JavaScript

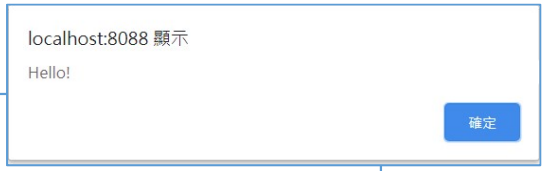
陳銀華 yinhua0999@gmail.com

https://reurl.cc/XVjNYe

HTML 、CSS 、JavaScirpt 關係

HTML 語法

```
<tag attribute="value" attribute="value">content</tag>

<h1 style="color:red" onclick="window.alert('Hello!');">This is heading 1</h1>
```



css 語法
attribute:value

屬性事件（動作）

JavaScript 語法
object.method(parameter);

具有互動功能

物件成員由屬性(property)和方法(method)組成
事件--預先定義好的特定動作，通常由使用者或系統啟動

JavaScript 語法

```
object.property
object.method()
```

JavaScript課程大綱

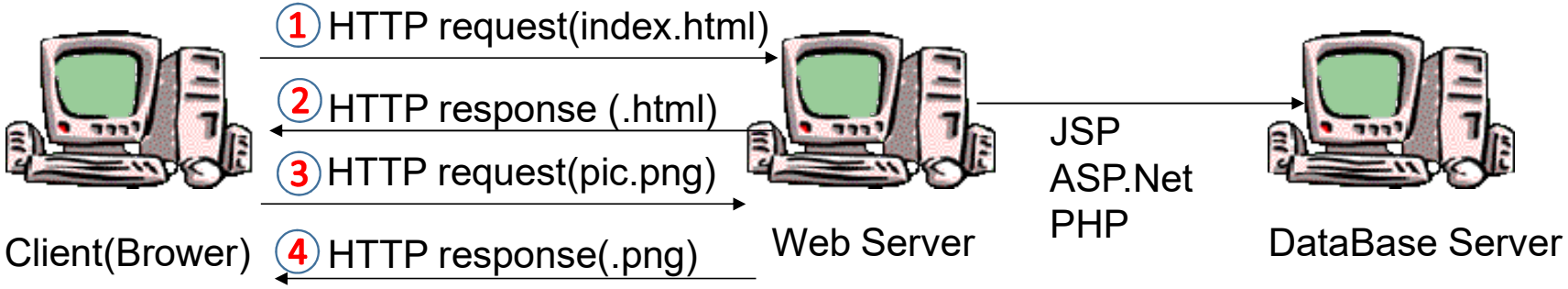
- 單元一：JavaScript簡介
- 單元二：資料型態與變數
- 單元三：運算子與敘述
- 單元四：函數
- 單元五：JavaScript物件
- 單元六：JavaScript的事件處理
- 單元七：Dynamic HTML
- 單元八：BOM 物件模型
- 單元九：DOM 物件模型
- 附錄一：Drag & Drop

單元一：JavaScript簡介

- Web 運作原理
- JavaScript 簡介
- JavaScript 的好處
- JavaScript 的基本架構
- JavaScript 的撰寫格式
- JavaScript 的輸出

Web 運作原理

- HTTP – HyperText Transfer Protocol(協定)
  - Protocol(協定)：電腦之間或網路上的設備之間溝通的語言
- HTML – HyperText Markup Language

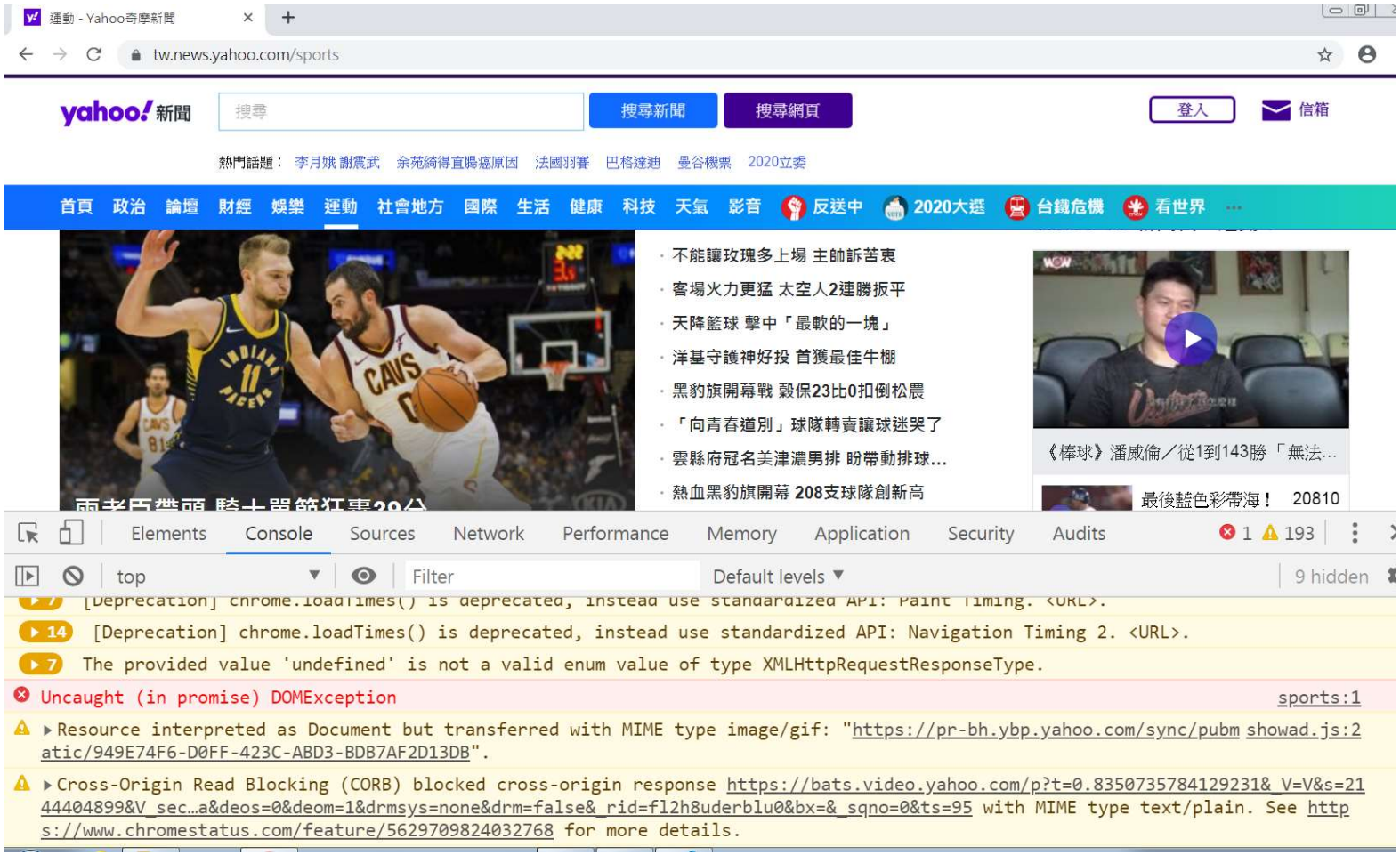



```
<h1 style="color:red" onclick="window.alert('Hello!');"> this is heading 1</h1>
<img src="pic.png">
```

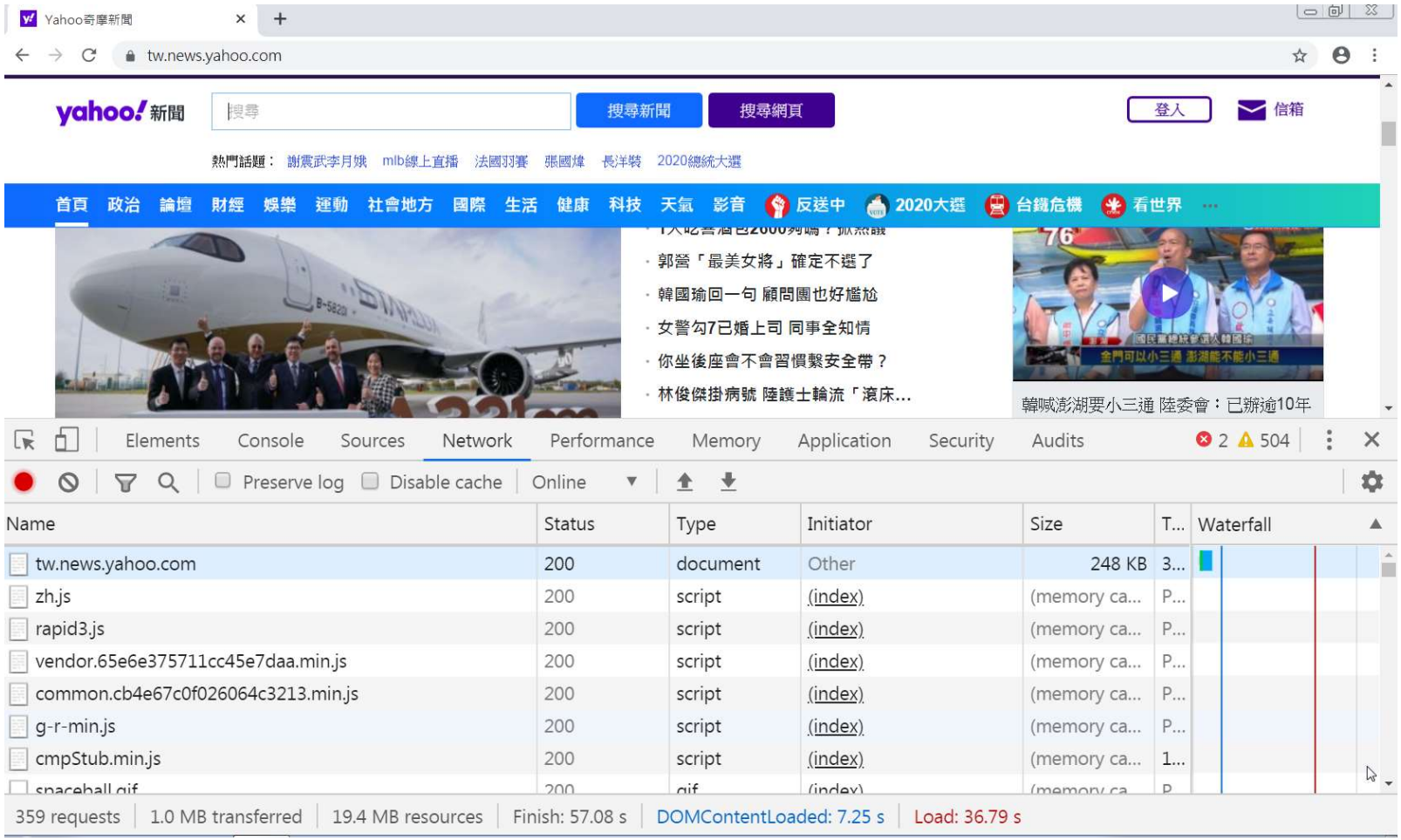
Chrome開發人員工具

- 按F12 、Ctrl+Shift+I或在網頁上滑鼠按右鍵，在選單上點選「檢查」，即可開啟開發人員工具(developer tools)
- Element：檢視網頁的元素(element)。
- Console：查看錯誤訊息或用來做JavaScript 除錯。
- Source：查看該頁面所使用的資源。
- Network ：檢視HTTP 的requests 和responses 狀況。
- Application ：查看頁面暫存資料，如cookies。

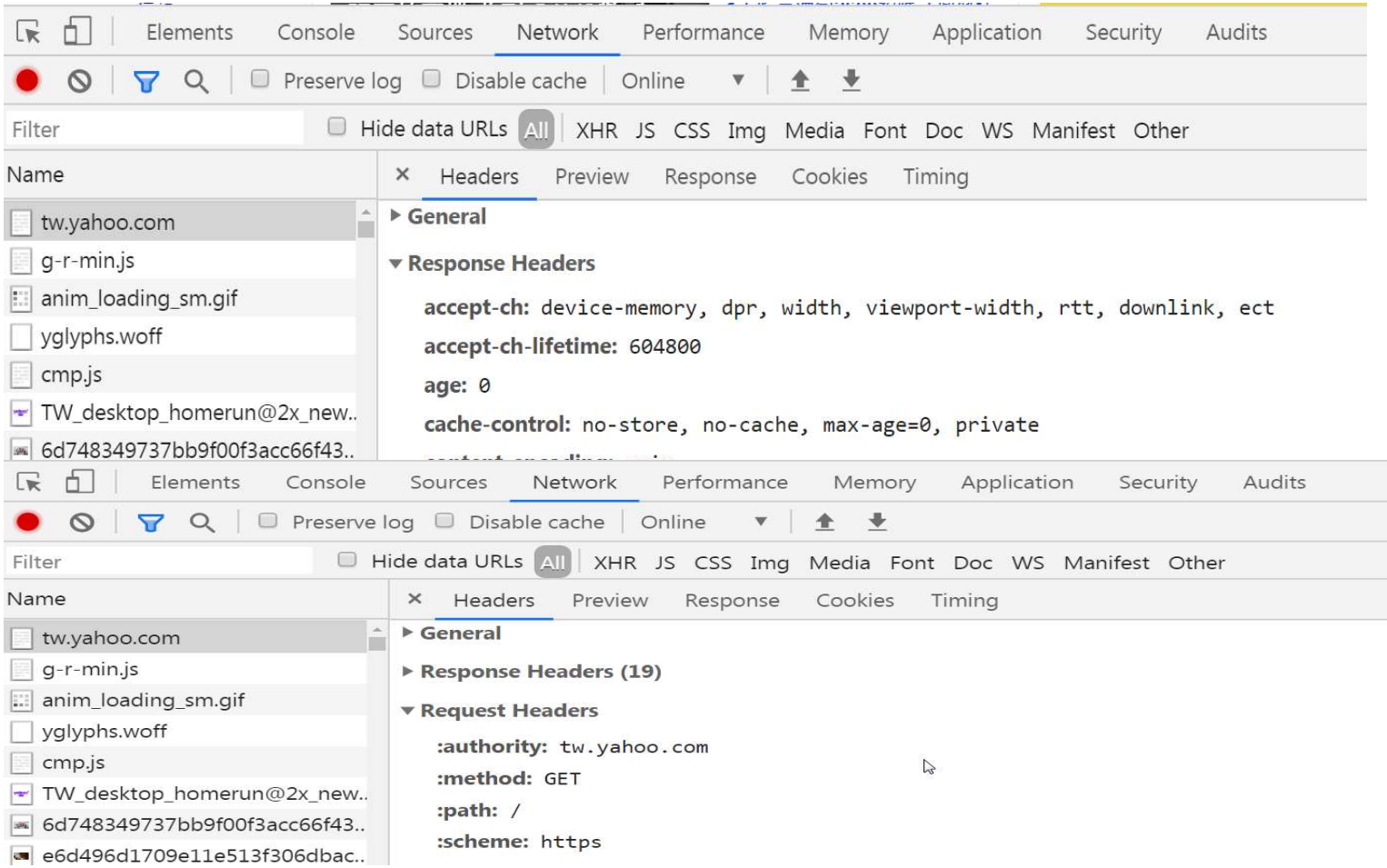
Console面板


運動 - Yahoo奇摩新聞
tw.news.yahoo.com/sports
yahoo! 新聞
搜尋
搜尋新聞
搜尋網頁
登入
信箱
熱門話題： 李月娥 謝震武 余苑綺得直腸癌原因 法國羽賽 巴格達迪 曼谷機票 2020立委
首頁 政治 論壇 財經 娛樂 運動 社會地方 國際 生活 健康 科技 天氣 影音 反送中 2020大選 台錢危機 看世界
・不能讓玫瑰多上場 主帥訴苦衷
・客場火力更猛 太空人2連勝扳平
・天降籃球 擊中「最歡的一塊」
・洋基守護神好投 首獲最佳牛棚
・黑豹旗開幕戰 穀保23比0扣倒松農
・「向青春道別」球隊轉責讓球迷哭了
・雲縣府冠名美津濃男排 盼帶動排球...
・熱血黑豹旗開幕 208支球隊創新高
《棒球》潘威倫／從1到143勝「無法...
最後藍色彩帶海！ 20810
Elements Console Sources Network Performance Memory Application Security Audits
1 193
top
Filter
Default levels
9 hidden
[Deprecation] chrome.loadTimes() is deprecated, instead use standardized API: Paint Timing. <URL>.
14 [Deprecation] chrome.loadTimes() is deprecated, instead use standardized API: Navigation Timing 2. <URL>.
7 The provided value 'undefined' is not a valid enum value of type XMLHttpRequestResponseType.
Uncaught (in promise) DOMException sports:1
Resource interpreted as Document but transferred with MIME type image/gif: "https://pr-bh.ybp.yahoo.com/sync/pubm showad.js:2
atic/949E74F6-D0FF-423C-ABD3-BDB7AF2D13DB".
Cross-Origin Read Blocking (CORB) blocked cross-origin response https://bats.video.yahoo.com/p?t=0.8350735784129231&_V=V&s=21
44404899&V_sec…a&deos=0&deom=1&drmsys=none&drm=false&_rid=fl2h8uderblu0&bx=&_sqno=0&ts=95 with MIME type text/plain. See http
s://www.chromestatus.com/feature/5629709824032768 for more details.

Network面板(1/2)



Network面板(2/2)

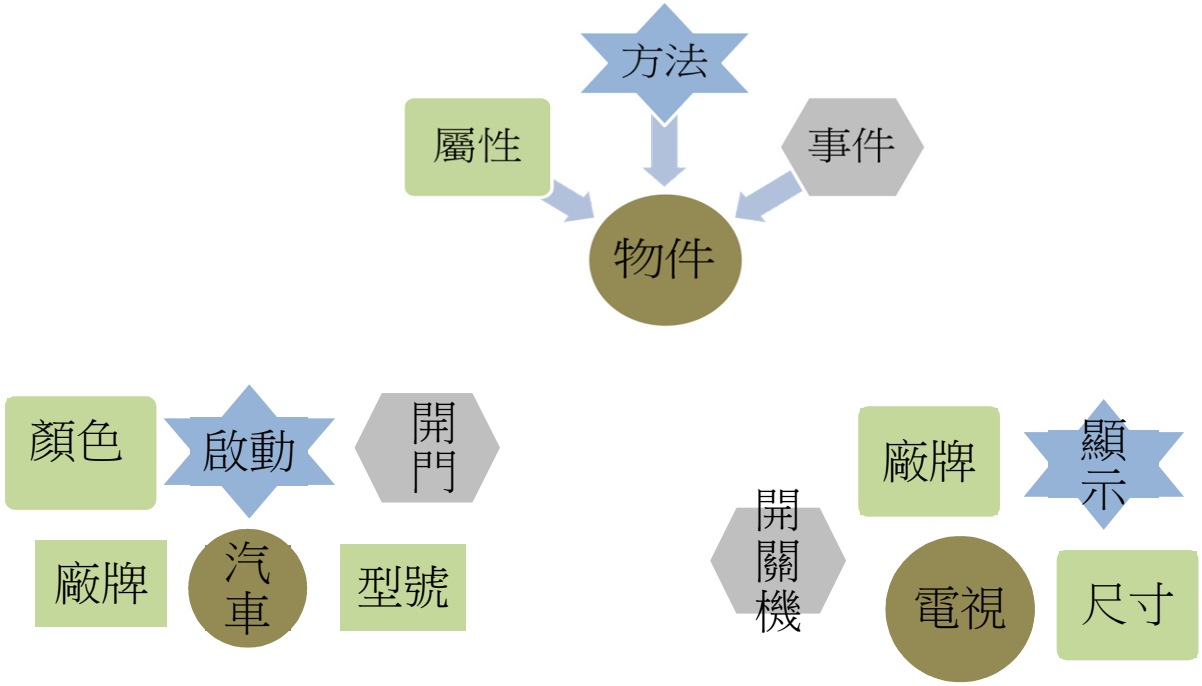


JavaScript 簡介

- 1995年由網景公司(Netscape co.)的Brendan Eich開發了一個名叫LiveScript的指令碼語言
- 1997年在歐洲電腦製造商協會(European Computer Manufacturers Association-ECMA International)協調下，確定統一標準:ECMA-262，稱為ECMAScript
- ECMAScript規格書:
  - https://www.ecma-international.org/ecma-262/
- 是一種以物件及事件驅動為基礎的程式語言
- 必須要直接嵌入HTML文件中，才可執行
- JavaScript包括三大部分:ECMAScript、Browser Object Model(BOM)與Document Object Model(DOM)

