

Browser Object Model & Document Object Model

Browser Object Model (BOM)

Global Object

- In truth, there is no such thing as a **global variables** or **global functions**. All of them are defined as **properties** of the **Global object**.
- Global object is a virtual built-in object. It actually doesn't exist.

Global Object Property 內建

- undefined, NaN, Infinity
- Object
- Array
- Function
- Boolean
- String
- Number
- Date
- RegExp
- Error
- EvalError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError

Global Object v.s Window Object

- The EMCA-Script doesn't indicate a way to access the Global object directly.
- Browsers implement it as **part of the window object.**

- e.g.

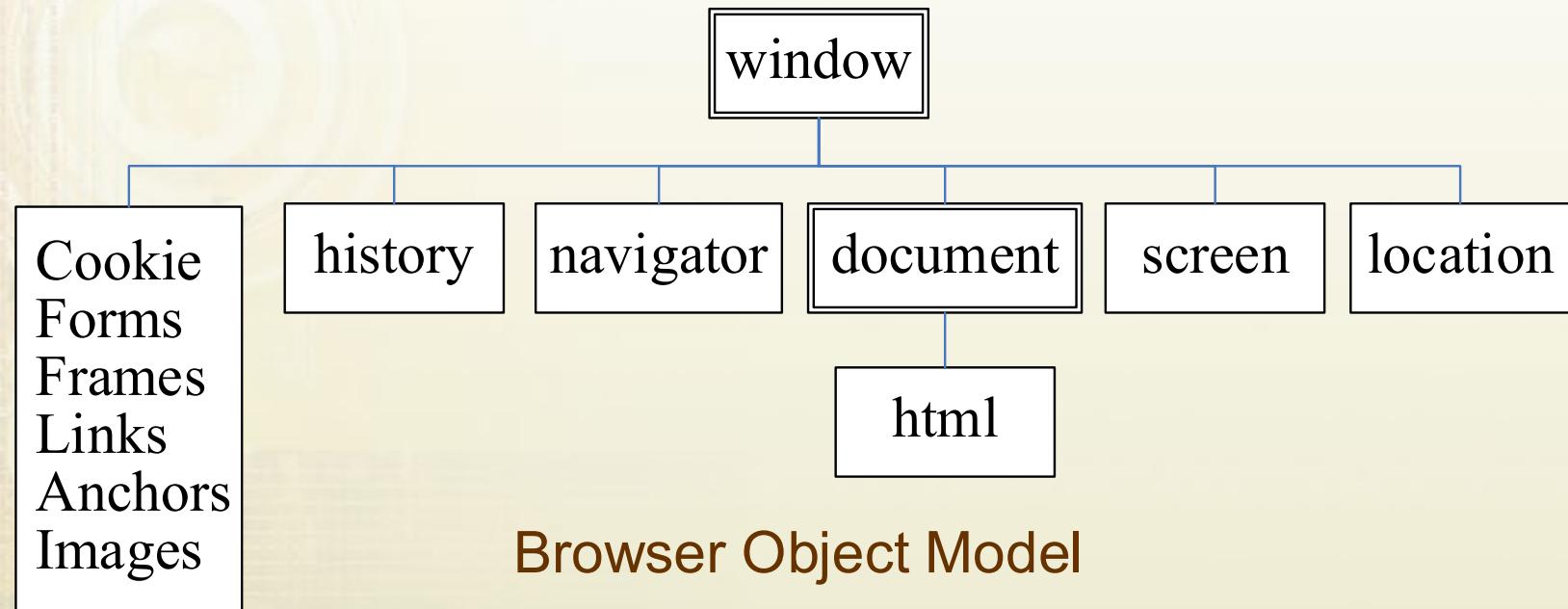
```
var color = "red"; // declare a variable
// declare a function
function sayColor() { alert(window.color); }
```

```
window.sayColor(); // "red"
```

- ps: can not *"delete window.color;"*

Browser Object Model

重力態系因頁面變：Browser Object Model



window Object

List window's Property

- e.g. [BOM/windowProperty.html](#)

```
for (var i in window)
  document.writeln("window."+i+"="+window[i]+"<br>
");
  P開、P關 - 來回用
```

- e.g. `window.open()`, `window.close()`, `window.moveTo()`, `window.resizeTo()`
- e.g. window size
 - `window.innerHeight` - the inner height of the browser window (in pixels)
 - `window.innerWidth` - the inner width of the browser window (in pixels)
- **dependent on browser!!**

window.open() / close()

- `winId = window.open(url, winName, optionsString, overwriteHistory);` 利用回傳值把頁面刪掉
 - `url` : the URL of the page
 - `winName` : the name of the window, or target attribute (`_self`, `_blank`, `_parent`, `_top`)
 - `optionsString`: e.g. `width, height` of window, whether or not to display scroll bar, status bar ...etc.
 - `overwriteHistory`: Specifies whether the URL creates a new entry or replaces the current entry in the history list.
- close windows: `winId.close()`

