

2. 開發工具

<http://webdesign.about.com/od/macintoshhtmleditors/tip/free-macintosh-editors.htm>

• 簡易工具

- Notepad++
- Sublime text 2 / 3
- TextWrangler
- Eclipse
- W3schools

• 參考網站：

<http://www.ecma-international.org/publications/standards/Ecma-262.htm>
<http://www.w3schools.com/js/default.asp>

3. 撰寫格式

- 放在HTML的<body>中，將JS當HTML的element來使用

```
<body>
```

```
    ...
```

```
</body>
```

- 放在HTML的<body>中，透過事件驅動來執行JS

```
<head>
```

```
    <script>...</script>
```

```
</head>
```

- 寫成外部檔案(xx.js)

```
<head>
```

```
    <script src="xx.js"></script>
```

```
</head>
```

(II). Syntax Basic

撰寫任何程式的原則：

- 易於維護
- 具可攜性(**portable**)
- 最好程式碼本身就能代表其意義
- 並適時加入註解(**comment**)，增加程式的可讀性
 - 單行註解 `//...`
 - 多行註解 `/* ... */`
- 常見錯誤：
大小寫字母、拼錯字、漏寫 " " 或 ' ' ...

1. 變數

- 意義：
 - 代表記憶體空間，可存放任何數值或物件
 - 其所代表的值可任意改變
- 命名原則：
 - 由英文數字大小寫、數字、\$及底線組成
 - 數字不能在開頭
 - 大小寫視為不同(case-sensitive)
 - 不能與關鍵字(保留字)相同
- 宣告時，可使用關鍵字 var
- 敘述(statement)結束，請以『；』結尾

關鍵字或保留字：

break	delete	function	return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	finally	instanceof	throw	while
default	for	new	try	with

未來保留字：

abstract	double	implements	private	throws
boolean	enum	import	protected	transient
byte	export	int	public	volatile
char	extends	interface	short	
class	final	long	static	
const	float	native	super	
debugger	goto	package	synchronized	

2. 資料型態

- Javascript 有幾個重要的資料型態：
 - 數字 Number
 - 日期時間 Date
 - 陣列 Array
 - 數學 Math
 - 字串 String
 - 正規表示式 RegExp
 - 布林 Boolean
 - 物件
- 也是資料型態：
 - null (空值)
 - undefined (未定義)
 - 宣告一個變數，沒立刻給值
 - 使用一個物件，但未聲明屬性
 - 定義了函數的參數，但並未傳值給該函數

數字型態--可以進行加減乘除等運算

```
var num = 25;
```

字串型態--加上'...'或"..."，讓JS知道正在處理的是字串

```
var str = 'I am a simple string.';  
var str = '25';
```

布林型態--只有兩個值：true和false

```
var boo = true;
```

物件型態--除了以上型態的其他型態

```
var id = document.getElementById('context');
```

·宣告變數時，無須宣告資料型態，JS會在執行時自動轉換。

·與變數有關的運算子：typeof

```
typeof variable;  
//顯示此variable是何種資料型態  
//若typeof 某函數;  
//會顯示function，函數實際上就是物件  
/*若結果為null，常常是表示物件不存在，所以null會被當成物件*/
```

·資料型態轉換

parser會視程式的前後文決定資料型態的轉換。

或使用強制轉型的語法：

Number(object)

parseInt(object)

object.toString()

變數的關鍵字 var

宣告變數時，不一定要加上此關鍵字。

宣告變數時，盡可能立刻給初值，否則是undefined

每個變數都有其scope和 life cycle

其scope可大概分成區域和全域

較好的程式，盡可能使用區域變數

• 定義區域變數：

- 在函數中使用var關鍵字
- 定義函數時的參數

• 定義全域變數：

- 在函數中**不要**使用var關鍵字
- 寫在函數之外

舉例來說：

• meg在此為區域變數

```
function doFirst () {  
    var meg = 'error';  
    alert (meg);  
}
```

• meg在此為全域變數

```
var meg = 'error';  
function doFirst () {  
    alert (meg);  
}
```

• meg在此為全域變數

```
function doFirst () {  
    meg = 'error';  
    alert (meg);  
}
```

