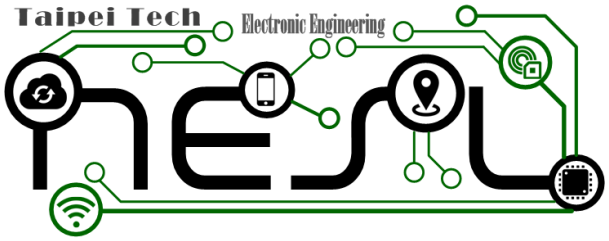
智慧整合感控系統概論
Introduction to Cyber-Physical Systems

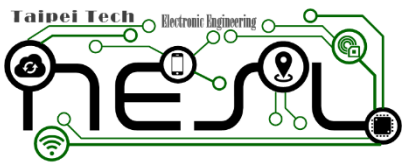
LAB: Web後端(Backend)入門

國立臺北科技大學電子工程系
授課教師：李昭賢 副教授
電子郵件：chlee@ntut.edu.tw
校內分機：2288



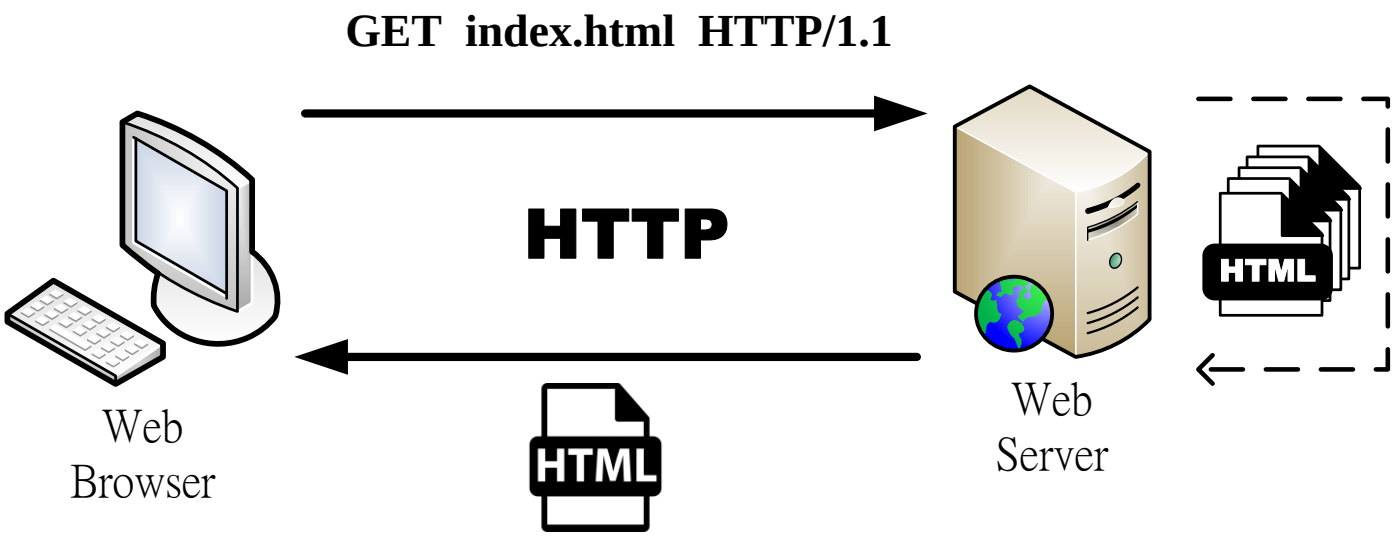
學習目標

1. Dynamic Webpage
2. Node.JS
3. Express
4. Ajax



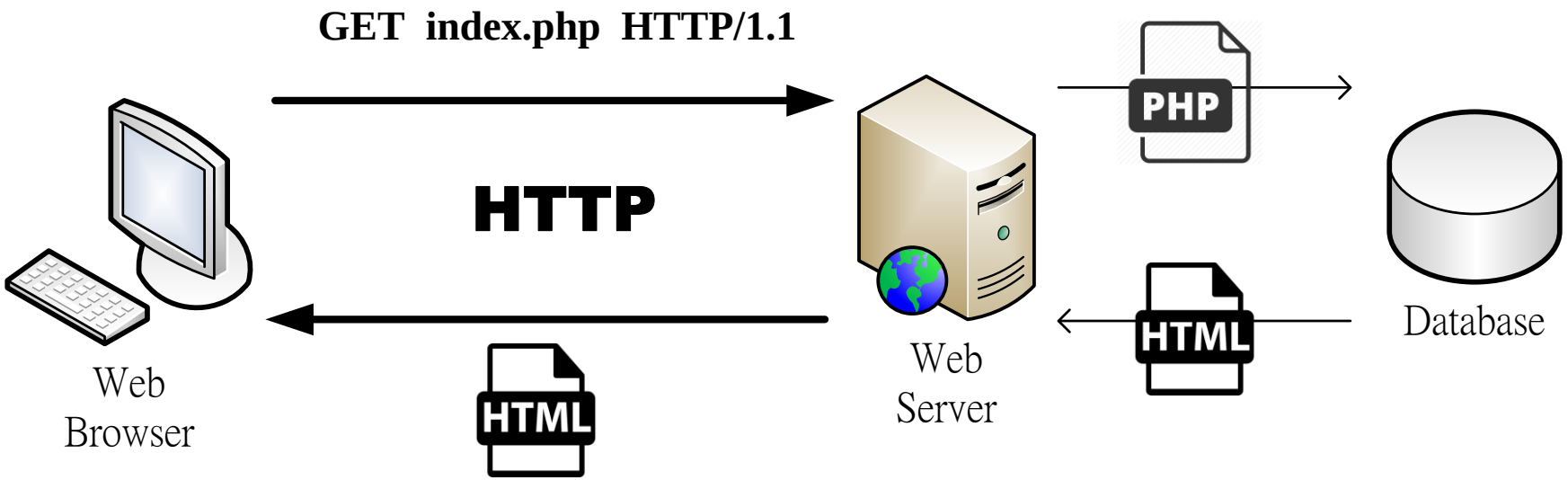
靜態網頁(Static Webpage)

