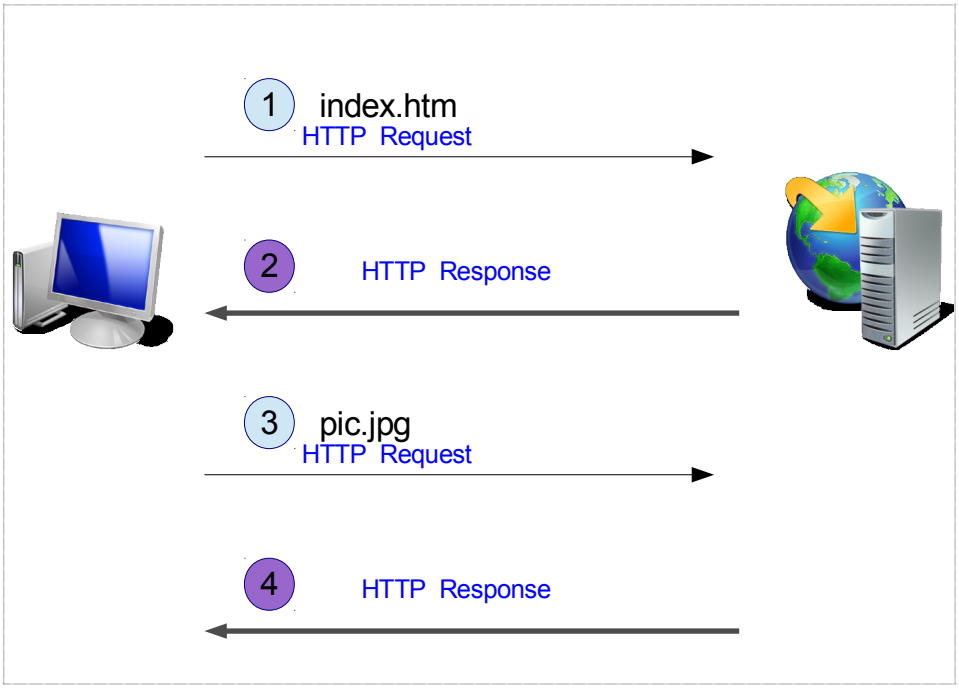
➢ 1　網頁概論

◎　網頁的顯示



◎　HTML-CSS-JavaScript：
結構-樣式-互動
資料-外觀-互動

◎　JavaScript 不是 Java

➢ 2　JavaScript 基礎語法

➢ 2-1　環境

➢ 2-1-1　編輯環境

1. Dreamweaver CS# (整合開發環境、視覺編輯)
2. Notepad++ (文字編輯器, http://notepad-plus-plus.org/)
3. Sublime Text 2 (文字編輯器, http://www.sublimetext.com/2)
4. Aptana Studio (整合開發環境, http://aptana.com/)
5. Netbeans (整合開發環境, http://netbeans.org/)
6. JSFIDDLE (線上, http://jsfiddle.net/)

還有許多... (選擇順手者)

➢ 2-1-2　執行和除錯

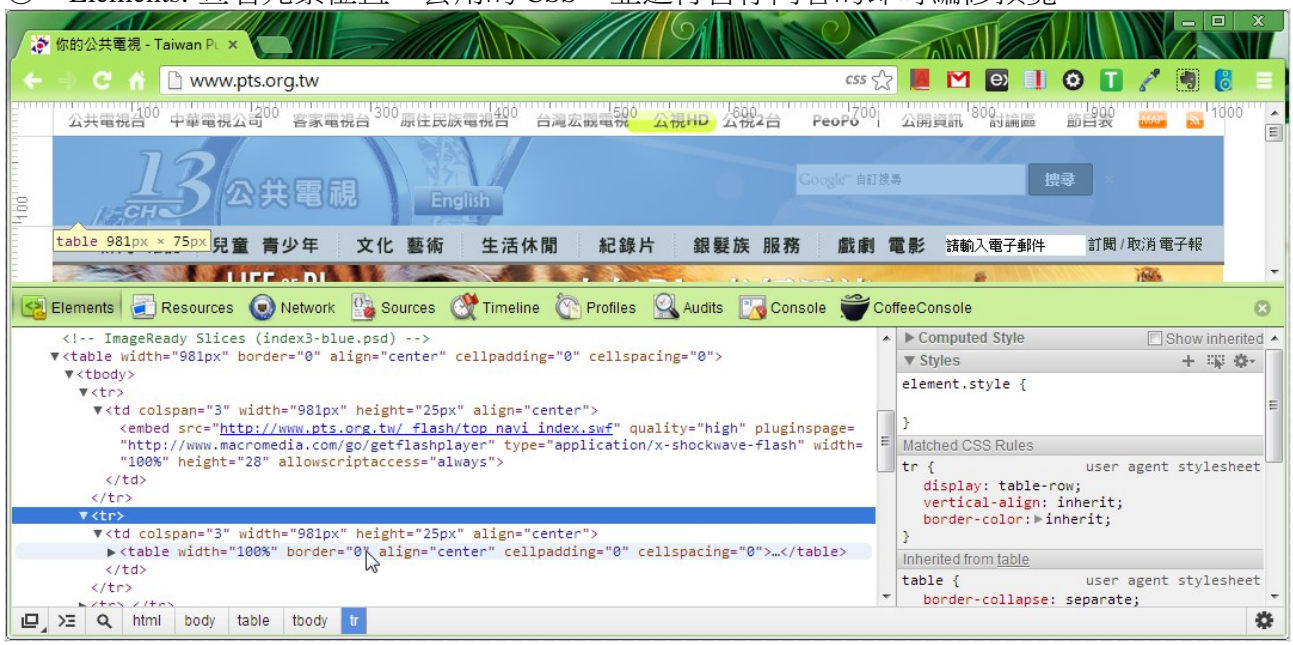
1. Google Chrome (按「Ctrl」+「Shift」+「i」啟用開發者工具)
2. Safari (啟用開發者工具)
3. Mozilla Firefox (外掛: FireBug , Web Developer)
4. Internet Explorer 6、7、8、9、10 (「F12」開發者工具)

