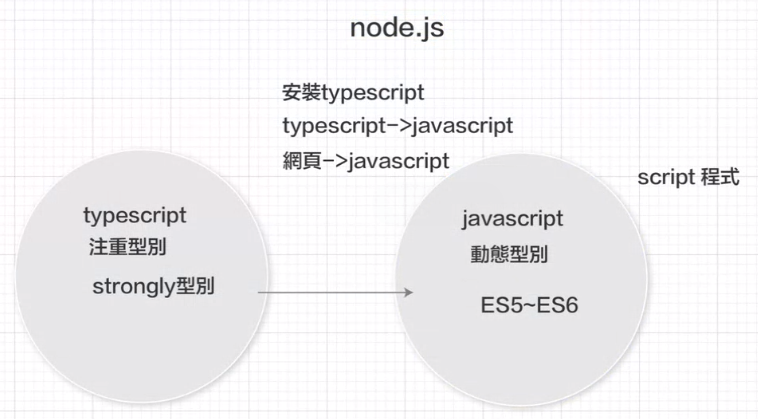
# Javascript



2023-05-08

## 1.安裝 node.js

<https://nodejs.org/zh-tw/download>

node --version 查詢是否安裝成功



npm 套件管理程式

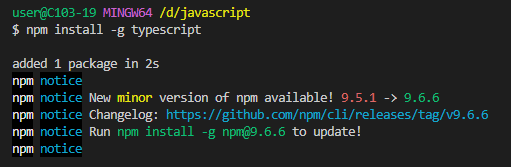
npm --version



## 2.透過npm安裝typescript

https://www.npmjs.com/package/typescript

npm install -g typescript



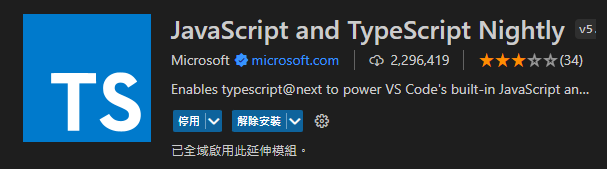
安裝完會有 tsc 命令=> typescript 指令

查詢版本

tsc --version



VS安裝套件 TypeScript



## 設定 tsconfig.json

tsc --init

產生 tsconfig.json

修改 ts 及輸出目錄

  "rootDir": "./c01/ts",

  "outDir": "./c01/js",

存檔後, 執行 tsc -w 偵測並自動產生 js

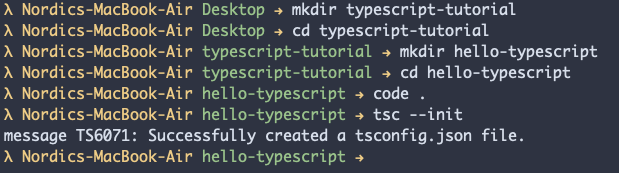
tsc -w

首先，沒有安裝過 TypeScript 的話，必須先安裝它的指令工具，記得用 -g，因為我們要讓該指令工具可以在任何地方使用喔：

$ npm install -g typescript

如果已經下載完成的話，我們可以下達：

$ tsc --init

  
圖八：終端機下達基本指令

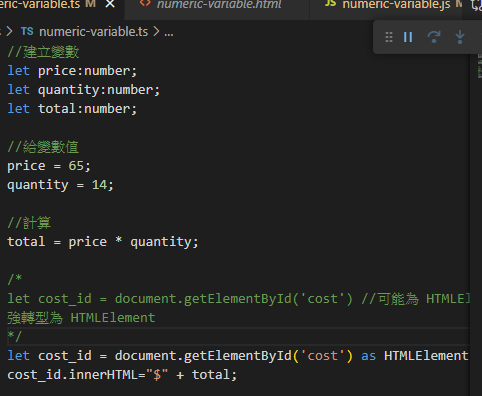
此時你的專案會出現一個名為 tsconfig.json 的檔案，這就是 TypeScript 編譯器的設定檔

js 會出現 使用嚴格形別

"use strict";

**DOM** 全名為 **Document Object Model** 中文翻譯為 **文件物件模型**，看起來很抽象但其實就是把一份 **HTML** 文件內的各個標籤，包括文字、圖片等等都定義成物件，而這些物件最終會形成一個[樹狀結構](https://zh.wikipedia.org/wiki/%E6%A8%B9%E7%8B%80%E7%B5%90%E6%A7%8B)，下面有一張示意圖可以參考。  


document.getnElementById(‘idname’)



let user\_name:string;

let message:string;

user\_name="Maple"

message="hello"

let user\_span = document.getElementById('name') as HTMLElement;

let note\_span = document.getElementById('note') as HTMLElement;