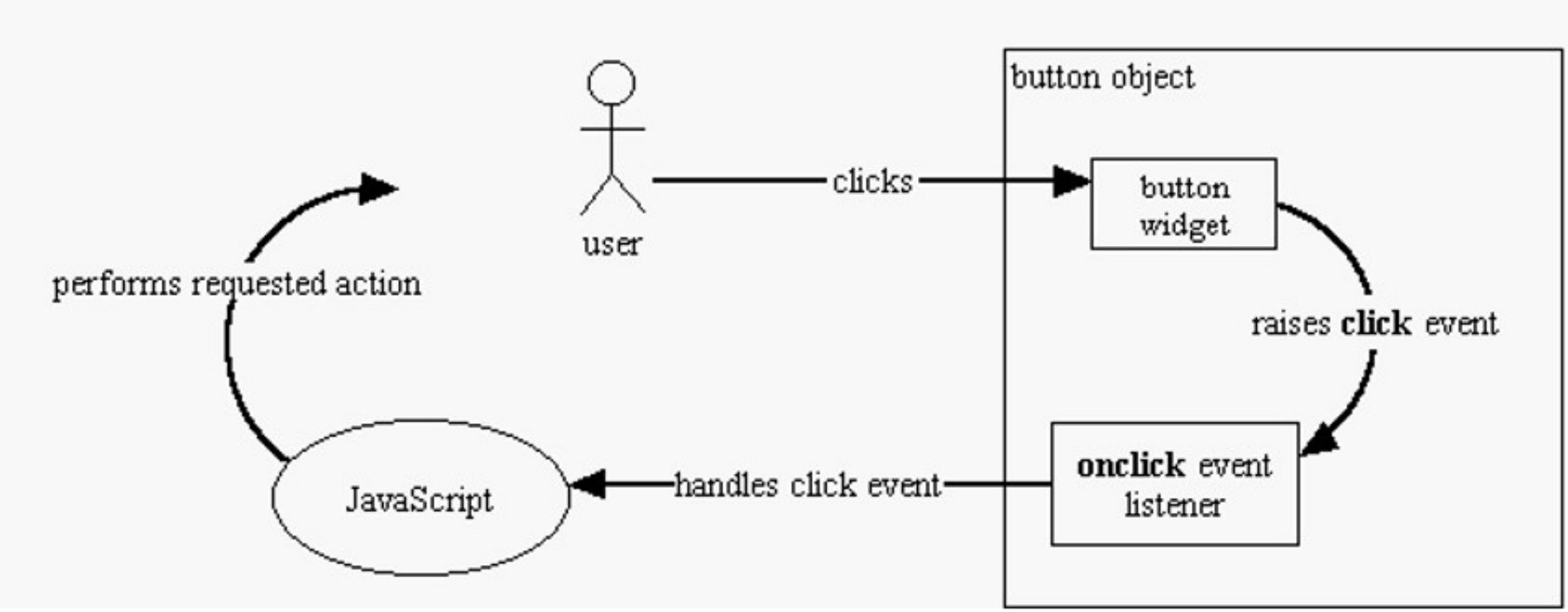
以物件為基礎的程式語言

- 物件成員由屬性(property)和方法(method)組成
- 事件--預先定義好的特定動作，通常由使用者或系統啟動



以事件驅動為基礎的程式語言

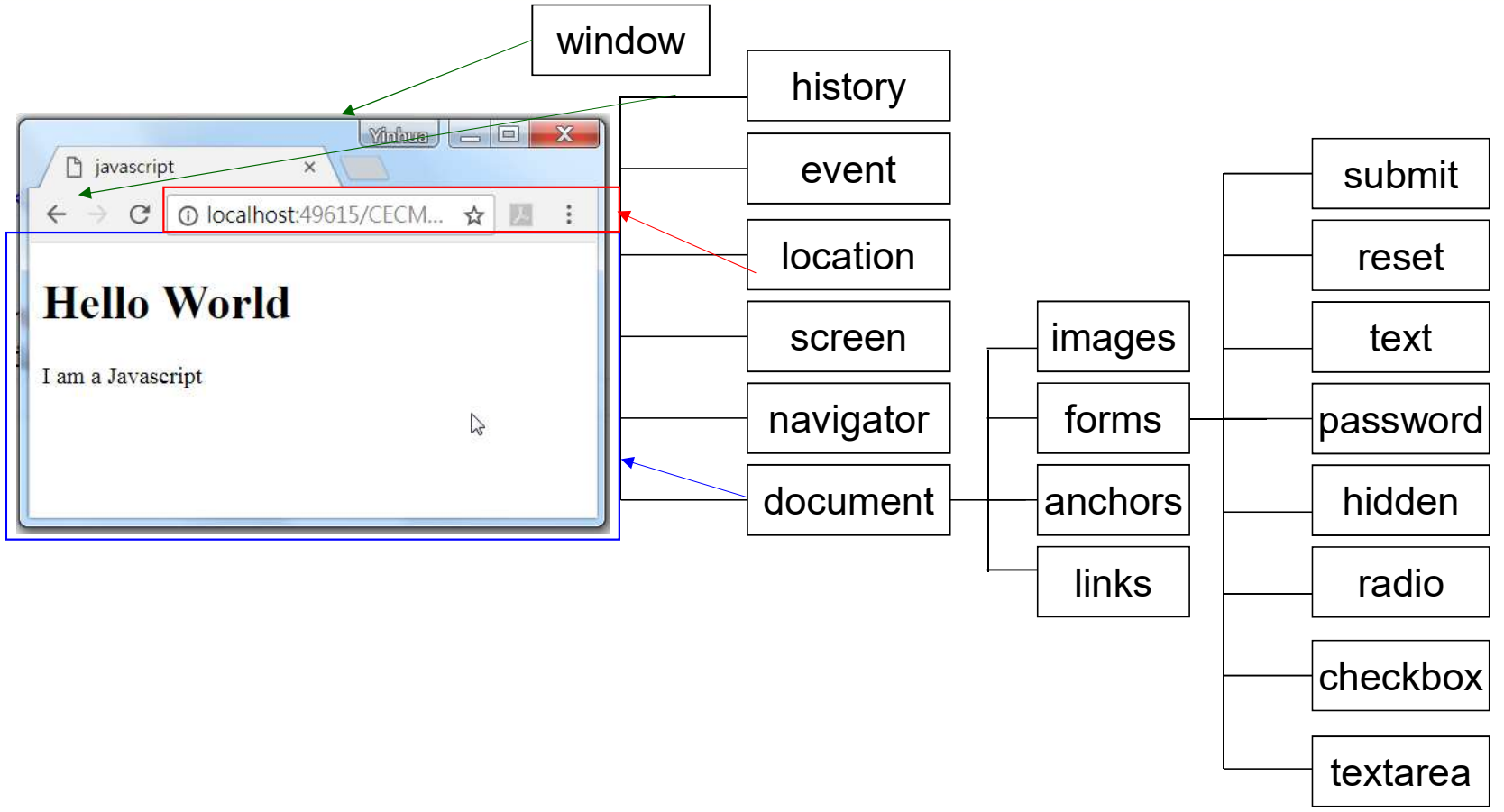
```
<input type="button" value="ClickMe" onclick="window.alert('JavaScript'); "/>
```



JavaScript 簡介

- 1995年由網景公司(Netscape co.)的Brendan Eich開發了一個名叫LiveScript的指令碼語言
- 1997年在歐洲電腦製造商協會(European Computer Manufacturers Association-ECMA International)協調下，確定統一標準:ECMA-262，稱為ECMAScript
- 是一種以物件及事件驅動為基礎的程式語言
- 必須要直接嵌入HTML文件中，才可執行
- JavaScript包括三大部分:ECMAScript、Browser Object Model(BOM)與Document Object Model(DOM)
- https://www.w3schools.com/
- https://developer.mozilla.org/zh-TW/docs/Web

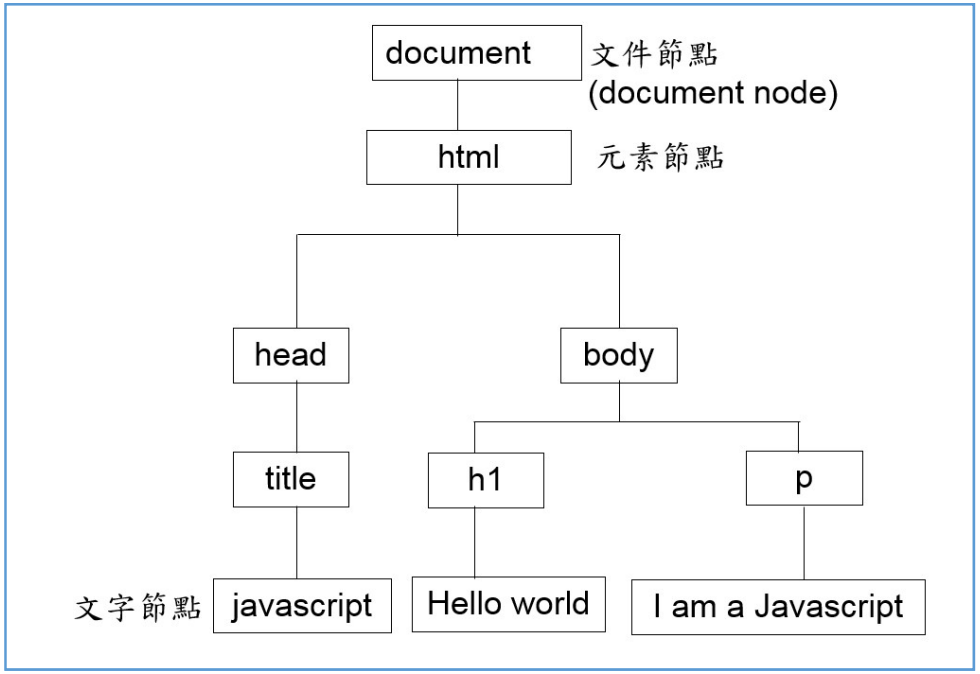
BOM 物件模型架構(Browser Object Model)



DOM(Document Object Model)

- DOM (Document Object Model，文件物件模型)，是一個以樹狀結構來表示 HTML 文件的模型，並定義讓程式可以存取及改變文件架構、樣式和內容的方法。
- https://dom.spec.whatwg.org/

```
<!DOCTYPE>
<html>
   <head>
         <title>javascript</title>
   </head>
   <body>
         <h1>Hello world</h1>
         <p>I am a Javascript</p>
   </body>
</html>
```



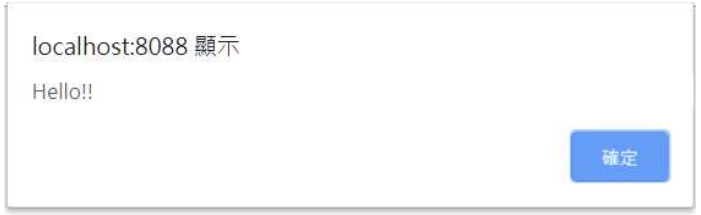
JavaScript 的好處

- 簡單易學
- 具跨瀏覽器之特性
- 提高網頁的互動性及趣味性
- 提供表單前端驗證
- 動態更新網頁部分內容

JavaScript 的基本架構

- 直接嵌入在HTML文件中的任何位置
- 建議寫法為

```
<body>
    <script>
        window.alert("Hello! ");
    </script>
</body>
```



- 建立Javascript檔案(first.js)

```
document.write("JavaScript");
```

```
<body>
  <script src="first.js"></script>
</body>
```

01first.html

01first.js

JavaScript 的撰寫格式

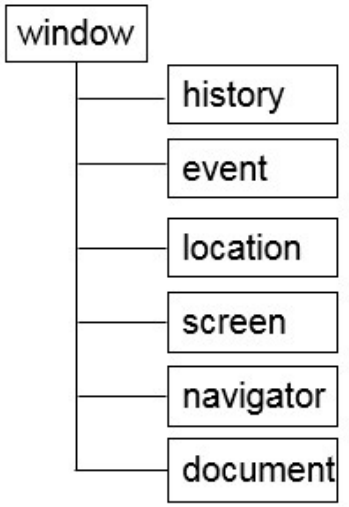
- 每個敘述不限制一定要寫成一行
- 行尾可加；號代表此敘述結束
- 程式區塊使用{}號包圍
- 程式的註解
  - 單行註解使用 //
  - 多行註解使用 /* */

JavaScript 的輸出(1/2)

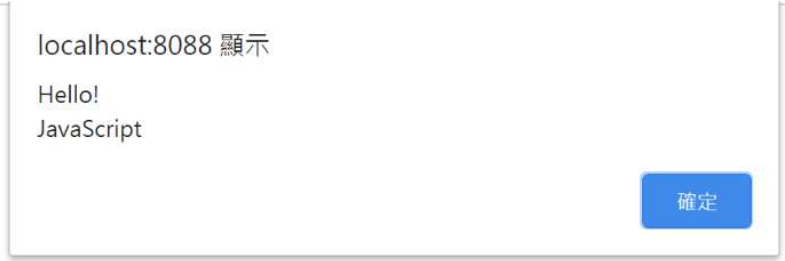
01first.html

01first.js

- 使用window.alert(message)
  - 顯示對話方塊及按鈕
- 使用document.write(exp1,exp2…)
  - 在文件上印出HTML 元素及字串
- 使用console.log(exp)
  - 在除錯模式(F12，Ctrl+Shift+I)顯示字串




```
<script>
    window.alert("Hello!\nJavaScript");
    document.write("Hello!<br>JavaScript");
   console.log("Hello!\nJavaScript");
</script>
```



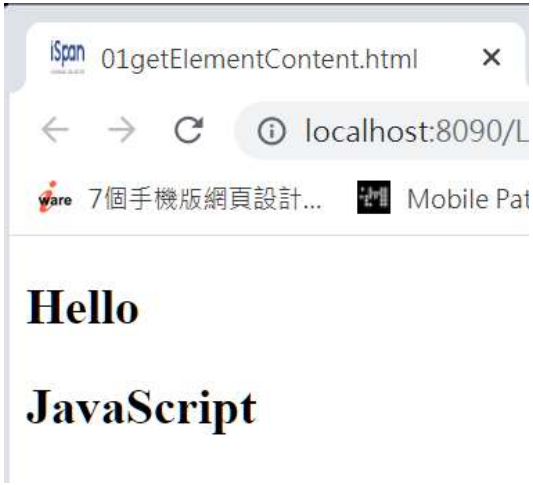
JavaScript 的輸出(2/2)

- 使用innerHTML
- var *elementObj*=document.getElementById(*elementId*)
  - 取得文件上標籤元素
- innerHTML：取得或設定元素內容

取得或改變元素的內容

content

```
<body>
    <div id="demo"></div>
    <script>
        var ele=document.getElementById("demo");
        ele.innerHTML="<h2>Hello</h2>";
        ele.innerHTML +="<h2>JavaScript</h2>";
    </script>
</body>
```



02getElementContent.html

單元二:變數與資料型別

- JavaScript字面值(literal)
- JavaScript變數
- JavaScript保留字
- JavaScript跳脫字元
- JavaScript資料型別
- 資料型別的轉換與取得

JavaScript字面值(literal)

- 各種資料類型中實際儲存值的表示法，永遠不變的值
- 數字字面值
  - 123.45,12e3 ， -6 ，0b1010(2進制) ， 0o12(8進制) ， 0x4e00(16進制)
- 字串字面值
  - "javascript"，'javascript'
- 布林字面值
  - true,false
- 函數字面值(function literal) (單元四)
  - function(){…}
- 物件字面值(單元五)
  - {…..}
- 陣列字面值(單元五)
  - […….]
- RegExp字面值 (單元五)
  - /……../

```
//literal(字面值)
console.log(0b1010);
console.log(0o12);
console.log(0x4e00);
```

01varDeclaration.html

JavaScript變數(identifier)

- 變數的宣告
  - 語法：var identifier [=value];
  - 用 var宣告變數 var intAge = 25;
  - 直接指定變數值 strName = "Sherman";
  - 一般變數宣告後不分資料型別(loose typed language), 真正的資料型別將視 assigned value 而定(dynamic language)

```
var intAge; //變數宣告
intAge=25; //變數定義

var intAge=25; //變數宣告及定義

intAge=25;

console.log(intAge);
```

01varDeclaration.html

識別字的規則

- 變數的命名規則
  - 第一個字母應為大小寫英文字母或下底線(_)或$
  - 大小寫視為不同變數 (Java , java)
  - 除了第一個字母外，變數從第2個字元以後可為大小寫英文字, 數字, 下底線_
  - 變數名稱不可使用保留字

```
var intAge;  //正確
var strName; //正確
var _sysName; //正確
var $price; //正確
var 3car;  //錯誤
```

JavaScript 保留字

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Words marked with* are new in ECMAScript 5 and 6.

let 和 const (ES6)

- let 宣告
  - 語法：let identifier [=value];
  - 不允許變數重複宣告
  - 區塊範圍

```
if(true){
   var username = "Jack";
}
console.log(username );  //會印出Jack?
```

```
if(true){
   let age = 26;
}
console.log(age);   //會印出26?
```

- const宣告
  - 語法：const identifier =value;
  - 宣告時必須給予值，之後不可重複指派
  - 區塊範圍，也不能重複宣告

```
const TAX=0.6;
console.log(TAX);
```

01varDeclaration.html

樣板字面值(Template Literals)(ES6)

- 在反引號(back-ticks (`))的符號中
  - 可以使用${變數名稱}讀取變數中的值
  - 程式碼可以換行

```
let username = "David";
let age = 25;

console.log("Hello! name is " + username + " , and I am " + age + " years
old ");

console.log(`Hello my name is ${username}, and I am ${age} years old.`);
```

01varDeclaration.html

跳脫字元(escape sequence)

- 在字串中若要表示特定字元，或該字元具有特殊函義

跳脫字元	代表字元
\t	水平定位字元(\u0009)
\n	換行(\u000A)
\"	雙引號(\u0022)
\'	單引號(\u0027)
\\	反斜線(\u005C)
\xXX	兩位數的十六進位
\uXXXX	以十六進位數指定 Unicode字元輸出

```
console.log("\x69\x53\x70\x61\x6E");
console.log("\u8CC7\u5C55\u570B\u969B");
```

01varDeclaration.html

JavaScript資料型別 Data types

- 基本資料型別
  - 數字(number)型別:雙精準度浮點數(精準度15位)，共64 bits
  - 字串(string)型別: 以雙引號或單引號括起來，如:"JavaScript"
  - 布林(boolean)資料型別: true , false
    - 若結合布林值作+、-、*、/等運算，true會被當作1，而false會被當作0
    - 若在真假判斷式中，任何值都可轉為布林值。**0 、 NaN、""、null、undefined轉為false，其它的值，包括所有物件與陣列則轉為true。**
  - null(object)型別：無值或無物件
  - undefined (undefined)型別：宣告時未指定值或不存在的物件
- 物件資料型別
  - 物件
  - 陣列
  - 函數(function)

```
let a=null;
let b=a+2;
console.log(b);
```

02DataTypes.html

資料型別的取得

- 取得變數的基本資料型別
  - 使用 typeof 運算子
  - 一般變數宣告後不分資料型別(loose typed language), 真正的資料型別將視 assigned value 而定(dynamic language)

```
var info=12;
info="JavaScript";
console.log (typeof info);

var a=true;
var b=a+1;
if(b)
   window.alert(true);
else
   window.alert(false);
```

02DataTypes.html

資料型別的轉換

- 字串轉換為數值
  - parseInt()方法 ： 將字串轉換成整數
  - parseFloat()方法：將字串轉換成浮點數
  - Number(string)方法：將字串轉換為數值

```
parseInt("33P");  //33
parseInt("12.522");
parseInt("P33");

parseFloat("5.22A22");
parseFloat("P5.22A22");

Number("33P");  //NaN
Number("12.522");
Number("P33");
```

- 數值轉換為字串
  - number + ""
    - 運算子的左右兩個運算元，只要一個為字串，會自動轉換為字串串接
  - number.toString()方法：數值轉換為字串

```
var intAge=23;
console.log(intAge+"years old");
var strAge=intAge.toString();
console.log(typeof strAge);
```

02DataTypes.html

單元三：運算子與敘述

- JavaScript 的運算子
- 增減數運算子
- 流程控制

JavaScript 的運算子

5 = 3 + 2

二元運算式= 運算元1 運算子 運算元2

expression =operand operator operand

- 算術運算子(arithmetic operators)
  - + , - , * , / , % , ++ , --
- 比較性運算子 (comparision operators)
  - = =, != , < , <= , > , >=
- 指派運算子
  - =
- 邏輯運算子(logical operators)
  - && , || , !
- 條件運算子(三元運算子)
  - （判斷式）?（真時回傳值）:（假時回傳值）
  - 範例: x > 0 ? x : -1

算術運算子

Operator 運算子	說明	範例
+	加法	10 + 10 = 20
-	減法	10 – 8 = 2
*	乘法	10 * 2 = 20
/	除法	10 / 2 = 10
%	取餘數	10 % 3 = 1
++	遞增	var a = 10; a++ ;
--	遞減	var a = 10; a-- ;

- 前置(prefix)型

b = 10 * ++a;

- 後置(postfix)型

b = 10 * a ++;

01OperatorPrecedence.html

關係運算子

運算子	說明	範例
==	比較是否相等(值)	10=="10" = true
===	嚴格比較是否相等(值及資料型別)	10==="10" = false
!=	比較是否不相等	10!="10" = false
!==	嚴格比較是否不相等	10!=="10" = true
>	大於	10>10 = false
>=	大於等於	10>=10 = true
<	小於	10<10 = false
<=	小於等於	10<=10 = true

01OperatorPrecedence.html

邏輯運算子

- 結果不一定為布林值(0 、 NaN、""、null、undefined轉為false，其它的值，包括所有物件與陣列則轉為true)

&& AND運算子

A	B	A&&B
false	false	false
false	true	false
true	false	false
true	true	true

|| OR運算子

A	B	A||B
false	false	False
false	true	true
true	false	true
true	true	true

! NOT運算子

A	!A
true	false
false	true

```
console.log(5>3 && 6<9);
console.log(2<0 && 6>3);
console.log( 2>8 || 7<20 );
console.log( 2 || 7 ); //2
console.log(! -2 );
```

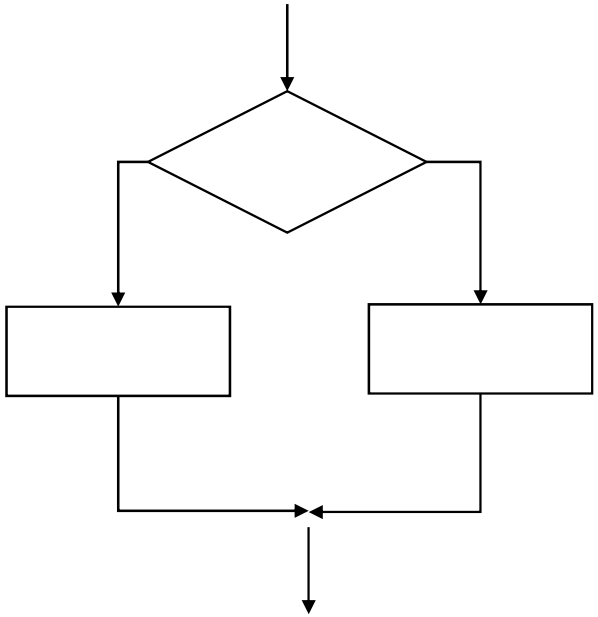
01OperatorPrecedence.html

流程控制

- 選擇敘述
  - If 單一選項敘述(區間值)
  - switch-case多重選項敘述(單一值)
- 重複結構
  - for 迴圈敘述
  - for in迴圈敘述(單元五)
  - for of 迴圈敘述(單元五)
  - while 迴圈敘述
    - Continue 與 break 敘述