- 網頁已以檔案形式固定儲存於伺服器上



動態網頁(Dynamic Webpage)

- 網頁內容是動態產生，可能是依照使用者請求(Request)或查詢資料庫(Database)即時轉成HTML形式。

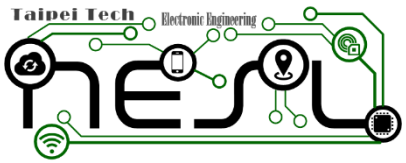


傳統動態網頁技術

- PHP
  - [Linux] Apache + PHP
  - http://www.php.net/
- JSP
  - [Linux] Tomcat
  - http://tomcat.apache.org/
- ASP/ASP.NET
  - [Windows] IIS
  - https://www.asp.net/

THE APACHE SOFTWARE FOUNDATION

php



傳統動態網頁技術

PHP示範

```
<!DOCTYPE html>
<html>
<body>

<?php
echo “<p>My first PHP
script!</p>";
?>

</body>
</html>
```

伺服器端

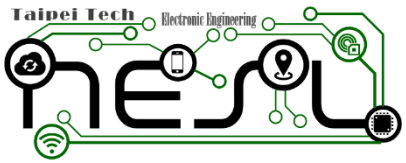
```
<!DOCTYPE html>
<html>
<body>

<p>My first PHP script!</p>

</body>
</html>
```

瀏覽器端

- 更多細節請參考
  - https://www.w3schools.com/php/default.asp
  - http://php.net/manual/en/

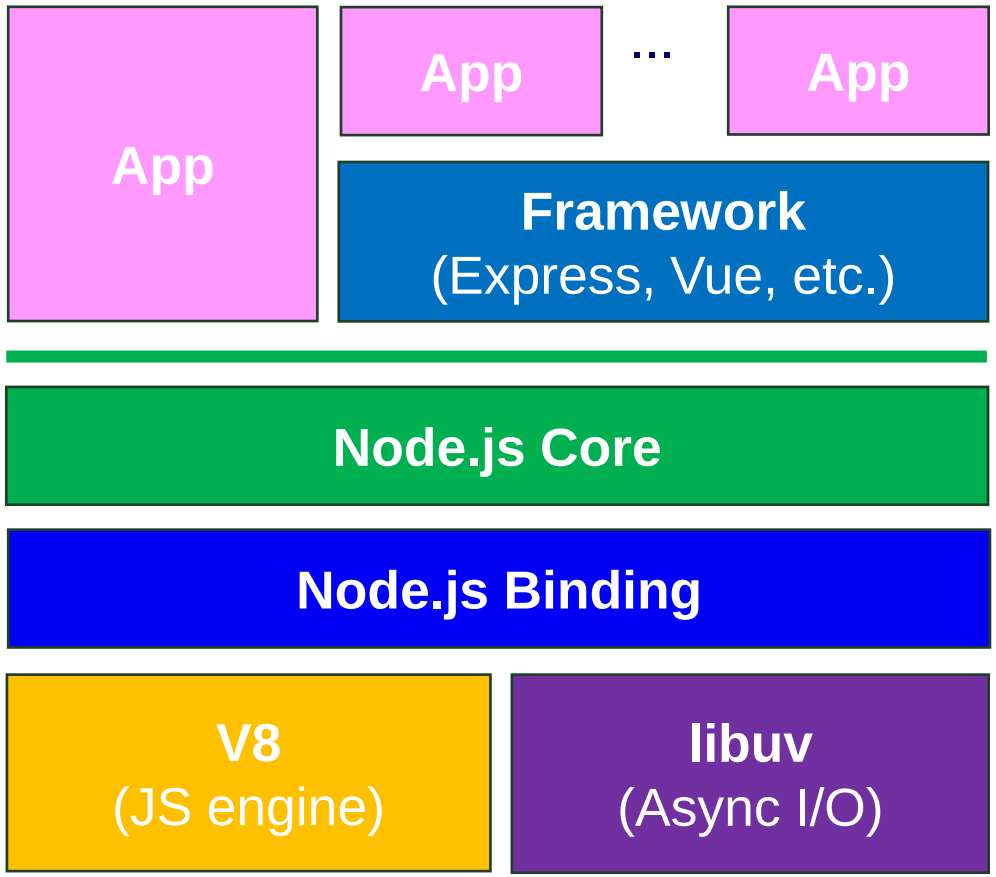
網頁框架(Web Framework)

- Node.js + Express
  - https://nodejs.org/
  - https://expressjs.com/
- Python + Django
  - https://www.python.org/
  - https://www.djangoproject.com/
- Ruby + Rails
  - https://www.ruby-lang.org/
  - https://rubyonrails.org/