window.open() - Example

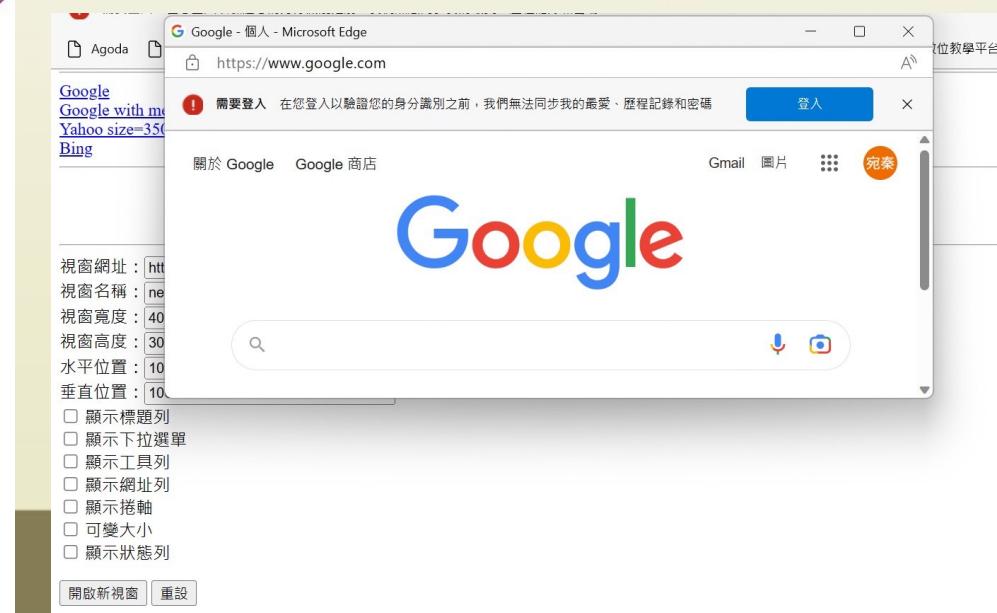
- e.g. BOM/windowOpen.html

```
window.open("http://www.google.com/", "myWin",
"width=800,height=600", false);
```

- note

- Opera open windows in a new tab instead a new window

```
30 <input type=checkbox id=status> 顯示狀態列<br>
31 <p>
32 <input type=button value="開啟新視窗" onClick="openWindow(this.form)">
33 <input type=reset>
34 <script>
35 function openWindow(form){
36     var winFmt="";
37     winFmt += "width="+form.width.value+ ",";
38     winFmt += "height="+form.height.value+ ",";
39     winFmt += "left="+form.left.value+ ",";
40     winFmt += "top="+form.top.value+ ",";
41     winFmt += "titlebar="+(form.titlebar.status?1:0)+ ",";
42     winFmt += "menubar="+(form.menubar.status?1:0)+ ",";
43     winFmt += "toolbar="+(form.toolbar.status?1:0)+ ",";
44     winFmt += "location="+(form.location.status?1:0)+ ",";
45     winFmt += "scrollbars="+(form.scrollbars.status?1:0)+ ",";
46     winFmt += "resizable="+(form.resizable.status?1:0)+ ",";
47     winFmt += "status="+(form.status.status?1:0);
48     status="控制字串 = " + winFmt;
49     window.open(form.url.value, form.title.value, winFmt);
50 }
51 </script>
```



window open() Options

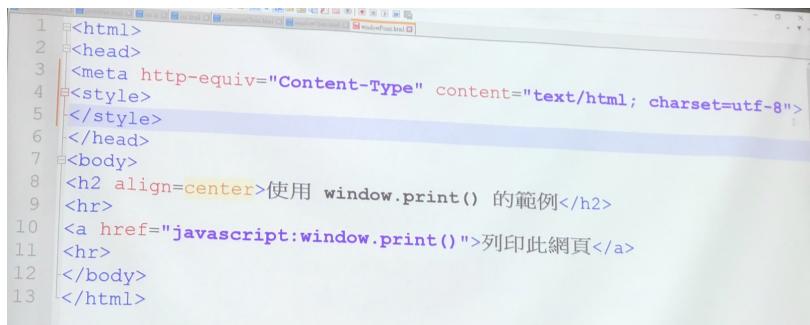
height=pixels	The height of the window. Min. value is 100
width=pixels	The width of the window. Min. value is 100
top=pixels	The top position of the window. Negative values not allowed
left=pixels	The left position of the window. Negative values not allowed
menubar=yes no 1 0	Whether or not to display the menu bar
scrollbars=yes no 1 0	Whether or not to display scroll bars. IE, Firefox & Opera only
status=yes no 1 0	Whether or not to add a status bar
titlebar=yes no 1 0	Whether or not to display the title bar. Ignored unless the calling application is an HTML Application or a trusted dialog box
toolbar=yes no 1 0	Whether or not to display the browser toolbar. IE and Firefox only
location=yes no 1 0	Whether or not to display the address field. Opera only
channelmode=yes no 1 0	Whether or not to display the window in theater mode. Default is no. IE only
directories=yes no 1 0	Obsolete. Whether or not to add directory buttons. Default is yes. IE only
fullscreen=yes no 1 0	Whether or not to display the browser in full-screen mode. Default is no. A window in full-screen mode must also be in theater mode. IE only
resizable=yes no 1 0	Whether or not the window is resizable. IE only

window.print()

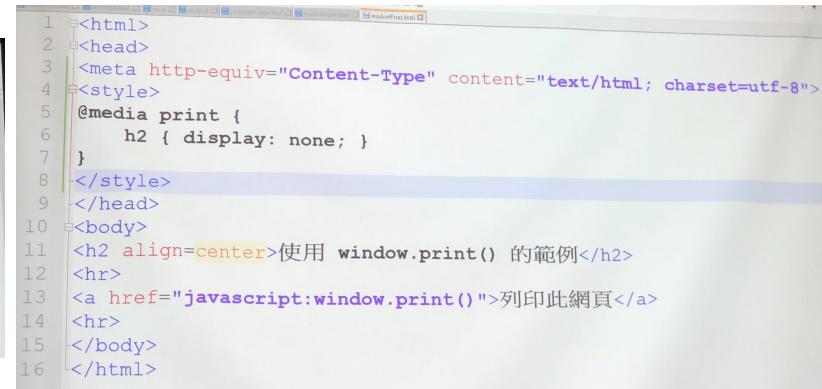
- 主題：利用 window.print() 列印網頁
 - 檔名：[BOM/windowPrint.html](#)
 - 程式碼重點
`列印此網頁`
- 說明
 - windows.print() 會呼叫瀏覽器提供的列印程式，將這個網頁列印下來。

```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
4 </head>
5 <body>
6 <h2 align=center>使用 window.print() 的範例</h2>
7 <hr>
8 <a href="javascript:window.print()">列印此網頁</a>
9 <hr>
10 </body>
11 </html>
```

