圈的技巧來處理陣列中的每一個資料。例:

```
for(i=0; i<=7; i++) {
    document.write(arr[i] + "<br>");
}
```

一維陣列的操作_2

- 陣列物件可以透過length屬性來取得陣列的長度。
  - 例:

```
for(i=0; i<arr.length; i++) {
      document.write(arr[i] + "<br>");
}
```

陣列的字面值表示法

- 陣列字面值：Array literal

- 語法:

  let陣列名稱 = [data1, data2, ……];

- 例:
  - var arr = [];
  - var arr = [11, 22, 33];

一維陣列的操作_3

- 常見的陣列操作
  - 如何將陣列轉成字串後輸出?
  - 如何將資料insert到陣列的最後面?
  - 如何將資料insert到陣列中的特定位置?
  - 如何得知資料是否在陣列之中?
  - More...

陣列所提供的方法

- 常用的方法

方法	說明
sort()	排序
toString()	將陣列中的元素以逗點結合起來,傳回一個字串
reverse()	將陣列中的資料順序反轉
join(字元)	將陣列中的元素以特定符號結合起來,傳回一個字串
concat(參數...)	將陣列結合成一個新的陣列，並傳回該陣列
push(資料)	將新的元素加到陣列的後面,並傳回陣列長度
pop()	傳回陣列中的最後一個元素,並移除該元素
slice(start,end)	拷貝自索引值start處到end-1處的子陣列後傳回(不會從原陣列中移除)
splice(start,n,ele1,ele2..)	將陣列中索引值start處後的n個元素移除並傳回,然後插入資料ele1,ele2,..
shift()	傳回陣列中的第一個元素,並移除該元素
unshift(資料)	將新的元素加入陣列的最前面
indexOf(data,start)	自索引start處找出data在array中第一次出現的位置之索引值
forEach	提供一個函式給陣列中的每一個元素去執行

- More...

陣列物件參考的操作

- 當把陣列變數A的物件參考指定給陣列變數B，則它們參考到同一個陣列物件。也就是透過任一陣列變數(如B) 去修改陣列中的值後，經由另一陣列變數(如A)去取值時，也會得到那個新的異動結果。
- 此亦稱為淺拷貝(shallow copy)多個物件指向的位址相同，彼此之間的操作會互相影響。
  - 例：

```
let arrA = new Array(11,22,33);
let arrB = arrA;
arrB[2] = 300;
document.write( arrA[2] ); //印出300
```

- 深拷貝(deep copy): 拷貝出來的物件和原物件具有各自的空間，互相獨立不會彼此影響。在JS中要自行設計。

二維陣列的建立

	ch
0	83
1	79
2	95
…	…
39	87

	en
0	85
1	77
2	91
…	…
39	85

	avg
0	84
1	78
2	93
…	…
39	86

將上面三個一維陣列改以一個二維陣列來存放，其中score[2][1]內的值為91

	0	1	2
0	83	85	84
1	79	77	78
2	95	91	93
…	…	…	…
39	87	85	86

二維陣列的建立

- 先建立一維陣列，再於其元素中放入另一個陣列
  - 例：

```
//先建一維陣列
let arr= new Array(3);

//於陣列元素中再放入另一個一維陣列
arr[0] = new Array(1,2,3,4);
arr[1] = new Array(11,12,13,14);
arr[2] = new Array(21,22,23,24);
```

二維陣列的操作

- 取得陣列中的元素：

  陣列變數[1維索引] [2維索引]，如：arr[2][3]

- 例：印出二維陣列中的每一個元素

```
for(i=0 ; i<3; i++) {//陣列中第i列
    for(j=0; j<4; j++){
        //陣列中第i列內第j行的元素
        document.write( arr[i][j] + " " );
    }
    document.write("<br>");
}
```


JavaScript
董淑蕙編著【版權所有，不得任意拷貝或引用】

函式

- 函式簡介
- 內建函式
- 自訂函式
- 變數的可用範圍

函式簡介

- 主要功能: 將一段具有某種特定功能的程式碼另外包裝成獨立的函式
- 可減少程式碼的重複性並增進程式的再用性
- 可增加程式的可讀性並讓程式更易於維護

- 通常函式分為兩類
  - 內建函式(Built-in functions)
    - Js的內建函式大部分放到相關的類別定義中，以方法的形式來呼叫
      - 可以透過類別來呼叫, Math.sqrt(9)
      - 可以透過物件來呼叫, array.toString()
    - 全域物件的方法可以直接呼叫方法名稱, 如:
      - window.alert("Hello~");
      - alert("Hello~");
  - 自訂函式(User-defined functions):由使用者自訂其功能

全域函式

- 全域函式：可直接在全域範圍中被呼叫，不用從某個物件取得後呼叫

方法	說明
parseInt(字串,基底)	將字串資料轉成整數資料
parseFloat(字串)	將字串資料轉成浮點數資料
eval(字串)	將字串中的資料當作敘述來執行
isFinite(testValue)	是否為有限大的數
isNaN(testValue)	是否不是一個有效的數值

自訂函式_1

- 語法

```
function 函式名稱( [參數1 [, 參數2 [,...] ] ] ){
  //函式主體...
  ...
  [ [return 值;]| [return;] ]
}
```

- 函式名稱取有意義的名稱
- 視函式的功能決定參數的個數及是否需傳回值

JavaScript

董淑蕙編著【版權所有，不得任意拷貝或引用】

自訂函式_2

- 不需參數的函式

```
function sayHello(){
  document.write("Hello world! <br>");
  //沒有傳回值
  return; //此例中，return可以省略
}
```

- 需要一個參數的函式

```
function sayHelloToSomeone(name){
  document.write("Hello "+name+"<br>");
  //沒有傳回值， return可以省略
}
```

自訂函式_3

- 需要a,b兩個參數的函數

```
function sum(a,b) {
  return a+b;  //有傳回值
}
```

- 其它.....
  - 根據案例的需要，做不同的設計

函式的特性_1

- 函式本身是一個物件
- 函式可以被重覆定義

```
function sum(a, b){
    let total = 0;
    total = a + b;
    return total;
}
function sum(a, b, c){
    let total = 0;
    total = a + b + c;
    return total;
}
```