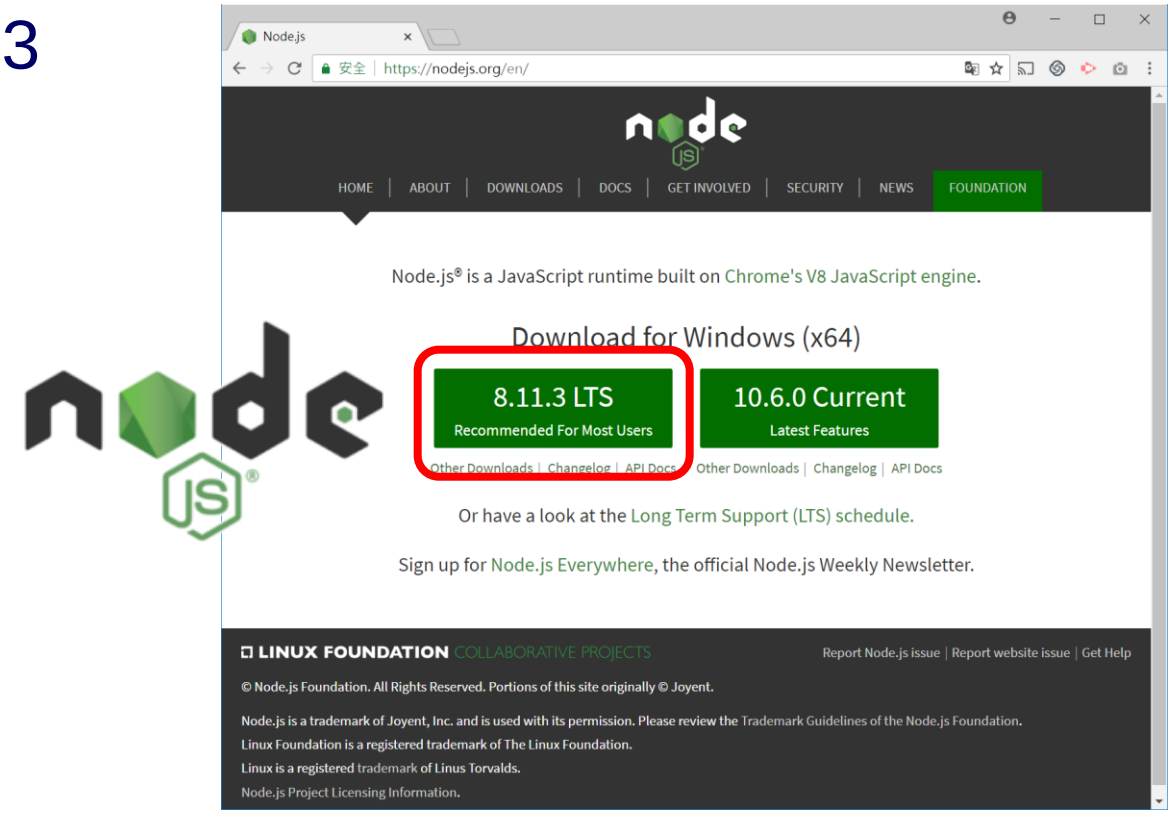
Node.JS

- 官方文件公布的內部組成

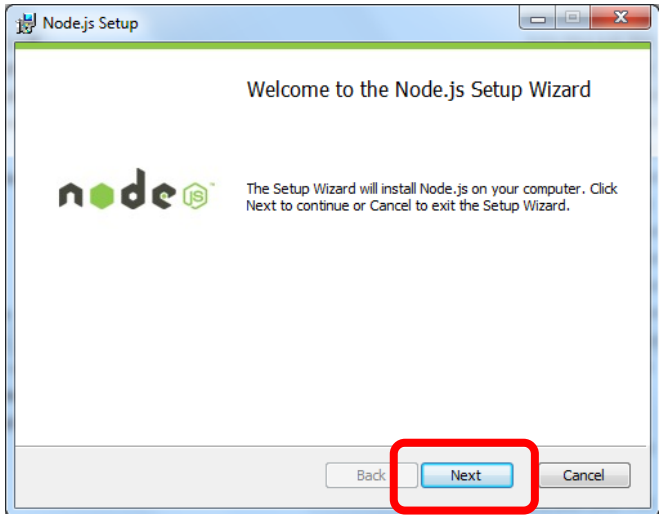


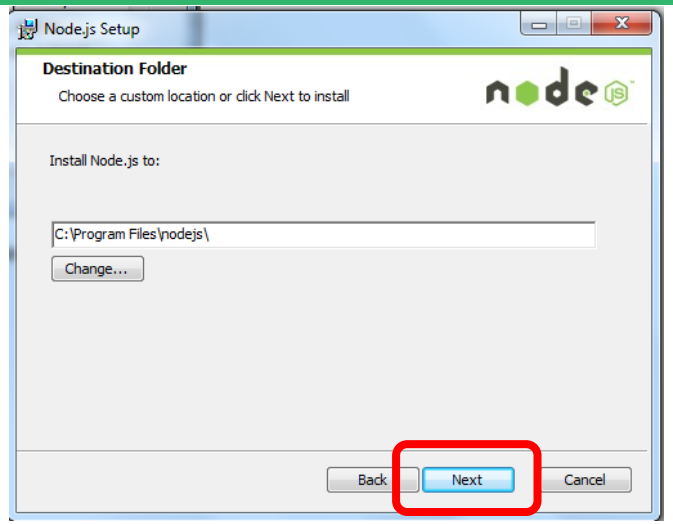
Node.JS

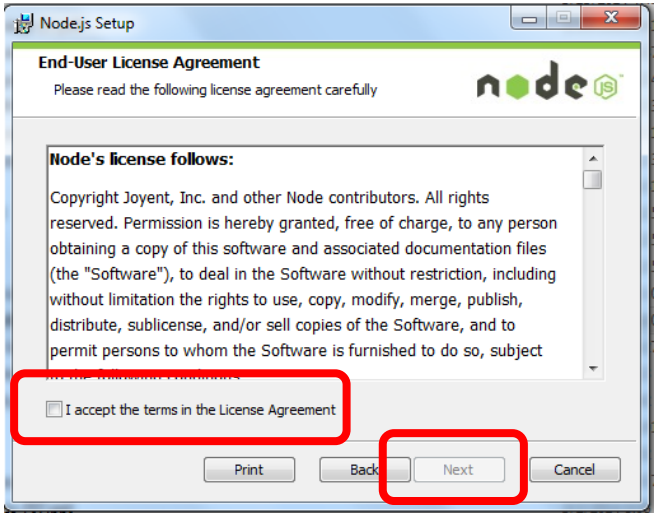
- Node.JS官方網站(https://nodejs.org/en/)下載
  - 目前長期支援版本(Long Term Support，LTS) v10.16.3

