JavaScript

楊道澄　Derek

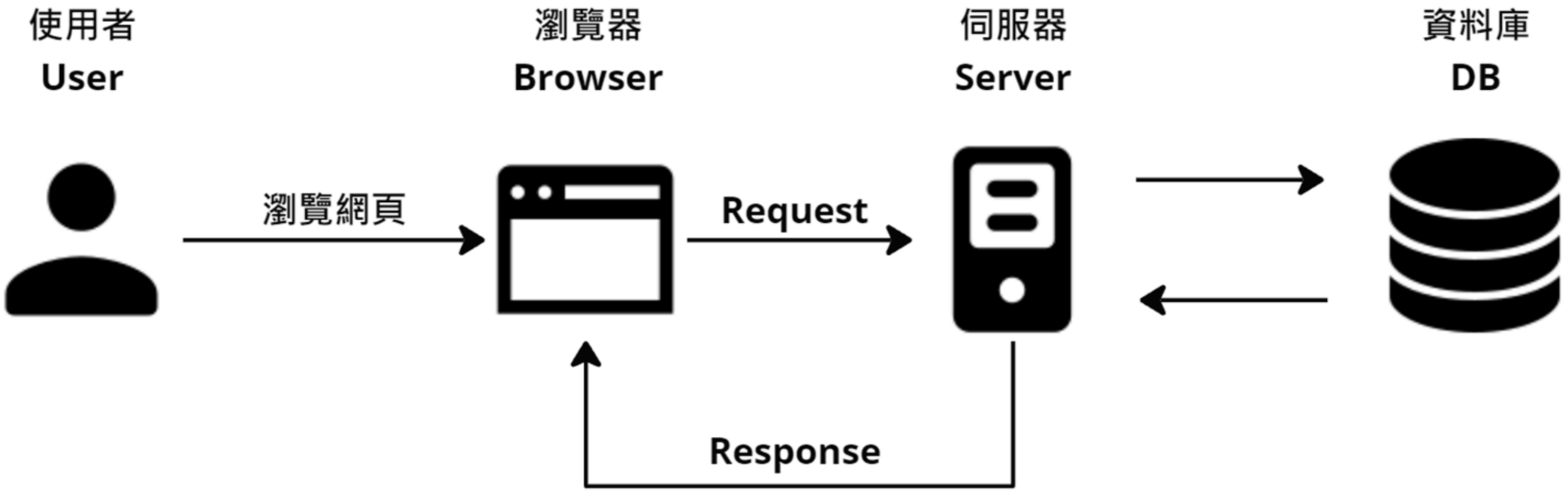
課程大綱

- 1.JS基本簡介
- 2.基本型別、變數、運算子
- 3.參考型別
- 4.BOM
- 5.流程控制
- 6.DOM
- 7.常用物件
- 8.事件處理
- 9.JSON格式

1.JavaScript簡介

網頁如何運作?

https://www.ispan.com.tw/



構成網頁的三大元素

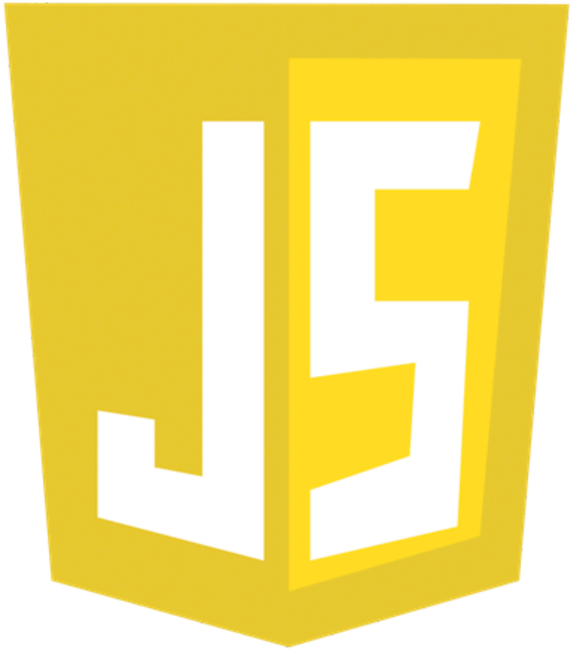
HTML

網頁基礎架構

網頁風格樣式

CSS

JavaScript



網頁動態效果與互動

JavaScript是什麼?

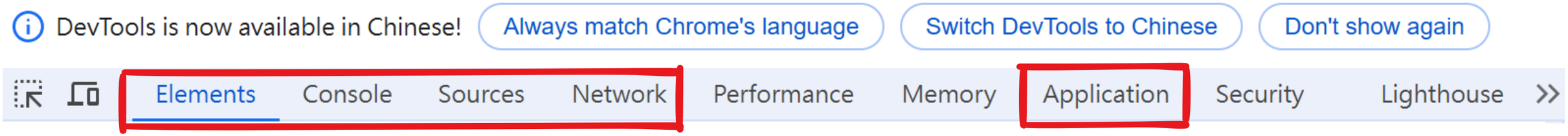
- 一種直譯程式，無須編譯器編譯。
- JavaScript(JS)，不是Java。

ECMA Script是什麼?