單一選項敘述(if)

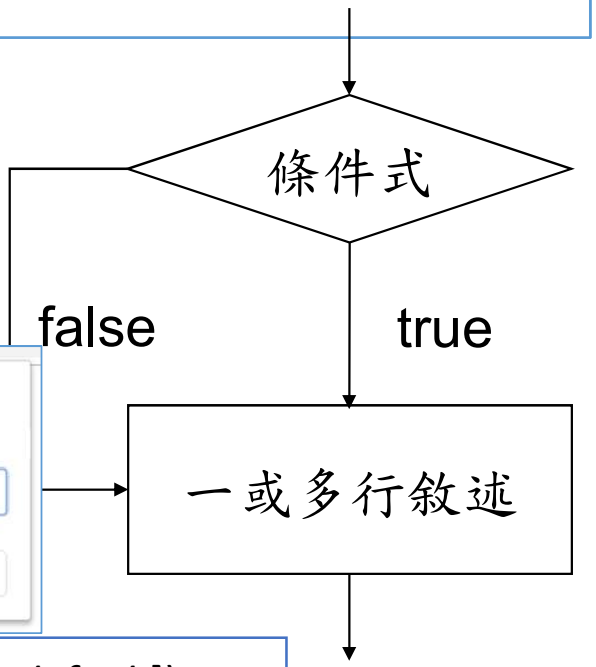
- 根據條件式的結果來決定執行不同的程式敘述
- 一般分為三種
  - 單一選擇結構
  - 雙向選擇結構
  - 多向選擇結構



單一選擇結構 if

```
if (條件式){
   // 條件式為true時執行，一行或多敘述
}
```

```
let score = parseInt(prompt("請輸入成
績?","90"));
if (score >= 60){
   console.log("及格了!!")
}
```

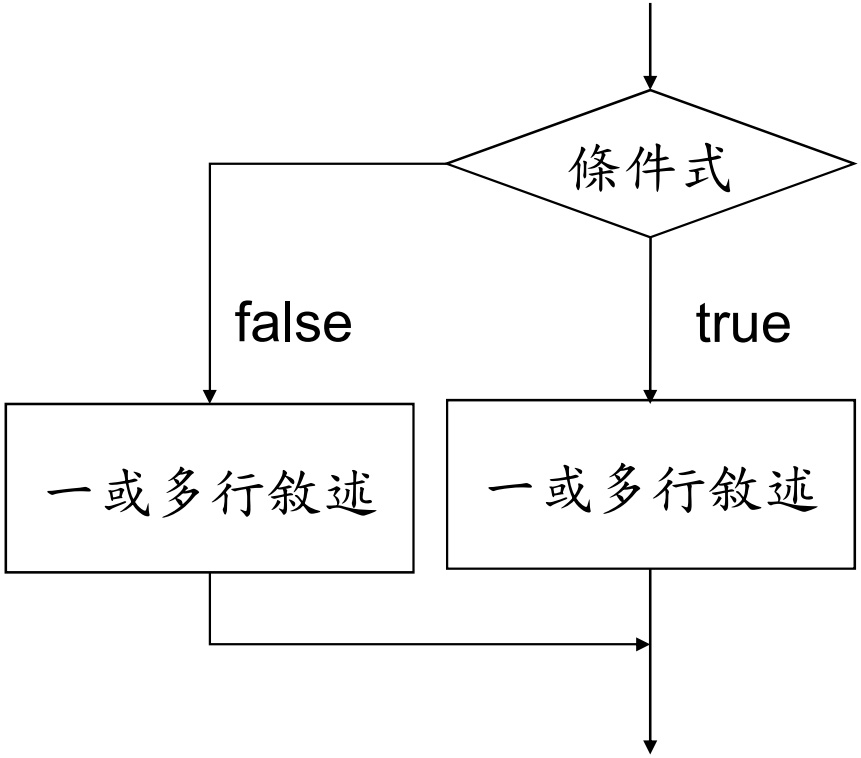


let *Stringvar* = window.prompt([String *message*],[String *default*])
顯示訊息對話方塊，讓使用者輸入值

02if.html

雙向選擇結構 if...else

```
if (條件式){
      // 條件式為true時執行
} else {
   一行或多敘述
}
```

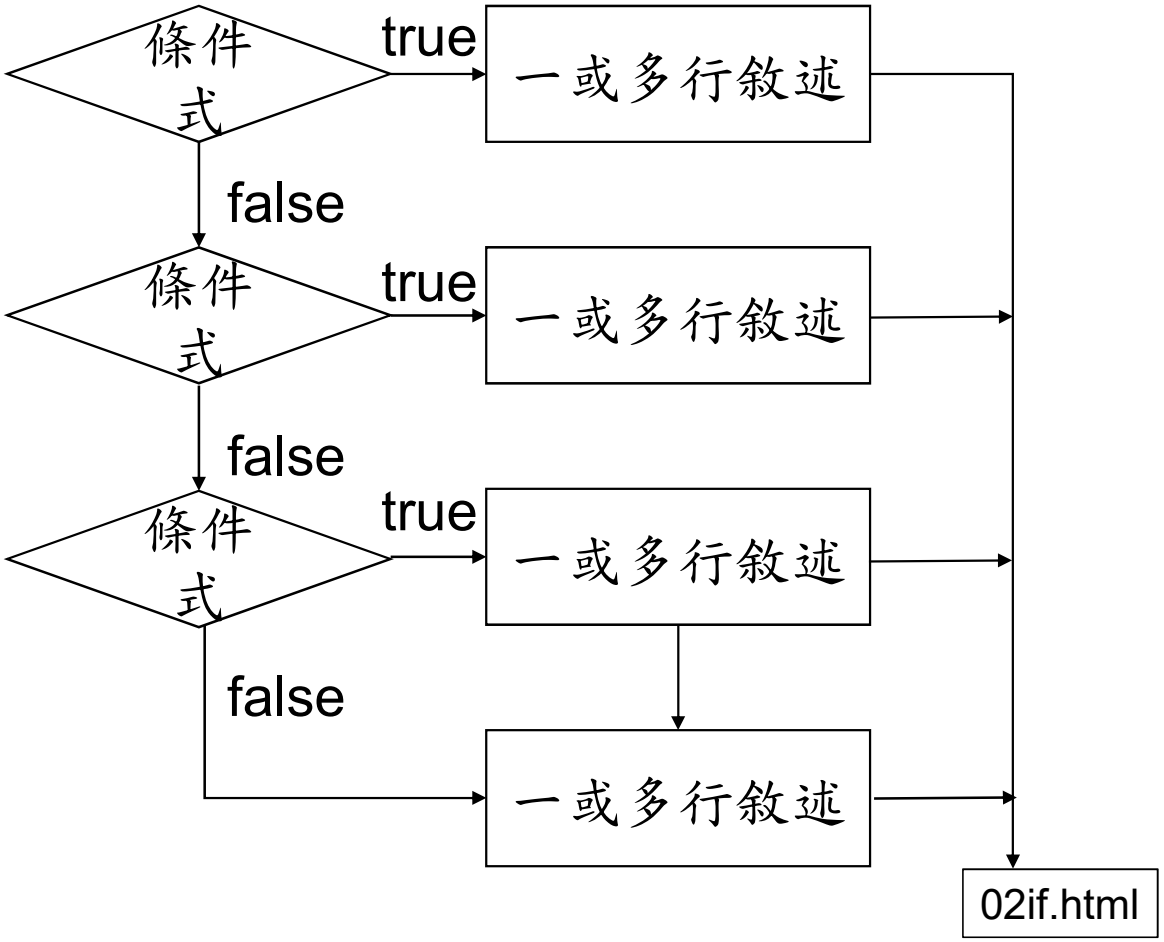



```
let score = parseInt(prompt("請輸入成績?"))
if (score >= 60){
   console.log("及格了!!")
} else {
   console.log("不及格!!")
}
```

02if.html

多向選擇結構 if...else if...else

```
if (條件式1){
 // 條件式為true時執行
}
else if (條件式2){
  // 條件式為true時執行
}
else if (條件式3){
  // 條件式為true時執行
}
...
else {
    一行或多敘述
}
```



if 範例

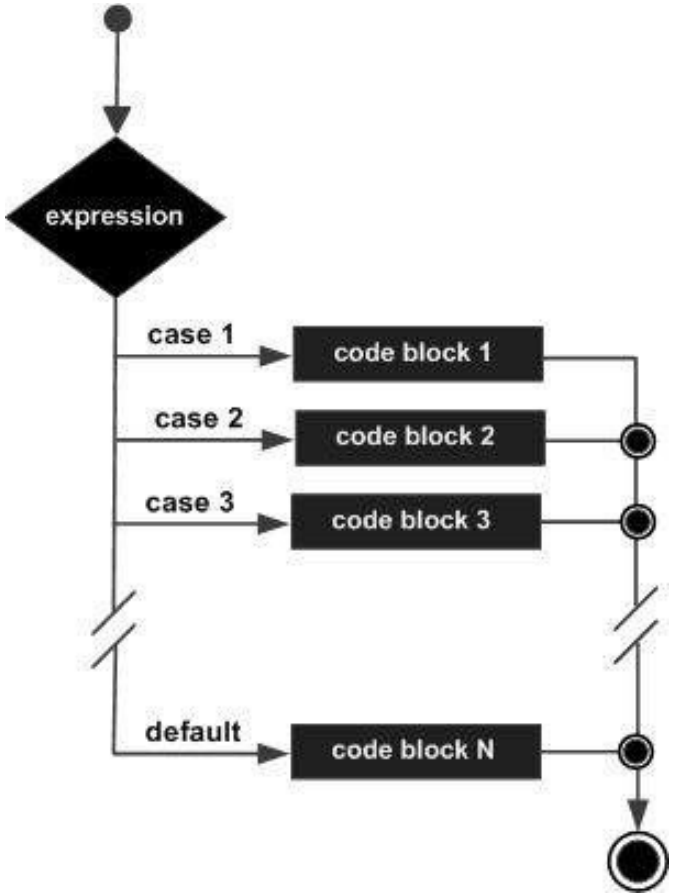
```
var Transportation = "",
money=parseInt(prompt("請輸入你有多少錢"));
if(money > 2500){
    Transportation = "飛機";
}else if(money > 1500){
   Transportation = "高鐵";
}else if(money > 1000){
   Transportation = "火車";
}else if(money > 500){
  Transportation = "統聯";
}else{
   Transportation = "摩托車";
}
console.log("台北到高雄坐 " + Transportation);
```

02if.html

多重選項敘述(switch)

- 多選一條件敘述

```
switch (expression) {
 case condition 1:
      statement(s)
      break;
 case condition 2:
      statement(s)
      break;
 default:
      statement(s)
}
```



switch 範例

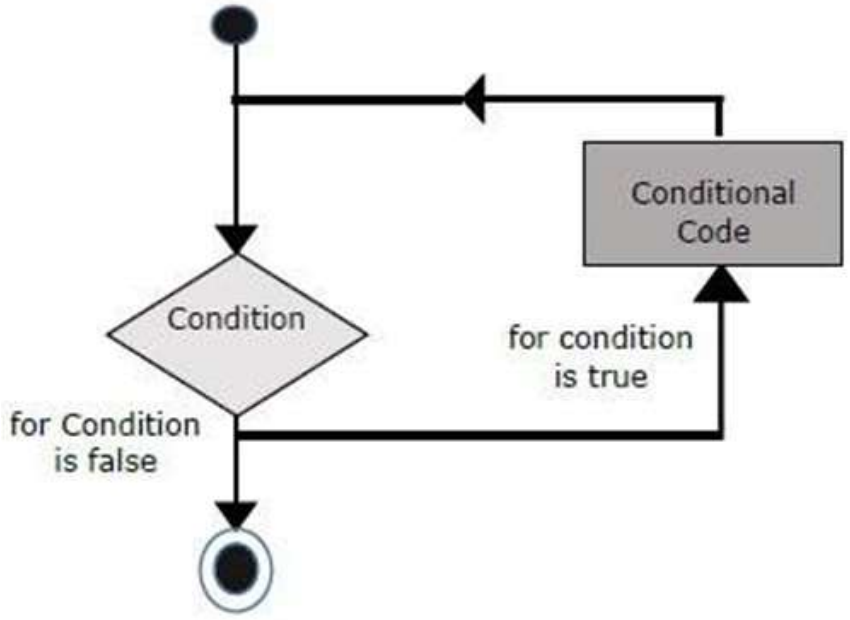
```
switch (theDay) {
   case 0:
      aDay = "日";
      break;
   case 1:
      aDay = "一";
      break;
   case …
   case 6:
       aDay = "六";
       break;
   default:
       aDay = "?";
}
```

03swicth.html

for 迴圈敘述

- 用迴圈結構來多次執行某個程式碼區塊
- 迴圈結構需要設定終止條件，每次執行時判斷是否符合終止條件，符合就離開迴圈

```
for (初值;  邏輯判斷式;  更新初值)
{
    //邏輯判斷式為true
    //執行這裡面的程式碼
    //更新初值
    //重新判斷是否為true
    //重複以上步驟
    //直到判斷式為false
    //離開迴圈
}
```



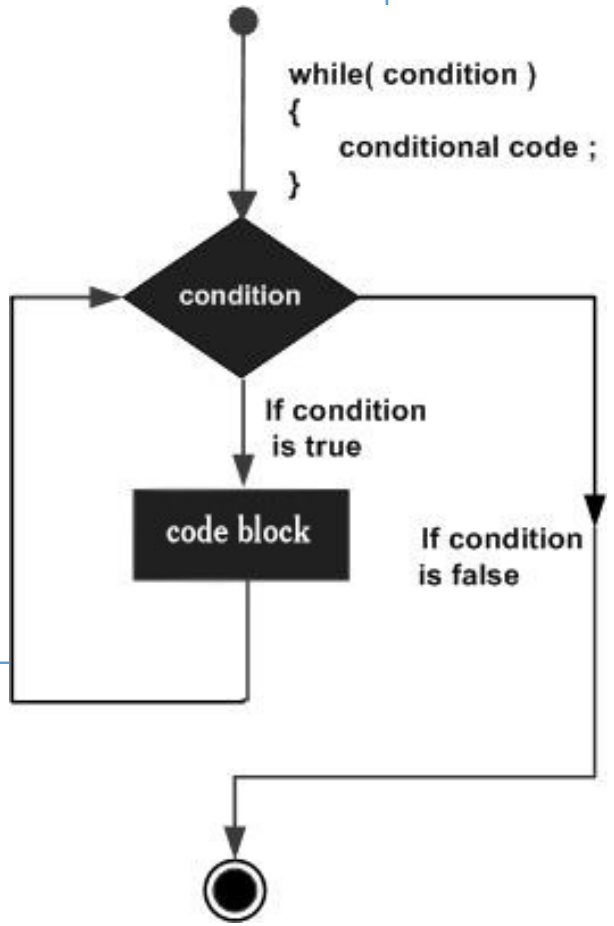
for 範例 1加到100

```
var i,sum=0
for (i=1; i<=100; i++)
{
    sum += i;
}
console.log("1+2+3+....+99+100=" + sum);
```

04for.html

while 迴圈敘述

```
while (邏輯判斷式){

    //邏輯判斷式為true
    //執行這裡面的程式碼
    //更新初值
    //重新判斷是否為true
    //重複以上步驟
    //直到判斷式為false
    //離開迴圈
}
```



while 範例 1加到100

```
var i=1,sum=0
while (i<=100)  //執行 1到100相加
{
   sum += i;
   i++;
}
console.log("1+2+3+....+99+100=" + Sum);
```

05while.html

break和continue

- break：跳離迴圈
- continue ：跳到迴圈結束點(})繼續執行

```
while (true){
    text = prompt("請輸入quit指令中斷程式：");
    if (text == "quit"){
        console.log("bye");
        break;
    }
    console.log(text);
}
```

```
for(var i=0;i<10;i++){
    if (i % 2 != 0){
        continue;
    }
    console.log(i);
}
```

05while.html

單元四：函數

- 函數的功能
- 函數寫法
- 變數的範圍
- 除錯功能

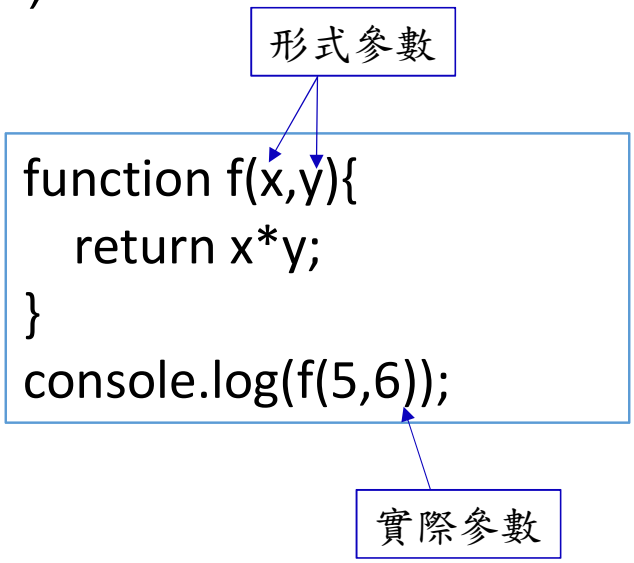
函數的功能

- 獨立完整的程式碼，可完成一個特定的功能
- 使用函數的好處
  - 避免重覆設定工作
  - 可將程式模組化，且易閱讀或修改
- 函數寫法分為
  - 函數宣告 (function declaration)
  - 匿名函數(anonymous function or function literal or function expression)
  - 箭頭函數(arrow function)

函數宣告

- 語法：

```
function 函數名稱(參數1,參數2, ..... parameters)
 {
     //將程式碼寫在這裡
   return 回傳值
 }
```



- function –保留字
- 函數名稱
  - 可自訂
  - 不可使用保留字
- 參數列(形式參數)– 可有可無 (參數1, 參數2, …)
- return 如果沒有加上回傳值，回傳值為undefined

01functionMethod.html

函數宣告與呼叫

- 函數定義

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

- 呼叫(calling)、使用、調用函數
- 函數名稱(參數值1,參數值2);

```
let R=fAdd(10,20);
console.log(R);

console.log(fAdd(10,20));
```

彈性的函數參數

- 函數的傳入參數，必須是與函數設定的參數一致，ES6提供了更彈性的變化，可以使用預設參數
  - 預設參數：參數名稱="參數值"

```
function f (x=10,y=20){
        return x*y;
}
console.log(f(5));
```

01functionMethod.html

匿名函數(anonymous function)

- 語法：

```
let variable =function(params){
statemenets
}
```

- function –保留字
- 沒有函數名稱

```
let f=function(x,y){
        return x*y;
  }
console.log(f(5,6));
```

01functionMethod.html

箭頭函數(arrow function)

- 語法:
  - let variable =(參數,…)=>{statemenets}
  - 函數沒有參數
    - () => { statements }
  - statement 只有一行，return 和 {} 可省略
    - (singleParam) => statements
  - 只有一個參數
    - (singleParam) => { statements }
    - singleParam => { statements }
  - 多個參數
    - (param1, param2, …, paramN) => { statements }
    - (param1, param2, …, paramN) => expression

```
let f=(x,y)=>{
        return x*y;
  }
console.log(f(5,6));
```