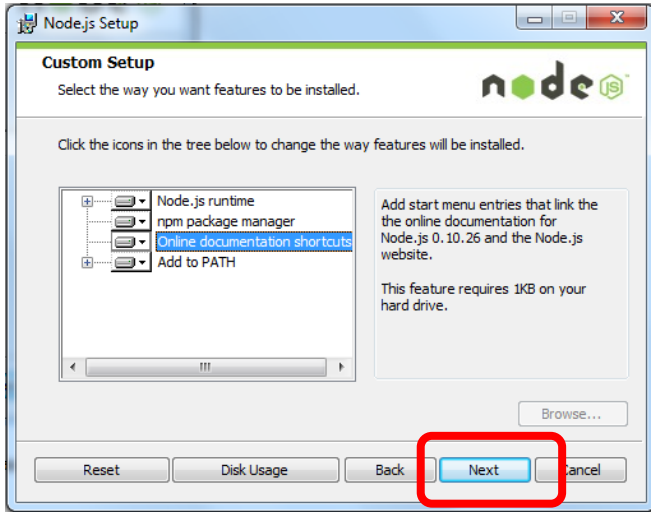


Node.JS軟體-安裝&設定

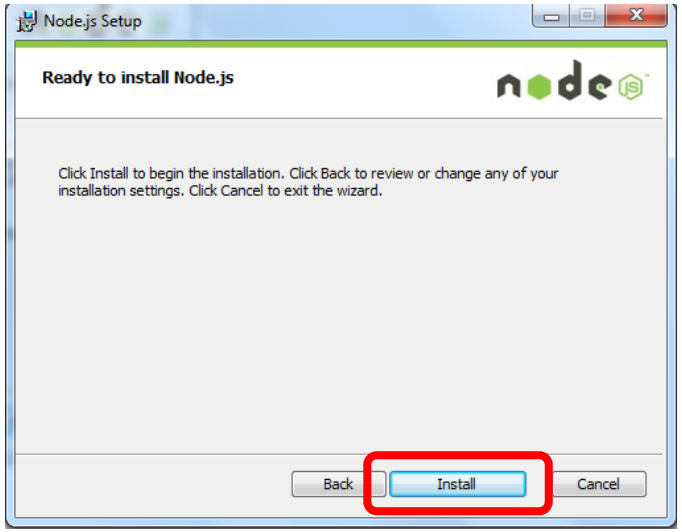


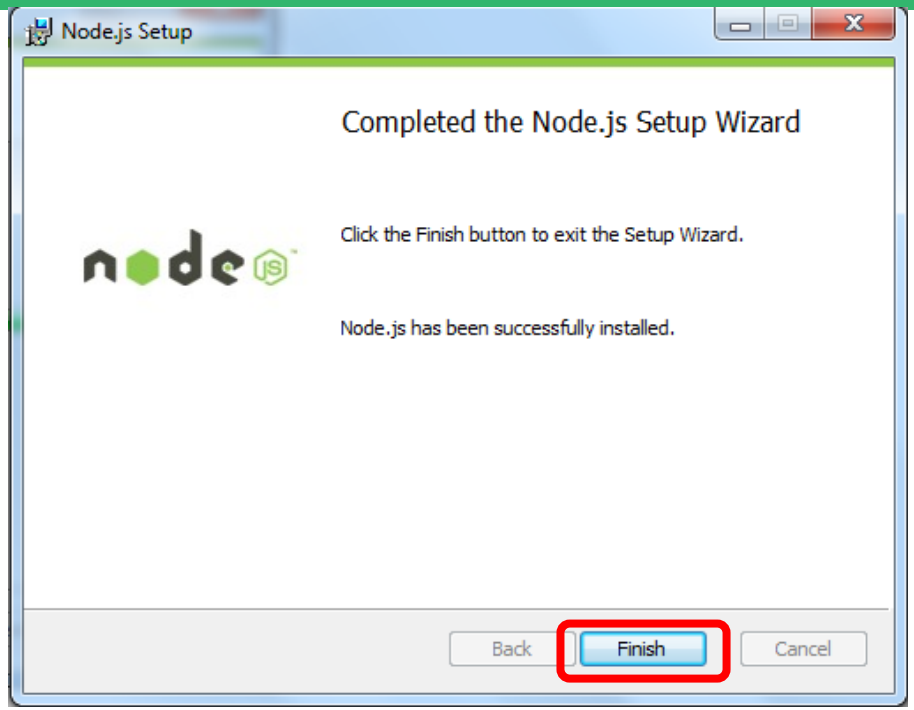


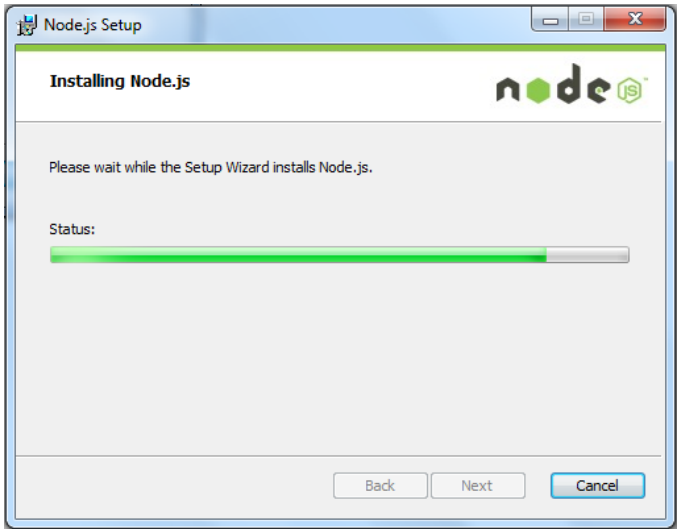


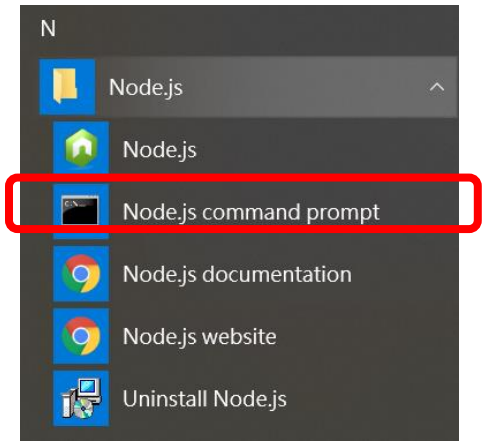


Node.JS軟體-安裝&設定





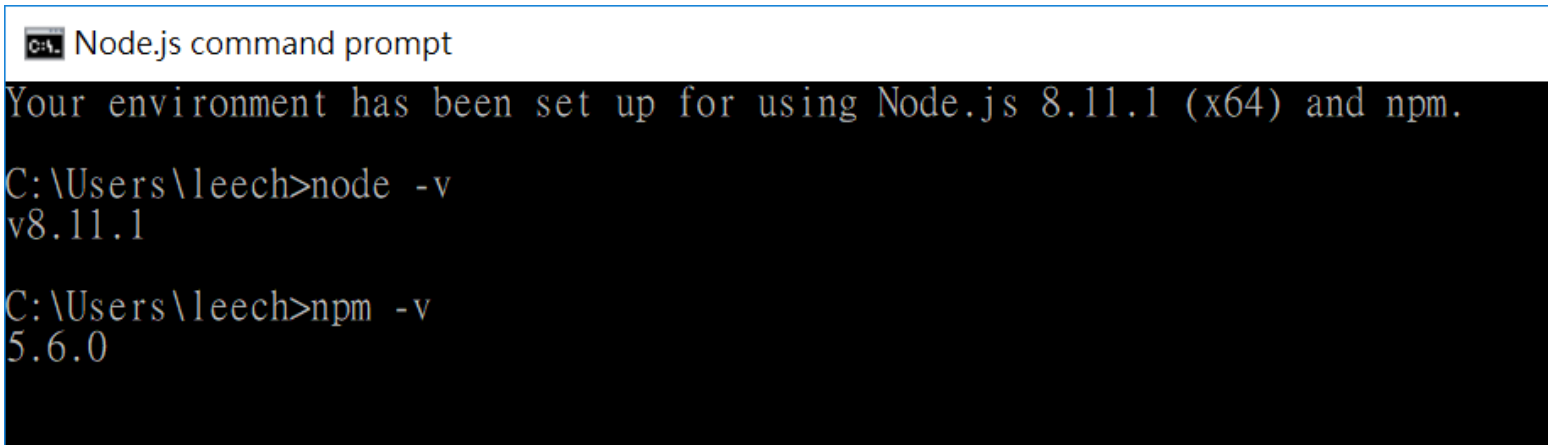