- JavaScript的一個標準。
- 避免我寫的JS在不同瀏覽器中會有不能執行的問題。

Chrome Developer Tools

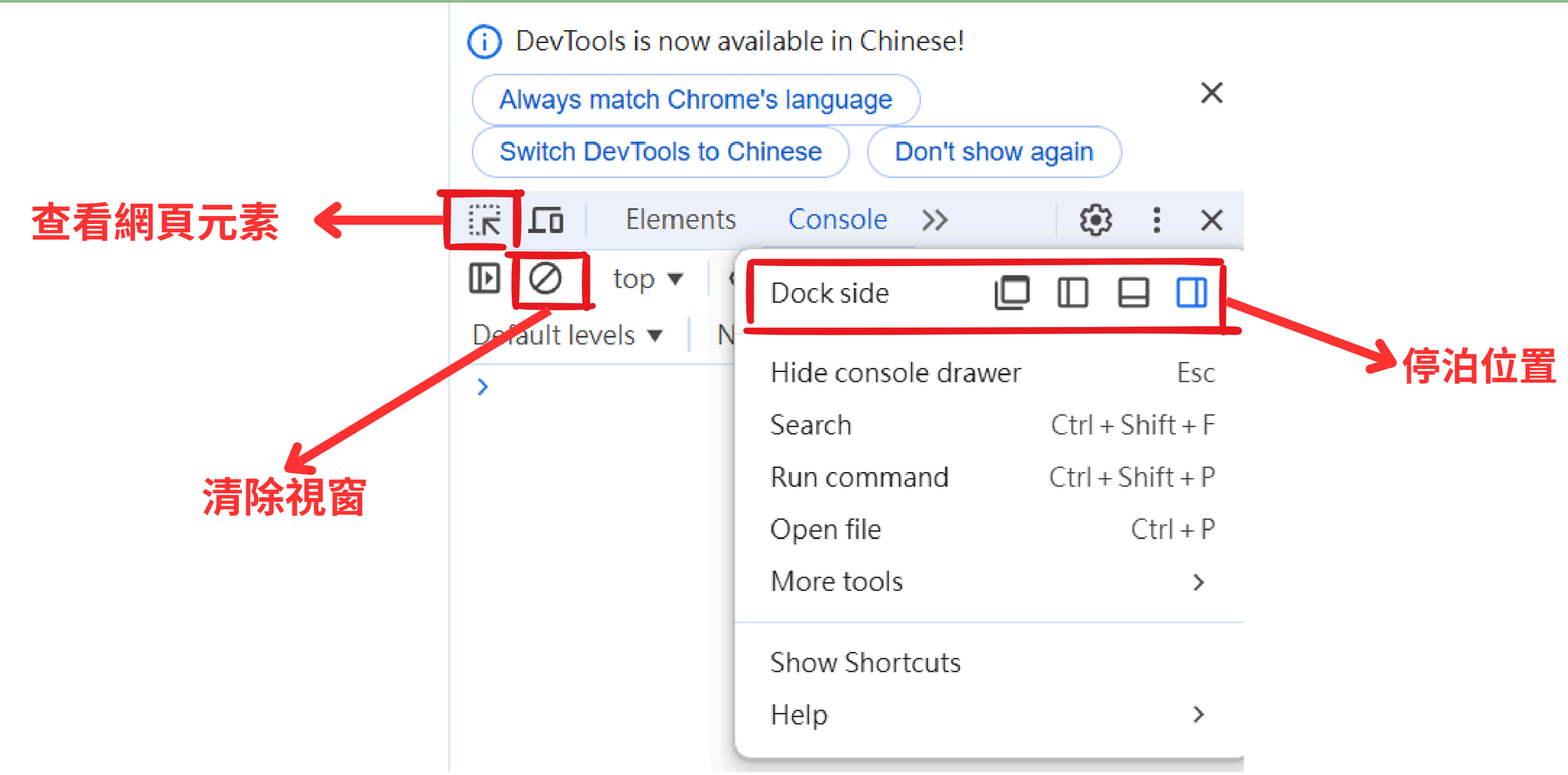
- Elements (元素)
- Console(主控台)
- Sources(原始碼)
- Network(網路)
- Application (應用程式)