01functionMethod.html

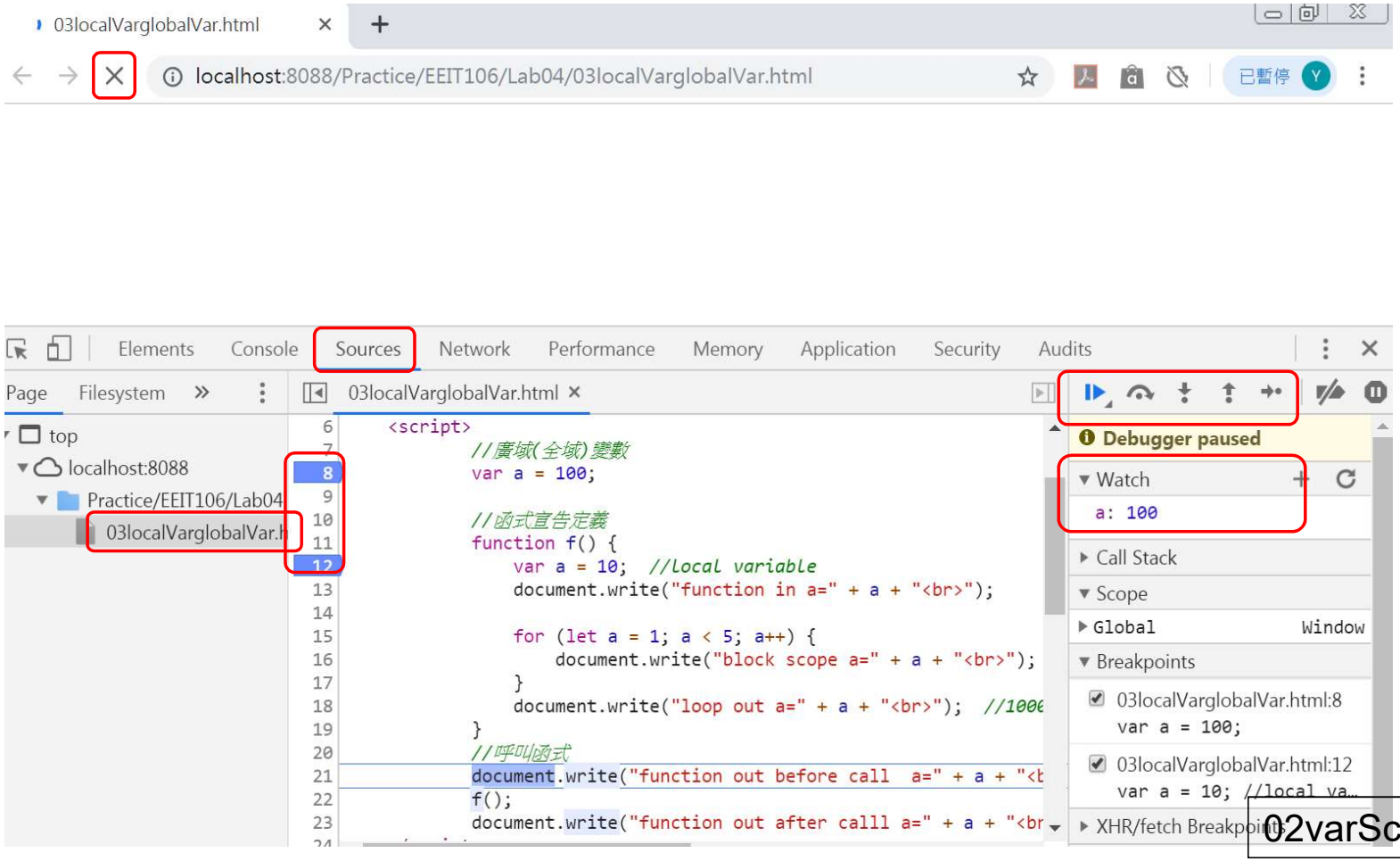
變數的範圍

- 區域(local)變數
  - 用 var 在函數內宣告
- 廣域(global)變數
  - 在函數外宣告
  - 在函數內宣告，但不使用 var
- 區塊(block)變數
  - 在{...}區塊用 let 宣告

```
var scope="global";
function f(){
   var scope="local";
   for(let scope=1;scope<=5;scope++){
      console.log(scope);
   }
   console.log(scope);
}

f();
console.log(scope);
```

02varScope.html

除錯(debug)功能

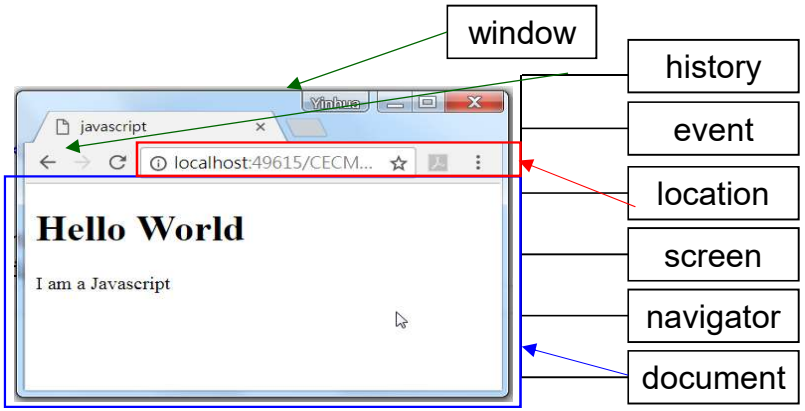

03localVarglobalVar.html
localhost:8088/Practice/EEIT106/Lab04/03localVarglobalVar.html
已暫停
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audits
Page
Filesystem
03localVarglobalVar.html
top
localhost:8088
Practice/EEIT106/Lab04
03localVarglobalVar.h
<script>
//廣域(全域)變數
var a = 100;
//函式宣告定義
function f() {
var a = 10; //local variable
document.write("function in a=" + a + "
");
for (let a = 1; a < 5; a++) {
document.write("block scope a=" + a + "
");
}
document.write("loop out a=" + a + "
");
}
//呼叫函式
document.write("function out before call a=" + a + "<b
f();
document.write("function out after calll a=" + a + "<br
Debugger paused
Watch
a: 100
Call Stack
Scope
Global
Window
Breakpoints
03localVarglobalVar.html:8
var a = 100;
03localVarglobalVar.html:12
var a = 10; //local va...
XHR/fetch Breakpoints
02varScope.html

單元五：JavaScript物件

- JavaScript物件種類
- JavaScript 物件的屬性與方法
- String 物件
- Date 物件
- Object 物件
- Array 物件
- JSON 物件
- Math 物件
- RegExp 物件

JavaScript物件種類

- 原生(Native)物件或內建物件
  - 是由ECMAScript規格所定義的物件或物件類別。例如，Date，Array，function，RegExp
- 主機(host)物件
  - 由JavaScript執行環境所定義的物件(例如Web瀏覽器)。
  - 客戶端用來表示網頁結構的HTMLElement物件。(https://developer.mozilla.org/en-US/docs/Web/API)
- 使用者定義(User-defined)物件
  - 使用JavaScript程式碼所建立的物件



JavaScript 物件的屬性與方法

- JavaScript 是一個以物件為基礎的程式語言
- 語法如下:
  - 物件.屬性[=設定值]
  - 物件.方法(參數[，參數]...)
- 原生物件
  - String 物件
  - Date 物件
  - Object 物件
  - Array 物件
  - JSON 物件
  - Math 物件
  - RegExp 物件

字串物件(String Object)1/3

01stringObject.html

- 字串變數與字串物件所運用的屬性與方法皆是一樣
- stringObj = new String(["stringLiteral"])
- 字串的方法：
  - charAt(*index*) ：回傳指定索引的字元，index從0開始
  - charCodeAt(index) ：回傳指定索引的Unicode 編碼
  - indexOf (*subString*) ：回傳字元第一次出現的索引
  - lastIndexOf (*subString*) ：回傳字元最後出現的索引
  - substr (*start[, length]*) ：從開始索引取得幾個字元； length省略，則取到字串結束
  - substring(*start, end*) : 從開始索引取到結束索引(不含結束索引)
  - slice(start, end) ：與substring()相同
  - split([*separator] [,limit]*) :用符號作分割，回傳分割後的字串陣列
  - toUpperCase() , toLowerCase() ：將字串轉換為大寫或小寫

字串物件屬性與方法範例

01stringObject.html

```
let strObj=new String("hello world"); //字串物件
console.log("strObj.charAt(3)="+strObj.charAt(3));  //l
console.log("strObj.charCodeAt(3)="+strObj.charCodeAt(3)); //108
console.log("strObj.indexOf('o')="+strObj.indexOf('o')); //4
console.log("strObj.lastIndexOf('o')="+strObj.lastIndexOf('o')); //7
console.log("strObj.substr(6,3)="+strObj.substr(6, 3)); //wor
console.log("strObj.substr(6)="+strObj.substr(6)); //world
console.log("strObj.substring(6,9)="+strObj.substring(6,9)); //wor
console.log("strObj.slice(6,9)="+strObj. slice(6,9)); //wor
console.log("strObj.split(' ')="+strObj.split(' ')); //hello,world
console.log("strObj.toUppeCase()="+strObj.toUpperCase());
```

取得元素的值

- var *elementObj*=document.getElementById(*elementId*)
  - 取得文件上指定Id屬性的元素物件
- var *Stringvar* = *elementObject*.value
  - 取得文件上元素的值

```
<body>
   <label>name:</label>
   <input type="text" id="idName" value= "Javascript" />
   <script>
          var  theNameObj=document.getElementById("idName");
          var  strVal=theNameObj.value;
          console.log(strVal);
          theNameObj.value="new Value" ;
   </script>
</body>
```

02checkPassword.html

字串物件應用

```
<body>
    <label>Password:</label>
    <input type="password" id="idPwd" size="6" /><!-- 瀏覽器執行到此標籤會建立一個物件-->
    <span id="idsp"></span><br/>
    <input type="button" id="idbut" value="checkPassword" onclick="checkPwd();" />

    <script>
        function checkPwd(){
            let PwdO=document.getElementById("idPwd");
            alert(typeof PwdO);
            let PwdOVal=PwdO.value;
            alert(typeof PwdOVal);
            alert(PwdOVal);

            if(PwdOVal=="")
                alert("you must enter");
            else
                alert("Password is :"+PwdOVal);
        }
    </script>
</body>
```

02checkPassword.html

String 物件(String Object)2/3

- 字元編碼：
  - ASC-II(American Standard Code for Information Interchange，美國資訊交換標準碼)是基於拉丁字母的一套電腦編碼系統。它主要用於英語系編碼。
  - unicode（萬國碼、國際碼、統一碼）是電腦系統的一項業界標準。對世界上大部分的文字系統進行整理、編碼，使得電腦可以用更為簡單的方式來呈現和處理文字。

String 物件(String Object)3/3

- UTF-8(8-bit Unicode Transformation Format)是一種針對unicode的可變長度字元編碼。
  - US-ASCII字元只需一個位元組編碼(U+0000至U+007F)。
  - 帶有附加符號的拉丁文、希臘文、西里爾字母、亞美尼亞語、希伯來文、阿拉伯文、敘利亞文及它拿字母則需要兩個位元組編碼(U+0080至U+07FF)。
  - 其他基本多文種平面（BMP）中的字元（這包含了大部分常用字，如大部分的漢字）使用三個位元組編碼(U+0800至U+FFFF)
  - 其他極少使用的Unicode輔助平面的字元使用四至六位元組編碼

- 字串的屬性
  - length：字串的長度

Date物件

- 建立日期物件語法：
  - dateObj = new Date()
  - dateObj = new Date(*dateString*)
  - dateObj = new Date (*year,month,date [,hour [, minutes[, seconds[, milliseconds]]]]*)

```
//方法一
let d = new Date();
//方法二
let d = new Date("2023-01-06T11:20:30");
//方法三
let d = new Date(2023,2,6,11,20,30);
```

03DateObject.html

Date物件

- 日期物件的方法:
  - getFullYear() ：回傳西元年份值
  - getMonth() ：回傳月份值(0-11)
  - getDate() ：回傳日數(1-31)
  - getDay() ：回傳星期數(0-6)
  - getHours() ：回傳時數(0-23)
  - getMinutes() ：回傳分數(0-59)
  - getSeconds() ：回傳秒數(0-59)
  - getTime() ：回傳自1970/1/1 0:0:0算起之毫秒數

```
let theYear=d.getFullYear()-1911;
let theMonth=d.getMonth()+1;
let theDate=d.getDate();
console.log(new Date().getDay());
```

03DateObject.html

Object 物件

- 語法:
  - obj = new Object()
  - obj={property:value,property:value…}
  - property：可用雙引號(")或單引號(')包起來，也可不用引號
- 存取屬性:
  - obj.property
  - obj["property"]
- 存取方法:
  - obj.method()

```
let person=new Object();
person.firstName="Mary";
person.lastName="Wang";

let peson={firstName:"Mary",lastName:"Wang"};
```

```
person.fullName=function(){return this.lastName+" "+this.firstName;}
```

  - this：指向呼叫該函數的物件

04Object.html

JavaScript字面值(literal)

- 各種資料類型中實際儲存值的表示法，永遠不變的值
- 數字字面值
  - 123.45,12e3 ， -6 ，0b1010(2進制) ， 0o12(8進制) ， 0x4e00(16進制)
- 字串字面值
  - "javascript"，'javascript'
- 布林字面值
  - true,false
- 函數字面值(function literal) (單元四)
  - function(){…}
- 物件字面值(單元五)
  - {…..}
- 陣列字面值(單元五)
  - […….]
- RegExp字面值 (單元五)
  - /……../

```
//literal(字面值)
console.log(0b1010);
console.log(0o12);
console.log(0x4e00);
```

01varDeclaration.html

for/in 迴圈

■ 語法:

```
for(變數名稱 in 陣列或物件){

}
```

```
<div id="myDiv"></div>
<script>
  let myDiv = document.getElementById('myDiv');
  let person={firstName:"Mary",lastName:"Wang"};

 for(let obj in person){
      myDiv.innerHTML += obj +": "+person[obj]+"<br>";
 }
</script>
```

內容為屬性名稱

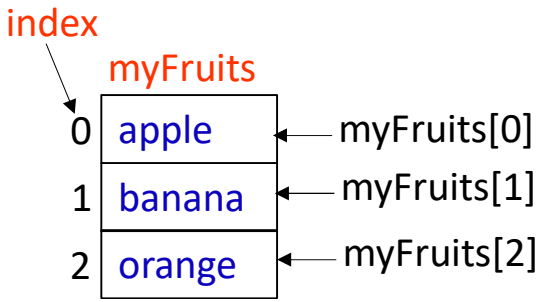
內容為屬性值

04Object.html

Array 物件

- 陣列是放置在連續記憶體的變數的集合
- 語法:
  - arrayObj = new Array([size])
  - arrayObj = [element0 [, element1[, ...[, elementN]]]])

```
//方法一
let myFruits= new Array();
myFruits[0]= "apple" ;
myFruits[1]= "banana" ;
myFruits[2]= "orange "

//方法二
let myFruits = ["apple", "banana", "orange"] ;
```



05ArrayObject.html

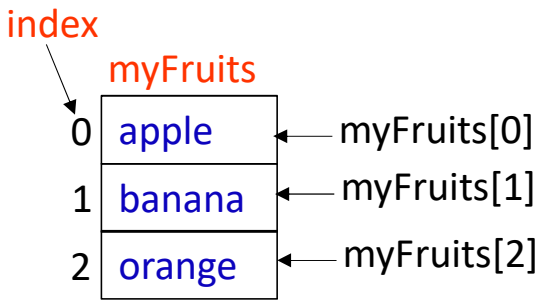
Array 物件的使用

- 索引值從0 開始，使用索引來取得元素(element)

```
for(變數名稱 of 可迭代物件){
        ....
}
```

for/of 使用在可迭代類型(如:array，字串)，不可以使用在object

```
let myFruits = ["apple", "banana", "orange"] ;
for (let i=0;i< myFruits.length;i++) {
      alert (myFruits[i])
 }
for(let fruit of myFruits){
        alert(fruit);
}
```

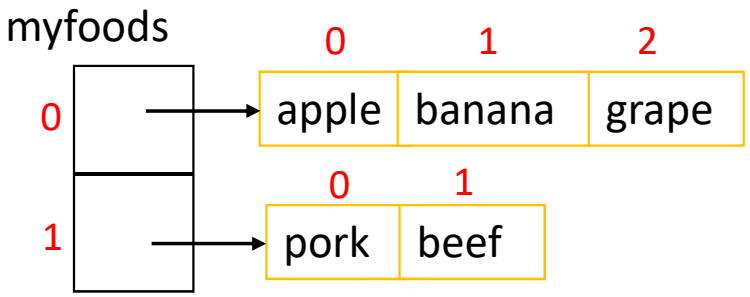


05ArrayObject.html

Array 物件範例

- myfoods Array

```
let myfoods = [["apple", "banana", "grape"], ["pork", "beef"]];
for (let i = 0, maxi = myfoods.length; i < maxi; i++) {
    for (let j = 0, maxj = myfoods[i].length; j < maxj; j++) {
        console.log(myfoods[i][j]);
    }
}
```



myfoods[0][0]="apple"

myfoods[1][0]="pork"

Array物件的方法

Array的方法	說明　　藍色方法表示原陣列會改變
concat(array)	和某陣列串接建立新的陣列。
join([separator])	將所有元素以分隔符號串連
reverse()	陣列元素順序反轉
sort()	陣列元素排序
push(item1,item2,…)	從陣列尾端加入多個元素值。
pop()	從陣列尾端取出一個元素值。
unshift(item1,item2,…)	從陣列前端加入多個元素值。
shift()	從陣列前端取出一個元素值。
slice(start, end)	從索引start到索引end複製部份陣列(不包含索引end)
splice(index [,delnum [,additem]])	插入、刪除、替換元素。

Array的屬性	說明
length	陣列元素個數

05ArrayObject.html

Array物件的方法範例

```
let myFruits = ["pineapple", "Water pear ", " Guava"];
let meat = ["beef", "pork", "chicken"];
let foods = myFruits.concat(meat);
myFruits.push("cherry");
myFruits.shift();
myFruits.splice(1,2, "grape", "orange");
console.log(myFruits);

//預設排序是以字串( unicode編碼)為順序
let br = [35, 6, 78, 12, 54, 9];
br.sort();
console.log(br);

//自訂排序規則，數值由小到大
let br = [35, 6, 78, 12, 54, 9];
br.sort(function (a, b) {
    return a - b; // 正值對調
});
console.log(br);
```

05ArrayObject.html

06Arraysort.html

JSON字串規則

- 全名是 JavaScript Object Notation
- javascript 下的一組物件描述方法
- 是一種資料交換的格式
- JSON 的文件格式(http://json.org/)
  - {} 來包住各物件(object)
  - [] 來包住各陣列(array)
  - "" 來包住各字串
  - 逗號來區隔各變數
  - 資料格式是:名稱/值成對
  - 名稱是字串要用雙引號包起來
  - 值可以是數字(整數或浮點數)、字串(在雙引號中)、邏輯值(true 或 false)、陣列(在方括號中)、物件(在大括號中)、null

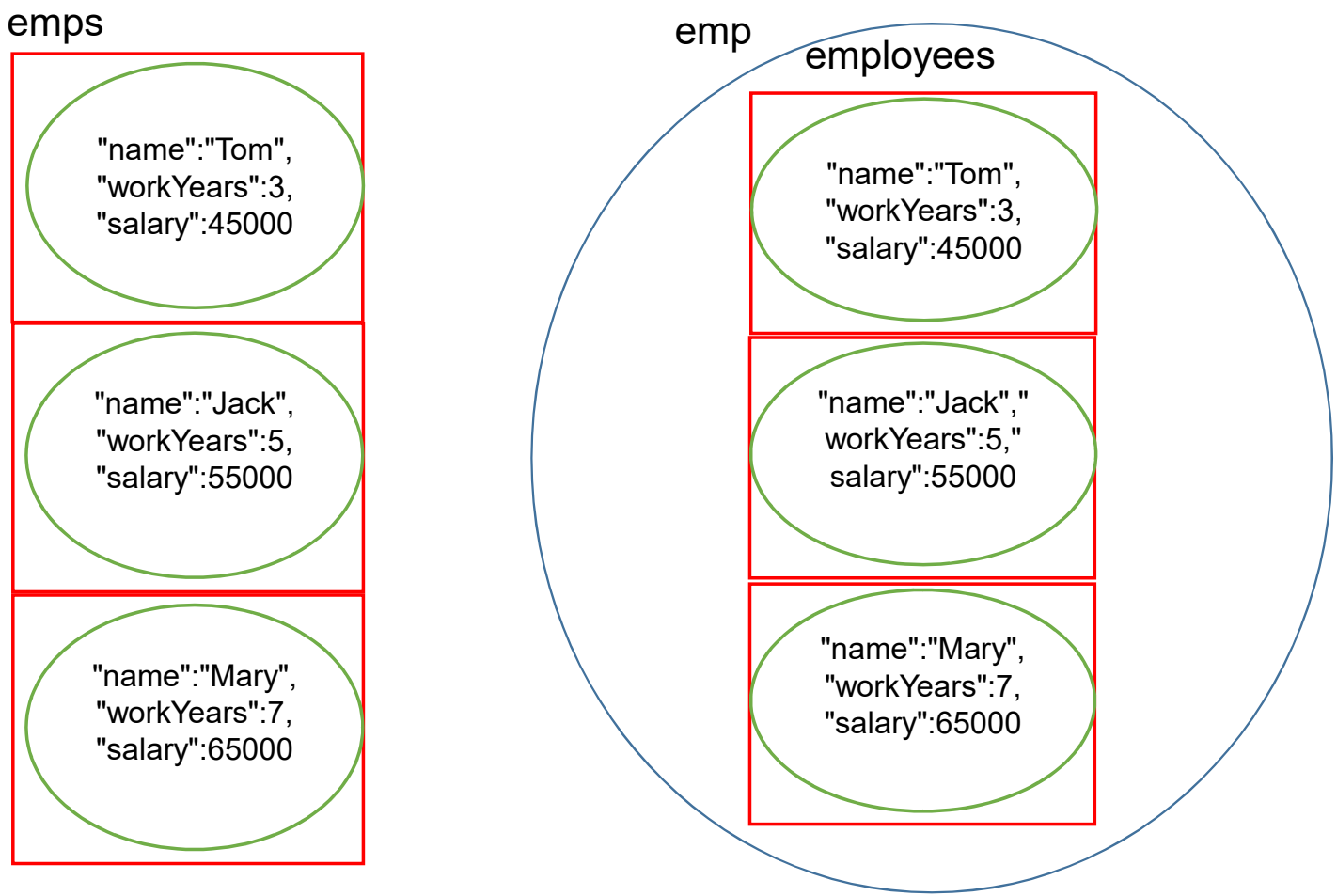
JSON範例

- JSON的格式，常用於網站上的資料呈現、傳輸 (例如將資料從伺服器送至用戶端，方便於網頁的顯示)。

```
var emps = [{"name":"Tom","workYears":3,"salary":45000},
            {"name":"Jack","workYears":5,"salary":55000},
            {"name":"Mary","workYears":7,"salary":65000}];
```

```
var emp = {"employees":[
            {"name":"Tom","workYears":3,"salary":45000},
            {"name":"Jack","workYears":5,"salary":55000},
            {"name":"Mary","workYears":7,"salary":65000}
          ]};
```

JSON範例


emps
"name":"Tom",
"workYears":3,
"salary":45000
"name":"Jack",
"workYears":5,
"salary":55000
"name":"Mary",
"workYears":7,
"salary":65000
emp
employees
"name":"Tom",
"workYears":3,
"salary":45000
"name":"Jack","
workYears":5,"
salary":55000
"name":"Mary",
"workYears":7,
"salary":65000