– 同名的函式，後面函式的會覆寫前面的函式

不支援overload

overload : 指定有多個名稱相同的方法，但參數列(parameter)不同，呼叫函式時可以依據參數列的比對而執行所要的函式。但JS不提供這項功能。

函式的特性_2

- 函式中有arguments屬性
  - 只能在函式中使用
  - arguments是一個物件 : 內含呼叫此函式時的所有實際引數資料
  - 取用引數資料: arguments[0], arguments[1], …
  - 有length屬性記錄其個數
- 應避免與瀏覽器的全域方法使用相同的名稱，否則會覆寫(Override)掉window所提供的功能

函式的進階特性

下面特性在後面介紹

- 函式物件可被當做參數放入函式中來傳遞
  - 如：
    - array.sort(compareFunction)
- 可以建立匿名函式
- 函數中的this用來參考到呼叫此函式的物件

插播--邏輯運算有short circuit的快捷運算法

- 當已預知運算式結果的情況下，停止評估其他運算
- 目的是來快速計算運算式的結果
- 通常用於邏輯運算，如&&和||
  - 運算元1 && 運算元2 ：
    - 如果運算元1為 false，則已知運算式結果為false。立即返回運算元1的值，且不評估運算元2
    - 如果運算元1為true，則繼續評估運算元2
    - 結果是最後一個被評估的值
  - 運算元1 || 運算元2 ：
    - 如果運算元1為 true ，則已知運算式結果為true。立即返回運算元1的值，且不評估運算元2
    - 如果運算元1為false，則繼續評估運算元2
    - 結果是最後一個被評估的值

Default parameter value

- 問題：

宣告函式時，若參數列中有宣告參數，但在呼叫函式時卻未提供實際引數值，則在函式中用到參數時會得到一個undefined的值。若程式没有處理好，可能會造成函式的運行結果不正確。

- 解決方法：
  - 提供參數給預設值的方法
  - 可以在參數列中指定參數的預設值

- 例：

```
function sayHello(name="buddy", word="Hi~"){
   document.write( name , "說 : ", word, "<br>" );
}
sayHello("Adam", "午安~");
sayHello();
```

Rest parameters

- ES6提供rest parameters (剩餘參數)的寫法，讓我們可以將特定參數之後的所有實際引數資料直接指定給另一個參數

- 語法：

```
function fucName(a, b, ...restArgs) {  ... }
```

- rest parameters和arguments不同
  - rest parameters只包含没有對應到參數的所有實際引數資料，而arguments包含所有的實際引數資料
  - rest parameters是陣列，arguments不是陣列

變數的可用範圍-全域變數

- 全域變數
  - 使用var或let 宣告在函式外皆為全域變數
  - 函式外，未宣告直接使用之變數亦為全域變數

- 例：

```
var  a;
a = 10;

var  b = 20;

c = 30;
```

變數的有效範圍-區域變數_1

- 區域變數
  - 區域變數是在進入函式時被建立出來的，只能在此函式中使用，函式結束時若不再被使用便會將此變數空間收回
  - 使用var或let宣告在函式中者皆為區域變數
  - 函式中，參數列中的變數也是區域變數
  - 函式中使用變數時，會先從函式內部開始找
    - 若找得到，直接使用此函式中的變數
    - 若找不到則會往外找

變數的有效範圍-區域變數_2

- 例：

```
function sum(a,b){   //a,b是區域變數
    let c ;          //c是區域
    c= a+b;
    return c;
}
let x=10, y=20;      //x,y是全域變數
document.write( sum(x,y), "<br>");
```

BOM

Browser Object Model

BOM

- 幾乎所有的瀏覽器都為 JavaScript 提供了一組物件及相同的方法和屬性，通常將其統稱為 BOM
- 整個BOM的源頭是window物件
- 透過window物件可以存取許多的屬性、方法和物件, 如:
  - window.navigator
  - window.history
  - window.location
  - window.document
  - window.console
  - window.alert()
  - More…

navigator物件

- 主要提供了很多的唯讀屬性,透過這些屬性我們可以取得瀏覽器和系統資源的相關資訊。

屬性名稱	說明
appCodeName	瀏覽器代碼
appName	瀏覽器名稱
appVersion	瀏覽器版本
userAgent	使用者代理器
platform	瀏覽器所在的作業平台
browserLanguage	瀏覽器所使用的語系
product	瀏覽器引擎

window物件_1

- window物件參考到瀏覽器本身

屬性名稱	說明
name	視窗的名稱
document	文件物件
history	歷史物件
location	位置物件
screen	螢幕物件
console	主控台物件
innerHeight	視窗內的高度(含捲軸)
innerWidth	視窗內的寬度(含捲軸)
outerHeight	視窗的高度(含捲軸、工具列)
outerWidth	視窗的寬度(含捲軸、工具列)
pageXOffset	文件左上角被捲出的寬度
pageYOffset	文件左上角被捲出的高度
More...	

window物件_2

方法名稱	說明
alert(訊息)	顯示訊息對話盒
confirm(訊息)	顯示訊息對話盒,並要求使用者選擇是或否
prompt(訊息[,預設值])	顯示訊息對話盒,並要求使用者輸入資料
open(URL,視窗名稱)	開啟一個新視窗
close()	關閉視窗
focus()	取得焦點
blur()	去除焦點
setInterval(指定程式,間隔時間)	設定指定的程式,經過間隔時間之後會被週期性地重覆被執行(時間單位:毫秒,會傳回計時器編號)
clearInterval(計時器編號)	取消setInterval所設定的重覆執行某函式的工作
setTimeout(指定程式,間隔時間)	設定指定的程式,經過間隔時間之後會被執行一次(時間單位:毫秒,會傳回計時器編號)
clearTimeout(計時器編號)	取消setTimeout所設定要執行某函式的工作
print()	列印網頁
More...	