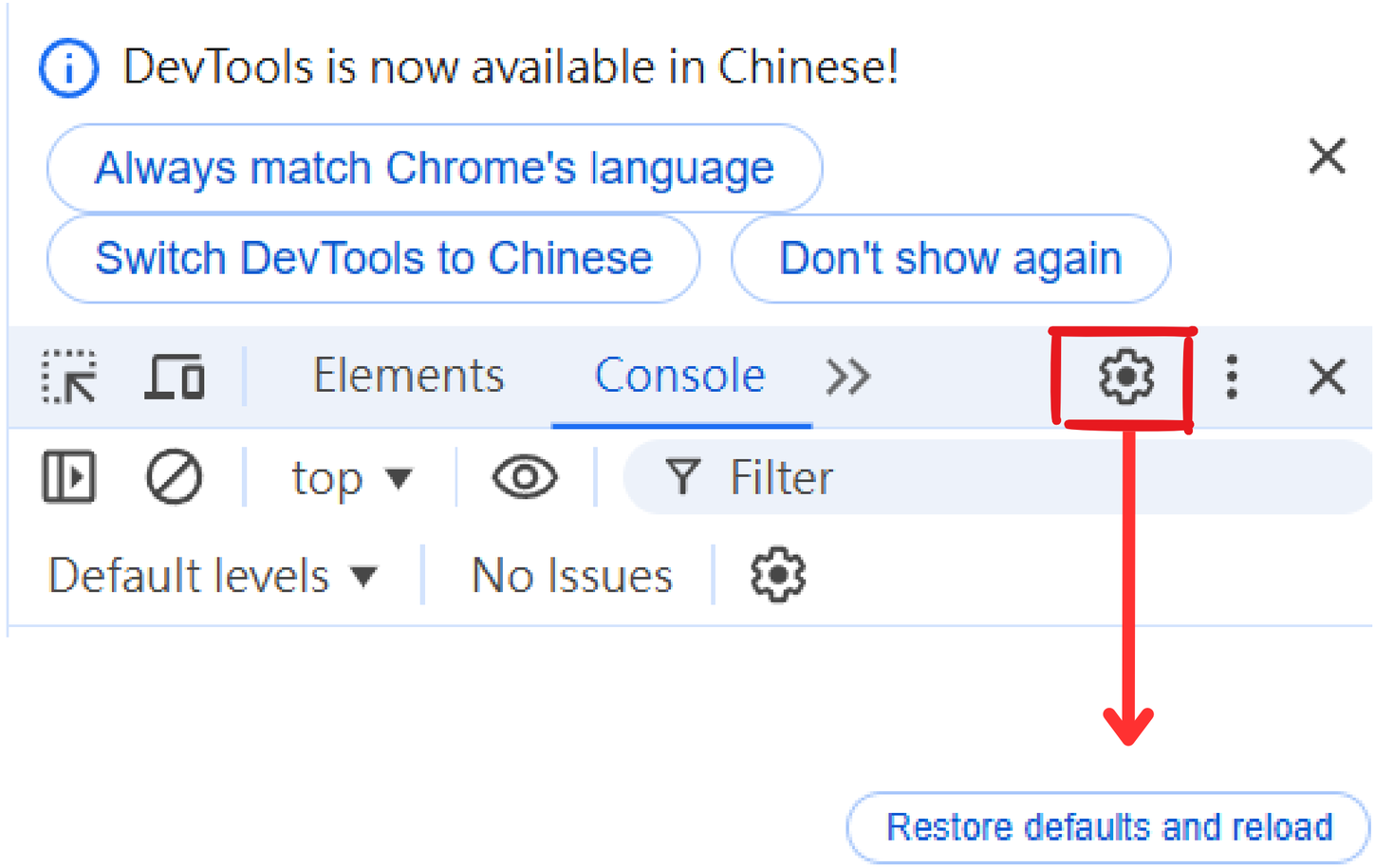
Chrome Developer Tools

- Elements (元素)
  - 檢視HTML結構與CSS樣式。
- Console(主控台)
  - 可執行JS指令，顯示錯誤與警告訊息。
- Sources(原始碼)
  - 檢視使用資源，引用到的檔案。
  - 設置中斷點，JavaScript偵錯。
- Network(網路)
  - 檢視Http載入資源的狀況。
- Application (應用程式)
  - 檢視瀏覽器上儲存的資料，ex：cookies、localStorage、sessionStorage。

Chrome Developer Tools



Chrome Developer Tools



還原最初設定

2.基本型別、變數、運算子

2-1.基本型別 (Primitive Types)

- number 數字
- string 字串
- boolean 布林
- undefined 未定義
- null 空值
- symbol *ES2015(ES6)
- bigInt *ES2020(ES11)

Number 數字

- 數字都是IEEE754 64位元 雙精度浮點數 (Float)。
- 不區分整數與浮點數。
- Try It
  - 10+10
  - 1/8
  - 0/0
  - -1/0
  - 1.5e10
  - 0.1 + 0.4
  - 0xA

Number 數字

- NaN (Not a Number)
- Infinity 無限大
- -Infinity 負無限大

浮點數的地雷

- 0.1 + 0.2 > 0.3
- 無法被二進制精確表示的小數相加會有誤差出現。
- (0.1+0.2).toFixed(1)

String 字串

- 用單引、雙引、反單引號包住的內容。
- Try It
  - 'Hello'
  - "Hello"
  - `Hello`
  - 10 + '1'

Boolean 布林

- 兩個關鍵字 true、false
- Try It
  - true
  - false
  - false == 0
  - '1' == 1
  - '1' !== 1

Undefined (未定義)
Null (空值)

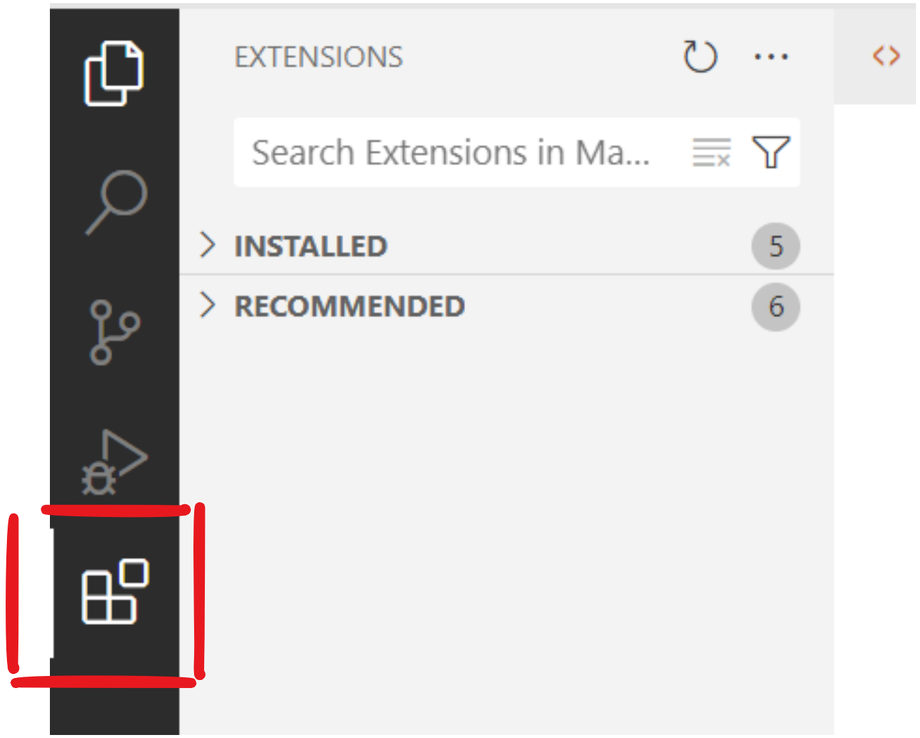
- 沒有指派值，所以是一個全空的狀態。
- 有指派值，所以是有一個值，這個值是空。
- Try It
  - `let a ; typeof a;`
  - `let b = null ; typeof b;`

Visual Stuido Code

- ctrl + / (註解/解註解)
- alt + shift + F (排版)
- ctrl + [+] (放大)
- ctrl + [-] (縮小)

Visual Stuido Code

- 擴充套件
  - Live Server
  - indent-rainbow

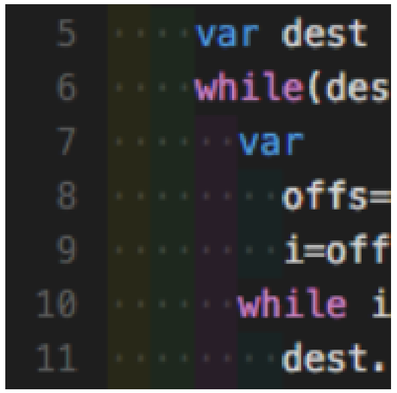


Live Server v5.7.9

Ritwick Dey | 56,986,791 | ★★★★½ (493)