讀取 JSON 的資料

```
var myDiv = document. getElementById("div1");
var str = "<ul>";

var name,wy,s;
for (var i=0;i<emp.employees.length;i++){
  name = emp.employees[i].name;
  wy = emp.employees[i].workYears;
  s = emp.employees[i].salary;
  str += "<li>" + name + ", ";
  str += wy + "," + s + "</li>";
}
str += "</ul>";

myDiv.innerHTML = str;
```

07JSON.html

JSON 的方法

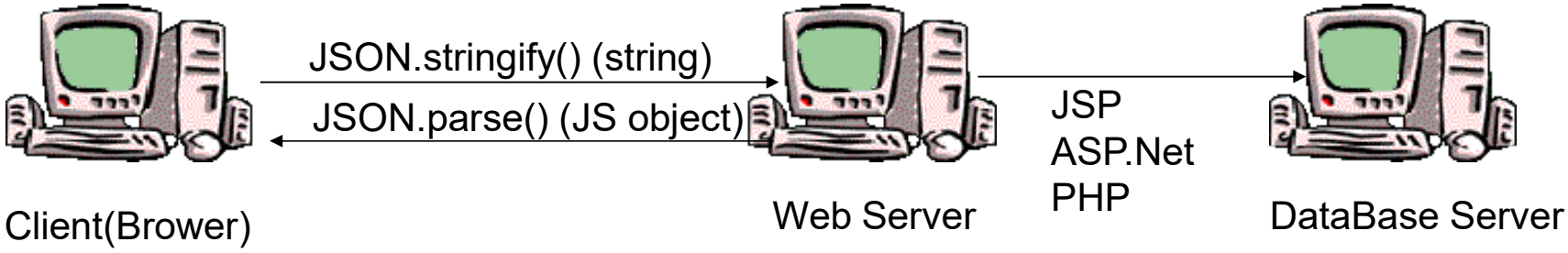
- JSON.stringify() 目的：
  - 當在 Clent 端操作的 JavaScript 物件要傳遞給 Server 端時，使用 JSON.stringify()將物件(或陣列，甚至是原始型別)序列化為一個 JSON 字串，然後將這個 JSON 字串傳送給 Server。
- JSON.parse() 目的：
  - 從 Server 端取得 JSON 字串後，利用JSON.parse() 將 JSON 字串剖析為 JavaScript 物件來操作使用。目前 Client 端與 Server 端溝通的方式大多以 JSON 字串格式來溝通。

```
let obj1 = {name: "Tom", b: 20, c: {d:40,e:50} };
let str = JSON.stringify(obj1);  //把物件轉成字串
let obj2 = JSON.parse(str);   //把字串轉成JavaScript原生物件
console.log("obj1:",obj1);
console.log(" str :", str );
console.log("obj2:", obj2);
```

08JSONMethod.html

Web 運作原理

- HTTP – HyperText Transfer Protocol(協定)
  - Protocol(協定)：電腦之間或網路上的設備之間溝通的語言
- HTML – HyperText Markup Language



Math 物件

Math的方法	說明	範例
abs(x)	求絕對值。	Math.abs(-3.25)=3.25
ceil(x)	取大於等於x的最小整數。	Math.ceil(6.1)=7
floor(x)	取小於等於x的最大整數。	Math.floor(6.9)=6
max(x, y, z, ..., n)	取最大值	Math.max(3,4,5)=5
min(x, y, z, ..., n)	取最小值。	Math.mix(3,4,5)=3
pow(x, y)	x的y次方。	Math.pow(2,3)=8
random()	0 到1 (不含1)之間的亂數	Math.random()=0.3345946351174467
round(x)	四捨五入求整數	Math.round(2.5)=3
sin(x弧度)	三角函數正弦	Math.sin(3)=0.1411200080598672
cos(x弧度)	三角函數餘弦	Math.cos(3)=-0.9899924966004454
PI	回傳PI值	Math.PI=3.141592653589793

```
let no = Math.floor(Math.random() * 6 + 1);
console.log(no);
```

09MathObject.html

RegExp 物件

- Regular Expression 語法:
  - re = new RegExp("pattern", "flag")
  - re = /pattern/flag
    - pattern 是正規表示法的字串，flag 則是比對的方式
    - flag 的值可能有三種
      - g：全域比對（Global match）
      - i：忽略大小寫（Ignore case）
      - gi：全域比對並忽略大小寫
- 方法:
  - *re*.test(*string*)：以字串 string 比對物件 re，並回傳比對結果(true 代表比對成功，false 代表比對失敗)

10regExpObject.html

正規式字元

字元	說明	簡單範例
\	避開特殊字元	/A*/ 可用於比對 "A*"，其中 * 是一個特殊字元，為避開其特殊意義，所以必須加上 "\"
^	比對輸入列的啟始位置	/^A/ 可比對 "Abcd" 中的 "A"，但不可比對 "aAb"
$	比對輸入列的結束位置	/A$/ 可比對 "bcdA" 中的 "A"，但不可比對 "aAb"
*	比對前一個字元零次或更多次	/bo*/ 可比對 "Good boook" 中的 "booo"，亦可比對 "Good bk" 中的 "b"
+	比對前一個字元一次或更多次，等效於 {1,}	/a+/ 可比對 "caaandy" 中的 "aaa"，但不可比對 "cndy"
?	比對前一個字元零次或一次	/e?l/ 可比對 "angel" 中的 "el"，也可以比對 "angle" 中的 "l"
.	比對任何一個字元（但換行符號不算）	/.n/ 可比對 "nay, an apple is on the tree" 中的 "an" 和 "on"，但不可比對 "nay" .
xy	比對 x 或 y	/a*b*/g 可比對 "aaa and bb" 中的 "aaa" 和 "bb"
{n}	比對前一個字元 n 次，n 為一個正整數	/a{3}/ 可比對 "lllaaalaa" 其中的 "aaa"，但不可比對 "aa"
{n,}	比對前一個字元至少 n 次，n 為一個正整數	/a{3,}/ 可比對 "aa aaa aaaa" 其中的 "aaa" 及 "aaaa"，但不可比對 "aa"
{n,m}	比對前一個字元至少 n 次，至多 m 次，m、n 均為正整數	/a{3,4}/ 可比對 "aa aaa aaaa aaaaa" 其中的 "aaa" 及 "aaaa"，但不可比對 "aa" 及 "aaaaa"
[xyz]	比對中括弧內的任一個字元	/[ecm]/ 可比對 "welcome" 中的 "e" 或 "c" 或 "m"
[^xyz]	比對不在中括弧內出現的任一個字元	/[^ecm]/ 可比對 "welcome" 中的 "w"、"l"、"o"，可見出其與 [xyz] 功能相反。（同時請同學也注意 /^/ 與 [^] 之間功能的不同。）
\b	比對英文字的邊界	例如 /\bn\w/ 可以比對 "noonday" 中的 'no' ; /\wy\b/ 可比對 "possibly yesterday." 中的 'ly'

正規式字元

\d	比對任一個數字，等效於 [0-9]	/[\d]/ 可比對 由 "0" 至 "9" 的任一數字 但其餘如字母等就不可比對
\D	比對任一個非數字，等效於 [^0-9]	/[\D]/ 可比對 "w" "a"... 但不可比對如 "7" "1" 等數字
\f	比對換頁(\u000C)	若是在文字中有發生 "換頁" 的行為 則可以比對成功
\n	比對換行符號(\u000A)	若是在文字中有發生 "換行" 的行為 則可以比對成功
\r	比對 游標返回(\u000D)	
\s	比對任一個空白字元（White space character），等效於 [\f\n\r\t]	/\s\w*/ 可比對 "A b" 中的 "b"
\S	比對任一個非空白字元，等效於 [^ \f\n\r\t]	/\S/\w* 可比對 "A b" 中的 "A"
\t	比對定位字元（Tab）(\u0009)	
\w	比對數字字母字元（Alphanumerical characters）或底線字母（"_"），等效於 [A-Za-z0-9_]	比對數字字母字元（Alphanumerical characters）或底線字母（"_"），等效於 [A-Za-z0-9_]
\W	比對非「數字字母字元或底線字母」，等效於 [^A-Za-z0-9_]	/\W/ 可比對 ".A _!9" 中的 "."、" "、"!"，可見其功能與 /\w/ 恰好相反。
\xxx	比對八進位，其中*xxx是八進位數目*	/\123/ 可比對 與 八進位的ASCII中 "123" 所相對應的字元值。
\x*hh*	比對兩位數的十六進位，其中*hh是十六進位數目*	/\x57/ 可比對 與 16進位的ASCII中 "57" (W)所相對應的字元。
\uhhhh	比對四位數的十六進位	/\u4E2D/ 可比對 與 16進位的unicode "4E2D" (中)所相對應的字元。
\0	比對NULL字元(\u0000)	

正規式字元(範例)

```
function checkEmail()
{
   let theEmail=document.getElementById("idEmail").value;
   re = /^.+@.+\..{2,3}$/;
   if (re.test(theEmail))
         alert("成功！符合「" + re + "」的格式！");
   else
         alert("失敗！不符合「" + re + "」的格式！");
}
```

10regExpObject.html

跳脫字元(escape sequence)

- 在字串中若要表示特定字元，或該字元具有特殊函義

跳脫字元	代表字元
\t	水平定位字元(\u0009)
\n	換行(\u000A)
\"	雙引號(\u0022)
\'	單引號(\u0027)
\\	反斜線(\u005C)
\xXX	兩位數的十六進位
\uXXXX	以十六進位數指定 Unicode字元輸出

01varDeclaration.html

JavaScript字面值(literal)

- 各種資料類型中實際儲存值的表示法，永遠不變的值
- 數字字面值
  - 123.45,12e3 ， -6 ，0b1010(2進制) ， 0o12(8進制) ， 0x4e00(16進制)
- 字串字面值
  - "javascript"，'javascript'
- 布林字面值
  - true,false
- 函數字面值(function literal) (單元四)
  - function(){…}
- 物件字面值(單元五)
  - {…..}
- 陣列字面值(單元五)
  - […….]
- RegExp字面值 (單元五)
  - /……../

```
//literal(字面值)
console.log(0b1010);
console.log(0o12);
console.log(0x4e00);
```

01varDeclaration.html

單元六：JavaScript的事件處理

- 事件處理程序
- 事件處理方式
- 常見事件

事件處理程序

- 事件是使用者對瀏覽器或網頁內容所做的某個動作，例如，滑鼠點擊(click)、鍵盤輸入(keydown)等，同時會產生event物件。
- 事件處理程序(Event Handlers)定義事件發生時要做的事
- 事件處理方式
  - HTML屬性事件處理程序
  - JavaScript屬性事件處理程序
  - W3C DOM 處理程序

事件處理方式(1/2)

- HTML屬性事件處理程序
  - <element event="function();">

```
<input type="button" onclick="check() " value="Try it"/>
<script>
        function check(){  }
</script>
```

- JavaScript屬性事件處理程序
  - *object.event=function;*

```
<input type="button" id= "myBtn" />
<script>
        document.getElementById("myBtn").onclick=check;
        function check(){   }
</script>
```

01eventHandler1.html

事件處理方式(2/2)

- W3C DOM處理程序
  - *object*.addEventListener(*event*, *function*,*usecapture*)

1.可省略
2. 預設為false

```
<input type="button" id= "myBtn" />
<script>
    document.getElementById("myBtn"). addEventListener("click", check,false);
    function check(){   }
</script>
```

01eventHandler1.html

事件處理方式範例

```
<h1 onclick="f1(); ">方法 1</h1>
<h1 id="s2">方法 2</h1>
<h1 id="s3">方法 3</h1>

<script>
    function f1() { alert("方法 1");   }
    function f2() { alert("方法 2");   }
    function f3() { alert("方法 3");   }
    document.getElementById("s2").onclick=f2;
    document.getElementById("s2").onclick=function(){
          alert("方法 2x");
    }
    document. getElementById("s3").addEventListener("click",f3);
    document. getElementById("s3").addEventListener("click",function(){
          alert("方法 3x");
    });
 </script>
```

01eventHandler1.html

函數(function)寫法

- 函數宣告

```
function fAdd1(a,b){
        return (a+b);
}
```

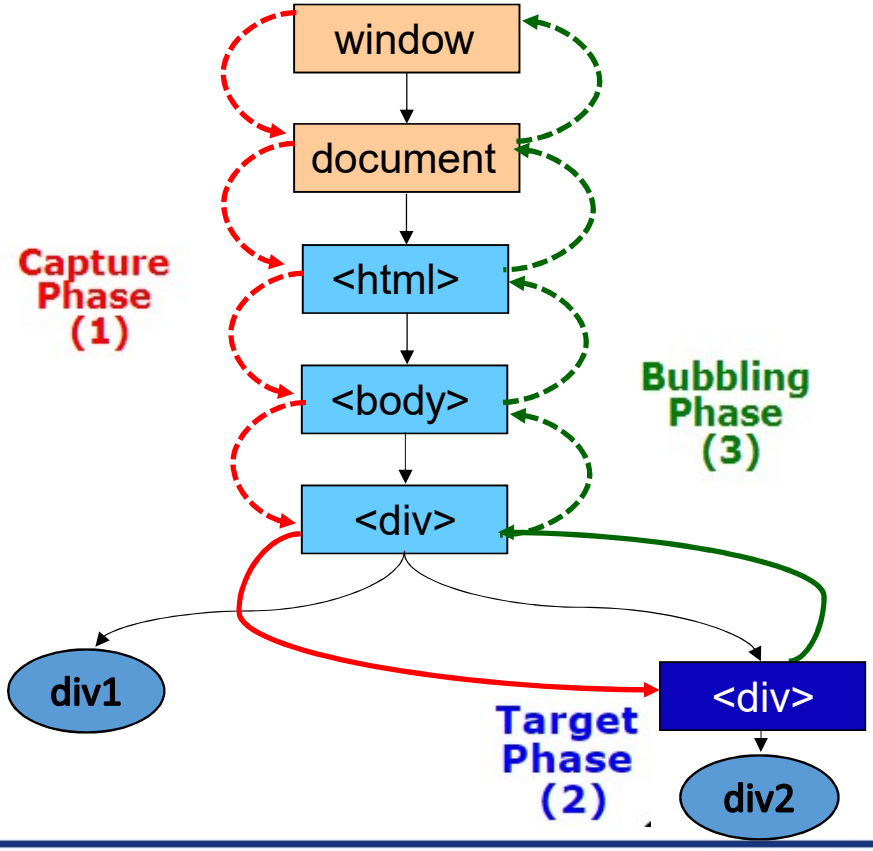
- 匿名函數

```
let fAdd2=function(a,b){
        return (a+b);
}
```

DOM2級事件(W3C)

階層式標籤，事件流程分三階段，
事件捕獲(capture)階段(true)==>處理目標階段==>事件氣泡(bubbling)階段(false)

```
object.addEventListener(event,handler,usecapture)
```



02eventHandlerAttachAdd.html

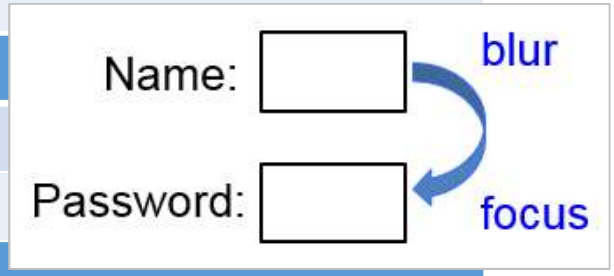
常用事件

03eventHandlerMouseoverout.html

04eventHandlerFocusBlur.html

滑鼠事件處理	說明
click、dblclick	單擊滑鼠左鍵
mousedown、mouseup	按下滑鼠左鍵時、放開滑鼠左鍵時
mouseover、mouseout	滑鼠移入元素、滑鼠移出元素
mousemove	滑鼠在元素上移動

鍵盤事件處理	說明
keydown，keyup	鍵盤按下、鍵盤放開
keypress	鍵盤按下放開



其他事件處理	說明
load	視窗載入所有資源(文件、圖檔、樣式檔)後觸發
DOMContentLoaded	文件全部讀取及解析後觸發，不需等待圖檔、樣式檔讀取
focus	選取元素取得焦點
blur	游標離開選取元素
change	改變文字方塊內容且離開或選取下拉式選單

單元七： Dynamic HTML

- Dynamic Styles 動態樣式
- Dynamic Content 動態內容

動態樣式

- 方法一
  - 透過 物件.style.css屬性 = css值 來改變樣式
- 方法二
  - 先預先定義class的樣式，再使用
    物件. className屬性=class樣式 來改變樣式

```
let theP = document.getElementById("myP");
theP.style.fontSize="20";
theP.style.textDecoration="underline";

theP.className = "s2";
```

01dynamicStyle.html

02dynamicStyleImg.html

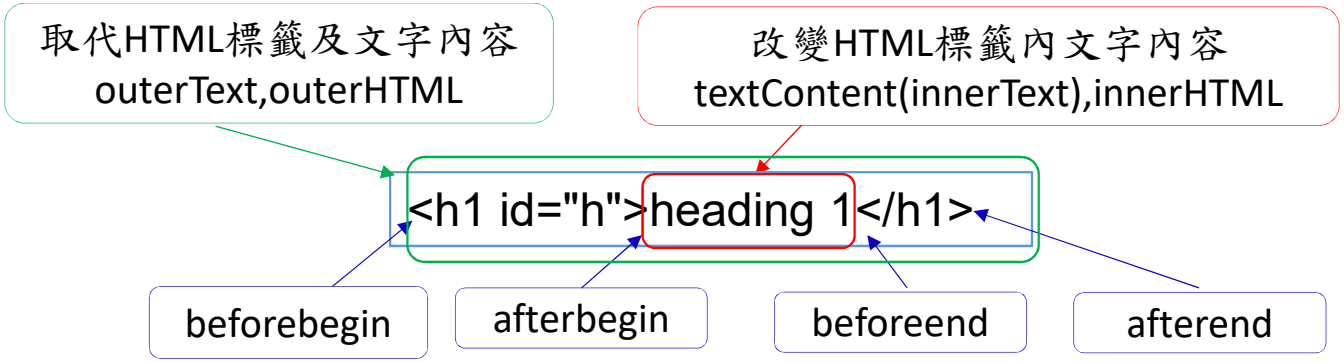
動態修改顯示內容

03dynamicContent.html

04checkPassword.html

05dynamicInsert.html

- 改變HTML標籤內的文字內容(成對標籤)
  - textContent ：讀取元素內容或將取代的內容視為純文字(plain)
  - innerHTML：讀取元素內容或將取代的內容視為HTML標籤解譯

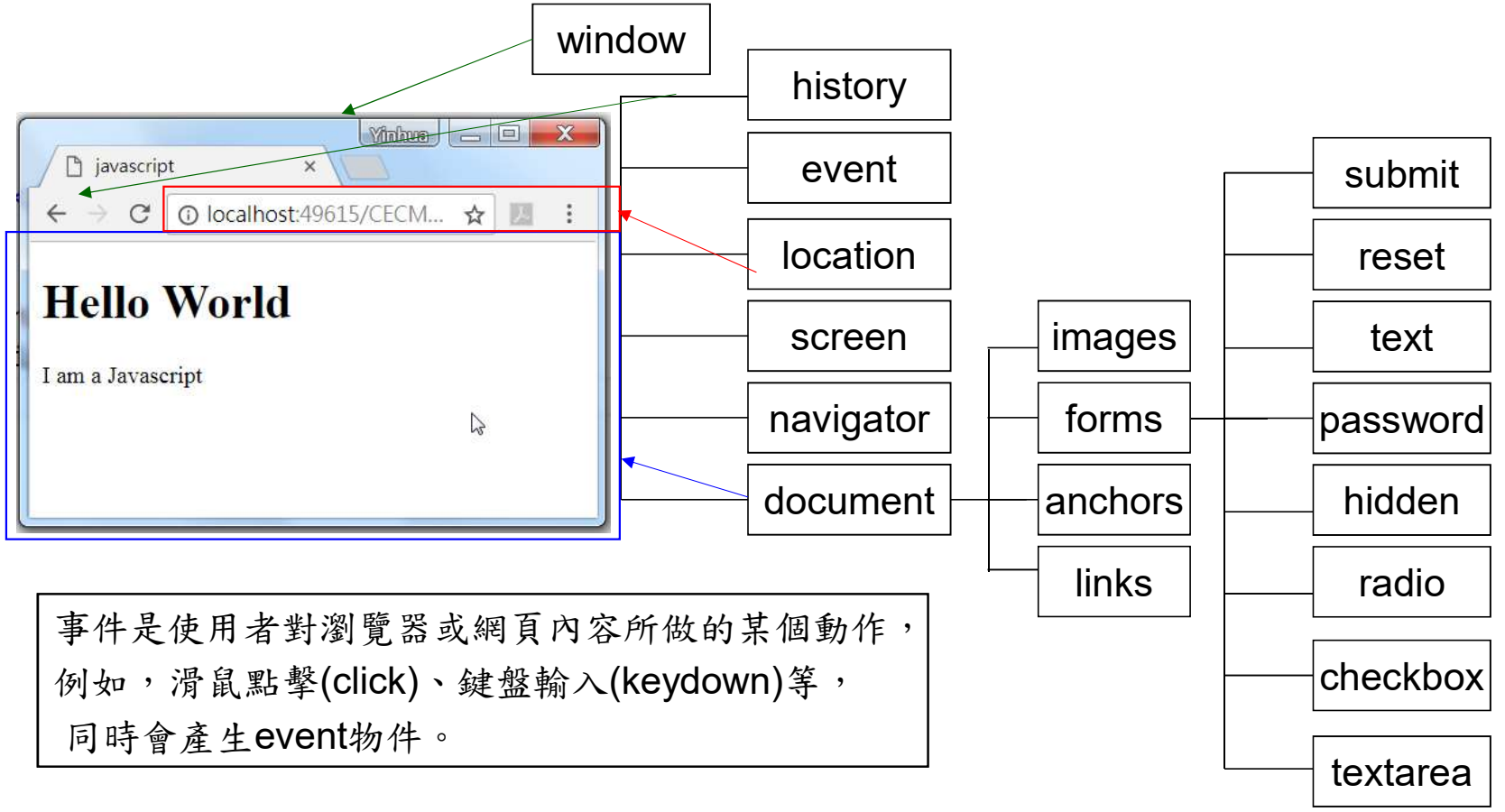


- 自訂文字加入的位置
  - insertAdjacentHTML(where, text)
  - insertAdjacentText (where, text)
    - Where :
    - afterbegin , beforebegin(開始標籤之前) , beforeend , afterend (結束標籤之後)