5. Opera (http://tw.opera.com/)
6. IETester (IE 各版本除錯測試, http://www.my-debugbar.com/wiki/IETester/HomePage)

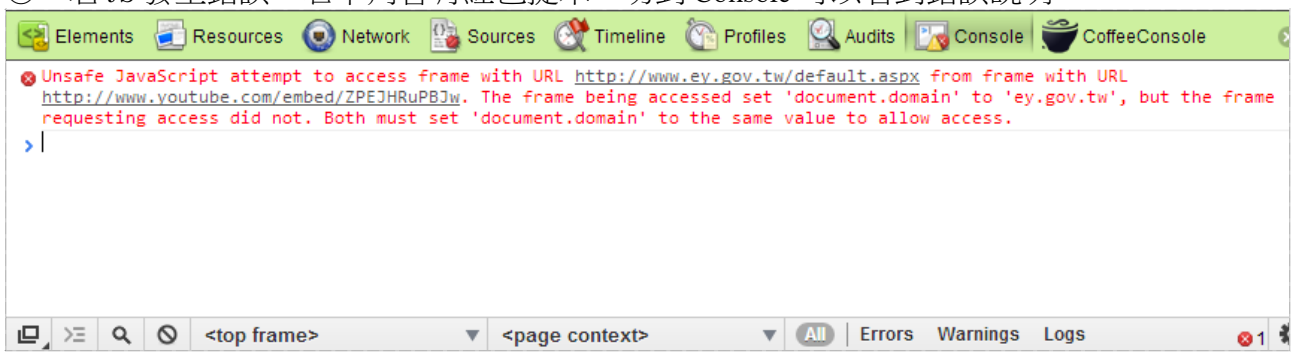
➢ 2-1-3 Chrome

◎ 原始碼：在頁面上按滑鼠右鍵，並點選「檢視網頁原始碼」。

◎ Elements: 查看元素位置、套用的 CSS，並進行暫存內容的即時編修預覽。



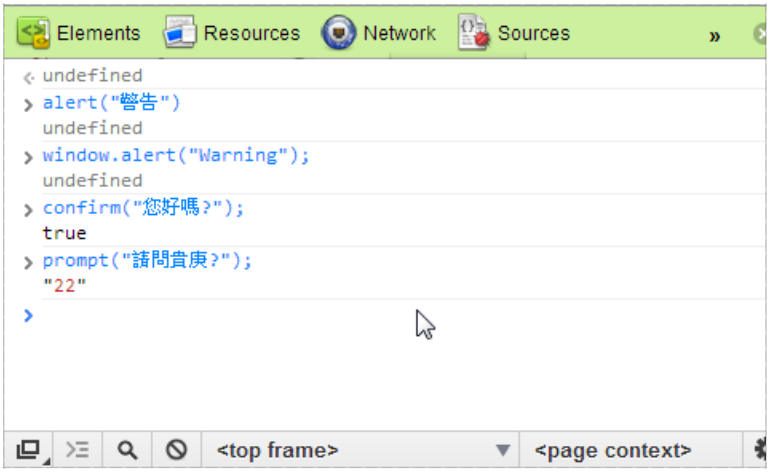
◎ Resources: 查看使用的資源檔 (`.js`, `.css`, `.jpg` 等)。
◎ Network: 查看頁面及資源檔載入的時間差，以利效能改進。亦可查看單一資源 HTTP 的 header、method 和 response 等載入情況。
◎ Sources: 查看 JS 和 CSS 檔內容。
◎ Console: JS 主控台，可測試 JS 並查看 `console.log()` 所輸出的訊息。
◎ 若 JS 發生錯誤，右下角會有紅色提示。切到 Console 可以看到錯誤說明。



➢ 2-3　對話視窗 alert, confirm, prompt

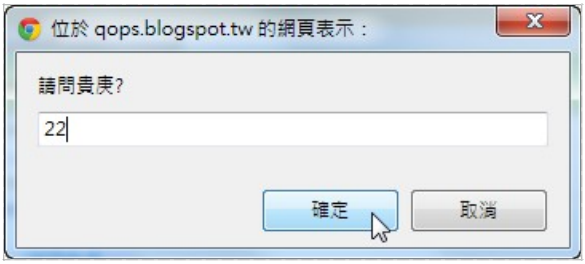
◎　<script></script> 標籤

◎　使用 alert() 警示對話框



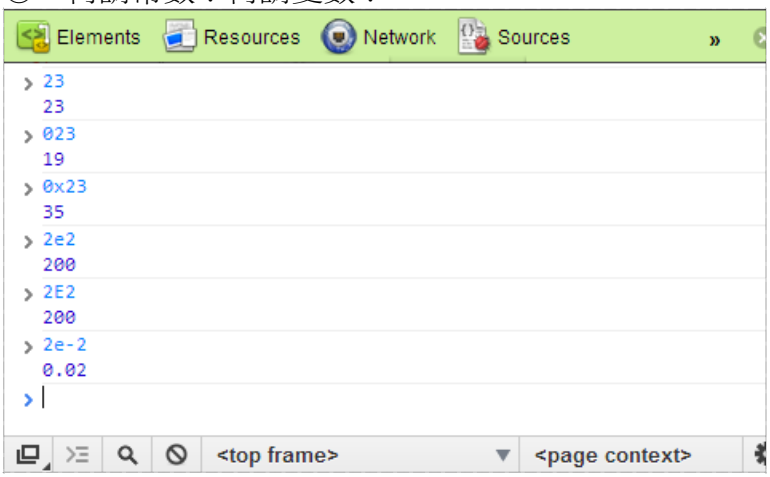
◎　使用 confirm() 讓用戶確認的對話框

◎　使用 prompt() 詢問的對話框



➢ 2-2　變數

◎　何謂常數？何謂變數？



8 進位、16 進位、科學表示法

◎ 變數宣告與設定 (=)

```
var i = 123;
var b = true;
var s1 = '字串',
    s2 = "my string",
    n = 3.456;
```