【開始】選單

Node.JS軟體-安裝&設定

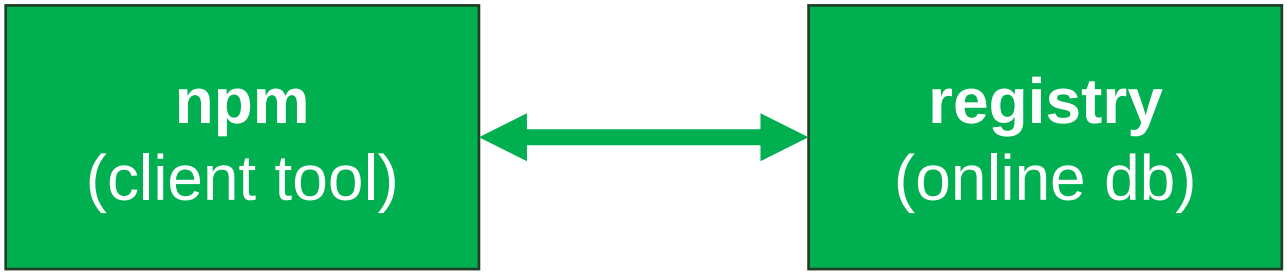
- 確認安裝成功
  - 顯示Node.JS & NPM安裝版本
    - 指令：node –v
    - 指令：npm -v



Node.JS & NPM

Node Package Manager (NPM)

- 官方提供有預設的套件管理程式
  - 使用者端，可以透過npm指令，進行套件管理、安裝
  - 一旦npm指令被執行後，將會觸發連接到網際網路中的線上資料庫(即registry)，尋找所需套件，並自動下載後安裝在使用者端



