
# JavaScript 程式設計新手村

## 單元17 - BOM & DOM 操作基礎

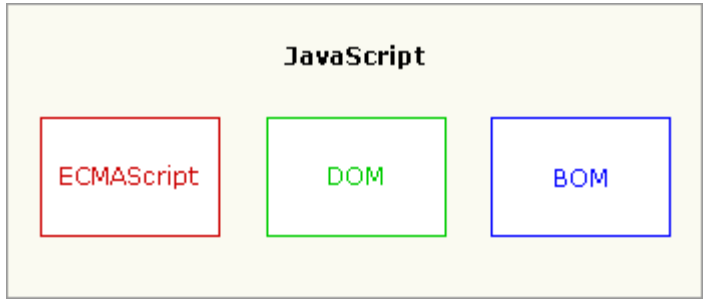
@kdchang

# Outline

1. BOM & DOM 基礎概念
2. BOM 基礎概念
3. BOM API 操作
4. DOM 基礎概念
5. DOM API 操作

# BOM & DOM 基礎概念

# 廣義 JavaScript 包含 BOM & DOM



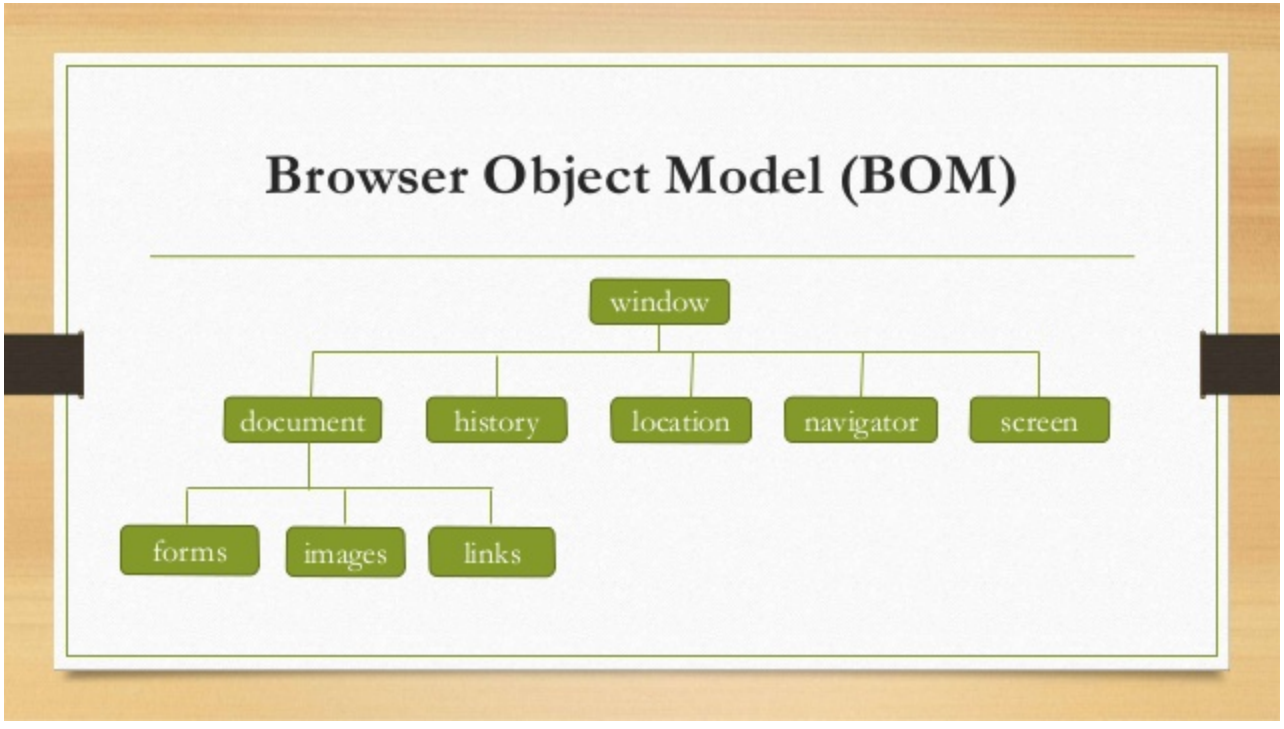

1. JavaScript 的核心 ECMAScript 描述了語言的語法和物件
2. BOM 描述了和瀏覽器互動的方法和 API
3. DOM 描述了網頁文件的內容和 API

# BOM & DOM 基礎概念

所謂的物件模型（Object Model）對於HTML 網頁來說，是一種規範如何存取HTML 元素、CSS 樣式和 JavaScript 程式碼的一種機制，可以將HTML元素、CSS樣式和 JavaScript 程式碼視為物件