➢ 2-4 基本類型

◎ 弱型別，使用 typeof 運算子

```
'use strict';
var a=10;
alert( typeof a );            // 'number'
alert( typeof true );         // 'boolean'
alert( typeof 'shinder' );    // 'string'
alert( typeof 5.5 );          // 'number'
alert( typeof undefined );    // 'undefined'
alert( typeof null );         // 'object'
```

◎ 數值和運算

算術運算： * / % + -

關係運算： > >= < <= == != === !==

※ 可以使用 ( ) 標示優先運算。特殊值：NaN、Infinity。

◎ 布林值

邏輯運算： ! && ||

※ 常用在流程控制

◎ 字串（字串物件）

length：字串長度（字元數量）

String 的方法	說明
charAt(索引)	取得某位置的字元。
charCodeAt(索引)	取得某位置字元的字碼。
concat(字串)	字串串接。通常使用 + 運算子。
indexOf(子字串, [索引])	子字串出現的位置。
lastIndexOf(子字串)	子字串最後出現的位置。
slice(索引B, 索引E)	從索引B到E(不包含)建立一個新字串回傳。
split(分割符號)	以分割符號(字串)切割產生陣列。
substr(索引B, 字元個數)	依字元個數，從索引B取得建立一個新字串回傳。
substring(索引B, 索引E)	同 slice()。
toLowerCase()	轉換成小寫字母回傳。
toUpperCase()	轉換成大寫字母回傳。

◎ 型別轉換
- ○ 字串轉數值：parseInt()、parseFloat()。
- ○ 轉成字串：+ 、toString()、String()。
- ○ 轉成布林值：0、undefinded、null、NaN 和空字串視為 false。

➢ 2-5 流程控制（選擇敘述、迴圈）

◎ if：依條件執行或不執行某程式區塊（block）。

```
if(條件式)
{
        // 條件式為 true 時執行
}
```

◎ if/else：依條件選擇執行第一個區塊或第二個區塊。

```
if(條件式)
{
        // 條件式為 true 時執行
}
else
{
        // 條件式為 false 時執行
}
```

◎ if/else if：巢狀結構，依序判斷多個條件。

```
if(條件式一) {
    // 條件式一為 true 時執行
} else {
    if(條件式二) {
        // 條件式二為 true 時執行
    } else {
        if(條件式三) {
            // 條件式三為 true 時執行
        } else {
            // 皆為 false 時執行
        }
    }
}
```

```
if(條件式一) {
        // 條件式一為 true 時執行
} else if(條件式二) {
        // 條件式二為 true 時執行
} else if(條件式三) {
        // 條件式三為 true 時執行
} else {
        // 皆為 false 時執行
}
```

◎ switch/case：比對以選擇開始執行的位置。

```
switch(變數) {
        case 值一:
                // 變數為值一時
                break;
        case 值二:
                // 變數為值二時
                break;
        case 值三:
                // 變數為值三時
                break;
        default:
                // 都不是時執行
}
```

◎ for：迴圈，通常會有控制變數。注意，小刮號裡的兩個分號一定要有。

```
for(起始式 ; 條件式 ; 步進式) {
      // 迴圈內容
}
```

通常控制變數的值從 0 開始，小於「次數值」。

```
for(var i=0; i<8; i++) {
      console.log(i);
}
```

◎ while：迴圈。

```
while(條件式) {
      // 迴圈內容
}
```

◎ do/while：迴圈，先執行一次再說。

```
do {
      // 迴圈內容
} while(條件式) ;
```

◎ for/in：迴圈，變數會取得物件的屬性名稱（字串）。

```
for(變數  in 物件) {
      // 迴圈內容
}
```

```
var obj = {a:12, d:'dog', b:true};

for(var s in obj){
      console.log(s + ':' + obj[s]);
}
```

➢ 2-6 複雜類型（Object、Array）

非基本類型者（Number, Boolean, String）為複雜類型。

➢ 2-6-1 Object

為鍵值對（key-value pair）的資料型態，一個屬性**名稱**對應一個屬性**值**。也稱為 hash table。「屬性名稱」為字串，定義時不應重複，否則會發生覆蓋。

```
var obj1 = {a:12, d:'dog', b:true};
console.log(obj1);
var obj2 = {a:12, d:'dog', b:true, a:56};       // 重複使用相同的 key :(
console.log(obj2);
```

Object 的功能通常用來當作「字典」。其 key-value pair 的 value 也可以是 function，而成為有「方法」的物件。

可動態新增或刪除屬性。

```
var obj = {a:12, d:'dog', b:true};
obj['m'] = 'monkey';    // 中括號表示法
obj.t = 'tiger';        // 點表示法
delete obj.d;           // 用 delete 刪除
console.log(obj);
```

Object 的特性是一個屬性名稱對應一個屬性值，重點不是順序，無需理會名稱的順序。
◎ 比較「中括號表示法」與「點表示法」。

➢ 2-6-2 Array

Array 為有序的資料集合，以索引為取值的 key。

```
var ar = [12, 'abc', 77, -6];
for(var i=0; i<ar.length; i++) {
      console.log( i + ':' + ar[i] );
}
ar['3'] = 100;
for(var s in ar) {
      console.log( s + ':' + (typeof s) + '::' + ar[s] );
}
```

Array 的方法（陣列本身改變）	說明
concat(陣列)	和某陣列串接建立新的陣列。
indexOf(元素值, [索引])	取得元素值的索引，若找不到該元素值回傳 -1。
join([黏著符號])	陣列以黏著符號串接為字串。
lastIndexOf(元素值)	由尾端取得元素值的索引，若找不到該元素值回傳 -1。
pop()	彈出，從尾端取出一元素值。
push(元素值)	推入，從尾端加入一元素值。
reverse()	陣列順序反轉。
shift()	從前端取出一元素值。
slice(索引B, 索引E)	由索引 B 到索引 E 複製部份陣列（不包含索引 E）。
sort()	排序。
splice(索引, 刪除量, [加入])	轉換成大寫字母回傳。
unshift(元素值)	從前端加入一元素值。