Node.JS & NPM

- 使用NPM管理套件，在目錄中會有其特殊的結構與檔案

/node_modules

- 負責儲存下載並安裝的套件

package.json

- 紀錄npm在此目錄管理的所有資訊，像是下載哪些套件、套件版本、相依關係等

Node.JS & NPM

- 常見的NPM指令

npm init

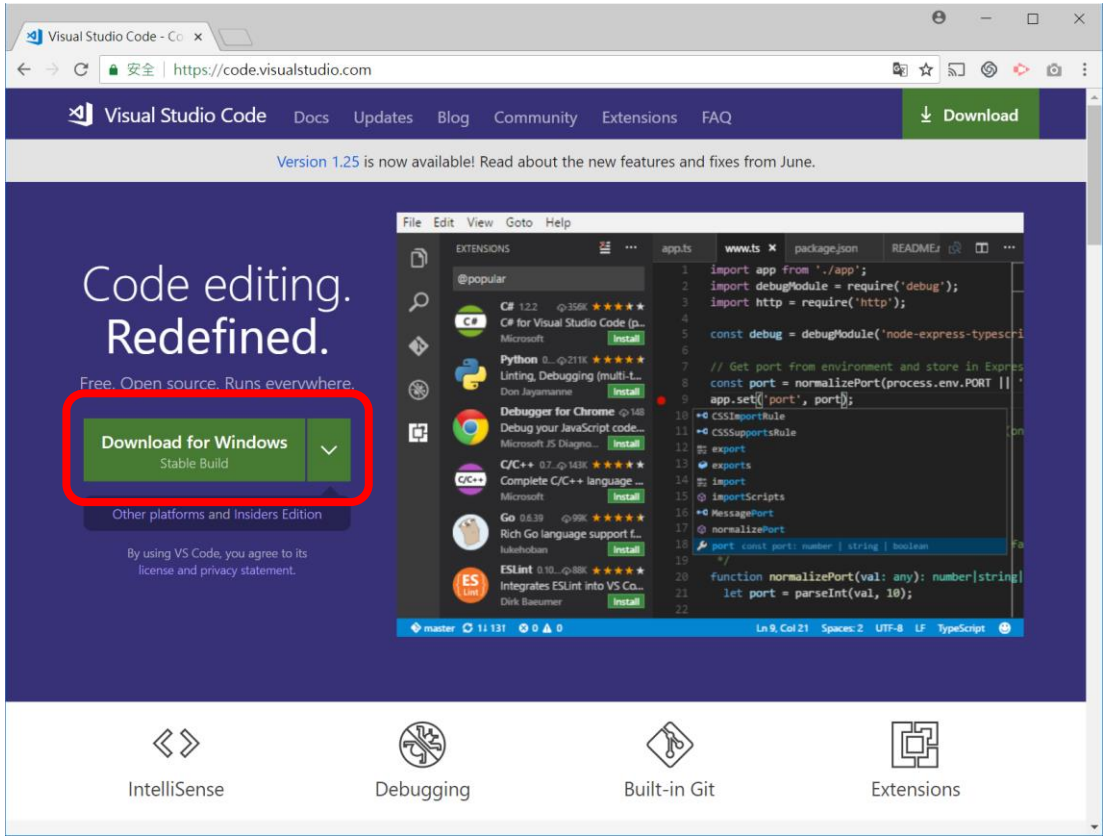
- 初始化目錄
- 產生package.json

npm install <name>

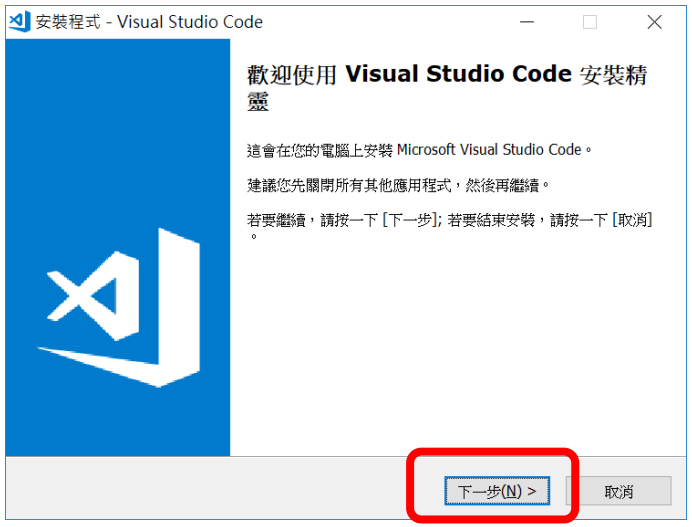
- 依照後面加上的套件名稱，到網際網路中的registry查詢，若有匹配的套件，就會自動下載並安裝