Launch a development local Server with live reload feature for static & dynamic pages

Disable | Uninstall | Auto Update



indent-rainbow v8.3.1

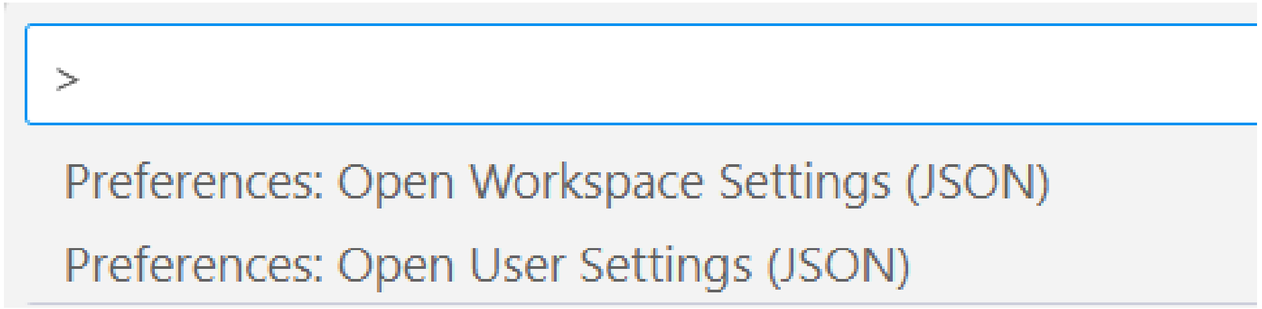
oderwat | 9,196,162 | ★★★★★ (135)

Makes indentation easier to read

Disable | Uninstall | Auto Update

Visual Stuido Code

- **F1 開啟**
- **Workspace Settings**
  - **針對專案設定**
- **User Settings**
  - **針對使用者設定**



"liveServer.settings.CustomBrowser": "chrome"
啟動時使用 Chrome 作為預設的瀏覽器

JavaScript要放在網頁的哪?

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        /* CSS放置處 */
    </style>
</head>

<body>
    <!-- HTML放置處 -->
    <script>
        // JS 放置處
    </script>
</body>

</html>
```

JavaScript 撰寫與使用

- 以分號(;) 結束
- // 單行
  - 快捷鍵：ctrl + /
- /* 多行註解 */
  - 快捷鍵：shift + alt + A
- 外部讀取方式
  - <script src='main.js'></script>

2-2.變數 (Variable)

- 要有宣告，才能使用。

- 宣告方式(let、var)
  - let 變數名稱
  - let 變數名稱 = Value
  - var 變數名稱
  - var 變數名稱 = Value

let與var

- let 要先宣告才能使用。
- var 只要有宣告就能使用。
- 使用var時若變數已經存在並不會有錯誤。

```
// let
console.log(text);
let text = 'Hello';

// var
console.log(text);
var text = 'Hello';
console.log(text);
```

Hoisting
提升、抬升

- 簡單說就是先用後補，撰寫時先使用後宣告。
- 宣告的動作一律都會被提升到函式的最頂端。
- 建議都用let

常數
(Constant)

- 設定後不能再次修改其值。
- 宣告方式
  - const 常數名稱 = Value
- 常會用全部大寫來表示加上底線分隔表示。

```
const PI = 3.14;
PI = 1; // Error:Assignment to constant variable.

const MY_CONST = 888;
```

命名規則

命名法	範例	用於
Camel Case	camelCase	JS
Kebab Case	kebab-case	CSS
Screaming Case	SCREAMING_CASE	DB Schema、JS(const)
Pascal Case	PascalCase	C#
Snake Case	snake_case	Python

保留字

var	let	if	for	else
const	break	do	switch	case
continue	import	export	in	true
false	finally	function	instanceof	new
this	try	catch	typeof	while

模板字串

- 字串使用反引號``，變數使用 ${}
- ``可輸出多行文字

```
let name = 'Derek';
let outputText = `Hello ${name}!!`
```

跳脫字元

替代字元	說明
\0	Null字元（\u0000）
\'	單引號（\u0027）
\"	雙引號（\u0022）
\\	反斜槓（\u005C）
\r	carriage return 回車符（\u000D）
\n	new line 換行符（\u000A）
\t	tab 水平製表符（\u0009）
\v	vertical tab 垂直製表符（\u000B）
\b	backspace 退格符（\u0008）
\f	form feed 換頁符（\u000C）
\uXXXX	Unicode 字元
\u{XXXX}	Unicode 字元

2-3.運算子

- 算數運算子
- 指定運算子
- 比較運算子
- 單元運算子
- 邏輯運算子

算數運算子

四則運算

- 加法：+
- 減法：-
- 乘法：*
- 除法：/
- 取餘數：%

指定運算子

將右側值賦值給左邊的變數

- 搭配算術運算子

運算子	範例	等同於
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5

比較運算子

運算子	範例	說明
==	5 == '5' true	是否相等(值)
===	5 === '5' false	是否不相等 (值與型別比較)
!=	5 != '5' false	是否不相等(值)
!==	5 !== '5' true	是否不相等 (值與型別比較)
>	5 > 5 false	大於
<	5 < 5 false	小於
>=	5 >= 5 true	大於等於
<=	5 <= 5 true	小於等於

單元運算子

針對單一資料運算

- 若x = 5 , a = true

運算子	範例	說明
+	+x	x = 5
-	-x	x = -5
++	x++	x = 6
--	x--	x = 4
!	!a	a = false

邏輯運算子

AND 且：&&

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

OR 或：||

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

三元運算子

Q ? A : B

- 條件式 ? 語句1 : 語句2
- 條件式若為true 回傳語句1，反之回傳語句2

3.参考型別
(Reference Types)

型別

- 基本型別
  - number 數字
  - string 字串
  - boolean 布林
  - undefined 未定義
  - null 空值
- 參考型別
  - Object 物件
  - Array 陣列
  - Function 函式

物件Object

- {}
- 可以儲存多個屬性或方法的資料型態。
- 由key和value組成(key-value pairs)
- {key：value,key：value,...};
- 物件的屬性也可以是物件