排序是以字串（unicode 字碼）為順序。

```
var str = '到底需要日曬多久才能幫助人體獲得足夠的維生素 D';
var ar = str.split('');
ar.sort();
console.log(ar);
var br = [35, 6, 78, 12, 54, 9];
br.sort();
console.log(br);
```

自訂排序規則，數值由小到大：

```
var br = [35, 6, 78, 12, 54, 9];
br.sort(function(a,b){return a-b;});      // 正值對調
console.log(br);
```

➢ 2-7 自訂函式

```
function 函式名 ( 參數列 ) {
      // 內容
      return 回傳值;
}
```

◎ 簡單函式

```
function myFunc() {
      console.log('hi');
}
```

◎ 參數

```
function myFunc(a, b) {
      console.log( a*b );
}
```

◎ 函式內包含所有參數的陣列 － arguments

```
function myFunc(a, b) {
      for(var i=0; i<arguments.length; i++) {
            console.log( arguments[i] );
      }
}
myFunc(7, 9, 12);
```

◎ 複雜類型為參數時

```
function myFunc(ar) {
      ar.push('abc');
}
var bc = [5,6,7];
myFunc(bc);
console.log(bc);
```

◎ 回傳值

```
function myFunc(a, b) {
      return a*b;
}
```

◎ 回傳值為複雜類型時，機制同參數傳遞和設定（=）

◎ 區域變數與全域變數

區域變數的領域（scope）只存在函式裡面，函式結束時區域變數也會被清除。參數也是區域變數。

全域變數屬於 window 物件的屬性。儘量避免使用全域變數，以避免變數相互干擾。

◎ 匿名函式（閉包）

```
var func = function (ar) {
      ar.pop();
}
var bc = [5,6,7];
func(bc);
console.log(bc);
```

◎ 執行但不影響其它程式時

```
var ar = [4,5,6];
(function(){
      var ar = [9,8,7];
      while(ar.length){
            console.log( ar.pop() );
      }
})();
console.log(ar);
```

◎ 函式可以看成是特殊的物件

◎ Scope

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

◎ 避免 console.log 造成 IE 發生錯誤

```
(function(){
      if (typeof console === "undefined" || typeof console.log === "undefined")
      {
            console = {};
            console.log = function(msg) {
                  alert(msg);
            };
      }
      trace = function(msg) {
            console.log(msg);
      };
})();
```

➢ 2-8 時間和數學物件

➢ 2-8-1 Date

```
var d = new Date();
trace( d );
trace( d.getFullYear() );
trace( d.getMonth() );  // from 0 to 11
trace( d.getDate() );
trace( d.getDay() );
trace( d.getHours() );
trace( d.getMinutes() );
trace( d.getSeconds() );
```

➢ 2-8-2 Math

◎ 數學常數

```
trace( Math.PI );
trace( Math.E );
```

Math 的常用方法	說明
abs(x)	求絕對值。
atan2(y, x)	三角函數反正切（垂直距離和水平距離求角度）。
ceil(x)	大於等於 x 的最小整數。
cos(x)	三角函數餘弦。
floor(x)	小於等於 x 的最大整數。
max(x, y, z, ..., n)	最大值。
min(x, y, z, ..., n)	最小值。
pow(x, y)	x 的 y 次方。
random()	0 到 1 之間的亂數（大於等於 0，小於 1）。
round(x)	四捨五入求整數。
sin(x)	三角函數正弦。

➢ 2-9 間隔時間觸發

◎ setTimeout()：間隔一段時間後觸發一次

```
trace( new Date() );
setTimeout(function(){
      trace( new Date() );
}, 3000);
```

◎ setInterval()：間隔一段時間後重複觸發

```
var iid = setInterval(traceTime, 1000);
var count = 6;
function traceTime() {
      count--;
      trace(count, new Date() );
      if(count <= 0) {
            clearInterval(iid);
      }
}
```

➢ 3　瀏覽器相關的物件

➢ 3-1　Window 物件

Window 物件為 JavaScript 裡，網頁的根物件（root）。

◎　Window 物件的常用屬性

```
var val;
var attr_ar = [
      "document",       "frames",         "history",        "innerHeight",
      "innerWidth",     "length",         "location",       "name",
      "navigator",      "outerHeight",    "outerWidth",
      "pageXOffset",    "pageYOffset",    "parent",         "screen",
      "screenLeft",     "screenTop",      "screenX",        "screenY",
      "self",           "status",         "top"
];
for(var i=0; i<attr_ar.length; i++) {
      val = window[ attr_ar[i] ];
      // val = this[ attr_ar[i] ];
      trace( attr_ar[i] + ' : ' + typeof val + ' : ' + val);
}
```

◎　查看 Window 物件的所有屬性

```
var val;
var attr_ar = [];
for(var attr in window) {
      val = window[attr];
      if( typeof val != 'function' && val != null)
            attr_ar.push( attr );
}
attr_ar.sort();
trace( attr_ar );
```

Window 物件的常用方法	說明
alert()	顯示警示對話框。
blur()	讓視窗失焦。
clearInterval()	清除重複觸發某函式。
clearTimeout()	清除觸發一次某函式。
close()	關閉視窗。
confirm()	顯示確認對話框。
focus()	讓視窗取得焦點。
moveBy()	以相對位置移動視窗。
moveTo()	移動視窗。
print()	執行列印。
prompt()	顯示詢問對話框。
resizeBy()	以相對尺寸調整視窗大小。
resizeTo()	調整視窗大小。

scrollBy()	以相對位置捲動視窗內容。
scrollTo()	捲動視窗內容。
setInterval()	間隔一段時間後重複觸發函式。
setTimeout()	間隔一段時間後觸發一次函式。

➢ 3-2 Navigator 物件

```
appCodeName : Mozilla
appName : Netscape
appVersion : 5.0 (Windows NT 6.1) AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.64 Safari/537.11
cookieEnabled : true
geolocation : [object Geolocation]
language : zh-TW
mimeTypes : [object MimeTypeArray]
onLine : true
platform : Win32
plugins : [object PluginArray]
product : Gecko
productSub : 20030107
userAgent : Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.64 Safari/537.11
vendor : Google Inc.
vendorSub :
```