/\*

 innerHTML 是 W3C 規定的標準寫法，

 而 innerText 則是除了可以用來取得 HTML 元素之外，

   還會把元素的 HTML 標籤去除掉，

   但 innerText 並非 W3C 所規定的標準寫法，

   而且僅適用於 IE 瀏覽器

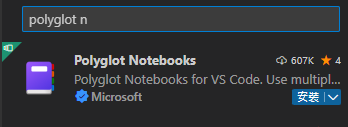
 \*/

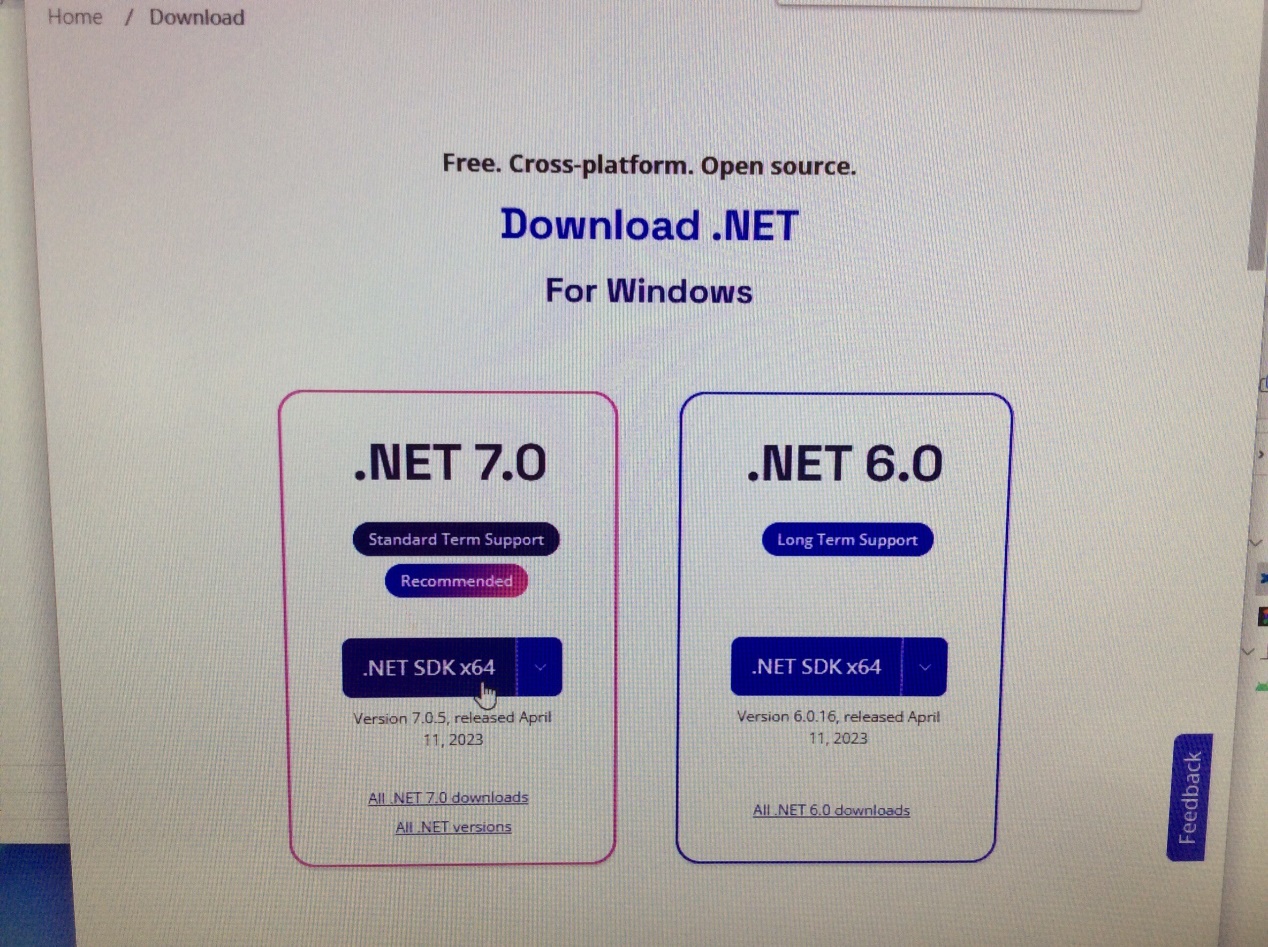
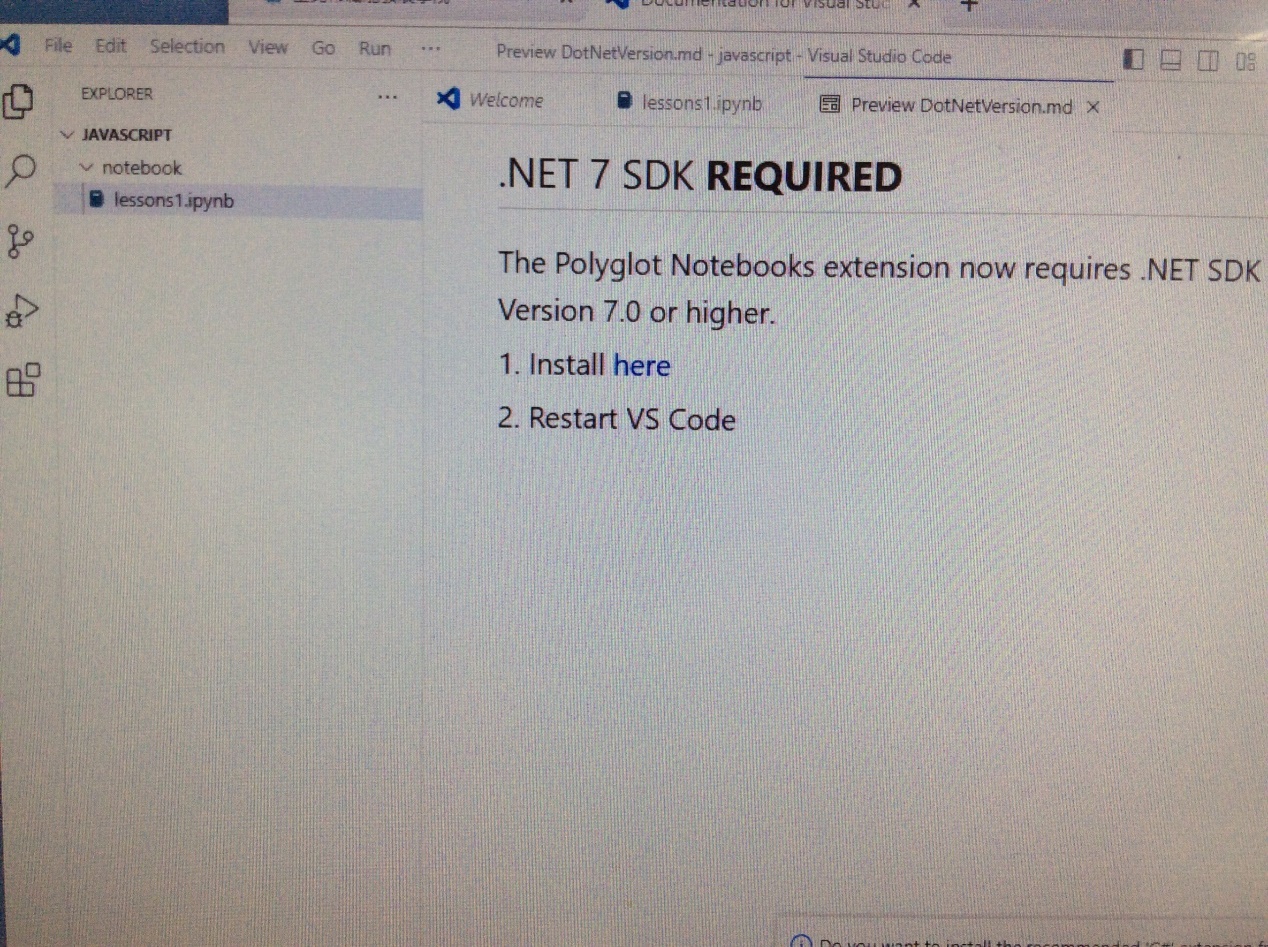
user\_span.innerHTML = user\_name;