# BOM 基礎概念

# BOM 基礎概念




BOM 就是 Browser Object Model 中文叫做瀏覽器物件模型

window 物件是瀏覽器最頂層物件，其下有 document（DOM）、history、location、navigator、screen 子物件

# BOM 小秘密

`window` 物件不須經過宣告，可直接使用，代表目前瀏覽器視窗

所有的全域變數、函式、物件，其實都是屬於 window 物件

BOM 物件的使用可讓我們操作包含開啟/關閉視窗，改變視窗大小，計時器與取得網址、存取瀏覽器屬性等

# BOM API 操作

# 視窗內容的捲動

window 物件提供三種方式可以捲動視窗：

- `scroll(x, y)`：移動指定位置
- `scrollTo(x, y)`：移動到指定位置
- `scrollBy(offsetx, offsety)`：從目前視窗移動到參數位移量

範例程式

# 視窗交談窗

- `alert(message)`：警示窗
- `confirm(message)`：確認窗
- `prompt(msg, default value)`：問答窗

# setTimeout

`setTimeout()` 是設定一個指定等候時間 (千分之一秒為單位)，時間到了，瀏覽器就會執行一個指定的函數，使用完記得用 `clearTimeout()` 停止 `setTimeout()` 方法

第一個參數為要執行的 `method` 或 `function`
第二個參數為等候的時間

```
function helloWord(){
 console.log("Hello");
}
setTimeout("helloWord()", 1000) ;
var id = setTimeout("alert('Hello')", 1000) ;
clearTimeout(id);
```

範例程式

# setInterval

`setInterval()` 每隔「時間長度」所指定的時間（以千分之一秒為單位）後，瀏覽器就會去執行指定函數 ，和`setTimeout` 不同是會其會每隔段時間執行一次，同樣可用 `clearInterval(id)` 清除

第一個參數為要執行的 `method` 或 `function`
第二個參數為每隔的時間

```
var id = setInterval("showTime()", 1000); //每一秒更新一次
clearInterval(id) //傳入清除欲id
```

# screen 物件

`screen` 物件屬性

`height` 螢幕解析度高度

`width` 螢幕解析度寬度

`availHeight` 螢幕解析度排除工具列的高度

`availWidth` 螢幕解析度排除工具列的寬度

```
console.log(window.screen.height);
```

# history 物件

window 物件中的 History 子物件可以取得瀏覽歷史紀錄

- `length`：傳回物件歷史記錄數
- `back()`：上一頁
- `forward()`：下一頁
- `go(num)`：移動第num頁

# location 物件

location 物件可以取得 URL 網址資訊

- `window.location.href`：為 URL 網址，可以用於轉址
- `reload()`：如同重新整理
- `replace(url)`：轉到參數網址，但會取代歷史紀錄(無法上一頁)

# DOM 基礎概念

# DOM 基礎概念

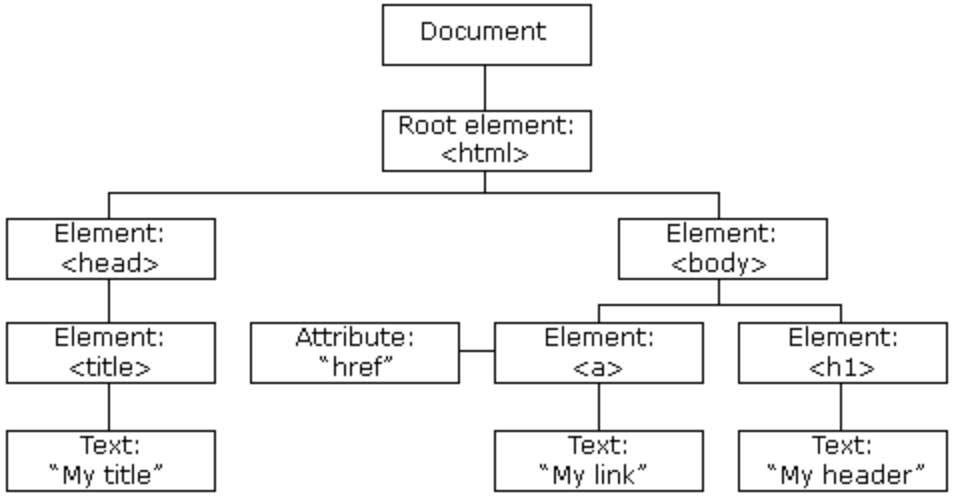
文件物件模型（Document Object Model，DOM）是給 HTML 與 XML 文件使用的一組 API

簡單的說就是將文件（文件可以想成單一網頁）物件化，以便提供一套通用存取的方式來處理文件內容。DOM 提供 HTML 網頁一種存取的方式，可以將 HTML 元素轉換成一棵節點樹，每一個標籤和文字內容是為一個節點，讓我們可以走訪節點 (Nodes) 來存取 HTML 元素