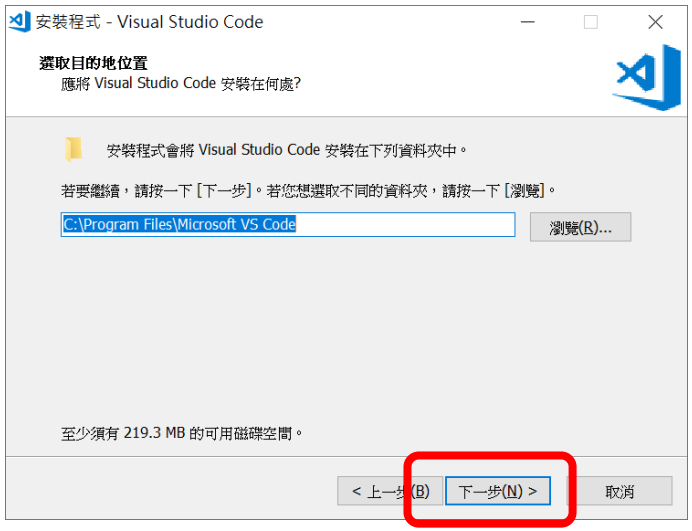
JS程式碼編輯器-開發介面

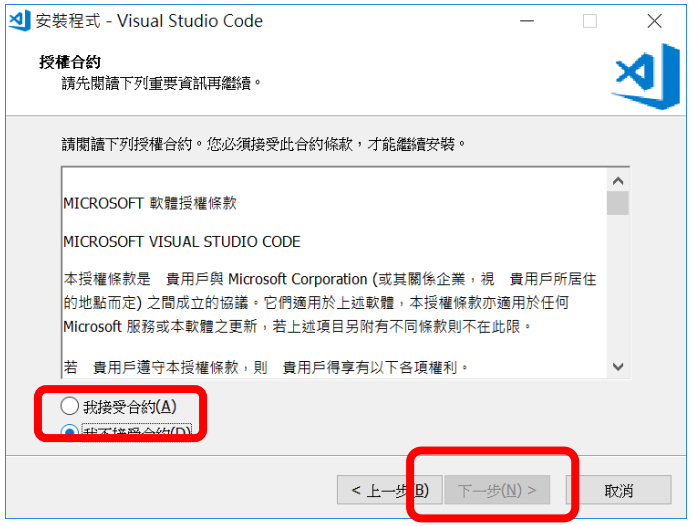
- Visual Studio Code (VS Code) (https://code.visualstudio.com/)