➢ 3-3 Screen 物件

```
availHeight : 1040
availLeft : 0
availTop : 0
availWidth : 1920
colorDepth : 32
height : 1080
pixelDepth : 32
width : 1920
```

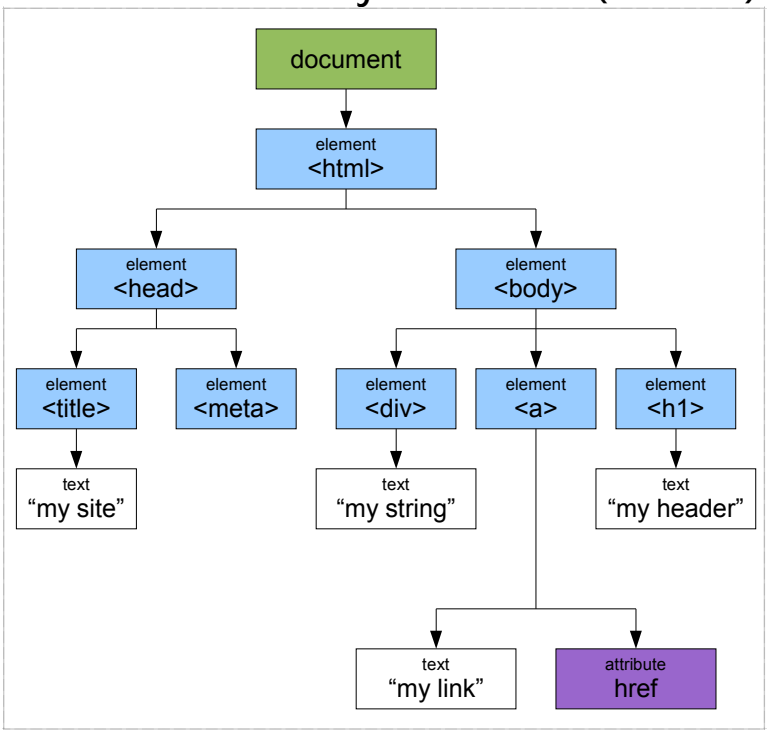
➢ 3-4 History 物件

```
back : function () { [native code] }
forward : function () { [native code] }
go : function () { [native code] }
length : 1
```

➢ 3-5 Location 物件

```
ancestorOrigins : [object DOMStringList]
assign : function () { [native code] }
hash :
host : dl.dropbox.com
hostname : dl.dropbox.com
href : http://dl.dropbox.com/u/9545926/2012/test_console_log_fallback2.html
origin : http://dl.dropbox.com
pathname : /u/9545926/2012/test_console_log_fallback2.html
port :
protocol : http:
reload : function () { [native code] }
replace : function () { [native code] }
```

➢ 4　Document Object Model（DOM）



◎　模型（http://www.w3schools.com/htmldom/）
◎　節點物件（http://www.w3schools.com/jsref/dom_obj_node.asp）

➢ 4-1　Document 物件

HTML 物件載入至瀏覽器之後變成 Document 物件。

Document 的常用屬性	類型（Chrome）	說明
URL	String	網址。
anchors	HTMLCollection	Anchor 集合。
characterSet	String	網頁使用的編碼，同 charset。
cookie	String	Cookies。
doctype	DocumentType	文件類型。
domain	String	網域名稱。
forms	HTMLCollection	表單集合。
head	HTMLHeadElement	Head 元素。
images	HTMLCollection	圖片集合。
links	HTMLCollection	連結集合。
referrer	String	從哪兒來。
title	String	標頭名稱。

Document 的常用方法	說明
getElementById()	依 id 取得元素。
getElementsByClassName()	依 css 類別名稱取得元素。
getElementsByName()	依 name 屬性名稱取得元素。
getElementsByTagName()	依標籤名稱取得元素。

◎ getElementById()

```
<!DOCTYPE html>
<html>
    <head>
        <title>Shinder's Test</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="console_log_fallback2.js">
        </script>
    </head>
    <body>
        <div id="myDiv">subject</div>
        <div class="myClass">item 1</div>
        <div class="myClass">item 2</div>
        <div class="myClass">item 3</div>
    </body>
    <script type="text/javascript">
        var d = document.getElementById('myDiv');
        d.innerHTML = "New Subject";
        for(var s in d) {
            trace(s + ' : ' + d[s]);
        }
    </script>
</html>
```

◎ getElementsByClassName()

```
        var d = document.getElementsByClassName('myClass');
        for(var i=0; i<d.length; i++) {
            d[i].innerHTML = d[i].innerHTML.toUpperCase();
        }
```

◎ getElementsByTagName()

```
        var d = document.getElementsByTagName('div');
        for(var i=0; i<d.length; i++) {
            d[i].innerHTML = (i+1) + '. ' + d[i].innerHTML.toUpperCase();
        }
```

◎ 寫成工具

```
(function(){
    window.SD = function(what) {
        var firstChar = what.slice(0, 1);
        var others = what.slice(1);
        switch(firstChar) {
            case '#':
                return document.getElementById( others );
            case '.':
```

```
                return document.getElementsByClassName( others );
            default:
                return document.getElementsByTagName( what );
        }
    }
})();
```

```
setInterval(function(){
      var now = new Date();
      SD('#myDiv').innerHTML = now.getHours() + ':' +
                               now.getMinutes() + ':' + now.getSeconds();
}, 1000);
```