• meg在此為區域變數

```
doFirst ('error');  
function doFirst (meg) {  
    alert (meg);  
}
```

3. 輸出 / 輸入

- 輸出到畫面 (少用)

```
document.write('Hello World!');
```

- 輸出到狀態列，目前 (非常少用)

```
window.status = "出現在狀態列";
```

- 跳出視窗

- alert 視窗
- confirm 視窗
- prompt 視窗

4. 運算子、運算式與敘述

· JS的運算子

運算子優先順序	描述	結合性
() 、 []		從左到右
++ 、 -- 、 + (正) 、 - (負) 、 !	一元運算子	從右到左
* 、 / 、 %	算術運算子	從左到右
+ (加) 、 - (減)	[同上]	從左到右
> 、 >= 、 < 、 <=	關係運算子	從左到右
== 、 !=	[同上]	從左到右
&&	邏輯運算子	從左到右
	[同上]	從左到右
? :	條件運算子	從左到右
= 、 += 、 -= 、 *= 、 /= 、 %= 、 ...	指定運算子	從右到左

· 字串運算子：字串可用 "+" 將字串串接

5. 迴圈敘述

·結構化的程式設計有三種結構：

- 循序式：即逐行執行
- 迴圈式：反覆執行到不想執行為止
- 選擇式：多重選項，擇一而行

·常見的迴圈形式：

迴圈	執行次數	使用時機
<code>while (條件判斷) {敘述;}</code>	$0 \sim N$	不固定的執行次數與狀態
<code>for (初值; 條件判斷; 計次) {敘述;}</code>	$0 \sim N$	固定的執行狀態
<code>for (...in...) {敘述;}</code>		
<code>do {敘述;} while (條件判斷);</code>	$1 \sim N$	執行起碼一次

·迴圈的好幫手**break**和**continue**

- break**;敘述除了在迴圈有作用以外，還可使用在**switch**
- continue**;敘述只在迴圈有作用

6. 選擇性敘述

選擇性敘述	說明	執行
<code>if(條件) { . . . }</code>	<ul style="list-style-type: none">· 執行0或1次	符合條件就執行
<code>if(條件) { (1) } else { (2) }</code>	<ul style="list-style-type: none">· 條件大多是在一個數值範圍內；且可以放浮點數的比較	符合條件就執行(1)，不符合就執行(2)
<code>switch(比對條件) { case XX: ... break; case OO: ... break; ... default; } //end of switch</code>	<ul style="list-style-type: none">· 比對條件控制在整數形態或字元形態· 不適合使用條件在一個範圍內，例如1~1000之間	多重選擇

• 雖然JavaScript的語法與C語言十分類似，但以下卻有些不同。

• try / catch

◦ try / catch不是迴圈

◦ 目的：希望user不會看到錯誤訊息(做錯誤處理)

◦ 語法：

```
try{  
    //這裡的程式碼可能會導致錯誤  
}  
  
catch (e){  
    //如果有錯誤就執行此處  
}  
  
finally{ //不一定要有finally  
    //不論是否有錯誤，此處都會被執行  
}
```

• for...in

◦ 用在物件的Traversal

7. 函數

- 內建函數--直接使用即可，不需要宣告

內建函數	說明
parseInt (value,base)	傳回 string 轉成 10 進制後的整數； base 不一定要寫，default 是10 進制。
parseFloat (value)	傳回 string 轉成浮點數後的數目
isNaN (testValue)	檢查 testValue 是否為非數字型態 (is Not a Number)
escape (string)	傳回string 的 16 進制編碼得出來的字串
unescape (string)	傳回以 16 進制編碼字串的原本字串
eval (string)	把 string 當作 JavaScript 的敘述來執行
Number (object)	將物件轉換成數值，遇到不合法傳回NaN