單元八：BOM 物件模型

- BOM 物件模型架構
- 視窗(window)物件
- 位址 (location)物件
- 事件(event)物件
- 文件(document)物件
- 圖片(image)物件
- 表單 (form) 物件

BOM 物件模型架構(Browser Object Model)



事件是使用者對瀏覽器或網頁內容所做的某個動作，例如，滑鼠點擊(click)、鍵盤輸入(keydown)等，同時會產生event物件。

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

視窗(window)物件

- window物件是瀏覽器物件模型的最高階層
- 代表目前正在使用的視窗
- window物件語法：

  window.方法()

  window.屬性

window 物件的計時器

- 計時器可以建立動態的網頁內容
- 方法
  - let timeoutID=setTimeout(*function*, *milliseconds*) 時間到之後，只執行一次
  - clearTimeout(timeoutID)停止 setTimeout 方法啟動的計時器
  - let IntervalID=setInterval(*function*, *milliseconds*) 時間到之後，每隔多少時間執行一次
  - clearInterval (IntervalID)停止setInterval方法啟動的計時器
- 範例
  - 電子鐘

01setTimeout.html

02setclock.html

window 物件的計時器(範例)

```
function f(){
    location.href="https://www.ispan.com.tw";
}
window.setTimeout(f, 2000);
```

```
    let d=new Date();
    let s1=parseInt(d.getSeconds()/10); //秒數的十位數
    let s2=d.getSeconds()%10;  //秒數的個位數
    //顯示圖片，對應秒數
    document.getElementById("picS1").src="WinImages/"+s1+".gif";
    document.getElementById("picS2").src="WinImages/"+s2+".gif";

   window.setInterval(setClock,1000);
```

01setTimeout.html

02setclock.html

Date物件

- 日期物件的方法:
  - getFullYear() ：回傳西元年份值
  - getMonth() ：回傳月份值(0-11)
  - getDate() ：回傳日數(1-31)
  - getDay() ：回傳星期數(0-6)
  - getHours() ：回傳時數(0-23)
  - getMinutes() ：回傳分數(0-59)
  - getSeconds() ：回傳秒數(0-59)
  - getTime() ：回傳自1970/1/1 0:0:0算起之毫秒數

```
let theYear=d.getFullYear()-1911;
let theMonth=d.getMonth()+1;
let theDate=d.getDate();
console.log(new Date().getDay());
```

03DateObject.html

位址 (location)物件

- 儲存載入網頁URL的相關資訊
- 屬性
  - href：轉向連結到其它網址
  - protocol , host , hostname , port , pathname, hash
- 方法
  - reload() ：重新載入目前網頁
  - replace(url)：使用新網頁取代目前網頁
- 範例

```
location.href="https://www.ispan.com.tw";
location.replace("https://www.ispan.com.tw");
```

03locationObject.html

事件(event)物件(1)

04eventObject.html

05eventObjectKeyboard.html

06eventTargetcurrentTarget.html

- 主要功能是用來取得事件發生的類型與位置
- 屬性
  - type ：事件發生的類型
  - button：回傳滑鼠按下的按鈕
  - key：回傳鍵盤的按鍵
  - altKey , ctrlKey , shiftKey ：回傳true 或 false
  - target：回傳事件觸發的元素
  - currentTarget：回傳事件正在處理時所在的元素，即事件處理程序所繫結的元素

以event物件作為參數來使用，IE8以前是透過全域變數window.event來取用這個物件

鍵盤事件處理	說明
keydown，keyup	鍵盤按下、鍵盤放開
keypress	鍵盤按下放開

事件是使用者對瀏覽器或網頁內容所做的某個動作，例如，滑鼠點擊(click)、鍵盤輸入(keydown)等，同時會產生event物件。

事件(event)物件範例

```
function mouseDown(event) {  //以event物件作為參數來使用
        console.log(event);
        alert(event.type);
        alert(event.button);
    }

function keyDown(event) {
        alert(event.key);
        if (event.altKey) alert("ALT");
        if (event.ctrlKey) alert("CTRL");
        if (event.shiftKey) alert("SHIFT");
    }
```

事件處理程序

- HTML屬性事件處理程序

```
<input id="myBtn"  type="button" onclick="check() " value="Try it"/>
```

- JavaScript屬性事件處理程序
  - *object.event=function;*

```
<script>
      document.getElementById("myBtn").onclick=check;
      function check(){   }
</script>
```

- W3C DOM處理程序
  - *object*.addEventListener(*event*, *function,usecapture*

1.可省略
2.預設為false

```
<script>
    document.getElementById("myBtn"). addEventListener("click", check,false);
</script>
```

JavaScript資料型別 Data types

- 基本資料型別
  - 數字(number)型別:雙精準度浮點數(精準度15位)，共64 bits
  - 字串(string)型別: 以雙引號或單引號括起來，如:"JavaScript"
  - 布林(boolean)資料型別: true , false
    - 若結合布林值作+、-、*、/等運算，true會被當作1，而false會被當作0
    - 若在真假判斷式中，任何值都可轉為布林值。**0 、 NaN、""、null、undefined**轉為**false，其它的值，包括所有物件與陣列則轉為true**。
  - null(object)型別：無值或無物件
  - undefined (undefined)型別：宣告時未指定值或不存在的物件
- 物件資料型別
  - 物件
  - 陣列
  - 函數(function)

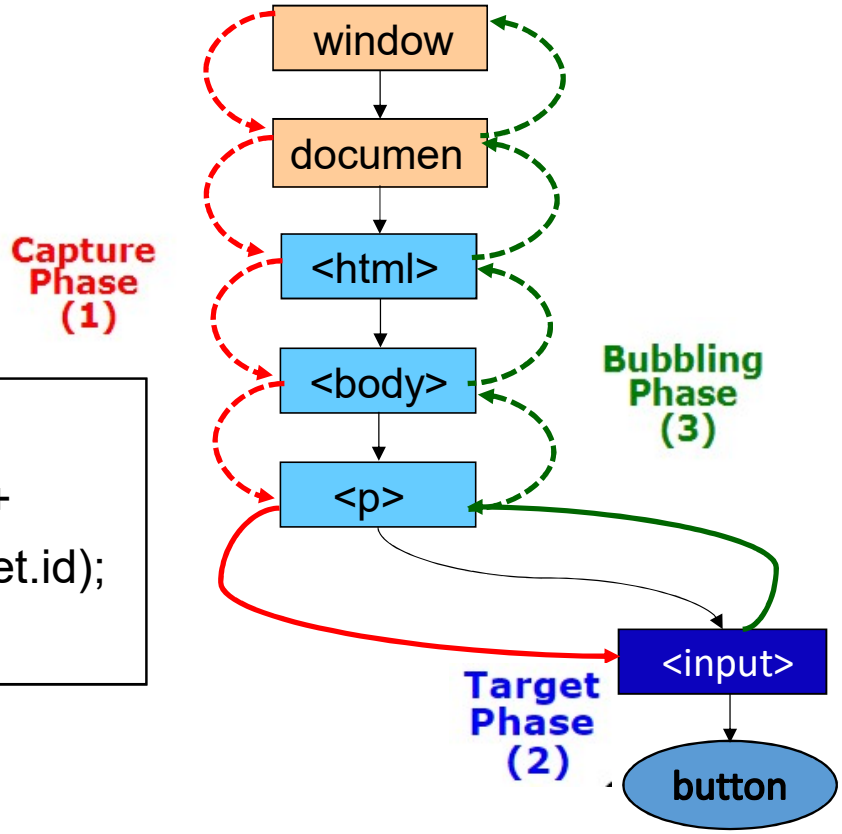
```
let a=null;
let b=a+2;
console.log(b)
```

02DataTypes.html

DOM2級事件(W3C)

階層式標籤，事件流程分三階段，
事件捕獲(capture)階段(true)==>處理目標階段==>事件氣泡(bubbling)階段(false)

```
object.addEventListener(event,handler,usecapture)
```




```
function clickInput(event) {
    alert("input target="+event.target.id+"\n"+
    "input currentTartet="+event.currentTarget.id);
}
```

06eventTargetcurrentTarget.html

事件(event)物件(2)

- 方法
  - stopPropagation() ：取消事件氣泡(event bubbling)
  - preventDefault() ：停止(取消)預設的動作

```
function clickf(event) {
        event.stopPropagation();
        event.preventDefault();
}
```

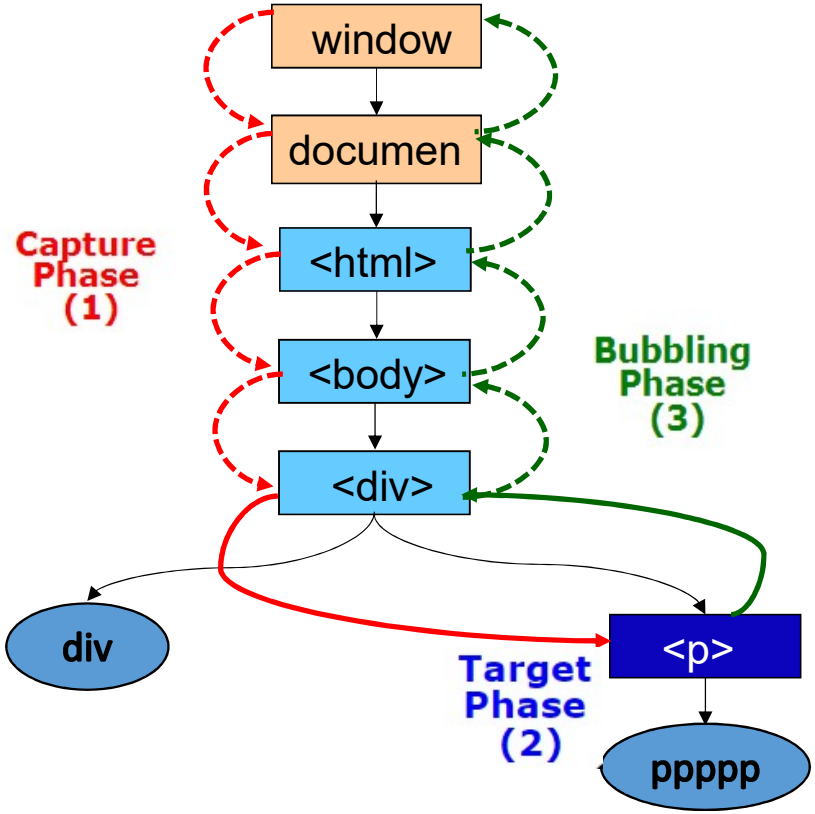
07eventBubble.html

08eventObjectDemo.html

DOM2級事件(W3C)

階層式標籤，事件流程分三階段，
事件捕獲(capture)階段(true)==>處理目標階段==>事件氣泡(bubbling)階段(false)

```
object.addEventListener(event,handler,usecapture)
```



07eventBubble.html

Document 物件(1/2)

document的屬性	說明
URL	回傳文件上的網址
forms	文件上所有<form>元素集合
images	文件上所有<img>元素集合
links	文件上含有href屬性的所有<a>及<area>元素集合
title	網頁上標頭名稱

09document.html

Document物件範例

```
function urlF() {
    var url = document.URL;
    document.getElementById("demo").innerHTML = url;

    var forms = document.forms.length;
    document.getElementById("demo").innerHTML = forms;

    var images = document.images.length;
    document.getElementById("demo").innerHTML = images;

    var links = document.links.length;
    document.getElementById("demo").innerHTML = links;

    document.title="iSpan";
}
```

09document.html

Document 物件(2/2)

document的常用方法	類型	說明
getElementById()	element	回傳指定id的元素
getElementsByName()	NodeList	回傳name屬性名稱的元素
getElementsByTagName()	HTMLCollection	回傳標籤名稱的元素
getElementsByClassName()	HTMLCollection	回傳css類別名稱的元素
querySelector()	element	回傳符合選擇器的第一個元素
querySelectorAll()	NodeList	回傳符合選擇器的所有元素

NodeList 與 HTMLCollection：這些物件是唯讀的類陣列物件(array-like objects)
1.NodeList物件：有length 屬性和item(index)方法
2.HTMLCollection物件：有length 屬性和item(index)方法，namedItem(name)

10NodeListHTMLCollection.html

Document 物件範例

```
let theInputs = document.getElementsByTagName("input");

for (let i = 0; i < theInputs.length; i++) {
    console.log(theInputs.item(i).value);
    console.log(theInputs[i].value);
}
console.log(theInputs.namedItem("txtName").value);  //abc
console.log(theInputs.namedItem("hobby").value);  //reading 傳回第一個
console.log(theInputs["hobby"].value);
```

querySelector()

- 使用CSS 選擇器取得DOM文件內的元素
- document.querySelector(*selector*) ：取得符合*selector* 的第一個元素(element)

```
<ul>
   <li id="li1">item1</li>
   <li class="myli">item2</li>
   <li>item3</li>
   <li class="myli">item4</li>
   <li>item5</li>
</ul>
<script>
   var li1 = document.querySelector("li");
   var li2 = document.querySelector(".myli");
   var li3 = document.querySelector("#li1");
   console.log(li1);
</script>
```

selector(選取器)
- 標籤(Type)選取器
- 類別(Class)選取器
- 物件(Id)選取器

10querySelector.html

querySelectorAll()

- document.querySelectorAll(*selector*) ：取得符合*selector* 的所有元素，取得的元素為NodeList 類型

```
<ul>
   <li id="li1"> item1</li>
   <li class="myli">item2</li>
   <li>item3</li>
   <li class="myli">item4</li>
   <li>item5</li>
</ul>
<script>
   var lis1 = document.querySelectorAll ("li");
   var lis2 = document.querySelectorAll(".myli");
   var lis3 = document.querySelectorAll ("#li1");
   console.log(lis1);
</script>
```

selector(選取器)
- 標籤(Type)選取器
- 類別(Class)選取器
- 物件(Id)選取器

10querySelector.html

圖片(image)物件

11imageObject.html

- image物件代表網頁中使用<img>標籤的圖片
- document.images：表示image物件的集合，是唯讀的類陣列(array-like objects)，具有length屬性，也可以像真正的陣列被索引(但只能讀無法寫)
- 屬性:
  - src 如,document.images[0].src= "images/Map"+this.id.substr(2)+".gif";
  - useMap：設定或取得客戶端影像地圖

https://www.image-map.net/

```
<img src="map00.gif" usemap="#image-map">
<map name="image-map">
    <area target="" alt="" title="" href="" coords="142,19,152,35" shape="rect">
    <area target="" alt="" title="" href="" coords="61,173,30" shape="circle">
</map>
```

image 範例

- https://www.image-map.net/

id	包含各縣市地圖	各縣市地圖
idTaipei	MapTaipei.gif	Taipei.gif
idTaoyuan	MapTaoyuan.gif	Taoyuan.gif
idTaichung	MapTaichung.gif	Taichung.gif
idKaohsiung	MapKaohsiung.gif	Kaohsiung.gif

```
"Map" + " Taipei " + ".gif"

"Map" +this.id.substr(2) + ".gif"

`Map${this.id.substr(2)}.gif`
```

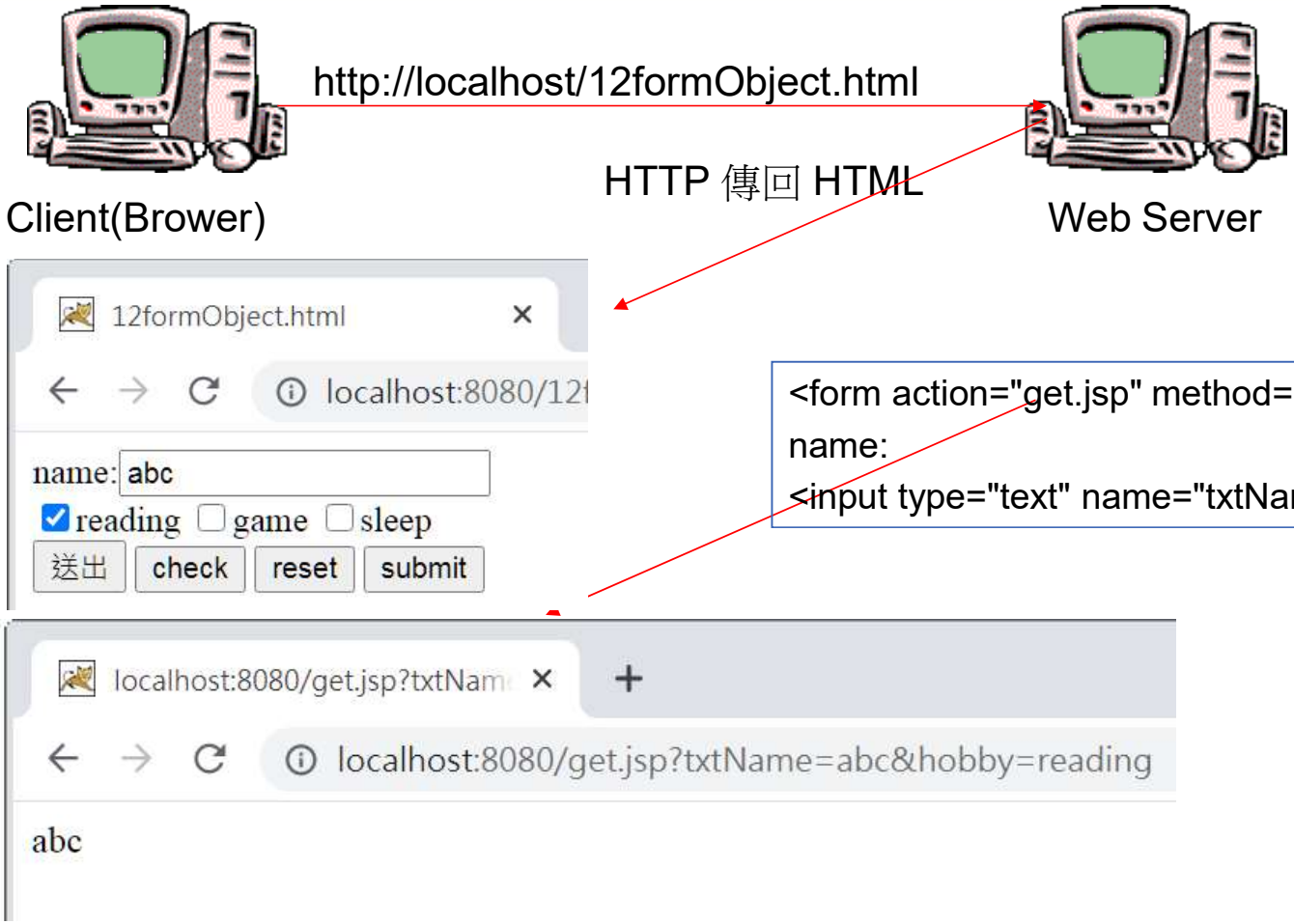
表單 (form) 物件

- 子集合(collections)
  - elements ： 傳回表單的所有元素的集合
- 屬性：
  - length ： 傳回表單擁有的欄位數
  - action ： 傳回或設定表單送出時，表單內資料要傳送的網址
  - method ： 傳回或設定表單送出時，表單內資料的格式
- 方法：
  - reset() ：將表單內資料，清除為預設值
  - submit() ：將表單內資料送出

```
console.log(document.forms.length);
console.log(document.forms[0].length);
console.log(document.forms[0].elements.length);
document.forms[0].submit();
```