location物件_1

- 參考到瀏覽器的網址列

屬性名稱	說明
hostname	伺服器網址
href	完整的URL字串
host	URL的hostname:port
port	所使用的通訊埠
pathname	網頁檔案的名稱和路徑
search	查詢字串

- 設定location.href = url，可以令瀏覽器跳轉到另一頁(url)

location物件_2

- 例:若網址列中的資料為

http://Tibame.com:8080/phpLab/index.php?type=A&id=11

  - hostname : Tibame.com
  - href : http://Tibame.com:8080/phpLab/index.php?type=A&id=11
  - host : Tibame.com:8080
  - port : 8080
  - pathname : /phpLab/index.php
  - search : ?type=A&id=11

location物件_3

方法名稱	說明
reload	重新載入目前所瀏覽的文件
replace	換掉目前所瀏覽的文件

- location.replace(url) : 會將目前瀏覽的這一頁換成url這個網頁

history物件

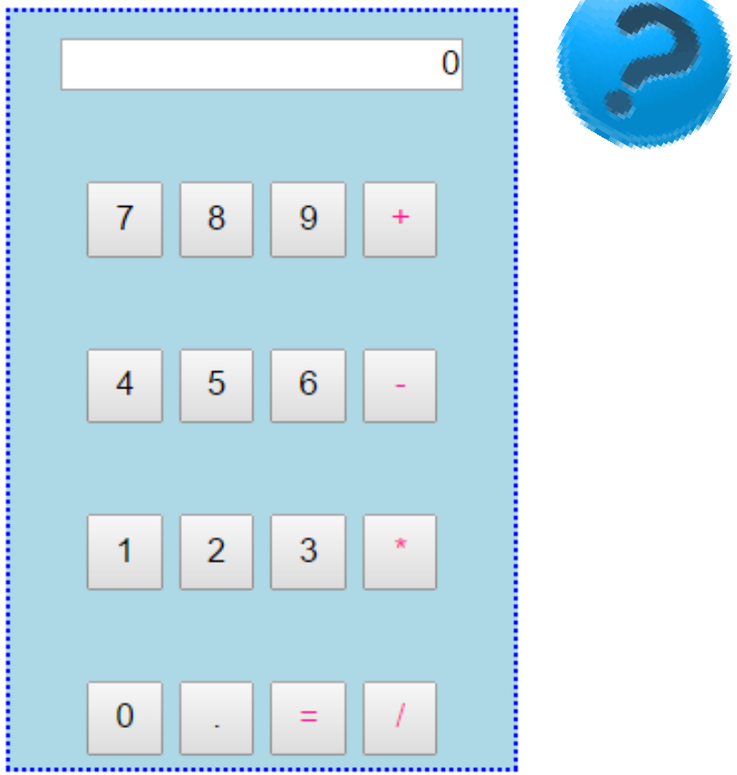
- 用來記錄最近瀏覽過的歷史網頁資訊
- 屬性:length
- 方法:

方法名稱	說明
back()	回到上一頁
forward()	移到下一頁
go(n)	n>0,往下移n頁;n<0,回到上n頁

DOM 簡介_1

當使用者點按頁面上的按鈕時，如何讓JS程式碼可以取得頁面上的元素，進行各項互動機制及動態效果??

範例一
範例二



DOM 簡介_2

- Document Object Model
- 瀏覽器將網頁文件以一群物件的方式架構起來
- 我們可以透過對這些物件的操作, 動態的來改變網頁的內容或網頁的外觀,使網頁有豐富的變化,而不只是一份靜態的網頁而已。
- 當瀏覽器將我們所要瀏覽的網頁文件打開來後，便將該網頁文件**物件化**成為 document 物件
- 使用document物件
  - window.document
  - document (window.可省略)

DOM 簡介_3

- 網頁文件一如下:

<html><head><title>DOM world</title></head><body><h1>Welcome to W3c DOM</h1> Treated as a tree. </body></html>

- 網頁文件二如下:

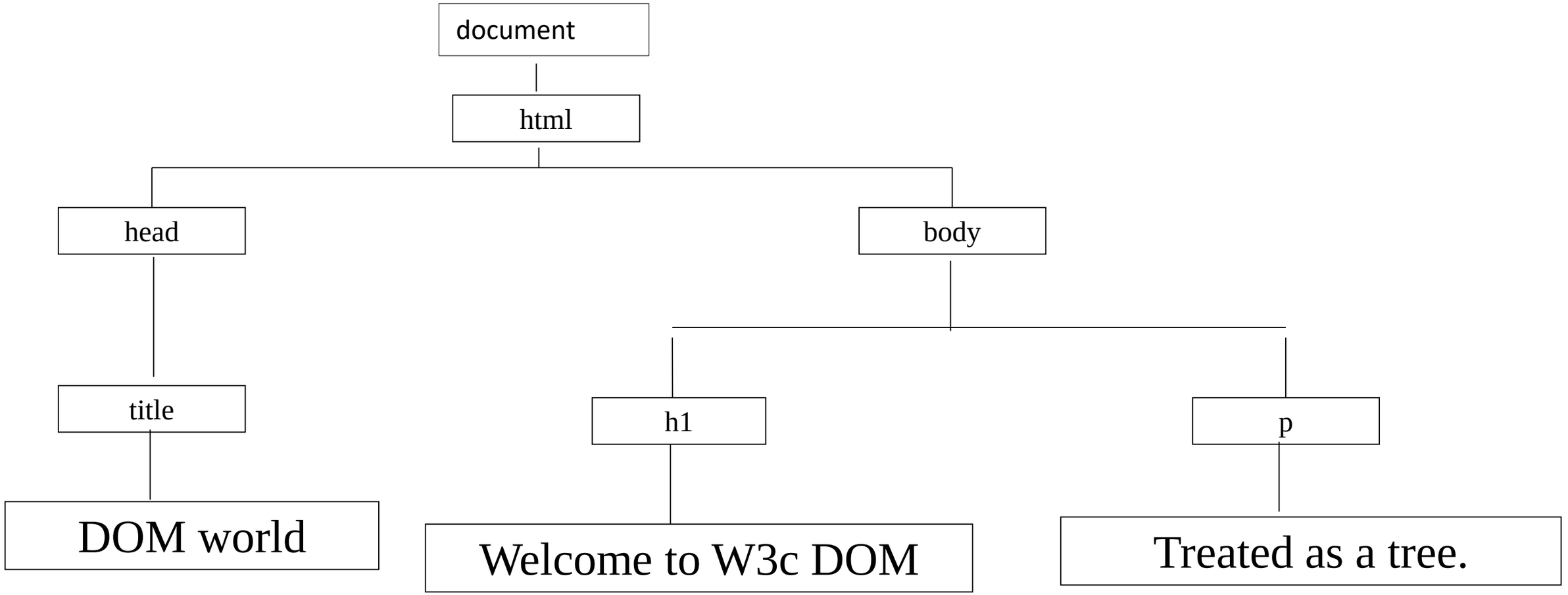
```
<html>
  <head>
    <title>DOM world</title>
  </head>

   <body>
    <h1>Welcome to W3c DOM</h1>
    <p>Treated as a tree. </p>
  </body>
</html>
```