note\_span.innerText = message;

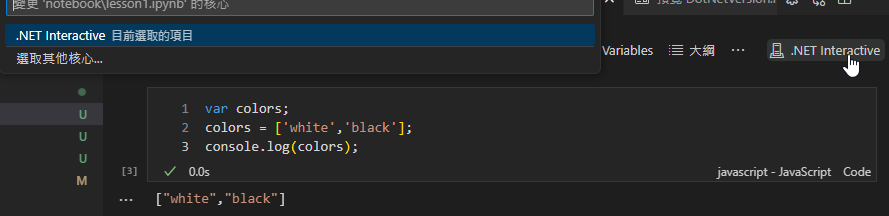


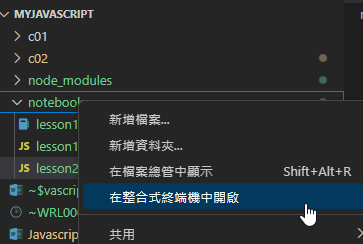
vsc安裝 Polyglot Notebooks





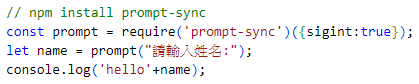
下載安裝後, 重啟vsc







3種javascript 開發環境

1. 網頁環境
2. node.js環境 : 測試輸入 require prompt
3. 筆記本環境 (.ipynb) : 測試

2023-05-09

## 數學運算

console.log(Math.round(.6))//4捨5入

console.log(Math.floor(3.9)) //無條件捨去

console.log(Math.ceil(.95)) //四捨五入並返回大於或等於給定數字的較小整數

console.log(Math.pow(2,2))  //次方

console.log(Math.log(10))  //對數

console.log(Math.pow(Math.E,2.302585092994046))