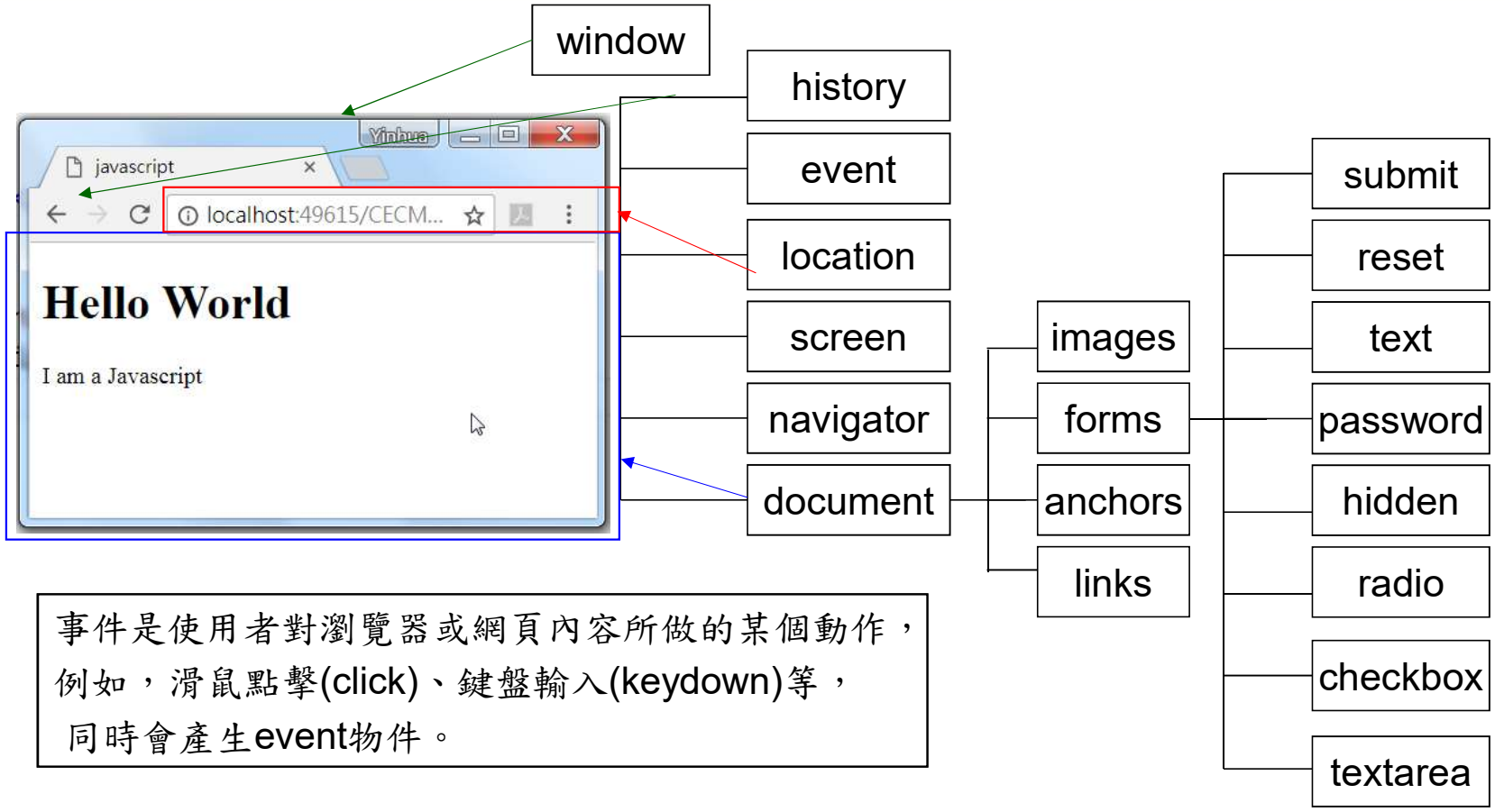
12formObject.html

表單 (form) 物件



12formObject.html

BOM 物件模型架構(Browser Object Model)



事件是使用者對瀏覽器或網頁內容所做的某個動作，例如，滑鼠點擊(click)、鍵盤輸入(keydown)等，同時會產生event物件。

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

單元九：DOM (Document Object Model)

- DOM 的標準化
- 什麼是DOM
- HTML網頁
- HTML網頁解析出的樹狀結構
- 瀏覽節點
- 尋找及存取節點資料
- 選取文件元素
- 如何新增及移除元素
- 關於屬性資料

DOM 的標準化

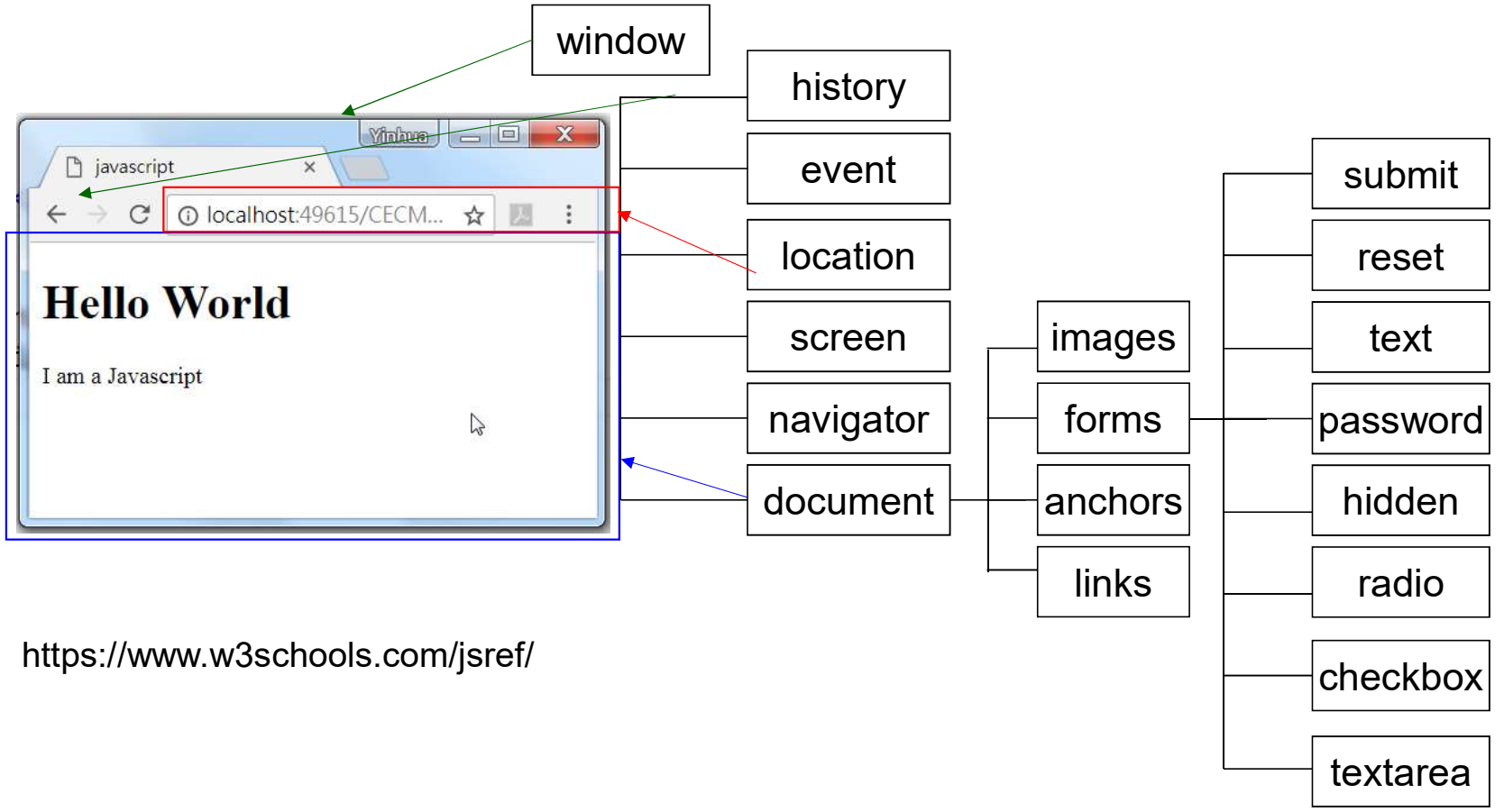
- 是W3C組織推薦的處理可延伸標示語言的標準程式介面，W3C對DOM進行標準化的動作，目前已經推行至第四代。

- DOM Level 0 (BOM)：
  - 不是W3C規格，目的是要整合Netscape和Internet Explorer 3.0版瀏覽器的物件模型而建立一個通用的物件模型。
- DOM Level 1：
  - 在1998年成為W3C推薦標準，
    - DOM Core ：提供HTML網頁和XML文件的瀏覽和處理元素內容的物件模型。
    - DOM HTML：HTML專屬的DOM API介面，目的是為了相容DOM Level 0 。

DOM 的標準化

- DOM Level 2：
  - 在2000年成為W3C推薦標準， 針對DOM Level 1新增樣式表物件模型及事件處理機制。
- DOM Level 3：
  - 在2004年成為W3C推薦標準，包含5種規格，即DOM3 Core、Event 、Load and Save 、Validation和XPath。
- DOM Level 4：
  - 在2015年成為W3C推薦標準，處理HTML及XML文件的程式介面。
- DOM Living Standard(動態標準) ：
  - https://dom.spec.whatwg.org/
- WHATWG
  - https://en.wikipedia.org/wiki/WHATWG

BOM 物件模型架構(Browser Object Model)



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

什麼是DOM

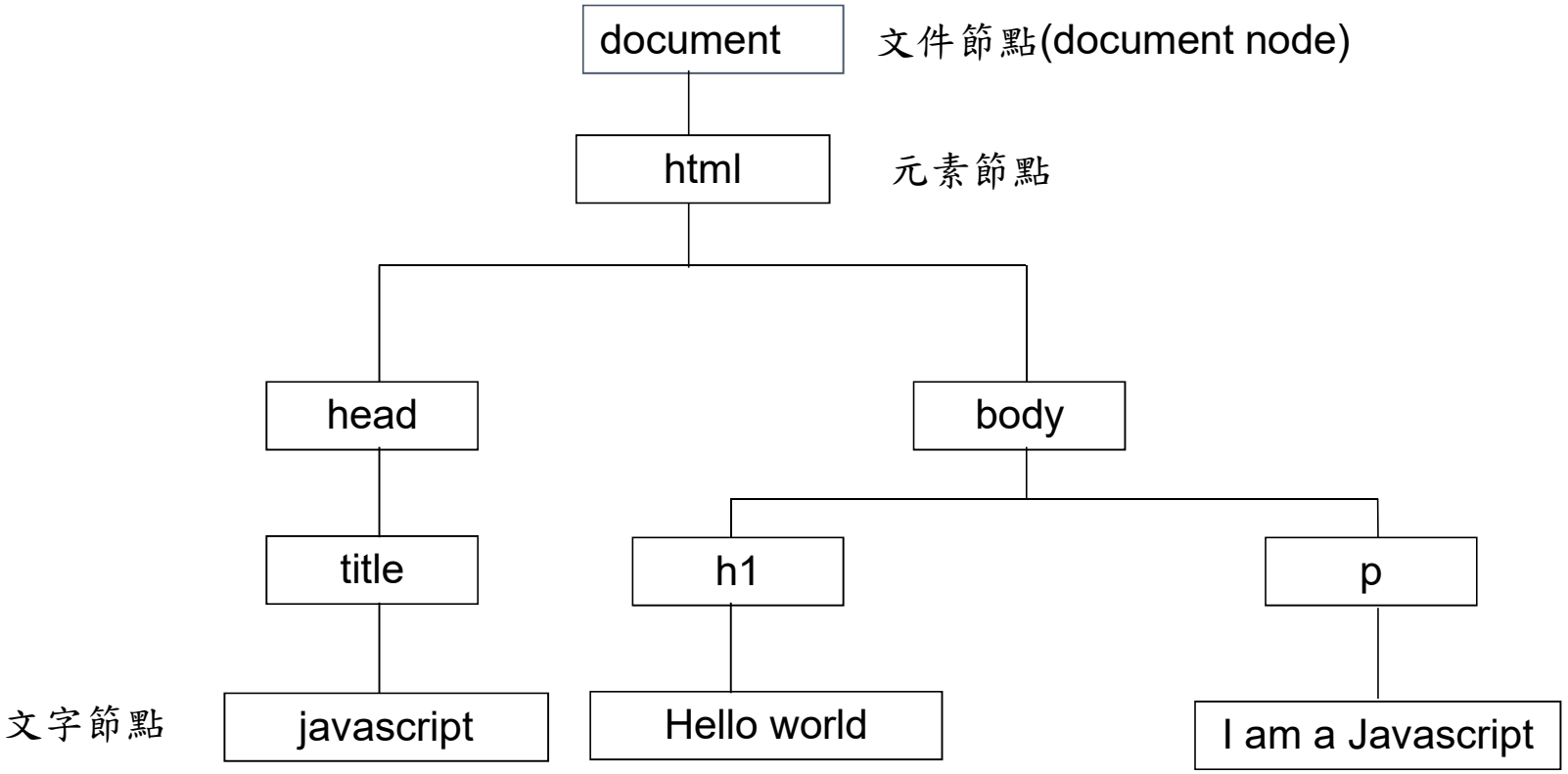
- 文件物件模型(Document Object Model)
  - 用來表示和操作HTML及XML文件內容的應用程式介面(API，Application Programming Interface)
  - 將文件解析為結構化表示法(樹)，並定義讓程式可以存取並改變文件架構、樣式和內容的方法。
  - 文件中的所有資料，皆可透過DOM來存取、修改、新增及刪除

- DOM Living Standard(動態標準)：
  - https://dom.spec.whatwg.org/

HTML網頁

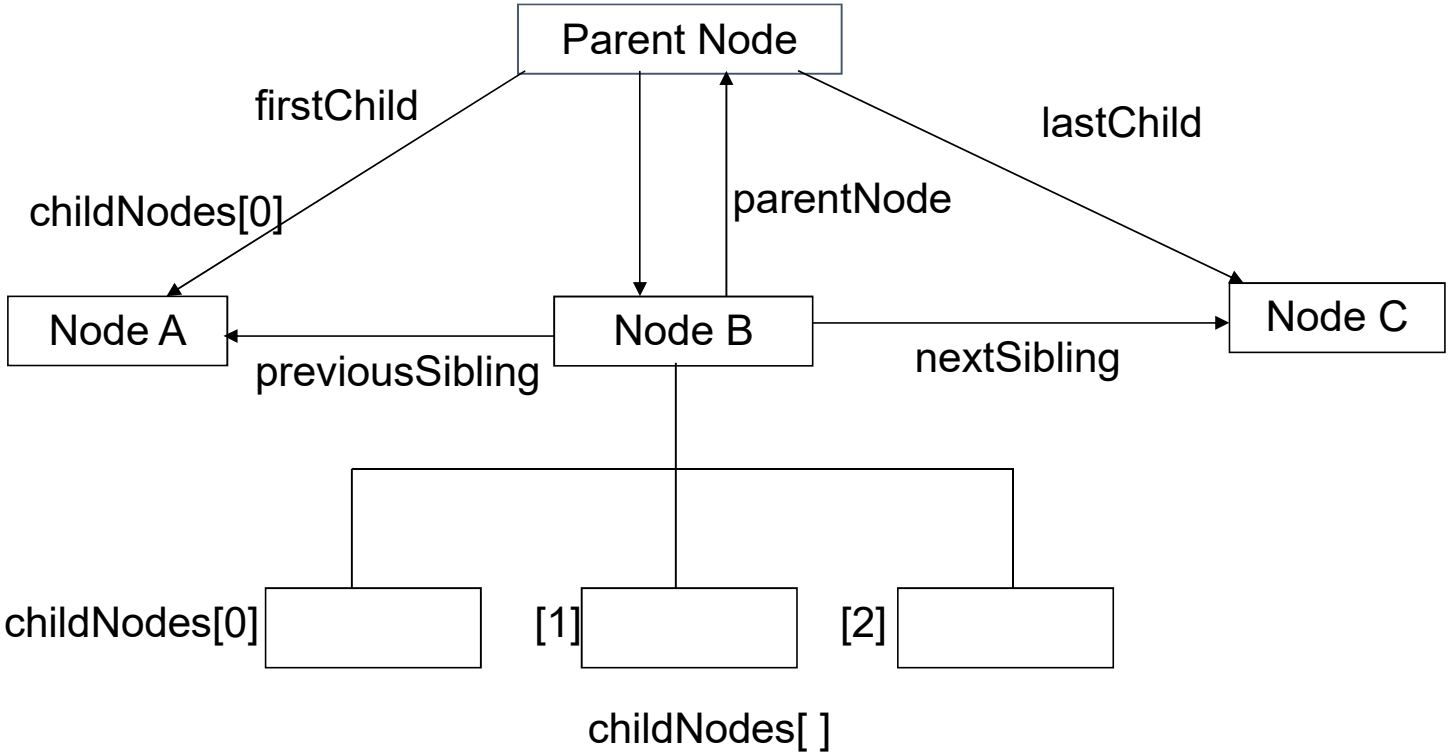
```
<!DOCTYPE>
<html>
  <head>
        <title>javascript</title>
  </head>
  <body>
        <h1>Hello world</h1>
        <p>I am a Javascript</p>
  </body>
</html>
```

HTML網頁解析出的樹狀結構



參考文件：http://www.w3schools.com/xml/dom_nodetype.asp

瀏覽節點(DOM node Tree)

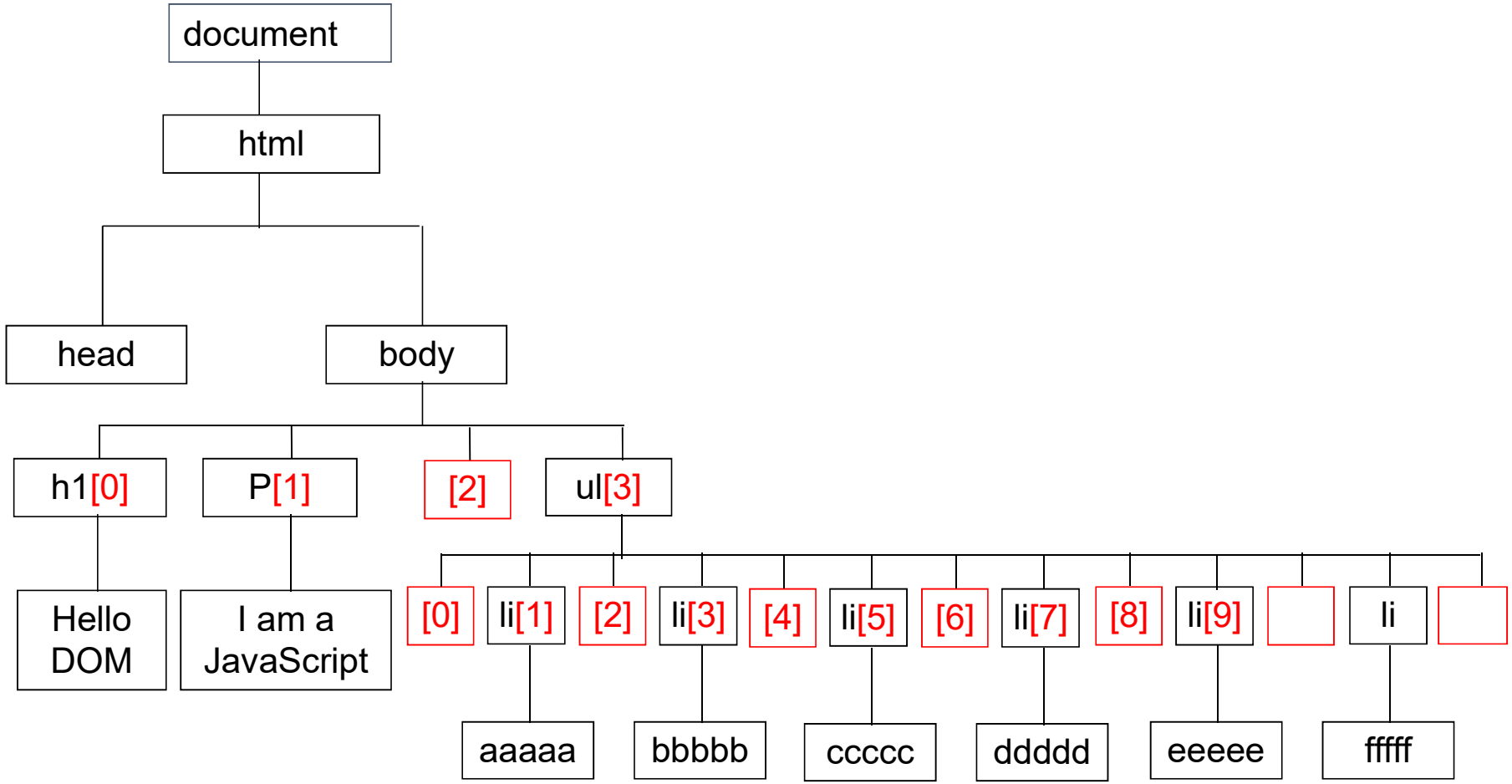


尋找及存取節點資料

- 透過document物件來使用節點物件的屬性及方法
- 節點彼此關係
  - documentElement
  - firstChild , lastChild , parentNode , childNodes
  - previousSibling , nextSibling
- 節點屬性
  - nodeType：節點類型
    - 9(Document)、1(Element)、3(Text).....
    - http://www.w3schools.com/xml/dom_nodetype.asp
  - nodeName :大寫標籤名稱或#document 或 #text
  - nodeValue : Text或Comment節點的文字內容或null

DOM 範例

01DOM1.html



選取文件元素

- getElementById()：選取指定id屬性的元素
- getElementsByName()：選取指定name屬性元素
- getElementsByTagName()：選取指定標籤名稱元素
- getElementsByClassName()：選取指定類別名稱元素
- 根據CSS選擇器
  - querySelector(selector)
  - querySelectorAll(selector)
  - https://www.w3.org/TR/css3-selectors/
  - https://www.w3.org/TR/2020/SPSD-selectors-api-20201103/