117

DOM 簡介_4

- W3C將HTML文件視為文件樹, 上列的文件一



118

DOM 簡介_5

- 透過document 物件，我們可以動態操作網頁的各項功能：
  - 改變頁面中的所有 HTML 元素
  - 改變頁面中的所有 HTML 屬性
  - 改變頁面中所有的 CSS 樣式
  - 可以刪除現有的 HTML 元素和屬性
  - 可以添加新的 HTML 元素和屬性
  - 可以對頁面中現有的 HTML 事件做出回應

DOM 簡介_6

- 網頁文件上放置了眾多的元素，一些常見並且我們經常會去操作的元素主要有
  - 非表單欄位物件，如：
    - <h1> 、<p> …
    - 、 …
    - <section> 、<nav>…
  - 表單欄位物件，如：
    - <input type="text"> 、 <input type="password" > 、 <input type="radio"> 、 <input type="checkbox"> …
    - <select>、<textarea>
- 我們可透過document物件所提供的方法來找到它們

document物件_1

- 屬性

屬性名稱	說明
title	網頁文件的標題
bgColor	背景顏色
fgColor	前景顏色
linkColor	可連結且未連結過的顏色
vlinkColor	已被連結過的顏色
alinkColor	作用中連結的顏色
lastModified	檔案的最新更新日期
documentElement	文件的根元素
head	文件中的head元素
body	文件中的body元素
More...	

document物件_2

- 方法

方法名稱	說明
write(data1,data2,..)	顯示資料到網頁文件中
getElementById(idname)	經由id取得某特定元素
getElementsByName(name)	經由name取得一些元素
getElementsByTagName(tagName)	經由標籤名稱取得一些元素
getElementsByClassName(className)	經由類別名稱取得一些元素
querySelector()	經由css selector來取得某元素
querySelectorAll()	經由css selector來取得一些元素們
createElement()	創建新元素
createTextNode()	創建文字節點
More...	

找到文件上元素物件的參考

- 可使用id、name 、標籤名稱、類別名稱或css選擇器等，

找到網頁文件上的物件參考：

  - document.getElementById("id名稱")
  - document.getElementsByName("name名稱")
  - document.getElementsByTagName("標籤名稱")
  - document.getElementsByClassName("class名稱")
  - document.querySelectorAll("css selector")
  - document.querySelector("css selector")

- 其中**getElementById()**和**querySelector()**只傳回**單一物件的參考**，其他的方法則傳回類似集合物件的參考
- 範例

找到文件上元素物件的參考_案例一：document.getElementById( )

– 範例：

```
<h3 id="curPos"></h3>

<script>
    let curPos = document.getElementById("curPos");
</script>
```

找到文件上元素物件的參考_案例二：document.getElementsByName( )

- 若只要取得某特定元素且有用id來命名者, 可以document.getElementId()來取得元素物件的參考
- 此例使用document.getElementsByName( )來取得name為特定值的的所有元素集合
- 傳回值: 將找到的元素依它們在樹中出現的先後順序排列傳回類似集合的NodeList。
  - NodeList
    - 是類陣列(array-like)的元素集合
    - 可用索引值0, 1, 2, …來存取
    - 有length屬性記錄它的個數.
- 範例：

```
搜尋<input type="text" name="searchData" id="searchData">
<script>
 let searchDataObjects =  document.getElementsByName("searchData"); //類似集合
 let searchDataObject = document.getElementsByName("searchData")[0];//單一元素
</script>
```

存取HTML元素物件的內容

- 非表單欄位元素，有起始標籤和終止標籤的元素
  - innerHTML：元素內的子標籤和文字內容
  - innerText：元素內的文字
    - 只取得元素內可見的文字內容(display: none或visibility: hidden的內容不會取得)
  - textContent: 元素內的文字
    - 可取得元素內的所有文字內容(包括隱藏和不可見的內容)
- 表單欄位元素(以輸入盒為例)，可透過value屬性取得或設定輸入盒內的值

存取HTML元素物件的內容_案例一

- 非表單欄位元素範例

```
<h3 id="curPos"></h3>

<script>
    let curPos = document.getElementById("curPos");
    curPos.innerHTML="首頁>>會員專區";
</script>
```

存取HTML元素物件的內容_案例二

- 表單欄位元素範例(以輸入盒為例)

```
搜尋<input type="text" name="searchData">

<script>
 let searchDataObject=document.getElementsByName("searchData")[0];
searchDataObject.value="請輸入搜尋關鍵字";
</script>
```

改變HTML元素物件外觀的樣式

- 透過『物件.style.樣式屬性名稱』可取得或改變標籤物件的css樣式值

  – 範例

```
<div>歷程 :
<span id="curPos" >1111</span>
</div>

<script>
document.getElementById("curPos").style.color="blue";
</script>
```