```
// 建立方式1
let obj1 = { '屬性名稱': Value };
// 建立方式2
let obj2 = new Object();

// 設定方式
obj2.屬性名稱 = value;

// 讀取方式
let getValue = obj.屬性名稱;
let getValue  = obj[屬性名稱];
```

陣列Array

- []
- 儲存連續的資料。
- 用索引(index)取值，從0開始。

```
// 建立方式
let array = ['value1', value2];

// 設定方式1
array[index] = value;
// 設定方式2
array.push(value); // 加到最後一筆

// 讀取方式
let getValue = array[index];
```

函式Function

- 一段獨立程式碼，完成指定工作。
- 讓程式可以簡化與重用。

- 定義函式
  - 函式宣告式(Function Declaration)
  - 函數表達式(Function Expression)
  - 箭頭函式(Arrow Function)

函式宣告式

```
function functionName(params1, params2) {
        // 執行邏輯;
        return result;
    }
```

- return：回傳結果，若沒有return，回傳 undefined

函數表達式

```
const functionName = function (params1, params2) {
    // 執行邏輯;
    return result;
}
```

- Hoisting 特性會消失或被降級
- Hoisting函數優先度比變數高

試著練習觀察一下發現什麼

```
console.log(test);
function test() {
}
var test = 'Hello';

console.log(test);
var test = 'Hello';
function test() {
}
```

箭頭函式

```
const functionName = (params1, params2) => {
    // 執行邏輯;
    return result;
}

const functionName = params1 => {
    // 執行邏輯;
    return result;
}
```

- 只有一個參數括號可省略。
- 會影響到「this」的使用。(不會產生this)

this保留字

- 會根據執行式的上下文做改變

位置	this指向
全域	window物件
函式	window物件
建構子	建立的實體
方法	使用的物件
事件監聽	事件觸發者

class保留字

- ES6
- 有建構子constructor(保留字)

```
public class Member
{
    /// <summary>
    /// 建構子
    /// </summary>
    public Member() { }

    public string Name { get; set; } = default!;
    public int Age { get; set; } = default!;
}
```

```
class Person {
    constructor(name, age) {
        this.name = name;
        this.age = age;
    }
}
let person = new Person("Derek", 32);
```

為什麼叫參考型別

- 儲存的是指向對象的參考（地址），而不是對象本身的值。

試著練習觀察一下發現什麼

```
let person = new Object();
person.name = 'Derek';
let otherPerson = person;
otherPerson.name = 'Peter';
console.log(person === otherPerson);
console.log(otherPerson.name);
```

型別

- (補充)都在描述同一種型別
  - 物件型別(Object Types)
  - 參考型別(Reference Types)
  - 非基本型別(Non-Primitive Types)

4.BOM

BOM

Browser Object Model

瀏覽器物件模型

- 與瀏覽器進行互動，並控制瀏覽器窗口、頁面導航等基本操作。
- winodw物件是BOM的核心。

window物件

- 屬性
  - innerWidth
  - innerHeight
  - location
  - screen
  - console
- 方法
  - alert('提示訊息')
  - prompt('提示', '預設值')

window物件

- location、screen、console...等物件是window中的一個屬性
- 這些物件中還可以包含別的屬性或方法
- location
  - href
- screen
  - width
  - height
- console
  - log()
- history
  - forward()
  - back()

document物件

- 是window中的一個屬性
- window.document
- 屬性
  - title
  - body
    - innerText
    - innerHTML
    - insertAdjacentText('位置','內容')
    - insertAdjacentHTML('位置','內容')

insertAdjacentXXX

5.流程控制

流程控制

- 循序結構
  - if...else...
- 選擇結構
  - switch...case...break
- 重複結構
  - while
  - do...while...
  - for loop

if...else...

```
if (條件) {
    // 條件為true 執行此區塊
} else {
    // 條件為false 執行此區塊
}

if (條件1) {
    // 條件1為true 執行此區塊
} else if (條件2) {
    // 條件2為true 執行此區塊
} else {
    // 條件1與2都為false 執行此區塊
}
```

if...else...

練習題

輸入文字

1.若為數字：
顯示「是數字，數字是：${inputText}」

2.若為文字：
顯示「不是數字，輸入的是：${inputText}」

3.若為空白：
顯示「未輸入」

switch...case...break

```
switch (判斷值) {
    case 符合值1:
        // 符合值1 執行此區塊
        break;

    case 符合值2:
        // 符合值2 執行此區塊
        break;

    default:
        // 都不符合 執行此區塊
        break;
}
```

- 做嚴格比較 (===)
- break沒寫不會跳出

switch...case... break

練習題

依照輸入票種顯示對應的票價。

電影版本	全票	優待票	早場票	愛心票
2D數位電影	320	290	270	160

while
do...while

```
while (條件) {
    // 條件為true時，迴圈執行此區塊
}

do {
    // 先執行此區塊，再檢查條件
    // 條件為true時，迴圈執行此區塊
} while (條件);
```

while
do...while

練習題

猜數字遊戲

設定猜數字的範圍(1~100)

先輸入答案後，讓玩家開始猜答案。

for loop

初始化 ； 條件 ；增減量