➢ 4-2　事件處理

➢ 4-2-1　標籤的事件處理器屬性

標籤的事件處理器	說明
onclick ondblclick onmousedown onmousemove onmouseover onmouseout onmouseup	單擊滑鼠左鍵。 雙擊滑鼠左鍵。 按下滑鼠左鍵時。 滑鼠在元素上移動時。 滑鼠移入元素時。 滑鼠移開元素時。 放開滑鼠左鍵時。
onkeydown onkeypress onkeyup	按下按鍵時，用於<body>。 按住按鍵時重複觸發，用於<body>。 放開按鍵時，用於<body>。
onload	內容載入後，使用於<body>、<object>。
onresize	文件大小改變時。
onscroll	文件捲動時。
onblur	失焦時，用於表單內的元素。
onchange	內容改變時，用於<input>、<select>、<textarea>。
onfocus	取得焦點時，用於表單內的元素。
onreset	重置時，用於表單。
onselect	選取部份內容時，用於<input>、<textarea>。
onsubmit	送出表單時。

◎　滑鼠事件

```
<!DOCTYPE html>
<html>
    <head>
        <title>Shinder's Test</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="sd_utils.js">
        </script>
```

```
    </head>
    <script type="text/javascript">
        function eventHandler(event) {
            SD('#myDiv').innerHTML = event.type;
            trace(event.type+': x='+event.x+', y='+event.y);
        }
    </script>
    <style type="text/css">
    #myDiv {
        width: 200px;
        height: 150px;
        background-color: #F00;
    }
    </style>
    <body>
        <div id="myDiv"
        onclick="eventHandler(event)"      ondblclick="eventHandler(event)"
        onmousedown="eventHandler(event)" onmouseover="eventHandler(event)"
        onmouseout="eventHandler(event)"  onmouseup="eventHandler(event)"
        >subject</div>
    </body>
</html>
```

◎ onload

```
<!DOCTYPE html>
<html>
    <head>
        <title>Shinder's Test</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="sd_utils.js">
        </script>
    </head>
    <script type="text/javascript">
        function pageLoaded() {
            var marks = ['一 ','二 ','三 '];
            var divs = SD('.myClass');
            for(var i=0; i<divs.length; i++) {
                divs[i].innerHTML = marks[i] + divs[i].innerHTML;
            }
        }
    </script>
    <body onload="pageLoaded()">
        <div id="myDiv">subject</div>
        <div class="myClass">item 1 </div>
        <div class="myClass">item 2 </div>
        <div class="myClass">item 3 </div>
    </body>
</html>
```

◎ 動態設定事件處理器

```
        var divs = SD('.myClass');  // HTML同上個範例
        for(var i=0; i<divs.length; i++) {
            divs[i].onclick = function(event){
                trace(event.target.innerHTML);
            };
        }
```

◎ 在 <a> 標籤的 href 屬性使用 JavaScript

```
<a href="javascript:do_something()">text</a>
```

➢ 4-2-2 事件的偵聽

◎ 使用 addEventListener()

```
<script type="text/javascript">
    var divs = SD('.myClass');
    for(var i=0; i<divs.length; i++) {
        divs[i].addEventListener('click', function(event){
            trace(event.target.innerHTML);
        });
    }
</script>
```

➢ 4-3 搭配 CSS

◎ 使用元素的 style 屬性。

```
<!DOCTYPE html>
<html>
    <head>
        <title>Shinder's Test</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="sd_utils.js">
        </script>
    </head>
    <body>
        <div id="myDiv">subject</div>
        <div class="myClass">item 1 </div>
        <div class="myClass">item 2 </div>
        <div class="myClass">item 3 </div>
    </body>
    <script type="text/javascript">
        var divs = SD('.myClass');
        for(var i=0; i<divs.length; i++) {
            divs[i].style.cursor = "pointer";
            divs[i].onmouseover = function(event){
                event.target.style['background-color'] = "#FF0000";
            };
            divs[i].onmouseout = function(event){
                event.target.style['background-color'] = "#FFFFFF";
            };
        }
    </script>
</html>
```

◎ CSS 為優先考量。

```
<style type="text/css">
    .myClass {
        cursor: pointer;
        background-color: #FFFFFF;
```

```
        }
        .myClass:hover{
            background-color: #0000FF;
        }
    </style>
```

◎ 個別選取效果

```
    var divs = SD('.myClass');
    for(var i=0; i<divs.length; i++) {
        divs[i].style.cursor = "pointer";
        divs[i].onclick = function(event) {
            var color = event.target.style['background-color'];
            // trace( color );
            if( color == 'rgb(255, 0, 0)' ) {
                event.target.style['background-color'] = 'rgb(255, 255, 255)';
            } else {
                event.target.style['background-color'] = 'rgb(255, 0, 0)';
            }
        };
    }
```

◎ 多選一效果

```
    var divs = SD('.myClass');
    for(var i=0; i<divs.length; i++) {
        divs[i].style.cursor = "pointer";
        divs[i].onclick = myClick;
    }
    function myClick(event) {
        for(var i=0; i<divs.length; i++) {
            if(divs[i] == event.target) {
                divs[i].style['background-color'] = 'rgb(255, 0, 0)';
            } else {
                divs[i].style['background-color'] = 'rgb(255, 255, 255)';
            }
        }
    }
```

◎ 套用 CSS 類別

```
<!DOCTYPE html>
<html>
    <head>
        <title>Shinder's Test</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="sd_utils.js">
        </script>
    </head>
    <style type="text/css">
        .classOne {
            padding: 3px;
            margin: 2px;
            background-color: #F99;
            color: #FFF;
            border: 1px #F00 solid;
        }
    </style>
```

```
    <body>
        <div id="myDiv">subject</div>
        <div class="myClass">item 1 </div>
        <div class="myClass">item 2 </div>
        <div class="myClass">item 3 </div>
    </body>
    <script type="text/javascript">
        var divs = SD('.myClass');
        for(var i=0; i<divs.length; i++) {
            divs[i].style.cursor = "pointer";
            divs[i].onclick = function(event) {
                event.target.className += ' classOne';
                trace( event.target.className );
            };
        }
    </script>
</html>
```