友善列印用 media query



```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
4 <style>
5 </style>
6 </head>
7 <body>
8 <h2 align=center>使用 window.print() 的範例</h2>
9 <hr>
10 <a href="javascript:window.print()">列印此網頁</a>
11 <hr>
12 </body>
13 </html>
```



```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
4 <style>
5 @media print {
6   h2 { display: none; }
7 }
8 </style>
9 </head>
10 <body>
11 <h2 align=center>使用 window.print() 的範例</h2>
12 <hr>
13 <a href="javascript:window.print()">列印此網頁</a>
14 <hr>
15 </body>
16 </html>
```

history Object

- no standard that applies to the history object, but all major browsers support it.

History Object Properties	
Property	Description
<u>length</u>	Returns the number of URLs in the history list
History Object Methods	
Method	Description
<u>back()</u>	Loads the previous URL in the history list
<u>forward()</u>	Loads the next URL in the history list
<u>go(num)</u>	Loads a specific URL from the history list

history - Example

- 主題：展示 history 的方法
- e.g. [BOM/historyGo01.html](#)

document.writeln(history.length)

回前一頁

重新整理

跳下一頁

- 說明

- history.length 記錄了以前瀏覽網頁的頁數。

- history.go(n) 可以跳到之前瀏覽過的網頁。

- $n < 0$ 表示往前， $n > 0$ 表示往後， $n = 0$ 表示目前網頁

location Object

- **location** provides information about current document.
- **location** is a property of **both window** and **document** object. It means **window.location** and **document.location** refer to the same object.

location 屬性與方法

Location Object Properties	
hash	Returns the anchor portion of a URL
host	Returns the hostname and port of a URL
hostname	Returns the hostname of a URL
href	Returns the entire URL
pathname	Returns the path name of a URL
port	Returns the port number the server uses for a URL
protocol	Returns the protocol of a URL
search	query portion of URL, after and including the question mark.
Location Object Methods	
assign()	Loads a new document <i>History</i> 
reload()	Reloads the current document
replace()	Replaces the current document with a new one <i>History</i> 

- [URL format] protocol://hostname[:port]/[path][?search][#hash]

location Assignment

- 使用 assign(), replace() 指定新網址
- e.g. [BOM/locationAssign.html](#)

```
location.assign("http://www.google.com/");
```

```
location.replace("http://www.google.com/");
```

- assign() 會在 history 記錄新網址，replace() 則不會
- 也可以直接修改 location 的任一屬性值(除了hash)，就會重新讀取新網頁

Refresh 頁面

- `reload()`:
 - `reload()`: reload from browser cache if it hasn't change
 - `reload(true)`: reload from server
- `location.reload()` 與 `history.go(0)` 都可以用來重整頁面
- e.g. [BOM/locationRefresh.html](#)
 - 程式重點

```
function softRefresh() { history.go(0); }
function hardRefresh() { location.reload(true); }
```
 - (IE) go(0) 會保留已進行的輸入

1. 傳入參數會被以為要重新整理

```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
4 <script>
5 location = "https://google.com/";
6 </script>
7 </head>
8 <body>
9 <h2 align=center>使用 window.print() 的範例</h2>
10 <hr>
11 <a href="javascript:window.print()">列印此網頁</a>
12 <hr>
13 </body>
14 </html>
```

直按這至 google

navigator Object

- navigator object 內存有與瀏覽器相關的資訊
 - 包括瀏覽器類別、版本、CPU 類別、平台、是否支援 cookie 之類的資訊。
 - e.g. [BOM/navigatorProperty.html](#)
- 為了撰寫跨各家瀏覽器平台的 JavaScript 程式碼，我們可以利用 navigator 資訊來偵測使用者所用的瀏覽器種類，以決定選用何種 JavaScript 程式。
 - e.g. [BOM/navigatorInformation.html](#)
- 我們只要使用 navigator 物件的各種屬性，就可以知道使用者的 navigator 物件的重要屬性

navigator 常見方法和屬性

Property	Description
appCodeName	Returns the code name of the browser
appName	Returns the name of the browser
appVersion	Returns the version information of the browser
cookieEnabled	Determines whether cookies are enabled in the browser
geolocation	Returns a Geolocation object that can be used to locate the user's position
language	Returns the language of the browser
onLine	Determines whether the browser is online
platform	Returns for which platform the browser is compiled
product	Returns the engine name of the browser
userAgent	Returns the user-agent header sent by the browser to the server
Method	Description
javaEnabled()	Specifies whether or not the browser has Java enabled

note: userAgent

- *Mozilla/version (OS) RenderEngine/version (comment) browser/version*
- e.g. 使用 Chrome 瀏覽器/107版
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/107.0.0.0 Safari/537.36
 - Mozilla/5.0 表示與 Mozilla 相容
 - 系統 Windows NT 10
 - 網頁渲染/排版引擎 WebKit 537.36
- e.g. 使用 FireFox 瀏覽器/版本103
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:100.0) Gecko/20100101 Firefox/103.0

screen Object

Property	Description
<u>availHeight</u>	Returns the height of the screen (excluding the Windows Taskbar)
<u>availWidth</u>	Returns the width of the screen (excluding the Windows Taskbar)
<u>colorDepth</u>	Returns the bit depth of the color palette for displaying images
<u>height</u>	Returns the total height of the screen
<u>pixelDepth</u>	Returns the color resolution (in bits per pixel) of the screen
<u>width</u>	Returns the total width of the screen