# 這是一張 HTML

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My title</title>
</head>
<body>
        <h1>My header</h1>
        <a href="">My link</a>
</body>
</html>
```

# DOM


Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"

# DOM API 選擇器

要操作 DOM 元素前要選取要操作哪個

- 根據ID名稱選取

  `document.getElementById(elementId)`

- 根據元素名稱選取

  `document.getElementsByTagName(tagName)`

- 根據名稱選取

  `document.getElementsByName(name)`

- 根據 Class 名稱選取

  `document.getElementsByClassName(classname)`

有很多元素符合，回傳的是 `NodeList` 物件集合，使用 item() 存取 (注意 Element's')，迭代使用 forEach 不然就要轉陣列

# querySelector 選擇器

document 物件有提供使用 CSS 選擇器來選取元素，效能較好

- `document.querySelectorAll()` 方法
  document 物件的 `querySelectorAll()` 方法可以取得 HTML 的節點陣列或清單，為一個 `NodeList` 物件（若要使用 map 方法需要轉陣列，不然只能用 forEach ）
- `document.querySelector()` 方法
  只會回傳一個符合的元素，沒有就回傳 null

# 元素物件 element 的屬性

- innerHTML
  存取元素的子標籤和內容但不包含標籤

## 選取元素插入元素標籤

```
<div class="info"></div>
<div class="info"></div>

<div id="danger"></div>

document.querySelector('#danger').innerHTML = '<h1>嘿嘿，是我</h

document.querySelectorAll('.info').forEach((value, index) => {
  value.innerHTML = '<h1>坐著打，普天之下排名第二</h1>';
});
```

範例程式

# 練習一

選取一個建立的 `<div>` 使用 DOM API 插入一張圖片 `<img src="圖片位置">`

# HTML element 的尺寸和位置

`offsetLeft`：節點物件距離左方邊界的距離

`offsetTop`：節點物件距離上方邊界的距離

`offsetHeight`：節點物件的高

`offsetWidth`：節點物件的寬

`offsetParent`：取得節點物件的上一層物件

# 走訪 DOM 元素

一般來說，我們在HTML元素中選擇特定元素後，通常我們還需要再找出 DOM 元素中的相關節點，如：父、子或兄弟節點

特別注意在 DOM 節點樹中，HTML 標籤是一節點，裡面的標籤內容則是其子節點

```
var node = document.getElementById('papa');
var nodeParent = node.parentNode;
console.log(nodeParent.nodeName);
```

範例程式

# 瀏覽 HTML 元素

事實上 DOM API 支援元素基礎的節點瀏覽，可以讓我們專注在HTML網頁元素的節點上，忽略文字和註解節點。基本上 parentNode 屬性一定取得的是元素節點

```
<div id="bro">
  <div>Hi 兄長</div>
</div>
```

```
var node = document.getElementById('bro');
var nodeFirst = node.firstElementChild;

console.log(nodeFirst.textContent);
```

範例程式

# 存取 HTML 屬性

- `getAttribute(attribute)` 取得傳入參數 attribute 屬性值
- `setAttribute(attribute, value)` 將 attribute 設為 value 值
- `removeAttribute(attribute)` 刪除傳入參數的 attribute 屬性

# HTML5 : 自定DATA - * 屬性

HTML5 可以讓使用者自訂屬性值，方便可以記錄一些屬性

```
<div data-num="1">HTML</div>
<div data-num="2">CSS</div>
```

範例程式

# 更改 CSS Style

CSS 樣式在 DOM API 中可以對應到Style 屬性，讓我們可以用 JavaScript 去動態更改 CSS 樣式

```
部分改變
document.getElementById('id').className= '動態加className';
document.getElementById('id').style.cssText = 'color: red';
document.getElementById('id').style.backgroundColor= '#003366'

全局改變(動態加入外部引入css檔)
<button onclick="javascript:document.getElementById('css').href
```

範例程式

# 更改 CSS 樣式

常用的 Style 屬性如下， CSS 樣式屬性和 Style 物件屬性差異在於 CSS 樣式屬性中的 "-" 在 Style 物件屬性會刪除後面字母轉大寫

CSS樣式屬性 / Style物件屬性

- color / color
- font-size / fontSize
- font-family / fontFamily
- background-color / backgroundColor
- background-image / backgroundImage
- display / display

# 練習二

使用 JavaScript 讓原來黑色 h1 文字變成紅色達到變換 CSS 效果

# 取出 input 值

透過 value 屬性可以取得所選取的input 標籤的值，但記得取出為字串，若希望進行數字運算，記得要使用 `parseInt()` 或是 `parseFloat()` 進行轉換

```
var num = document.getElementById('id').value;
```

範例程式

# 節點元素的新增

透過 JavaScript 可以針對 DOM 文件樹結構元素進行動態操作，包括動態建立元素，加入子元素或是移除指定元素等操作

## 新增節點元素

```
createElement('element name')
document.getElementById('parent');
var ele = document.createElement('div');
```

範例程式

# 插入元素

- 欲插入其後面的元素 `appendChild(新增的元素)`
- 欲插入元素所在的容器 `insertBefore(新增元素, 相對節點)`

```
<div id="dad">
  <div id="son"></div>
</div>

dad.appendChild(ele);

ele.className = "css";

dad.insertBefore(ele, son);
```

# 節點元素的刪除

移除元素 `removeChild()`

先取得欲移除元素的父元素，便可以透過 `removeChild()` ，移除其子元素

# 練習三

使用DOM API 製作一簡易計算機(含加減乘除)

# 總結

在這個單元我們學會了：

1. BOM & DOM 基礎概念
2. BOM API 操作
3. DOM API 操作