```
for (let index = 0; index < value; index++) {
    //  [index < value] 為true時，迴圈執行此區塊
}
```

for loop

練習題

九九乘法表

1的乘法	2的乘法	3的乘法	4的乘法	5的乘法	6的乘法	7的乘法	8的乘法	9的乘法
1 x 1 = 1	1 x 2 = 2	1 x 3 = 3	1 x 4 = 4	1 x 5 = 5	1 x 6 = 6	1 x 7 = 7	1 x 8 = 8	1 x 9 = 9
2 x 1 = 2	2 x 2 = 4	2 x 3 = 6	2 x 4 = 8	2 x 5 = 10	2 x 6 = 12	2 x 7 = 14	2 x 8 = 16	2 x 9 = 18
3 x 1 = 3	3 x 2 = 6	3 x 3 = 9	3 x 4 = 12	3 x 5 = 15	3 x 6 = 18	3 x 7 = 21	3 x 8 = 24	3 x 9 = 27
4 x 1 = 4	4 x 2 = 8	4 x 3 = 12	4 x 4 = 16	4 x 5 = 20	4 x 6 = 24	4 x 7 = 28	4 x 8 = 32	4 x 9 = 36
5 x 1 = 5	5 x 2 = 10	5 x 3 = 15	5 x 4 = 20	5 x 5 = 25	5 x 6 = 30	5 x 7 = 35	5 x 8 = 40	5 x 9 = 45
6 x 1 = 6	6 x 2 = 12	6 x 3 = 18	6 x 4 = 24	6 x 5 = 30	6 x 6 = 36	6 x 7 = 42	6 x 8 = 48	6 x 9 = 54
7 x 1 = 7	7 x 2 = 14	7 x 3 = 21	7 x 4 = 28	7 x 5 = 35	7 x 6 = 42	7 x 7 = 49	7 x 8 = 56	7 x 9 = 63
8 x 1 = 8	8 x 2 = 16	8 x 3 = 24	8 x 4 = 32	8 x 5 = 40	8 x 6 = 48	8 x 7 = 56	8 x 8 = 64	8 x 9 = 72
9 x 1 = 9	9 x 2 = 18	9 x 3 = 27	9 x 4 = 36	9 x 5 = 45	9 x 6 = 54	9 x 7 = 63	9 x 8 = 72	9 x 9 = 81

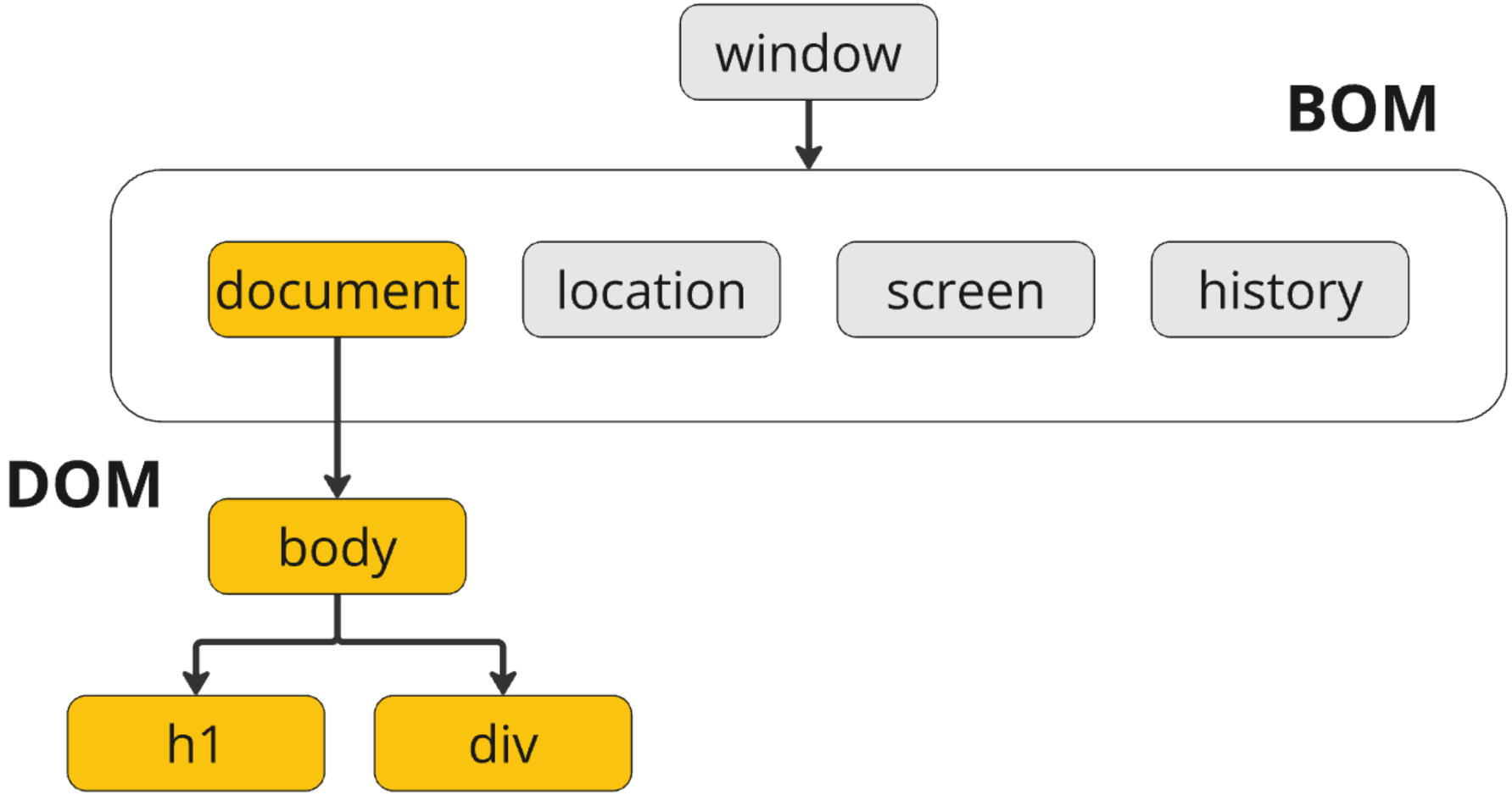
for...of...
for...in...