◎ 加入移除 CSS 類別

```
        var divs = SD('.myClass');
        for(var i=0; i<divs.length; i++) {
            divs[i].style.cursor = "pointer";
            divs[i].onclick = function(event) {
                if( hasClass(event.target, 'classOne') ) {
                    removeClass(event.target, 'classOne');
                } else {
                    addClass(event.target, 'classOne');
                }
            };
        }
        function hasClass(element, theClass) {
            var classStr = element.className;
            var ar = classStr.split(' ');
            if( ar.indexOf(theClass) > -1 ) {
                return true;
            } else {
                return false;
            }
        }
        function addClass(element, theClass) {
            if(! hasClass(element, theClass) )
                element.className += ' ' + theClass;
        }
        function removeClass(element, theClass) {
            var classStr = element.className;
            var ar = classStr.split(' ');
            if ( ar.lastIndexOf(theClass) > -1 ) {
                ar.splice( ar.lastIndexOf(theClass), 1);
            }
            element.className = ar.join(' ');
        }
```

◎ 加入移除 CSS 類別（使用 Regular Expression）

```
        function hasClass(element, theClass) {
            return element.className.match(
                new RegExp('(\\s|^)' + theClass + '(\\s|$)') );
```

```
        }
        function removeClass(element, theClass) {
            if (hasClass(element, theClass)) {
                var reg = new RegExp('(\\s|^)' + theClass + '(\\s|$)');
                element.className = element.className.replace(reg, ' ');
            }
        }
```

➢ 4-4 表單互動

◎ 以表單名稱和欄位名稱取得欄位物件

```
<!DOCTYPE html>
<html>
    <head>
        <title>Shinder's Test</title>
        <meta http-equiv="content-type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="sd_utils.js"></script>
    </head>
    <style type="text/css">
    .titleCol { float:left; width:100px; }
    </style>
    <body>
        <form name="form1" method="post" onsubmit="return form_check()">
            <div class="titleCol">Email:</div>
            <input type="text" id="email" name="email" /><br />
            <div class="titleCol">Password:</div>
            <input type="password" id="password" name="password" /><br />
            <div class="titleCol"> </div>
            <input type="submit" value="登入" />
        </form>
    </body>
    <script type="text/javascript">
    function form_check() {
        var pattern = /^([\w-]+(?:\.[\w-]+)*)@((?:[\w-]+\.)*\w[\w-]{0,66})\.([a-
z]{2,6}(?:\.[a-z]{2})?)$/i;
        if(! form1.email.value) {
            alert('請填寫 email !');
            return false;
        }
        if(! pattern.test(form1.email.value) ) {
            alert('請填寫正確格式的 email !');
            return false;
        }
        if(! form1.password.value) {
            alert('請填寫密碼 !');
            return false;
        }
    }
    </script>
</html>
```

◎ 以 id 取得欄位物件

```
<!DOCTYPE html>
<html>
    <head>
        <title>Shinder's Test</title>
```

```
        <meta http-equiv="content-type" content="text/html; charset=utf-8">
        <script type="text/javascript" src="sd_utils.js"></script>
    </head>
    <style type="text/css">
    .titleCol {
        float: left;
        width: 100px;
    }
    .info {
        display: inline;
        color: #F00;
        border: 1px #F90 dashed;
        padding: 2px 5px;
    }
    </style>
    <body>
        <form name="form1" method="post" onsubmit="return form_check()">
            <div class="titleCol">Email:</div>
            <input type="text" id="email" name="email" />
                <div class="info" id="email_info">請填寫 email</div><br />
            <div class="titleCol">Password:</div>
            <input type="password" id="password" name="password" />
                <div class="info" id="pass_info">請填寫密碼</div><br />
            <div class="titleCol"> </div>
            <input type="submit" value="登入" />
        </form>
    </body>
    <script type="text/javascript">
    form1.onkeyup = function(event) {
        var pattern = /^([\w-]+(?:\.[\w-]+)*)@((?:[\w-]+\.)*\w[\w-]{0,66})\.([a-
z]{2,6}(?:\.[a-z]{2})?)$/i;
        if(! SD('#email').value ) {
            SD('#email_info').style.display = 'inline';
            SD('#email_info').innerHTML = '請填寫 email';
        } else if(! pattern.test( SD('#email').value ) ) {
            SD('#email_info').style.display = 'inline';
            SD('#email_info').innerHTML = 'email 格式有誤';
        } else {
            SD('#email_info').style.display = 'none';
        }
        if( SD('#password').value ) {
            SD('#pass_info').style.display = 'none';
        } else {
            SD('#pass_info').style.display = 'inline';
        }
    };
    function form_check() {
        if( SD('#email_info').style.display != 'none' ||
            SD('#pass_info').style.display != 'none') {
            return false;
        }
    }
    </script>
</html>
```

➢ 5　深入 JavaScript

➢ 5-1　錯誤處理

```
function my_func1() {
    try {
        my_func2();
    } catch(err) {
        // trace(err);
        trace(err.message);
        trace(err.stack);
        trace(err.type);
    } finally {
        trace('一定會被印出');
    }
}
my_func1();
```

➢ 5-2　自訂物件

```
var MY = {
    ie_alert: true,
    trace: function(msg) {
        try {
            console.log(msg);
        } catch(err) {
            if(this.ie_alert)
                alert(msg);
        }
    },
    sd: function(what) {
        var firstChar = what.slice(0, 1);
        var others = what.slice(1);
        switch(firstChar) {
            case '#':
              return document.getElementById( others );
            case '.':
              return document.getElementsByClassName( others );
            default:
              return document.getElementsByTagName( what );
        }
    }
}
```

➢ 5-3　Regular Expression

正規表示法（regular expression）的目的是做文字的比對和尋找，在文字處理上非常重要，它是從善長文字處理的程式語言 Perl 上推廣而來。現在，JavaScript 和其它許多程式語言也都支援正規表示法。JavaScript 裡正規表示法的功能交給 RegExp 物件。

在一大段的文字中，你想找出 be 這個字而不是 bee 時，使用字串的 indexOf 方法似乎有些麻煩，這個時候使用 RegExp 物件是個不錯的選擇。RegExp 物件可以搭配 String 物件的 match、replace、search 和 split 方法一起使用。RegExp 物件可以直接使用「/」包裹的方式定義。

```
var re = /be/;
```