- e.g.
 - [BOM/screenProperty.html](#),
 - [BOM/screenInformation.html](#)

Document Object Model (DOM)

Document Object Model DOM

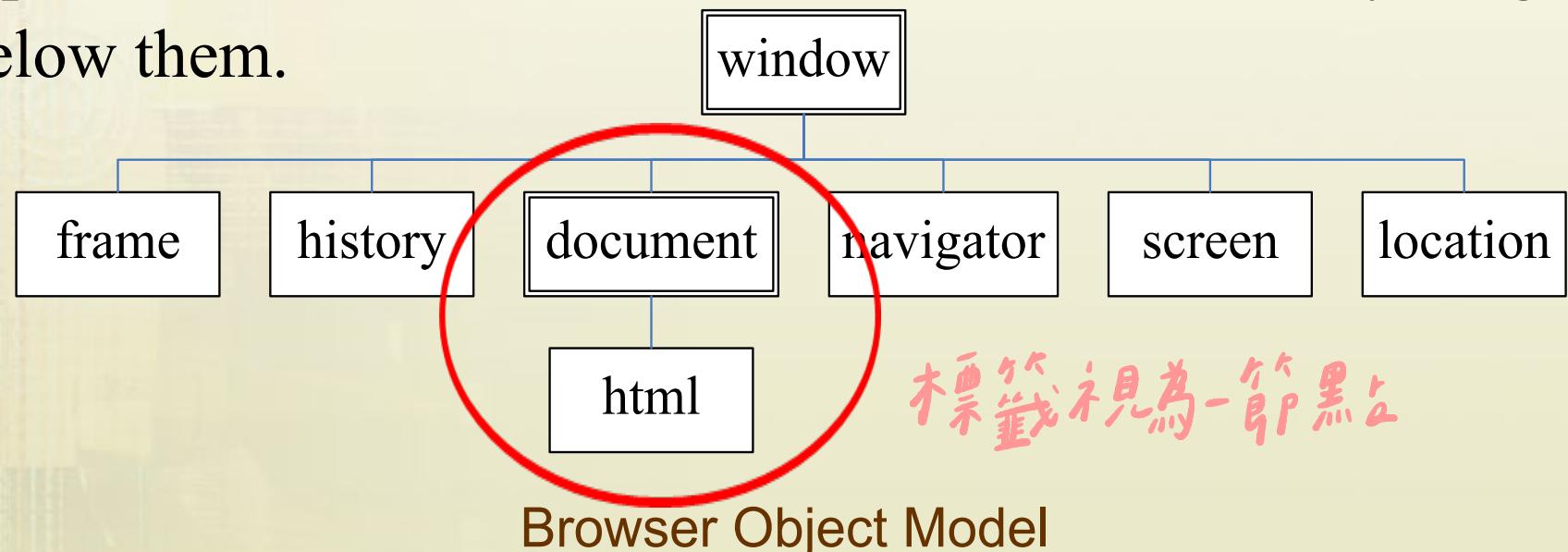
- The DOM is a W3C (World Wide Web Consortium) standard for accessing documents like HTML and XML.

"The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents." ---
<http://www.w3.org/DOM/>

- DOM是一個階層式的物件模型，包含了網頁的各種物件，經由調整這些物件的性質和方法，才能產生動態的網頁
- 當瀏覽器讀入一頁網頁時，就會根據此網頁的內容來建立相關的 Document Object

Document Tree

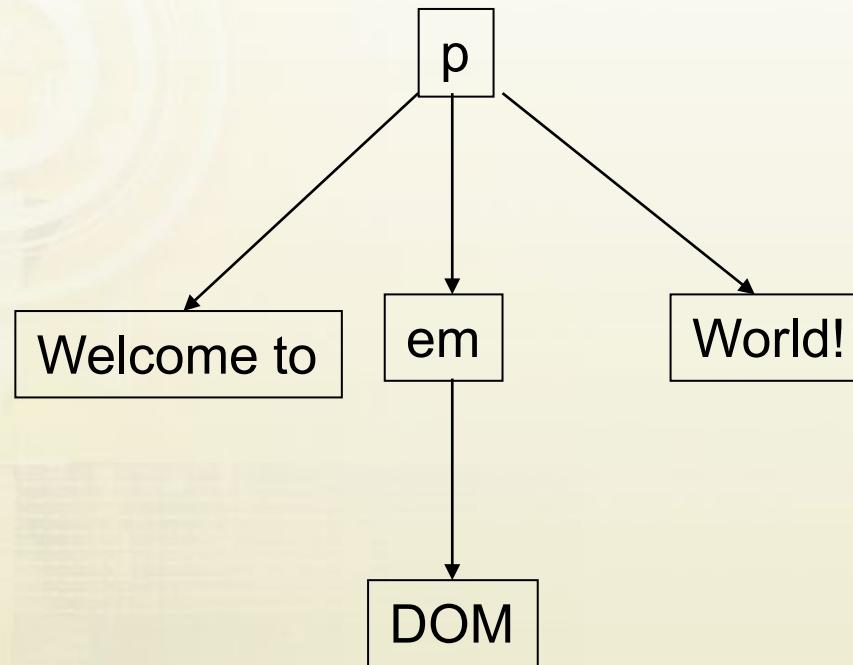
- The DOM presents documents as a hierarchy of Node (tree or forest).
- ***document*** object is the root of document tree
- Node represents a single object in the document tree.
- Some types of nodes may have child nodes of various types, and others are leaf nodes that cannot have anything below them.