找到文件上某些元素物件的集合

下列方法都會回傳類陣列的集合物件

- 集合物件中的元素物件以它們在網頁中的先後順序排列放在集合中
- 都可以使用索引值0、1、2、...取得在集合中的元素
- 此集合物件有length屬性，記錄集合內元素的個數

- document.getElementsByName("name名稱")
  - [object NodeList]
- document.getElementsByTagName("標籤名稱")
  - [object HTMLCollection]
- document.getElementsByClassName("class名稱")
  - [object HTMLCollection]
- document.querySelectorAll("css selector")
  - [object NodeList]

Element物件_1

Property / Method	Description
clientLeft	Returns the width of the left border of an element
clientTop	Returns the width of the top border of an element
clientWidth	Returns the width of an element, including padding
clientHeight	Returns the height of an element, including padding
scrollLeft	Sets or returns the number of pixels an element's content is scrolled horizontally
scrollTop	Sets or returns the number of pixels an element's content is scrolled vertically
scrollWidth	Returns the entire width of an element, including padding
scrollHeight	Returns the entire height of an element, including padding
tagName	Returns the tag name of an element

Element物件_2

Property / Method	Description
id	Sets or returns the value of the id attribute of an element
title	Sets or returns the value of the title attribute of an element
innerHTML	Sets or returns the content of an element
innerText	Sets or returns the text content of a node and its descendants
textContent	Sets or returns the textual content of a node and its descendants
style	Sets or returns the value of the style attribute of an element
className	Sets or returns the value of the class attribute of an element
classList	Returns the class name(s) of an element

Element物件_3

Property / Method	Description
blur()	Removes focus from an element
click()	Simulates a mouse-click on an element
addEventListener()	Attaches an event handler to the specified element
getElementsByClassName()	Returns a collection of all child elements with the specified class name
removeEventListener()	Removes an event handler that has been attached with the addEventListener() method
getElementsByTagName()	Returns a collection of all child elements with the specified tag name
querySelector()	Returns the first child element that matches a specified CSS selector(s) of an element
querySelectorAll()	Returns all child elements that matches a specified CSS selector(s) of an element

Element物件_4

Property / Method	Description
firstChild	Returns the first child node of an element
firstElementChild	Returns the first child element of an element
lastChild	Returns the last child node of an element
lastElementChild	Returns the last child element of an element
nextSibling	Returns the next node at the same node tree level
nextElementSibling	Returns the next element at the same node tree level
parentNode	Returns the parent node of an element
parentElement	Returns the parent element node of an element
previousSibling	Returns the previous node at the same node tree level
previousElementSibling	Returns the previous element at the same node tree level

Event-driven programming

事件驅動程式設計

事件驅動程式設計_1

- 事件(event)
  - 使用者在瀏覽網頁過程中由系統或使用者引發的一些狀況，如：網頁下載完畢或使用者點按送出按鈕
- 事件處理程序(event handler，事件處理器)
  - 針對所發生的事件予以適切的回應，撰寫相關的程式碼，即為事件處理程序

事件驅動程式設計_2

- 網頁開發者透過事件機制，讓使用者可以和頁面上的元素互動。
- 事件機制的主要概念：

  事件 + 物件 + 事件處理程序

  – 事件：什麼事發生了?
  – 物件 ：發生在哪一個物件上?
  – 事件處理程序：該做什麼事?

設定事件處理器

- 有下列方法：
  - 1. **在html標籤中以屬性指定事件處理器**

    <button on事件名稱= "js敘述 ">
  - 2. **以JavaScript程式碼使用事件屬性來設定事件處理器**
    - 物件.on事件名稱=事件處理器 (註)
  - 3.**以JavaScript程式碼使用addEventListener來設定事件處理器**
    - 物件. addEventListener (事件名稱,事件處理器, useCapture)
    - 物件. removeEventListener(事件名稱, 事件處理器, useCapture)
  - 4. **在IE瀏覽器中特有的方法來設定事件處理器**
    - 物件. attachEvent(事件名稱,事件處理器);
    - 物件. detachEvent(事件名稱,事件處理器);
    - 此種設定方式IE 11已不再支援
  - 註：事件處理器是必須是函式物件

在html標籤中使用屬性指定事件處理器

- 例:

```
<input type="button" name="btn" value="10+20=?" onClick="showAns()">

<script>
function showAns(){
    //事件發生時所要執行的程式碼放在這裡
    let a=10;
    let b=20;
    alert(a+b);
}
</script>
```

使用物件.on事件名稱設定事件處理器

- 例:

```
<input type="button" name="btn" value="10+20=?">
<input type="button" name="btn" value="10+20=?">

<script>
let btns = document.getElementsByName("btn");
btns[1].onclick = showAns;
</script>
```

使用物件.addEventListener()設定事件處理器_1

- **物件. addEventListener (事件名稱,事件處理器, useCapture)**
  - 使用addEventListener可以讓同一事件，有多個事件處理器
  - useCapture:可以設定事件處理程序是在捕捉階段執行或氣泡階段執行(參閱後面[事件的蔓延現象]課題)
    - true為capture階段
    - false為bubble階段
- **物件.removeEventListener(event, function[, useCapture])**
  - 可以在需要的時候使用removeEventListener移除事件處理器

使用物件.addEventListener()設定事件處理器_2

- 例：

```
<input type="button" name="btn" value="10+20=?">
<input type="button" name="btn" value="10+20=?">

<script>
    let btn1 = document.getElementsByName("btn")[1];
    btn1.addEventListener("click", showAns, false);
</script>
```

事件種類_1

事件名稱	說明
keydown	A keyboard key is pressed
keypress	A keyboard key is pressed or held down
keyup	A keyboard key is released
click	Mouse clicks an object
dblclick	Mouse double-clicks an object
mousemove	The mouse is moved
mouseout	The mouse is moved off an element
mouseover	The mouse is moved over an element
mousedown	A mouse button is pressed
mouseup	A mouse button is released

事件種類_2

事件名稱	說明
load	A page or an image is finished loading
unload	The user exits the page
beforeunload	It is fired just before the web page is unloaded.
error	An error occurs when loading a document or an image
abort	Loading of an image is interrupted
resize	A window or frame is resized
scroll	The event occurs when an element's scrollbar is being scrolled
blur	An element loses focus
focus	An element gets focus
change	The user changes the content of a field
select	Text is selected
reset	The reset button is clicked
submit	The submit button is clicked

More…

使用者與網頁間的互動特效

- 如：
  - 加入最愛、檢舉
  - 燈箱製作
  - 格位的亮化
  - More...