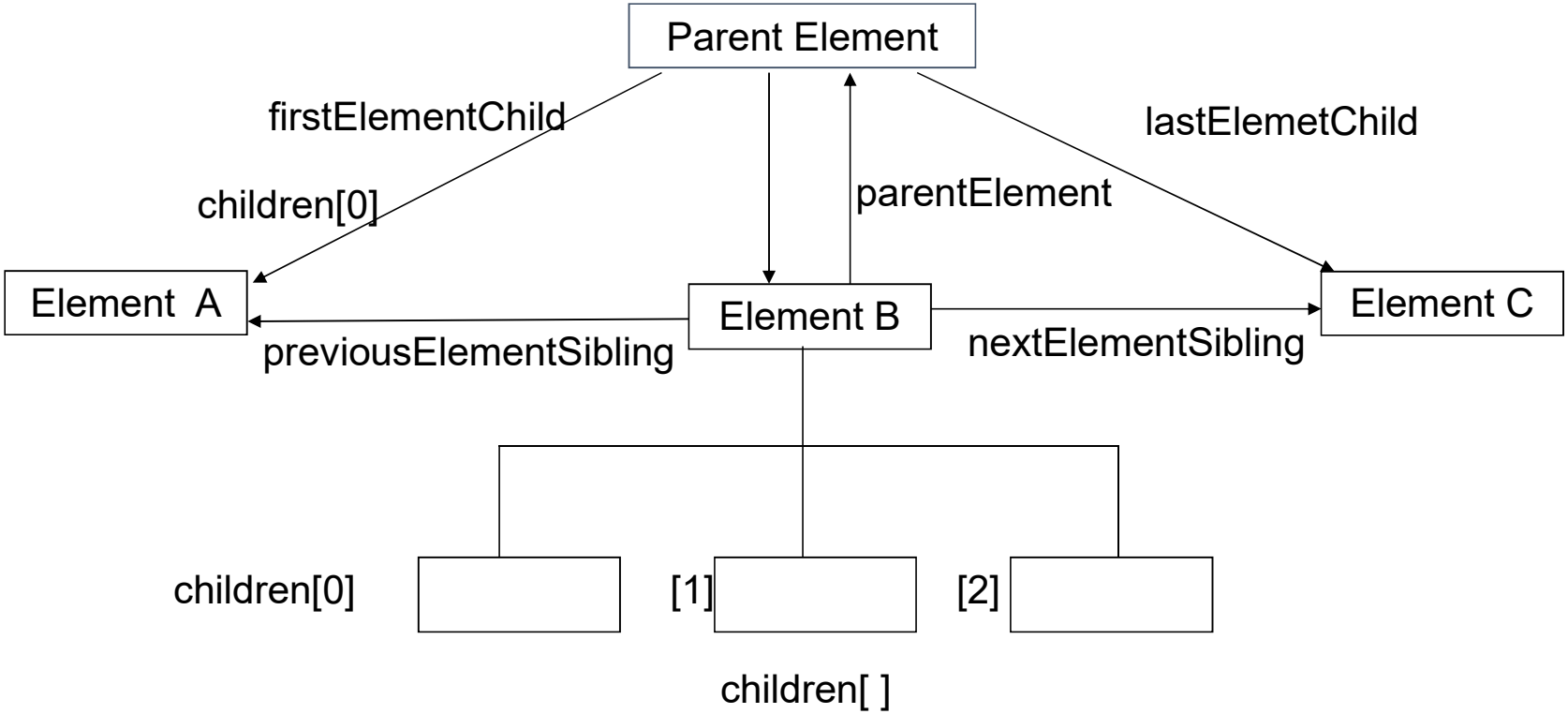
範例程式

```
alert (document.documentElement.childNodes(1).nodeName);



let ps = document.getElementsByTagName("p");

for (let i=0;i<ps.length;i++)
{
     alert (ps.item(i).firstChild.nodeValue);
}
```

- item(index)—回傳指定索引值的節點

01DOM1.html

瀏覽元素(DOM Element Tree)



尋找及存取元素資料

- 透過document物件來使用節點物件的屬性及方法
- 元素彼此關係
  - firstElementChild , lastElementChild
  - parentElement , children,childElementCount
  - previousElementSibling , nextElementSibling
- 元素內容存取
  - textContent(innerText)：存取內容純文字
  - innerHTML：存取內容及標籤

02DOM2.html

新增及移除元素

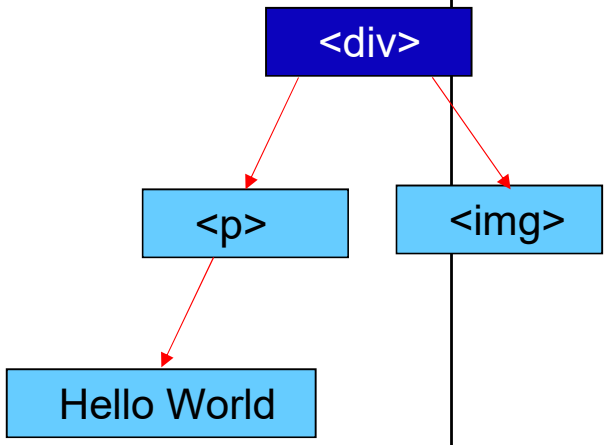
- 新增元素(成對標籤)
  - 步驟一 先建立元素名稱 createElement
  - 步驟二 再建立文字內容 createTextNode
  - 步驟三 使用 appendChild 將文字內容新增在元素之後
  - 步驟四 使用 appendChild 將新建立的元素，新增在已經存在的元素之後
- 移除元素
  - 步驟一 找到您要刪除的元素
  - 步驟二 使用 node.parentNode.removeChild(node) 的方式來刪除節點，或
  - 步驟二 使用 node.remove()方法刪除節點

關於屬性資料

- 關於屬性資料(成對、單一標籤)
  - 步驟一 找到相關的元素或建立元素名稱 createElement
  - 步驟二
    - 設定屬性
      - node.setAttribute ("屬性名稱", "屬性值")
    - 刪除屬性
      - node.removeAttribute ("屬性名稱")
    - 讀取屬性
      - node.getAttribute ("屬性名稱")

新增元素範例

```
let theDIV = document.getElementById("div1")

let eleP = document.createElement("p")
let txtP = document.createTextNode("Hello World")
eleP.appendChild(txtP)

theDIV.appendChild(eleP)

let eleImg = document.createElement("img")
eleImg.setAttribute("src","images/b.gif")

theDIV.appendChild(eleImg)
```



03DOMCreateDelete.html

DOM的使用

- 避免在迴圈中存取DOM
- 將DOM參考指派給區域變數，並使用此區域變數來操作
- 使用選擇器API
- DOM的更新次數要少
- 使用文件片斷 createDocumentFragment() 方法

```
let docFrag = document.createDocumentFragment();
document.querySelector("body").append(docFrag);
```

04DOMFrag.html

DocumentFragment(文件片斷)

- 與文件物件類似
  - 是一種沒有父層節點的「最小化文件物件」，DOM會用和標準文件一樣的方式來保存「片段的文件結構」。
- 與DOM 使用差別
  - 透過操作 DocumentFragment 與直接操作 DOM 最關鍵的區別在於 DocumentFragment 不是真實的 DOM 結構，所以說 DocumentFragment 的變動並不會影響目前的網頁文件，也不會導致回流（reflow）或引起任何影響效能的情況發生。
- 大量的 DOM 操作效能較好

Reflow 回流

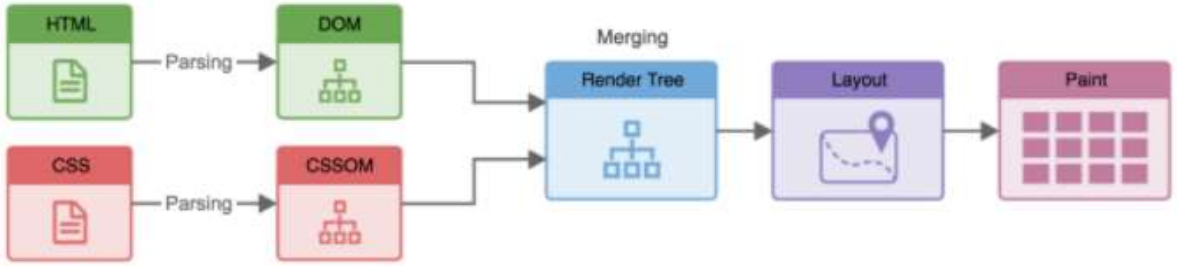


image by faressoft

- 從 HTML 檔解析出 DOM Tree
- 從 CSS 檔解析出 CSSOM(CSS Object Model)Tree
- 兩者合併後產生 Render Tree
- Reflow：計算出 Render Tree 上各個元素的物理屬性，如位置、大小、及是否看得見（visible）
- Repaint：將計算結果轉為實際的像素，畫到畫面上

附錄 一: Drag & Drop

- Drag & Drop API
- 元素拖曳
- 元素拖放

Drag & Drop API

- 屬性
  - Draggable
- 事件
  - Dragstart：開始拖放操作
  - Dragenter：被拖放的元素開始進入目標元素範圍內
  - Dragover：被拖放的元素正在目標元素範圍內移動
  - Dragleave：被拖放的元素離開目標元素範圍
  - Drop：其他元素被拖放到目標元素中
  - Dragend：拖放操作結束

Drag & Drop

event.dataTransfer 物件

event.dataTransfer.setData();

event.dataTransfer.getData();

1. draggable=true
2. dragstart() 事件

1. dragover()事件
2. drop() 事件

Drag & Drop API

- dataTransfer物件 -- 拖放操作時產生的事件物件的屬性，儲存拖放時攜帶的資料
  - *event.dataTransfer*.setData(*format*, *data*) ：

    加入拖放攜帶的資料
    - format : MIME(Multipurpose Internet Mail Extensions)類型或內容類型(content type) 或網際網路媒體類型(Internet media type) ，定義網際網路上傳輸內容的分類類型，其格式為:

      類型名稱/子類型名稱;參數

    - 如: text/plain， text/html， image/png ，image/webp，application/pdf
  - *data* = event.*dataTransfer*.getData(*format*) ：

    取得拖放攜帶的資料
  - ***event.dataTransfer*.files**：

    取得拖放的檔案集合(FileList)

元素拖曳

- 做法
  - 設定draggable屬性為true
  - 透過dragstart事件，開始拖曳的動作，取得要拖曳元素的資料
  - 將取得的資料放進dataTransfer物件中

```
function dragstartHandler(e){
    e.dataTransfer.setData("text/plain",e.target.textContent);
}
<div id="dragItem" draggable="true"
ondragstart="dragstartHandler(event)">Drag me!!</div>
```

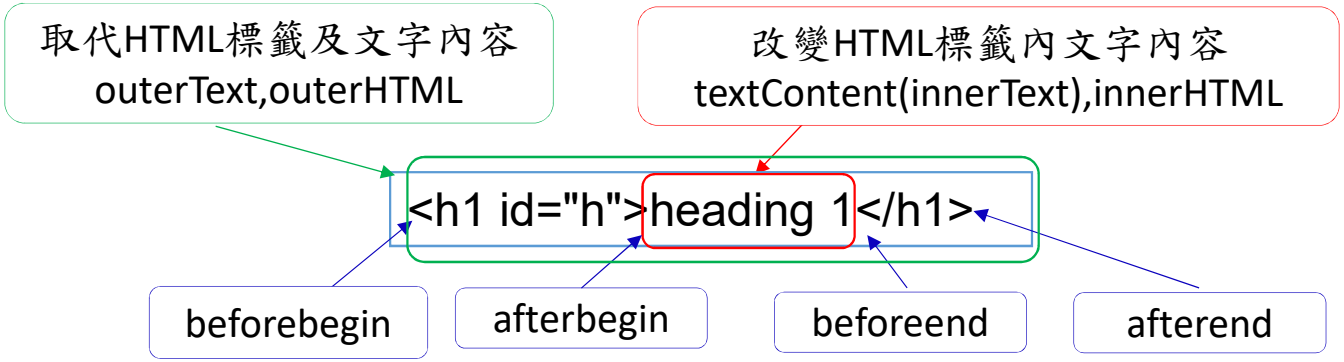
動態修改顯示內容

03dynamicContent.html

04checkPassword.html

05dynamicInsert.html

- 改變HTML標籤內的文字內容(成對標籤)
  - textContent ：讀取元素內容或將取代的內容視為純文字(plain)
  - innerHTML：讀取元素內容或將取代的內容視為HTML標籤解譯



- 自訂文字加入的位置
  - insertAdjacentHTML(where, text)
  - insertAdjacentText (where, text)
    - Where :
    - afterbegin , beforebegin(開始標籤之前) , beforeend , afterend (結束標籤之後)

元素拖放

- 元素的拖放，要處理二個事件
  - dragover事件，被拖放的元素拖放到目標元素邊框上方時觸發
  - drop事件，將資料拖曳進入目標元素後放開滑鼠時觸發，處理從dataTransfer物件取出的資料

```
<div id="dropZone" ondragover="dragoverHandler(event) "
                        ondrop="dropHandler(event)">
      Drop Zone
</div>
```

拖放範例

```
function dragoverHandler(e){
    e.preventDefault();
}
function dropHandler(e){
    e.preventDefault();
    e.stopPropagation();
    let sourceid = e.dataTransfer.getData('text/plain');
    let source = document.getElementById(sourceid);
    e.currentTarget.appendChild(source.parentNode.
                                        removeChild(source));
}
```

01dragdrop.html

事件處理程序

- HTML屬性事件處理程序

```
<input id="myBtn"  type="button" onclick="check() " value="Try it"/>
```

- JavaScript屬性事件處理程序
  - *object.event=function;*

```
<script>
      document.getElementById("myBtn").onclick=check;
      function check(){   }
</script>
```

- W3C DOM處理程序
  - *object*.addEventListener(*event*, *function,usecapture*

1.可省略
2.預設為false

```
<script>
    document.getElementById("myBtn"). addEventListener("click", check,false);
</script>
```

附錄二：Canvas 的使用

- <canvas>元素為一塊繪圖的區域，透過getContext()方法取得CanvasRenderingContext2D物件
- 繪圖的屬性及方法
  - 繪製線條
  - 繪製矩形
  - 繪製弧線
  - 處理圖像
  - 繪製文字

繪製線條方法與屬性

- beginPath()：開始一條新路徑
- moveTo(*x*,*y*)：設定一條新的子路徑的開始位置
- lineTo(*x*, *y*)：在目前位置新增一條直線
- strokeStyle：設定路徑的顏色、模式和漸變
- fillStyle：填滿路徑的顏色、模式和漸變
- lineWidth：設定線條寬度
- lineCap：設定線條端點
- lineJoin：設定線條連接點
- stroke()：沿著當前目前路徑繪製一條直線
- fill() :填滿路徑內部
- closePath():如果目前路徑開啟，則連接到路徑開始位置

繪製線條

```
context.beginPath();                    //開始繪製路徑
context.moveTo(100,50);                 //移到某一點上(起始點)
context.lineTo(50,100);                 //設定線條的位置(終點)
context.lineTo(150,100);
context.lineTo(100,50);
context.strokeStyle = '#ff0000';        //線條樣式設定(顏色)
context.lineWidth   = 10;               //線條寬度
context.lineCap = "round";              //端點 round,butt(default),square
context.lineJoin="round" ;              //連接點
round,bevel,miter(default)
context.fillStyle   = '#0000ff';        //填滿顏色
context.fill();                         //填滿
context.stroke();                       //開始畫線
context.closePath();                    //結束繪製路徑
```

01line.html

繪製矩形

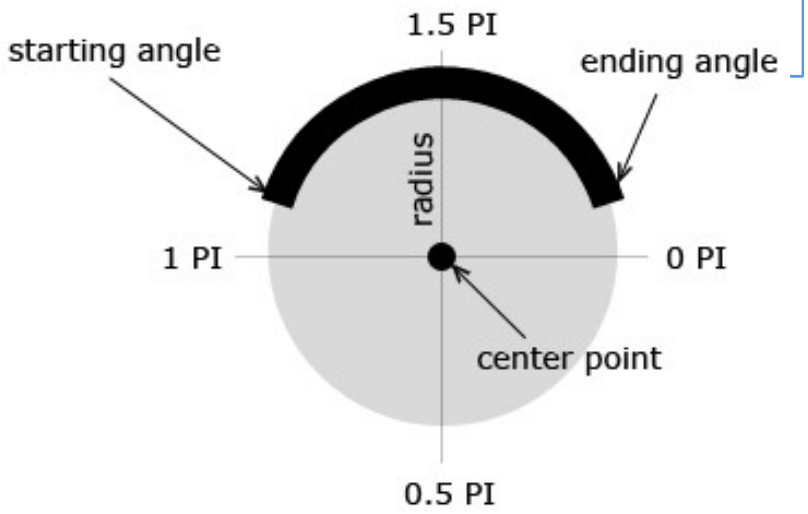
- 畫矩形的兩個方法
  - fillRect(x,y,width.height)
  - strokeRect(x,y.width,height)
- 清除矩形
  - clearRect(x,y.width,height)

```
context.fillStyle = "#FF0000";
context.strokeStyle = '#0000ff';
context.fillRect(0,0,150,50);
context.strokeRect(0,60,150,50);
context.clearRect(30,25,90,60);
```

02rect.html

繪製弧線

```
let centerX = 150;
let centerY = 100;
let radius = 75;
let startingAngle = 1 * Math.PI;
let endingAngle = 0 * Math.PI;
let counterclockwise = false;
context.arc(centerX, centerY, radius, startingAngle,endingAngle, counterclockwise);
context.lineWidth = 15;
context.strokeStyle = "black"; // line color
context.stroke();
```

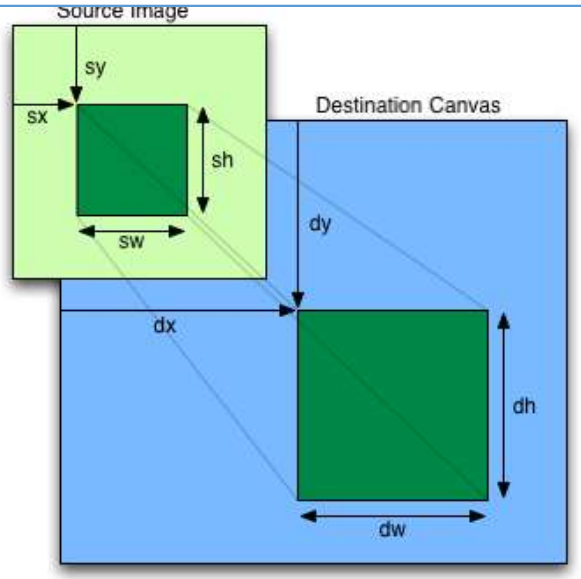


03arc.html

處理圖像

- Scale and crop images

```
drawImage(image, dx, dy)
drawImage(image, x, y, width, height)
drawImage(image, sx, sy, sWidth, sHeight, dx, dy, dWidth,
dHeight)
```



處理圖像範例

```
let imageObj = new Image();
    imageObj.onload = function(){
        let sourceX = 150;
        let sourceY = 150;
        let sourceWidth = 150;
        let sourceHeight = 150;
        let destWidth = sourceWidth;
        let destHeight = sourceHeight;
        let destX = 20;
        let destY = 20;

 //context.drawImage(imageObj, destX, destY);
 //context.drawImage(imageObj, destX, destY, destWidth, destHeight);
context.drawImage(imageObj, sourceX, sourceY, sourceWidth, sourceHeight, destX,
destY, destWidth, destHeight);

    };
    imageObj.src = "cars2_logo.jpg";
```

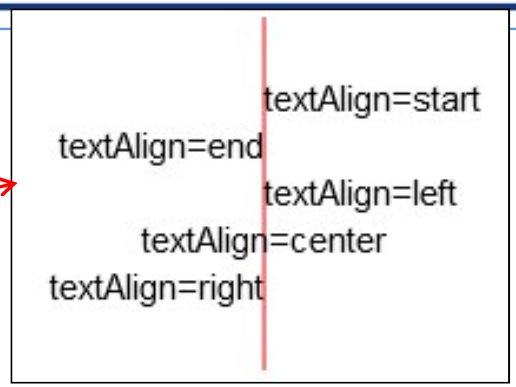
04image.html

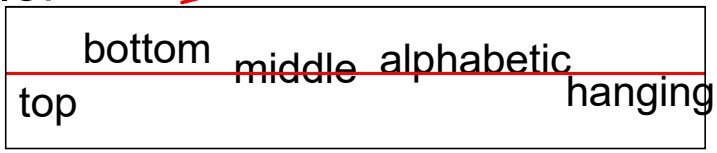
繪製文字方法與屬性

- font:設定文字的 *font-style*、*font-weight*、*font-size*、*font-family*
- textAlign : 設定文字內容的水平對齊(start 、left、center 、end 、right)
- textBaseline :設定文字內容的垂直對齊(top、hanging、middle、bottom、alphabetic)
- fillText(*text,x,y*) : 依據座標位置填滿文字內部
- strokeText(*text,x,y*): 依據座標位置繪製文字外框

繪製文字

```
let x = canvas.width / 2;
let y = canvas.height / 2;
context.font = "30pt Calibri";
context.textAlign = "center";
context.textBaseline = "middle";
context.fillStyle = "#0000ff"; // text color
context.fillText("Hello World!", x, y);

context.strokeStyle = "blue"; // stroke color
context.strokeText("Hello World!", x, y);
```





05font.html

附錄

- 正向前置(positive lookahead)
- this物件
- this物件範例
- 作為物件的方法

正向前置(positive lookahead)

- (?=p)-比對後面字元中相符於樣式p
- (?=.*[0-9])-至少有一個數字

/^(?=.*[0-9]).{2}$/

1. 判斷式，佔用的寬度為 0 。
2. 整個字串比對會以 .{2} 為主，但比對之前會先比對 (?=.*[0-9]) ，成功才會進行 .{2} 的比對。

.*[0-9]

1. *, 比對前一個字元0次以上
2. 0:[0-9]
   1:任[0-9]
   2:任任[0-9]

任 數
數 任
數 數
任 任

this物件

- 對於函數宣告及匿名函數，this 的用法

- 作為函數呼叫
  - 函數(); //函數內的this指向全域物件(window)
- 作為物件的方法
  - 物件.方法(); //函數內的this指向該物件
- 作為DOM事件處理
  - 物件. addEventListener(); // this指向觸發事件的元素

this物件範例

- 作為函數呼叫

```
var scope="global";
function f(){
   var scope="local";
   console.log(scope);
   console.log(this.scope);
}
f();
```

- 作為物件的方法

```
person.fullName=function(){return this.lastName+" "+this.firstName;}
```

- 作為DOM事件處理

```
let theH=document.getElementById("h");
function f(){
    console.log(this); //theH
}
theH.addEventListener("click",f,false);
```

作為物件的方法

- 傳統函數：依呼叫的方法而定(函數宣告,匿名函數)
- 箭頭函數：其宣告時所在的物件
- 箭頭函數當中的 this 是宣告時的物件，而不是呼叫時的物件