若要找出 be 而不是 bee 時：

```
var re = /\sbe\s/;
```

「\s」表示空白（space）。也可以使用 new 關鍵字建立 RegExp 物件：

```
var re = new RegExp('\\sbe\\s');
```

要注意的是在字串中，「\」必須使用「\\」跳脫表示。若字母大小寫不拘，則可以使用下列的其中一個敘述：

```
var re1 = /\sbe\s/i;
var re2 = new RegExp('\\sbe\\s', 'i');
```

「i」為旗標值，表示 ignore case。旗標值還可以為 g 和 m。g 表示 global，比對整個字串，若沒設定 g，只比對到第一個。m 表示 multiline，比對多行。建立 RegExp 物件的表示法，稱為「字模」（Pattern）。

```
var str = "b be bEAch bead Beaker BEAN bee being abbey abet";
var re = /\sbe/ig;  // remove 'g' and try again
trace(str.search(re));
trace(str.match(re));
trace(str.replace(re, "**"));
trace(str.split(re));
```

◎ 單一字元表示法

表示法	說明	範例
\d	數字 0~9	/\d\d/ 符合者為 '22' ; 'c2' 則不符合
\D	「非」數字	/\D\D/ 符合者為 'ac' ; '2c'則不符合
\s	一個空白 (space)	/a\sbar/ 符合者為 'a bar' ; 'abar' 則不符合
\S	「非」空白	/a\Sbar/ 符合者為 'a-bar' ; 'abar' 和 'a bar' 不符合
\w	字母、數字或底線 (_)	/c\w/ 符合者為 'c7' ; 'c#' 和 'c-' 不符合
\W	「非」字母、數字或底線	/c\W/ 符合者為 'c%' ; 'ca' 和 'c_' 不符合
.	任何字元 (不包含換行)	/a../ 符合者可為 'a12' 、'ap+' 、'a##'
[]	中括號中任一字元	/b[ae]d/ 符合者可為 'bad' 、'bed'
[^]	不包含中括號中任一字元	/b[^ae]d/ 符合者可為 'b-d' 、'bod' ; 'bad' 和 'bed' 不符合

◎ 多字元表示法。此類表示法中，該符號是重複前一個字元，所以和前一個字元是一體的。

表示法	說明	範例
*	重複 0 次或多次	/lo*p/ 符合者可為 'lp'、'lop'、'loop'、'looop'
?	重複 0 次或 1 次	/lo?p/ 符合者為 'lp'、'lop'
+	重複 1 次或多次	/lo+p/ 符合者可為 'lop'、'loop'、'looop'
{n}	重複 n 次	/ba{2}d/ 符合者為 'baad'
{n,}	重複 n 次或以上	/ba{2,}d/ 符合者可為 'baad'、'baaad'
{n,m}	重複 n 次至 m 次之間	/ba{1,2}d/ 符合者為 'bad'、'baad'

上表裡的表示符號又稱為「貪婪計量子（Greedy quantifiers）」，會儘量找尋較長的字串。例如表示式為「lo*」，當搜尋的對象為 "looop" 時，搜尋到的會是 "looo"，而不是 "loo"、"lo" 或 "l"。

「貪婪計量子」後面接個「?」時，會變成「自閉計量子（Reluctant quantifiers）」儘量找尋較短的字串。例如表示式為「lo+?」，當搜尋的對象為 "looop" 時，搜尋到的會是 "lo"，而不是 "loo" 或 "looo"。

◎ 位置及其它表示法。

表示法	說明	範例
^	字首	/^pos/ 符合者可為 'pose' ; 'apos' 不符合
$	字尾	/ring$/ 符合者為 'spring' ; 'ringer' 不符合
|	或	/jpg|png/
()	子表示法	/img\.(jpg|png)/

RegExp 物件有兩個方法 exec 和 test。exec 方法通常是用來搜尋字串中符合字模的子字串。test 方法是用來測試字串是否符合字模。

```
var str = "b bEAch bead Beaker";
var re = /\sbe/ig;
var obj;
while ( obj = re.exec(str) ) {
    trace( obj );
    trace(re.lastIndex + ' --------');
}
```

➢ 5-4 物件導向

◎ 自訂類型

```
function Person(name, age) {
    this.name = name;
    this.age = age;
    this.getInfo = function() {
        return this.name + ':' + this.age;
    }
}
var b = new Person('Bill', 32);
```

```
trace( b.getInfo() );
trace( b.name );
```

◎ 使用 prototype 擴充

```
function Person(name, age) {
    this.name = name;
    this.age = age;
    this.getInfo = function() {
        return this.name + ':' + this.age;
    }
}
Person.prototype.toString = function() {
    return JSON.stringify( this );
}
var b = new Person('Bill', 32);
trace( b.getInfo() );
trace( '' + b );
```

◎ 類型定義 setter/getter

```
function Person(name, age) {
    this.getInfo = function() {
        return name + ':' + age;
    }
    this.__defineGetter__("name",
        function() {
            return name;
        });
    this.__defineGetter__("age",
        function() {
            return age;
        });
    this.__defineSetter__("age",
        function(a) {
            age = a;
        });
}
var b = new Person('Flashman', 32);
b.name = 'Batman'
trace( b.name );    // read only
b.age = 25;
trace( b.age );
```

◎ 物件定義 setter/getter

```
var DATE = {
    _data: new Date(),
    set data(d) {
        if(d && d instanceof Date)
            this._data = d;
    },
    get data() {
        return this._data;
    },
    get fullYear() {
        return this._data.getFullYear();
    },
```

```
        get month() {
            return this._data.getMonth();
        },
        get date() {
            return this._data.getDate();
        },
        digi2: function(n) {
            n = '0'+n;
            return n.slice(n.length-2);
        },
        toString: function() {
            return this.fullYear + '-' + this.digi2(this.month+1) +
                    '-' + this.digi2(this.date);
        }
    }
    trace ( '' + DATE );
    DATE.data = new Date(2000, 9, 10);
    trace ( '' + DATE );
```