事件物件_1

- 事件發生時瀏覽器會將事件發生時的相關訊息放在**事件物件**中回傳。
  - 相關訊息包含滑鼠位置、壓下的按鍵.....等
- **我們的JavaScript程式碼如何取得此事件物件?**
- **說明**

  若我們使用JavaScript的方法設定事件處理器
  - 則瀏覽器在事件發生時，會產生一個事件物件
  - 並將此事件物件傳入事件處理器
  - 事件處理器可透過第一個參數來取得此事件物件的參考

事件物件_2

- 例如：
  - 假設我們以下列兩種方法設定btn物件被點按時，btnClick函式為事件處理器
    - btn.onclick = btnClick;
    - btn.addEventListener("click", btnClick, false);

  - 則事件處理器中的程式碼可透過第一個參數來取得事件物件

```
function btnClick( e ){    //e: 是變數名稱，可以自行決定
  //事件發生時所要執行的程式碼放在這裡
}
```

事件物件_3

- 事件物件(Event object)

屬性/方法	說明
cancelBubble	Sets or returns whether the event should propagate up the hierarchy or not
currentTarget	Returns the element whose event listeners triggered the event
preventDefault()	Cancels the event if it is cancelable, meaning that the default action that belongs to the event will not occur
stopPropagation()	Prevents further propagation of an event during event flow
target	Returns the element that triggered the event
timeStamp	Returns the time (in milliseconds relative to the epoch) at which the event was created
type	Returns the name of the event
More...	

事件物件_4

- e.target:事件發生在哪一個物件
- e.currentTarget:參考到處理事件的物件
- e.stopPropagation():停止事件的蔓延現象
  - 當文件上的元素有內外層關係時，事件有蔓延的現象
  - 若這些元素均有針對同一個事件撰寫事件處理器時，這些事件處理器會一一被執行之，除非使用程式碼去停止它的蔓延現象
- e.preventDefault():取消事件的預設行為

鍵盤事件

- keydown: when the user is pressing a key
- keypress : when the user presses a key
- keyup : when the user releases a key
- 產生 Keyboard事件時, 事件物件內可以取得
  - charCode: 按鍵的內碼(keypress事件)
  - keyCode : 按鍵的編碼(keydown, keyup事件)

滑鼠事件_1

- mousedown: when the user presses a mouse button over an element
- mouseup: when a user releases a mouse button over an element
- click: when the user clicks on an element
  - 上列三者的觸發順序: mousedown -> mouseup->click
- mouseover: when the mouse pointer is moved onto an element, or onto one of its children
- mouseout: when a user moves the mouse pointer out of an element, or out of one of its children
- mousemove: when the pointer is moving while it is over an element
- mouseenter: when the mouse pointer enters an element(沒有bubble現象)
- mouseleave: when the mouse pointer leaves an element
- More...

滑鼠事件_2

- MouseEvent Object
  - offsetX: relative to the position of the edge of the target element
  - offsetY: relative to the position of the edge of the target element
  - clientX　　: relative to the current window
  - clientY　　: relative to the current window
  - pageX　　: relative to the document
  - pageY　　: relative to the document
  - screenX: relative to the screen
  - screenY: relative to the screen

網頁動態效果的製作

- 有時我們想要重覆執行一些工作，若使用迴圈的技巧，其效果不佳，這時我們可使用計時器來達成。
- 如倒數計時、圖片淡出淡入或一些動態特效。

計時器

- setInterval(函式,間隔時間)
  - 指定的函式，經過間隔時間之後會被**週期性地重覆執行**
  - 時間單位：毫秒
  - 回傳值：計時器編號
- clearInterval(計時器編號)
  - 取消setInterval所設定的重覆執行某函式的工作
- setTimeout(函式,間隔時間)
  - 設定指定的函式，經過間隔時間之後會被**執行一次**
- clearTimeout(計時器編號)
  - 取消setTimeout所設定的執行某函式的工作

JS事件的蔓延現象_1

- JavaScript事件有蔓延開來的特性
- 舊式瀏覽器的事件處理機制
  - 只提供bubble的蔓延模式，事件處理器由內往外被執行
  - 如：button -> form -> body
- 截斷事件的蔓延現象
  - 若使用JS註冊事件處理器
    - 物件.on事件=事件處理器;
    - 物件.addEventListener(事件,事件處理器,useCapture);
  - 則可使用**事件物件.stopPropagation()**來截斷事件的蔓延現象

JS事件的蔓延現象_2

- 新式瀏覽器的事件也有蔓延的現象，若以addEventListener的方法來設定事件事理器，則**可以指定事件處理器是在capture階段或bubble階段執行**
  - **物件. addEventListener (事件名稱,事件處理器,useCapture)**
  - useCapture若為true，則使用**capture階段**執行的模式，事件處理器**由外往內**被執行
  - useCapture若為false，則使用**bubble階段**執行的模式， 事件處理器**由內往外**被執行
- 若要截斷事件的蔓延現象
  - e.stopPropagation() ;
- 註:
  - mouseover, mouseout有bubble現象
  - mouseenter, mouseleave没有bubble現象

使用者與網頁間的互動特效(二)

- 如：
  - 點小圖換大圖
  - More...