Document Tree - Example

- e.g.

```
<p>Welcome to <em> DOM</em> World!</p>
```



DOM Node 1/3

- Everything in an HTML document is a node.
- DOM 主要的 6 種節點類型如下：

nodeType	Type Name	說明	nodeName	nodeValue
1	Element	tag name, e.g. body		null
2	Attr	搭配 Tag 的屬性節點	e.g. href	e.g. "url"
3	Text	內容文字	#text	e.g. "Hello, World!"
8	Comment	註解節點	#comment	content of the comment
9	Document	#document	#document	null
10	DocumentType	document type name		e.g. <!doctype...>

- The entire document is a **document** node
- Every HTML element is an **element** node
- The text in the HTML elements are **text** nodes
- Every HTML attribute is an **attribute** node

DOM Node 2/3

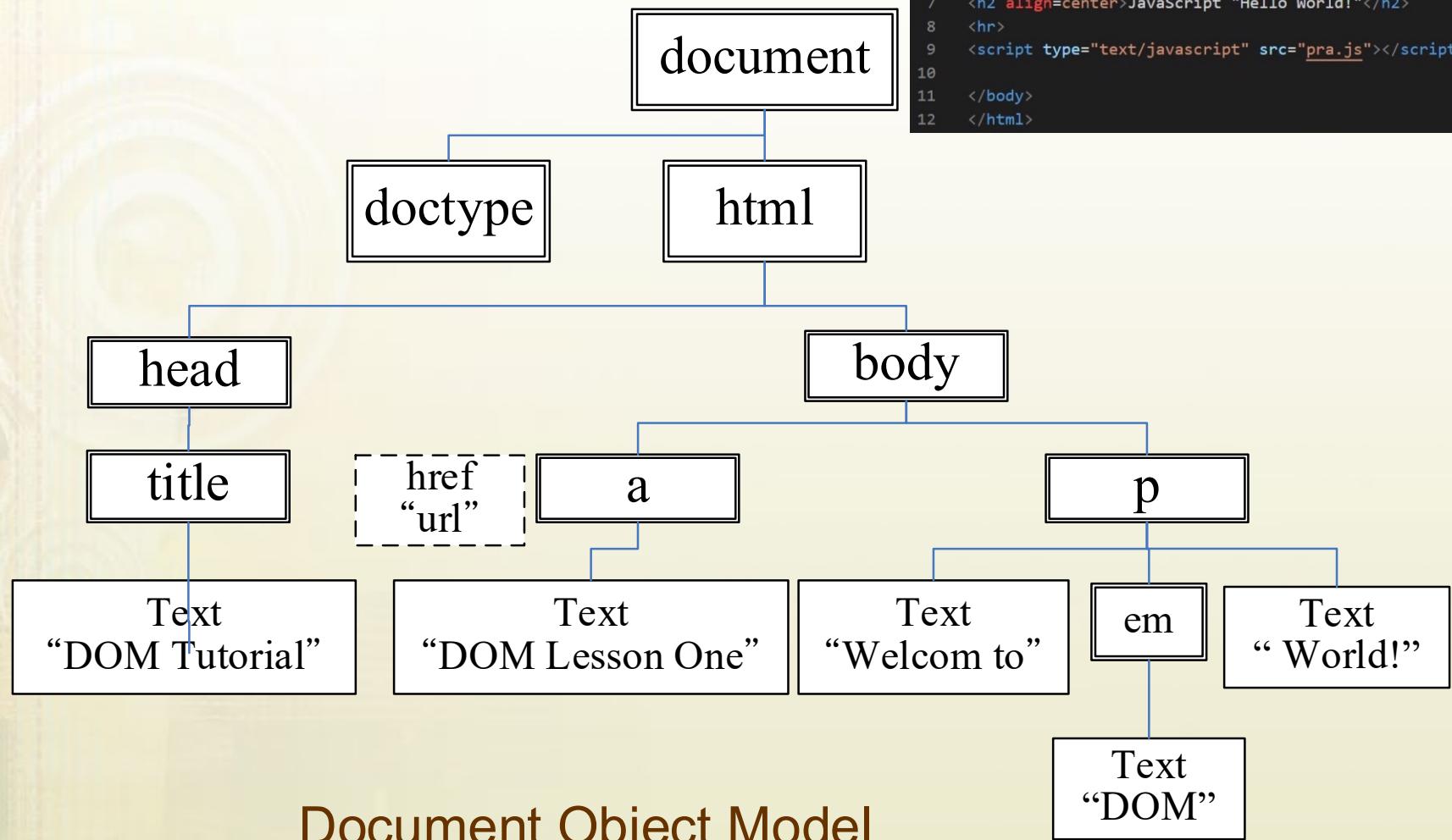
- e.g. [domFirst.html](#)

```
<html> <head> <title>DOM Tutorial</title> </head>
<body>
<a href="url">DOM Lesson one</a>
<p>Welcome to <em>DOM</em> world!</p>
</body>
</html>
```

- The root node in this HTML is `<html>`. All other nodes in the document are contained within `<html>`.
- The `<html>` node has two child nodes; `<head>` and `<body>`.
- The `<head>` node holds a `<title>` node. The `<body>` node holds a `<a>` and `<p>` node.