```
for (const 自訂變數 of 可列舉物件) {
    // 迴圈執行此區塊
}

for (const 自訂變數 in 物件 / 陣列) {
    // 迴圈執行此區塊
}
```

6.DOM

DOM 和 BOM

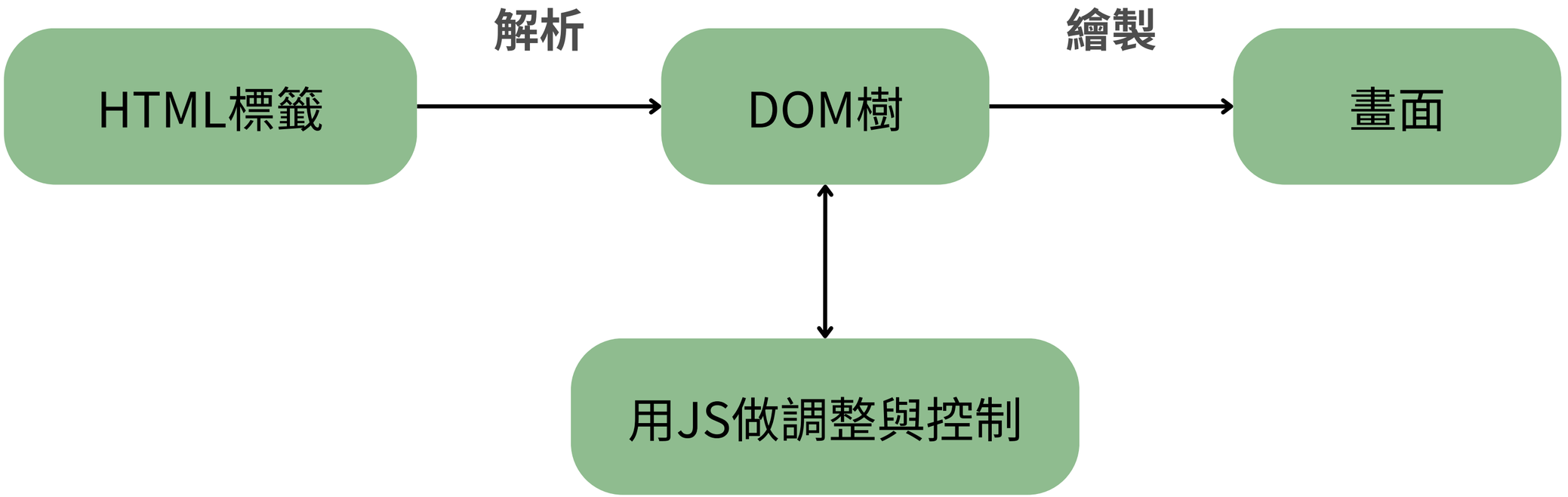

window
BOM
document
location
screen
history
DOM
body
h1
div

DOM

Document Object Model

- 瀏覽器載入HTML標籤時，會建立對應的DOM物件，再將其依照結構串在一起組成DOM樹。
- 瀏覽器根據DOM樹狀結構繪製畫面。
- 調整DOM，瀏覽器就會根據DOM結構重繪畫面。

瀏覽器運作

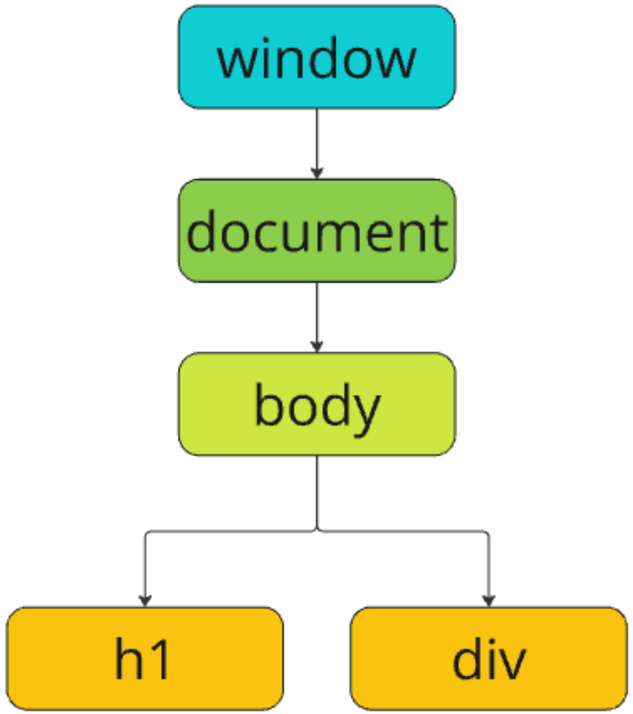


DOM

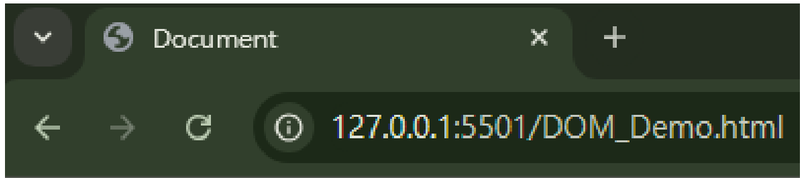
HTML標籤

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>我是Title</title>
</head>
<body>
    <h1>我是標題</h1>
    <div>我是內容</div>
</body>
</html>
```

DOM樹



畫面



我是標題

我是內容

DOM 和 BOM

- BOM
  - Browser Object Model。
  - 不處理頁面內容，只操作瀏覽器相關功能。
- DOM
  - Document Object Model。
  - 操作網頁內容。
  - 可透過JS動態修改HTML和CSS 。

document物件

- 是window中的一個屬性
- window.document
- 屬性
  - title
  - body
    - innerText
    - innerHTML
    - insertAdjacentText('位置','內容')
    - insertAdjacentHTML('位置','內容')
    - insertAdjacentElement('位置','內容')
  - createElement('標籤')
    - textContent
    - className

取得節點元素

- getElementById(id)
- getElementsByTagName(tag)
- getElementsByName(name)
- getElementsByClassName(class)
- querySelector(selector)
- querySelectorAll(selector)

取得節點元素

方法	回傳物件類型
getElementById(id)	Element 或 null
getElementsByTagName(tag)	HTMLCollection
getElementsByName(name)	NodeList
getElementsByClassName(class)	HTMLCollection
querySelector(selector)	Element 或 null
querySelectorAll(selector)	NodeList

元素的CRUD

- 新增
  - document.createElement(tag)
- 插入
  - element.append(nodes)
  - element.appendChild(node)
- 讀取
  - element.innerText
  - element.innerHTML
- 設定
  - element.innerText = content
  - element.innerHTML = content
- 刪除
  - element.remove();