Core JavaScript Built-in

匿名函式(anonymous function)

- 在JS中除了具名函式之外，我們也可以建立没有名稱的函式，稱之為匿名函式
- 具名函式
  - 以function statement的方式來宣告
  - 有hoist的現象，在宣告時被建立
- 匿名函式
  - 以function expression的方式來撰寫
  - 在執行到該expression時，才被建立
- 最常用在
  - 回呼函式(callback function)
  - IIFE(Immediately Invoked Function Expression)
- 可使用ES6新增的箭頭函式來簡化寫法

Arrow functions_1

- 當設定函式時不指定函式名稱，即為匿名函數
- 在es6中，匿名函數可省略function關鍵字, 改用箭頭指向所需執行的敘述區塊(即函式區塊)

- 基礎語法：

  (param1, param2, …, paramN) => { statements }

- 若只要執行一行敘述，且該敘述的目的是回傳一個值，則{}和return可以省略

  (param1, param2, …, paramN) => expression

JavaScript
董淑蕙編著【版權所有，不得任意拷貝或引用】

Arrow functions_2

- 參數列若只有一個參數，則左邊的()可以省略
  - (singleParam) => { statements }
  - singleParam => { statements }

- 參數列若沒有參數，則左邊的()不可省略
  - () => { statements }

Arrow functions _3

- 進階使用_可以傳回物件
  - (param1, param2, …, paramN) =>({ x:10,y:20 })
- 進階使用_可以設定參數的預設值
  - (param1 = defaultValue1, param2, …, paramN = defaultValueN) => { statements }
- 進階使用_可以搭配使用rest parameter
  - (param1, param2, ...rest) => { statements }

String_1

- 屬性:

  length:取得字串的長度

- 方法
  - 第一類:提供一些方法可以設定String物件字串的大小寫

方法	說明
toLowerCase()	將英文的大寫字母轉成小寫字母
toUpperCase()	將英文的小寫字母轉成大寫字母

String_2

– 第二類:提供一些方法可以做子字串的搜尋

方法	說明
indexOf(s)	從第0個位置開始往後找,找到s字串在字串中第一次出現的位置索引值,若找不到則傳回-1
lastIndexOf(s)	從最後一個位置開始往前找,找到s字串在字串中第一次出現的位置索引值,若找不到則傳回-1
indexOf(s,n)	從第n個位置開始往後找,找到s字串在字串中第一次出現的位置索引值,若找不到則傳回-1
lastIndexOf(s,n)	從第n個位置開始往前找,找到s字串在字串中第一次出現的位置索引值,若找不到則傳回-1
match(s)	若s字串在字串中找得到,則傳回s字串,否則,傳回null
search(s)	從第0個位置開始往後找,找到s字串在字串中第一次出現的位置索引值,若找不到則傳回-1

String_3

- 第三類:提供一些方法以取得字串中某字元的資料

方法	說明
charAt(n)	取得索引值為n的字元
charCodeAt(n)	取得索引值為n的字元內碼

- 第四類:提供一些方法處理字串中的子字串

方法	說明
substr(start,length)	從字串中的start處取出長度為length的子字串
substring(start,end)	取出字串中的start處到end-1處的子字串
replace(string1,string2)	將字串中的string1替換成string2
split(str,num)	以str來切割字串,並傳回num個子字串

- More...

Number

- 屬性

屬性	說明
MAX_VALUE	傳回數值的最大值
MIN_VALUE	傳回數值的最小值
NaN	傳回NaN表示不是一個數值(Not a Number)
NAGATIVE_INFINITY	傳回-Infinity
POSITIVE_INFINITY	傳回Infinity

- 方法

方法	說明
toString()	轉為字串
toFixed(x)	取小數位x位
toExponential()	轉為科學符號表示法
toPrecision(x)	將精確位數轉為x所指定的位數

Date_1

- Date物件用來指向某一個時間點
- 我們可以使用Date物件所提供的方法來輕易的取得該時間點的相關訊息，如年、月、日、時、分、秒等資訊
- Date 物件是使用UTC時間標準自1970-01-01 00:00:00 開始到某一個時間點的毫秒數值來儲存時間。
- 範例

Date_2

- 建立Date物件
  - new Date();
    - dateObj=new Date()
  - new Date(year, month, day, hours, minutes, seconds, milliseconds);
    - dateObj = new Date(2000,2,21);
    - dateObj = new Date(2000,2,21,12,10,30);
  - new Date(dateString);
    - dateObj = new Date("Mar 21,2000");
    - dateObj = new Date("Mar 21,2000 12:10:30");
  - new Date(milliseconds);
    - dateObj = new Date(168800000000);

Date_3

方法	說明
getDate()	傳回日期(1~31)
getDay()	傳回星期幾(0~6)
getMonth()	傳回月份(0~11)
getFullYear()	傳回完整的西元年(如:2004)
getHours()	傳回小時(0~23)
getMinutes()	傳回分(0~59)
getSeconds()	傳回秒(0~59)
getTime()	傳回自1970-01-01 00:00:00 UTC至某時間點的毫秒數

- toDateString() :將日期物件轉成日期字串
- toString() : 將日期物件轉成字串 (含時分秒)

Date_4

方法	說明
setDate()	設定日期(1~31)
setDay()	設定星期幾(0~6)
setMonth()	設定月份(0~11)
setFullYear()	設定完整的西元年(如:2004)
setHours()	設定小時(0~23)
setMinutes()	設定分(0~59)
setSeconds()	設定秒(0~59)
setMilliseconds()	設定時間,以毫秒為單位的數(0~999)
setTime()	設定Date物件的時間,自1970-01-01 00:00:00 UTC開始,到某一時間點的毫秒

Date_5

- 範例

```
let dateObj = new Date();
let str  =
      dateObj.getHours() + "時" +      //幾點
      dateObj.getMinutes() + "分" +  //幾分
      dateObj.getSeconds() + "秒";     //幾秒
document.write(str);
```