DOM Node 3/3

```
pra.html  X  JS pra.js  JS car.js  car.html
C: > Users > user > OneDrive > 桌面 > html > vscode > pra.html > html
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
4  </head>
5  |
6  <body>
7  <h2 align=center>JavaScript "Hello World!"</h2>
8  <hr>
9  <script type="text/javascript" src="pra.js"></script>
10
11 </body>
12 </html>
```



getAttribute node 要先去找至 9

DOM Node Tree - Example

■ e.g. DOM/nodeTree.html

DOM

Every document element are organized in a tree structure, called the DOM tree.

node type in the tree:

- element node
- attribute node
- text node

```
HTML (Type: element)
HEAD (Type: element)
META (Type: element)
TITLE (Type: element)
#text (Type: text) (Value: Show Document Tree)
LINK (Type: element)
SCRIPT (Type: element)
BODY (Type: element)
DIV (Type: element)
H2 (Type: element)
#text (Type: text) (Value: DOM)
P (Type: element)
#text (Type: text) (Value: Every document element structure, called the DOM tree.)
#text (Type: text) (Value: node type in the tree: )
UL (Type: element)
LI (Type: element)
#text (Type: text) (Value: element node)
LI (Type: element)
#text (Type: text) (Value: attribute node)
LI (Type: element)
#text (Type: text) (Value: text node)
```

Document Creation

- 我們可以利用 node creation 從無到有，建立一個完整的網頁
 - e.g. [DOM/nodeBorn.html](#)
- 只是這樣太麻煩了，所以可行的作法是先以建立一靜態網頁，再隨需求操作 document node，動態改變網頁的內容。

Find Elements in the DOM Tree

- If you want to manipulate elements, you have to find the elements first. The ways to find elements in the DOM tree:
 - by tag name: `getElementsByName()`
 - by id attribute: `getElementById()` *id 唯一較好找*
 - by name attribute: `getElementsByName()`
 - (HTML5)by class attribute: `getElementsByClassName()`
 - by CSS selector: `QuerySelector()`
 - 使用內建的 `document` 屬性來存取物件節點

Find Elements by Tag Name

- 每一個 HTML 的標籤在 Document Tree 下就是一個節點，可利用標籤名稱來存取這個節點
- e.g. [DOM/domAccess.html](#)

```
var pList=document.getElementsByTagName("li");
for ( i=0; i<pList.length; i++) {
    msg += pList[i].childNodes[0].nodeValue;
}
```

- 取得網頁所有 標籤的節點
- 回傳值 pList 為陣列
- childNodes[] 為 的子節點
- nodeValue 為節點值

Find Elements by id

- 每一個 HTML 的標籤可設定其 id 屬性 (`id="idValue"`)，可利用此一 id 屬性值來存取此節點
- `idValue` 值在同一文件中應唯一，只要 `idValue` 在整個網頁是唯一的，可以使用兩種方法來取得此物件
 - `document.getElementById(idValue)`：標準作法，符合 W3C 的標準。
 - `document.all(idValue)`：這是 IE4 自創的規格，目前並未被 W3C 的標準 DOM 所採納

Find Elements by name

- 對於每一個標籤，我們可以使用 name 屬性來存取節點，但 name 不唯一，不同標籤可以使用相同的 name
- `document.getElementsByName(nameValue)`
- 回傳值是一個陣列
- e.g.

```
var x=document.getElementsByName("x");
x[2].value = parseInt(x[0].value) + parseInt(x[1].value);
```

Find Elements by CSS selector

- The `querySelector()` method returns the **first** descendant element that matches a specified *CSS selector(s)*
- e.g. `document.querySelector("#content")`
- e.g. `var baseElement = document.querySelector("p");
document.getElementById("output").innerHTML =
baseElement.querySelector("div span").innerHTML`
- **Note:** The `querySelector()` method only returns the first element that matches the specified selectors. To return all the matches, use the [querySelectorAll\(\)](#) method instead.

使用內建的 `document` 屬性來存取

- 瀏覽器可能會自動建立一些屬性，方便直接存取網頁上的物件：
- e.g. 經由設定 `document.title` 來改變網頁的標題
(`document.title = "Access element by document property "`)
- e.g. 特殊陣列物件，如 `links`, `images`, `forms` 儲存網頁中所有 `link`, `image`, `form`
- e.g. [DOM/accessByProperty.html](#)
 - 列出文件中的 `links`, `images`, `forms` 物件
- 有些則使用標籤 `name` 屬性值做為 `document` 屬性值
e.g. 上例中，當 ``，存取第一張圖片檔名，可以用 `document.image1.src`
 - 此法 dependent on 瀏覽器，**不建議使用**

Change Element Node

- 改變物件(element node)的屬性值，就能動態改變網頁內容
- 存取物件的屬性值，有三種方法：
 - 用標籤的屬性名稱來存取物件的屬性(這是最常用到的方法)
語法：`object.propertyName`
 - 以屬性名稱做為索引字串來存取物件的屬性值
(此種方法好處是：可將屬性名稱以變數儲存)
語法：`object["propertyName"]`
 - 用數字索引來存取物件的屬性值(比較少用)
語法：`object[index]`

[pra.html](#)[JS nodeTree.js](#)[domAccess.html](#)[InnerText.html](#)[JS pra.js](#)[JS car.js](#)[car.html](#)C: > Users > user > OneDrive > 桌面 > html > .vscode > J5 > [domAccess.html](#) > html > body > script > accessByTagName

```
22  //accessByTagName();
23  //accessById();
24  //accessByName();
25  //accessByClass();
26  //accessByQuery();
27
28  function accessByTagName() {
29      let pList=document.getElementsByTagName("li"); //pList示一集合
30      /**集合沒有順序、pList的值唯一不重複(但是array可以重複)，[0]不是照順序排**/
31      console.log("getElementsByTagName() return an ",typeof pList); // Object
32
33      let msg="";
34      for (let i=0; i<pList.length; i++) {
35          msg += " " + pList[i].childNodes[0].nodeValue; // 集合的第i個元素，pList[i] == pList.item(i)
36      }
37      // for (let item of pList) { console.log(item); }
38      console.log("before changing pList[2]", msg);
39
40      pList[2].childNodes[0].nodeValue = s; //"Coffee";
41
42      msg="";
43      for (let i=0; i<pList.length; i++) {
44          msg += " " + pList[i].childNodes[0].nodeValue; // pList[i] == pList.item(i)
45      }
46      console.log("after changing pList[2]", msg);
47      /*
48      msg = "";
49      for (let i in pList) { // 0, 1, 2, lenght, item(), namedItem()
```