·自訂函數--

◦宣告 ◦定義 ◦呼叫

·JavaScript不用明顯的宣告，直接會寫『定義』，實際上也有『宣告』的意味

·定義

```
function functionName(){  
    // 敘述  
}
```

·呼叫

```
functionName();
```

帶有參數的函數：

·定義

```
function functionName(arg1,arg2,...){  
    // 敘述  
}
```

·呼叫

```
functionName(value1,value2,...);
```

有傳回值的函數：

·定義

```
function functionName(){  
    // 敘述  
    return value;  
}
```

·呼叫

```
variable = functionName();
```

(III). Object-Oriented

1. 物件的屬性與方法

- 所有能描述的東西都是由**物件**組成
 - 大的物件裡面可以包含小物件
 - 許多的小物件可以組成大物件
- 每一個物件都有其**特性**與其**使用方法**
 - 物件的特性即為**屬性 (attributes)** -- 名詞&形容詞
 - 物件的使用方法即為**方法 (methods)** -- 動詞
- 物件有層級
- 事件 (**events**) 的發生可以啟動物件的方法或屬性

2. 事件 (events) 處理--1 / 4

Javascript 是 event-driven language

(1). 何謂事件

- 系統會將每個發生的動作轉成事件，然後送到程式處理 user 的動作包括，按下滑鼠或載入某一頁等等。系統的回應可能是：當某一頁載入完。
- 我們就根據不同的事件來設計不同的工作，負責處理事件的就叫做事件處理程序 (Event Handler)。
- JavaScript 的事件通常與物件有關，所以不同的物件就支援不同的事件處理程序

2. 事件 (events) 聆聽功能 -- 2 / 4

(2). 建立事件聆聽功能

包括一個物件、一個事件、一個處理函數

1. 嵌入式事件處理 (.html)

```
<body onload="doFirst()">
```

2. 過去的事件處理方法 (.js)

```
window.onload=doFirst;
```

或 `window.onload=function () { ... }`

3. `window.addEventListener ('load', doFirst, false);`

4. for IE

```
window.attachEvent ('onload', doFirst);
```

2. 事件 (events) 分類--3/4

(3). 事件分類

- 輸入裝置

- 鍵盤

- Browser

- Form

2. 事件 (events) 分類--4 / 4

(4). 事件物件的屬性和方法

屬性：

方法：

`preventDefault()`

(5). 引用事件

3. 視窗(window)物件 -- 屬性(1/2)

屬性	說明
navigator	
location	
history	
document	
status	
name	視窗物件名稱
closed	視窗是否關閉
defaultStatus	預設的狀態列訊息
opener	用 <code>window.open()</code> 開啟的那個視窗
[event]	物件的屬性本身也可以是物件

3. 視窗(window)物件 -- 方法(2/2)

方法	說明	語法
<code>alert()</code>	警告訊息	
<code>confirm()</code>	確認訊息	
<code>prompt()</code>	提示訊息	
<code>open()</code>	開啟新視窗	
<code>close()</code>	關閉視窗	
<code>focus()</code>	成為焦點	
<code>blur()</code>	移開焦點	
<code>setTimeout()</code>	設定計時器	
<code>clearTimeout()</code>	取消計時器	
<code>setInterval()</code>	設定計時器	
<code>clearInterval()</code>	取消計時器	

視窗物件--window.navigator

- 提供瀏覽器的訊息

屬性	Result
appCodeName	Mozilla
appName	Netscape
appVersion	5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.111 Safari/537.36
cookieEnabled	true
language	zh-tw
onLine	true
userAgent	Win32
方法	說明
javaEnabled()	判斷是否支援JAVA

視窗物件--window.history

- 儲存該網頁曾到訪過的URL資料

屬性	說明	
length	到訪過URL的個數	history.length
方法	說明	
go()	載入指定的URL	history.go(number URL)
back()	載入上一個URL	history.back()
forward()	載入下一個URL	history.forward()

視窗物件--window.location

- 提供了很多讀取所在網址的相關資訊

屬性	說明	
hash	取得anchor	
host		
hostname		
href		
origin		
pathname		
port		
protocol	通訊協定，包括冒號	
search	頁面跳轉	location.search=querystring
方法	說明	
assign()		
reload()		
replace()		載入新的 URL

視窗物件--window.document--1/3

屬性	說明
lastModified	
domain	
URL	
cookie	
documentElement	
head	
scripts	
title	
body	

屬性	說明
anchors	
forms	
images	
links	
bgColor	
fgColor	
linkColor	
alinkColor	
vlinkColor	