RegExp

- 可用來描述字串應符合的樣式
- 產生RegExp物件的方法
  - /pattern/modifiers;
  - new RegExp(pattern[, modifiers]);
- modifier：
  - i ：case-insensitive matching
  - g ：global match
  - m ：multiline matching

Form Validation

表單驗證

form物件

- 屬性

屬性名稱	說明
name	表單名稱
action	指明要處理此表單的url
method	表單上的資料被送出的方法
enctype	表單的編碼方式
elements	表單上所有欄位物件的集合
length	表單上的欄位數

- 方法

方法名稱	說明
reset()	重設表單上的各欄位為原預設值
submit()	送出表單

表單上常用的element

- input
  - 配合不同的type有不同的輸入介面
- select
  - 內含option
- textarea

select元素_1

- 屬性

屬性名稱	說明
form	欄位屬於哪一個表單
name	欄位名稱
disabled	設定欄位是否有作用,true表沒作用,false表有作
length	清單中選項的個數
multiple	是否為複選
options	清單中所有option物件的集合
selectedIndex	被選到的項目之索引值
value	被選到的項目之值
size	等於1:下拉式選單,大於1:列表清單

select元素_2

- 方法

方法	說明
add(option物件,index)	將option物件加到select物件中,索引值為index的
remove(index)	將select物件中索引值為index的option物件移除

option元素

- 屬性

屬性名稱	說明
text	選項中使用者看到的文字內容
value	此選項被選到時的值
defaultSelected	此選項是否為預設選項
index	選項的索引值
selected	選項目前是否被選取

- 產生option物件的方法
  - option = new Option(text, value);

input元素_1

- 常用的屬性：
  - form：欄位屬於哪一個表單
  - name：欄位名稱
  - type：input標籤的type屬性
  - value：欄位值
  - disabled：設定欄位是否有作用，true表示沒作用，false表示有作用
  - maxLength：允許輸入的最大字元數
  - readOnly：欄位是否為唯讀
  - size：欄位面板的大小
  - checked：是否選取該項欄位
  - More…

input元素_2

- 常見的方法
  - blur()：失去焦點
  - focus()：取得焦點
  - select()：將文字內容反白
  - click()：點選該物件

Object

- 現實生活中我們常需要用一些複合的屬性來描述一個具體存在的物件。
- 如下列，無法用單一的原始資料型別來描述學生或車籍資料
  - 學生資料：學號、姓名、生日、地址、聯絡電話、班級代號、....
  - 車籍資料：車號、引擎號碼、出廠年月、車廠、排氣量、顏色、...
- 可以使用物件型式來封裝此物件所需的特性(屬性)及功能(方法)
- 建立物件的方法
  - 使用new Object()的方法
  - 使用物件字面值表示法
  - 使用new 搭配建構函式的方式

建立物件, 使用new Object()

- 使用new Object()產生物件之後
  - 可以添加屬性給物件，例：

```
let emp = new Object();
emp.empno = "7111";
emp.ename = "Adam";
```

  - 可以添加方法給物件，例：

```
emp.apply = function(money){
   document.write(`申請費用${money}元<br>`) ;
}
```

建立物件, 使用字面值表示法_1

- 使用{},內含所要的屬性名稱/值, 每一對屬性名稱/值之間以逗號隔開
  - 語法：

    let 物件 = { 屬性1: 值1, 屬性2: 值2, .., 屬性n: 值n };
  - 值：可以是所要的型別值，如number, string, boolean,陣列, 物件...等
- 例：

```
let emp = {
                empno: "7112",
                ename: "Ann",
                apply: function(money){
                                document.write(`申請費用${money}元<br>`) ;
                        }
                }
emp.apply(7000);
```

建立物件, 使用字面值表示法_2

- ES6提供簡化的字面值撰寫方法(short notation)。
  - 屬性：
    - {變數1，變數2，...}
  - 方法：

    方法名稱(參數1, 參數2, ...){
    //.....要執行的code
    }
- 例：

```
let empNo = "7118";
let ename = "SCOTT";
var emp = {empNo, ename,apply(money){
                                    document.write(`申請費用${money}元<br>`);
                            }
                    }
emp.apply(7700);
```

建立物件, 使用new搭配建構函式

- 例：

```
function Employee(empno, ename){
    this.empno = empno;
    this.ename = ename;
    this.apply = function(money){
                        document.write(` ${this.ename}申請費用${money}元<br>`) ;
                }
}
let emp = new Employee("7113" , "Alice");
emp.apply(8000);
```

建立物件, 使用建構函式搭配prototype屬性

- 令所有以new 建構函式產生的物件(instance)均有相同的方法與屬性的初始設定
- 例：

```
function Product(productNo, title, price){
    this.productNo = productNo;
    this.title = title;
    this.price = price;
}
Product.prototype.showInfo = function(){
    return `品名: ${this.price}, 單價: ${this.price}`;
}
Product.prototype.quantity = 0;
let pencil = new Product(1, "铅筆", 10);
```

destructuring assignment

- destructuring assignment：意指"解構賦值"
  - 是es6所提供的新功能
  - 可以透過這個機制取出陣列中或物件中的資料給變數
  - 運算式的左側接收右側的資料來源

- 有很多種解構賦值的方法：
  - 陣列解構賦值
  - 物件解構賦值
  - 配合預設值的使用
  - 配合剩餘結構的使用
  - 參數的解構賦值

陣列解構賦值

- 例 :

```
var arr = [11, 22, 33];
var [a, b, c, d, e=2000] = arr;
```

  - 對應不到會得到undefined
  - 對應不到, 可以給預設值
  - 可以跳過某些值不要
  - 可以利用陣列解構來做變數交換
  - ...在=左側(rest, 將解構剩餘元素給另一陣列)
  - ...在=右側(spread, 將陣列解構展開)

物件解構賦值

- 例 :

```
var obj = {x:10, y:20};
var {x, y, z} = obj;
```

  - 對應不到會得到undefined
  - 對應不到, 可以給預設值
  - 可以解構後指派給新的變數名稱
  - 可用在函式參數以物件解構的方式提取物件中某屬性的值