Example: List Document Property

- 列出網頁文件中的物件所具有的性質
- e.g. [DOM/documentProperty.html](#)

- 程式碼

```
for (var prop in document)
    document.writeln('document.'+prop+'='+document[prop]+'<br>');
```

- 說明
 - 區域變數 prop 則會依次等於物件的每一個屬性名稱，document[prop] 則是對應屬性的值。

改變大量節點值的快速方法

- 不論是經由設定節點 `nodeValue` 來改變節點值，或透過物件的內建屬性來改變節點值，在設定大量節點皆不方便，所以各家 browser (特別是 IE) 除了規格書上定義的方法與屬性外，新增一些功能較強的屬性來設定節點值
- e.g. IE 4 首先在 Element Node 加上下列屬性
 - `innerText`, `innerHTML`, `outerText`, `outerHTML`
 - (Supported by IE, Safari, Opera, Chrome, FireFox)
- DOM Level 3 定義了 `textContent`，相當於 `innerText`

innerText & innerHTML

- innerText
 - 傳回其下所有Text Node的值 (in Depth-First Order)
- innerHTML
 - 傳回其下所有節點的值，包括Element Node, Text Node, Comment Node
- consider following example
 - `some.innerText = "Hello, "World"!";`
 - `some.innerHTML = "Hello, "World"!";`
- e.g. [DOM/innerHTMLGet.html](#),
[DOM/innerHTMLSet.html](#),
[DOM/innerText.html](#)

[pra.html](#)[JS nodeTree.js](#)[domAccess.html](#)[InnerText.html](#)[●](#)[JS prajs](#)[JS car.js](#)C: > Users > user > OneDrive > 桌面 > html > .vscode > J5 > [InnerText.html](#) > [html](#) > [body](#) > [script](#) > [setInnerText](#)

```
15     <input type="button" value="Get InnerText" onclick="getInnerText()" />
16     <input type="button" value="Set InnerText" onclick="setInnerText()" />
17
18     <script type="text/javascript">
19         function getInnerText(){
20             var div = document.getElementById("content");
21             console.log(div.innerText);
22             console.log(div.innerHTML);
23         }
24
25         function setInnerText(){
26             var div = document.getElementById("content");
27             //div.innerText = "Hello World!";
28             //div.innerText = "Hello <b>\\"World\\\"!</b>";
29             div.innerHTML = "Hello <b>\\"World\\\"!</b>"; //<b>被解譯為element node
30             //div.outerHTML = "Hello <b>\\"World\\\"!</b>"; //希望被設成文字就用inner text
31         }
32
33         function getInnerText2(){
34             var div = document.getElementById("content");
35             console.log( (typeof div.textContent == "string") ?
36                         div.textContent : div.innerText);
37         }

```

outerText & outerHTML

- outerText
 - like innerText, except it includes the node itself
- outerHTML
 - like innerHTML, except it includes the node itself
- e.g. [DOM/outerHTMLGet.html](#),
[DOM/outerHTMLSet.html](#),
[DOM/outerText.html](#)
- ps: 這四個屬性很耗資源，應該盡量避免放在迴圈內使用

Access CSS Style

by DOM Operation

First Example - 隱藏圖片

- e.g. [DOM/cssFirst.html](#)

```
// 取得圖片
```

```
var my_imgs =
```

```
document.getElementsByTagName("img");
```

```
var my_img = my_imgs[0];
```

```
.....
```

```
// 隱藏圖片
```

```
my_img.style.visibility = "hidden";
```

空閒會留著

```
// or
```

```
my_img.style.display = "none";
```

空閒會消失

JavaScript 中的 CSS 屬性名稱

- CSS 的屬性名稱常會包含有連字號 -
■ e.g. font-family : *getComputedStyle / setComputedStyle 不管止比*
<h1 id="header" style="font-family: sans-serif;">這是標題文字</h1>
- 連字號 - 在 JavaScript 並非合法變數用字，會解讀成減號，因此在 CSS2Properties 物件中將連字號移除，然後將連字號後面的第一個字母改成大寫。
- e.g. font-family 變成 fontFamily : *一減非合清識別拿*
var h1 = document.getElementById('header');
h1.style.fontFamily = "serif";

拿 - 大寫

讀取數值

- 讀取元素中設定的 CSS 樣式值

- e.g.

<p id="para" style="padding-left: 20px; padding-right: 45px; padding-top: 35px; padding-bottom: 60px;"> 這是段落文字 </p>

相對的style屬性

```
var myP = document.getElementById('para');
myP.style.paddingLeft;
myP.style.paddingRight;
```

設定 CSS 屬性值

- 在使用 CSS2Properties 物件的屬性時，所有的屬性數值都必須寫成字串。
- e.g.

```
<p id="para">這是段落文字</p>
```

.....

```
var myP = document.getElementById('para');  
myP.style.fontFamily = "sans-serif";  
myP.style.fontWeight = "bold";  
myP.style.color = "#FF0000";  
myP.style.backgroundColor = "yellow";
```

設定 CSS 多個屬性值

- 有些 CSS 屬性可以一次設定許多個屬性值
- e.g.