7.常用物件

Array物件

方法	用途
push() / pop()	新增一或多個元素到末尾 / 移除最後一個元素
unshift() / shift()	新增一或多個元素到開頭 / 移除開頭第一個元素
map()	每一個元素依指示處理後，返回新陣列。
filter()	根據條件過濾，返回符合條件的新陣列。
forEach()	迴圈
find() / findIndex()	找到第一個符合的元素回傳 / 找到第一個符合的元素索引
includes()	檢查陣列是否包含某個值，返回布林值。
join()	將陣列元素轉換為字串，並以指定的分隔符連接。
sort()	對陣列元素進行排序
concat()	合併兩個或多個陣列，返回新陣列。
reverse()	反轉陣列元素順序。

Date物件

方法	用途
getDate() / setDate()	取得/設定日 (1~31)
getFullYear() / setFullYear()	取得/設定年(yyyy)
getMonth() / setMonth()	取得/設定月(0~11)
getDay()	取得星期(0~6)
toLocaleString()	本地化設置的日期和時間字串
toLocaleDateString()	本地化設置的日期字串
toLocaleTimeString()	本地化設置的時間字串
toUTCString()	世界協調時間(UTC)格式字串
toISOString()	國際標準ISO 8601格式字串

String物件

方法	用途
toLocaleUpperCase()	轉大寫
toLocaleLowerCase()	轉小寫
substring(start,end)	擷取索引從start到end(不含end)
trim()	移除字串兩端空白
charAt()	取得指定索引字串
indexOf()	第一次出現指定子字串的索引，如果沒有找到，則返回 -1
split()	切割指定字串轉換成陣列

Number物件

方法	用途
isNaN()	檢查是否為NaN
isInteger()	檢查是否為整數
parseInt()	字串轉為整數
parseFloat()	字串轉為浮點數

Math物件

方法	用途
abs()	絕對值
ceil()	無條件進位
floor()	無條件捨去
round()	四捨五入
pow()	指數運算
sqrt()	平方根
max()	取最大值
min()	取最小值
random()	產生一個0~1間的亂數

8.事件處理

事件

- 使用者與瀏覽器互動的行為
- 事件驅動程式(Event-driven programming)
  - 事件對象(Event Target)
  - 事件類型(Event Type)
  - 事件處理函式(Event Handler)
  - 事件監聽函式(Event Listener)
- 事件物件(Event Object)
- 事件傳播(Event Propagation)

常見事件類型

滑鼠事件
click
dblclick
mousedown
mouseup
mouseenter
mouseleave
mousemove
mouseover
mouseout

鍵盤事件
keypress
keyup
keydown

視窗事件
load
resize
scroll

表單事件
change
submit
input
focus
blur
reset
select

事件處理函式

綁定方式1

```
<標籤 onevent="eventHandler()" />
==================================
function eventHandler() {
    // do something
}
```

綁定方式2

```
element.onevent = eventHandler;
function eventHandler() {
    // do something
}
```

事件處理函式

移除方式

```
element.onevent = null;
element.onevent = undefined;
```

事件監聽函式

綁定方式

```
element.addEventListener('event', eventHandler, [capture = false]);
```

移除方式

```
element.removeEventListener('event', eventHandler)
```

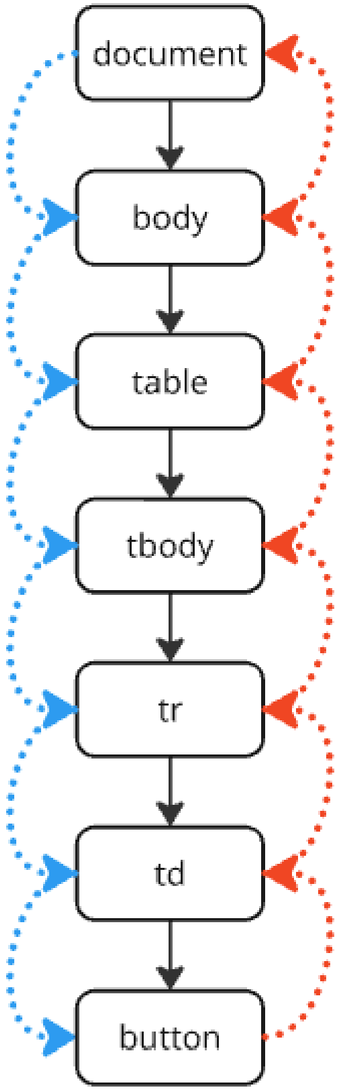
事件物件

- 網頁上的事件觸發的同時會產生event物件，記錄事件觸發的各種資訊。

屬性	說明
type	事件的類型
keyCode	回傳鍵盤輸入的編碼或滑鼠按鈕的代碼
pageX/pageY	滑鼠游標相對於文件左邊界或上邊界的位置
target	回傳觸發事件的DOM 元素

事件傳播

事件捕捉
Event Capturing



事件冒泡
Event Bubbling

- preventDefault()
  - 阻止事件的預設行為
- stopPropagation()
  - 阻止事件傳播

計時器

- window.setTimeout(function, time)
  - 時間到後只執行一次
- window.clearTimeout()
  - 停止 setTimeout 方法啟動的計時器
- window.setInterval(function, time)
  - 每隔多少時間執行一次
- window.clearInterval()
  - 停止setInterval方法啟動的計時器

9.JSON格式

JSON物件

```
{
    "name": "Derek",
    "age": 32,
    "address": {
        "city": "New Taipei City"
    }
}
```

JSON陣列

```
[
    "Alice",
    "Bob",
    "Cindy",
    "Derek"
]
```

```
[
    {
        "name": "Cindy",
        "age": 30,
        "address": {
            "city": "Taipei City"
        }
    },
    {
        "name": "Derek",
        "age": 32,
        "address": {
            "city": "New Taipei City"
        }
    }
]
```

JSON物件

- JSON.stringify()
  - JS物件轉換成JSON字串
- JSON.parse()
  - JSON字串轉成JS物件