console.log(Math.sqrt(100)) //平方根

console.log(Math.trunc(-10.123)); //刪除小數

console.log('取亂數:'+Math.floor(Math.random() \* 100));//取亂數

console.log(Math.max(1, 3, 2))//取最大值

console.log(Math.abs(-22)) //取絕對值

//建立array

console.log([1,2,3,4,5])

//建立二維array

console.log([1,2,5],[3,4],[5,6])

let a=[[1,2,5],[3,4],[5,6]]

console.log(a[2][1])

//object

//物件

let p = {x:2.3, y:-1.2,abc:333}

console.log(p['x'])

console.log(p.x)

console.log(p['y'])

console.log(p.y)

console.log(p['abc'])

//object 用大括號建立

var book={

    'main title':"javascript",

    'sub-title':"The Definitive Guide",

    for:"all audiences",

    author:{

        firstname:"Davie",

        surname:"Flanagan"

    }

}

console.log(book['main title'],book['sub-title'])

console.log(book.for,book.author.firstname)

//Object 用new方法建立

var book = new Object()

book['main title'] = "javascript"

book.author = new Object()

book.author.firstname="David"

book.author.Lastname="Bob"

console.log(book['main title'],book.author.firstname,book.author.Lastname)

console.log(book)

結果

javascript

David

Bob

{"main title":"javascript","author":{"firstname":"David","Lastname":"Bob"}}

## 動態取得瀏覽器的寬及高

//windows 代表整個瀏覽器視窗

//Document 代表物件框

//window.document.body.innerHTML = "hello world";

//{$window.innerHeight} 瀏覽器inner高度

//{$window.innerWidth} 瀏覽器inner寬度

let msg = `<h2>browser window</h2>`;

msg += `<p>brower\_width: ${window.innerWidth}</p>`;

msg += `<p>brower\_height: ${window.innerHeight}</p>`;

msg += `<p>使用者的螢幕寬度: ${window.screen.width}</p>`;

msg += `<p>使用者的螢幕高度: ${window.screen.height}</p>`;

let info\_element = document.getElementById('info');

info\_element.innerHTML = msg;

msg += `<p>brower\_width: ${window.innerWidth}</p>`;

msg += `<p>brower\_height: ${window.innerHeight}</p>`;

let info\_element = document.getElementById('info');

info\_element.innerHTML = msg;

## 跳出自定視窗說明

html

<dialog id="favDialog">

      <p>Greetings, one and all!</p>

      <form method="dialog">

        <button>OK</button>

      </form>

</dialog>

js

let favDialog\_element=document.getElementById('favDialog');

favDialog\_element.showModal();

//favDialog\_element.show();

## 新增querySelector 與 forEach 及 for迴圈

// dinamic insert li

let ul = document.querySelector('ul');

let li = document.createElement('li');

li.innerText = 'new li';

li.className='cool';

ul.appendChild(li);

let first\_hot\_element = document.querySelector('li.hot'); //只會抓第一個

first\_hot\_element.className = 'cool';

//一次取出多個

let all\_elements = document.querySelectorAll('li.hot');

console.log(all\_elements.length);

//querySelectorAll foreach

## /\* forEach 方法一

all\_elements.forEach(function(element){

    element.className = 'complete';

});

\*/

/\*forEach 方法二

all\_elements.forEach(

    // 箭頭表 anonymous function

    (currentElement) => {

        currentElement.className = 'complete';

    });

\*/

## //for 方法

for (let i = 0; i < all\_elements.length; i++) {

    all\_elements[i].className = 'complete';

}

## //匿名function 寫法

1.

function(element) {

        element.className = 'complete';

    }

2.

 currentElement => {

        currentElement.className = 'complete';

    }

## 取得鄰居的元件(Sibling)兄弟姐妹

let two\_element = document.getElementById('two');

// 取得鄰居的元件

var previous\_element=two\_element.previousElementSibling;

var next\_element=two\_element.nextElementSibling;

previous\_element.className='complete';

next\_element.className='cool';

let ul\_element=document.getElementsByTagName('ul')[0];

// get ul first child li element

let firstchild = ul\_element.firstElementChild;

let lastchlid = ul\_element.lastElementChild;

firstchild.className='cool';

lastchlid.className='complete';

/\*

ul\_element.firstElementChild.className='cool';

ul\_element.lastElementChild.className='complete';

\*/