< p id="para" style="padding: 80px 60px 40px 50px;"> 這是段落文字</p>
可寫成

```
var myP = document.getElementById('para');  
myP.style.padding = "80px 60px 40px 50px";
```

- myP.style.paddingLeft = "80px"

inlinestyle 設定物件屬性, 優先權最高

長度單位

- 在使用 CSS2Properties 物件的位置屬性時，所有的屬性數值都必須有單位(在standard mode時)。

- e.g. 

```
var para = document.getElementById('para');
```

```
para.style.position = "absolute";
```

```
para.style.left = "300px";
```

```
para.style.top = "250px";
```

getproperty, setproperty 2方法

Style Properties and Methods

- `style.cssText`
 - e.g. `my.style.cssText="width: 25px; height: 100px;"`
- `style.setProperty(propertyName, value, priority)`
 - e.g. `style.setProperty("display", "none");`
 - priority can be "important" or null
- `style.getPropertyValue(propertyName)`
- `style.removeProperty(propertyName)`
- `style.length`
- `style.item(index)`
 - return the name of CSS property at the given position
- [note] `style` 物件的內容僅止於標籤的 `style` 屬性設定

Iterating the CSS Properties

- e.g. `cssStyle.html`

```
for (var i=0; i<obj.style.length; i++) {  
    prop = obj.style.item(i); // or obj.style[i]  
    value = obj.style.getPropertyValue(prop);  
    document.writeln(...);  
}  
  
↓  
inline style
```

用[]存取時要確認是陣列 (凶毒不可以拿), 集合

存取 CSS Rules

- Document 物件的 styleSheets 屬性值是 CSSStyleSheet 物件的陣列
- CSSStyleSheet 物件的 cssRules 屬性值是樣式表中的所有規則的陣列
- e.g. // W3C DOM Level 2

```
var firstRule = document.styleSheets[0].cssRules[0];
// 第一個樣式表的第一條規則
```

- IE 不支援 cssRules 屬性，而是使用 rules 屬性

```
var firstRule = document.styleSheets[0].rules[0];
```

存取 CSS Rules - Example

- e.g. [cssRulesGet.html](#)

```
myStyleSheets[1].cssRules[0].selectorText // selector name
```

```
myStyleSheets[1].cssRules[0].cssText // declaration
```

- [note]

please display this example by FireFox.

For Chrome, cssRules is null when stylesheets loaded from local disk, so **do not open this sample by Chrome**

讀取 Effective CSS 樣式

- 下列範例讀取 p 元素的 color 樣式值，不管 color 樣式的來源為何：
 - e.g.

```
var p = document.getElementsByTagName("p")[0];
var color = window.getComputedStyle(p, null).color;
```
 - 語法 : `window.getComputedStyle(element[, pseudoElt]);`
- Internet Explorer 不支援 Windows 物件的 `getComputedStyle` 方法，而是使用元素的 `currentStyle` 屬性來取得元素的所有來源的樣式：

```
var p = document.getElementsByTagName("p")[0];
var color = p.currentStyle.color;
```
- e.g. [cssEffective.html](#)

附錄：預約程式碼的執行

`setTimeout()`, `setInterval()`

setTimeout() 使用方式

- Syntax

```
timer = setTimeout("JavaScript的命令列", 時間長度);
```

- 說明

- 設定使某段程式碼在 **指定時間後開始執行**。
- 第一個 **參數是執行的程式碼**，可以是一個函數；或者是一段 JavaScript 程式碼，以字串的形態傳入。
- 第二個 **參數是幾毫秒後開始**，單位是 **毫秒(ms)**，所以要用 **秒來計算時記得 *1000**。
- 用 `clearTimeout(timer)` 取消預約執行。

setTimeout() - Example

- e.g. [marquee.html](#)
- 利用 `setTimeout()` 反覆執行某段程式碼

```
function messageRotate(){  
    setTimeout(messageRotate, 400);  
}
```

- 當執行到 `setTimeout(messageRotate, 400);` 的時，會預約下一次的執行，即可定時重複執行這個函式
- `messageRotate()` 每次執行會把 `message` 的第一個字移至字串最末端，藉以製造跑馬燈的錯覺

```
15 <script>
16 text="微軟公司的XP作業系統走過十三個年頭，將於八日停止更新。業者估計，全球仍有百分之廿到百分之卅的使用者在用XP，這些未升級Windows 8的使用者未來得自己想辦法抵擋駭客攻擊。";
17 text=text+"      "; // 加一些全形空白
18 var myText1 = document.getElementById("myText1");
19 function messageRotate(){
20     var n = text.length;
21     text = text.substr(1, n-1)+text.substr(0, 1); // 將第一個字搬到最後面
22     myText1.innerText = text; // 設定至 <div> 標籤內文
23     setTimeout(messageRotate, 1000); // 0.1 秒閃一遍
24 }
25 messageRotate();
26 //var timer = setInterval(messageRotate, 100);
27 //setTimeout("clearInterval(timer);", 3000);
28 </script>
```

預頁到下次
setInterval 會存起來等
要結束時把 setInterval 刪掉

setInterval() 使用方法

- `timer = setInterval("JavaScript的命令列", 時間長度)`
 - 週期性執行某段程式碼
 - 也可以用 `setTimeout()` 達到此效果，但是 `setInterval()` 更為精簡。
 - 用 `clearInterval()` 取消預約。
- e.g. [clock.html](#) 使用 `setInterval()` 顯示數位時鐘
- 說明
 - 每次執行 `showTime()` 就重新取得最新時間。
- Note: w3schools 時鐘
https://www.w3schools.com/js/js_timing.asp