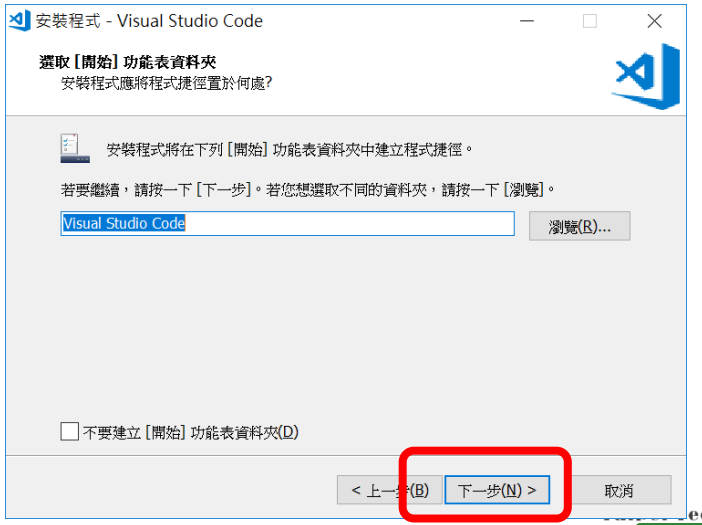


JS程式碼編輯器-開發介面

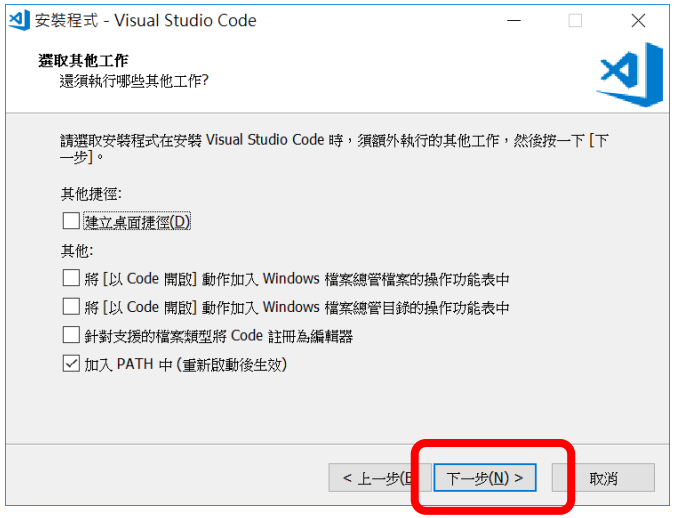


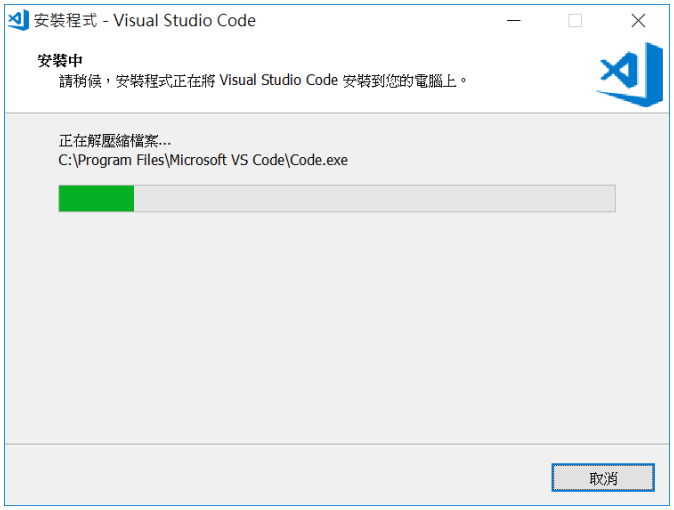


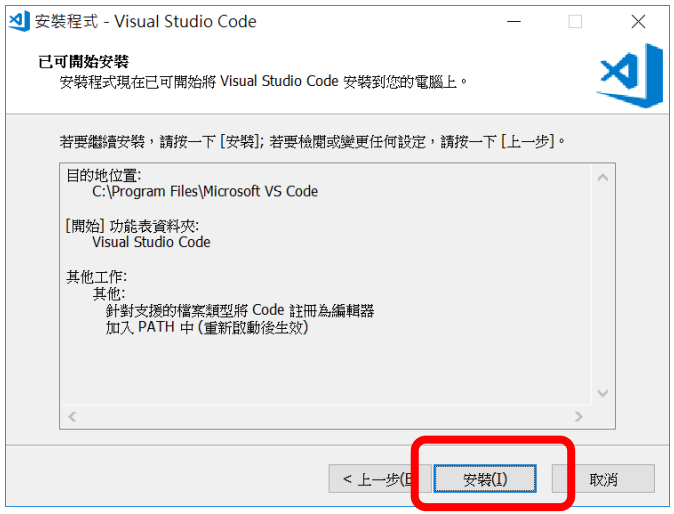


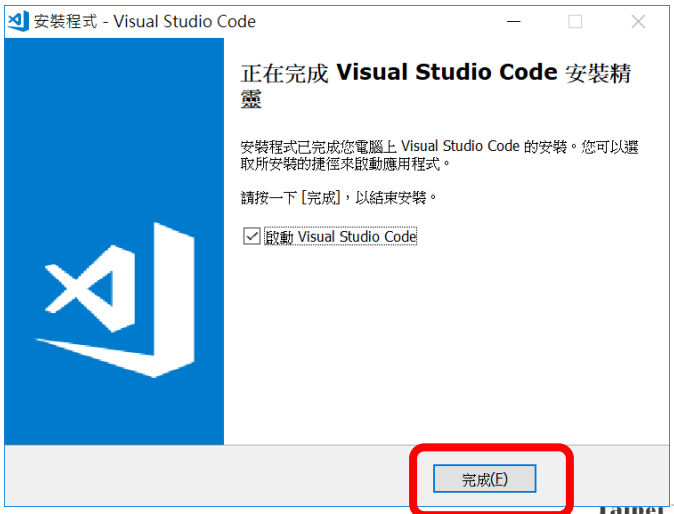


JS程式碼編輯器-開發介面

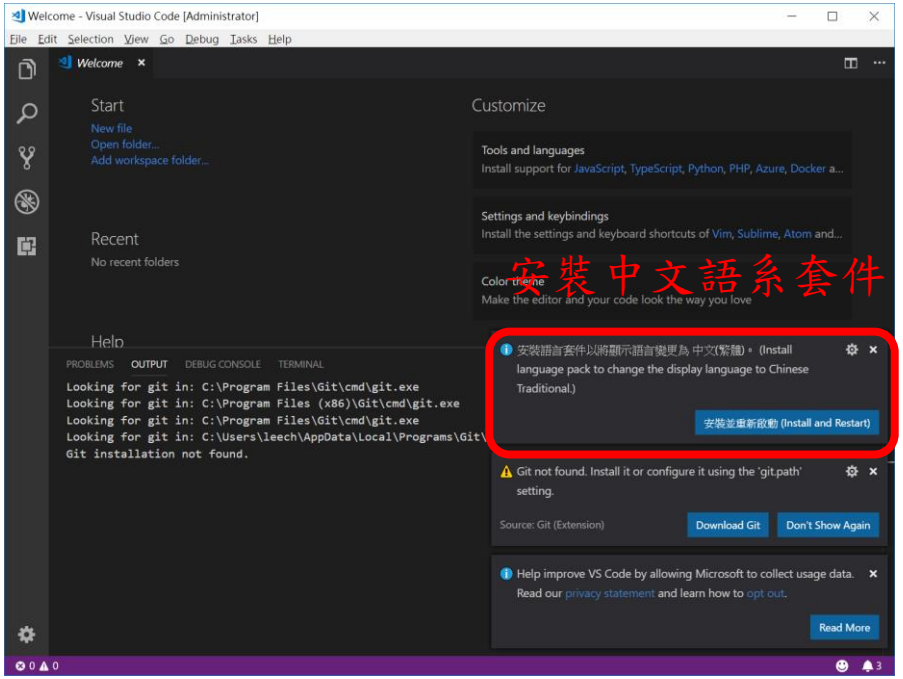


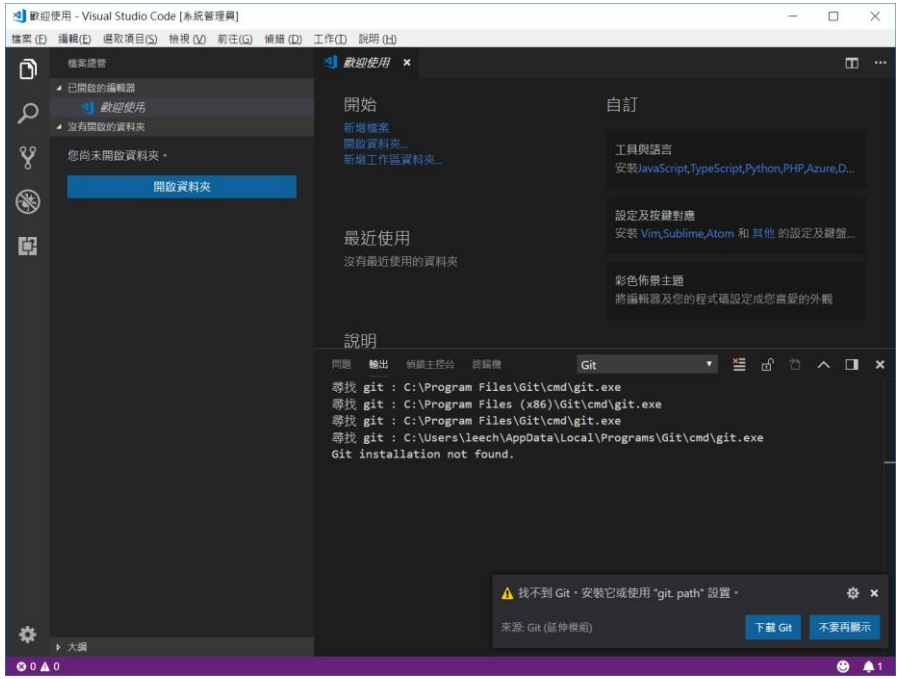






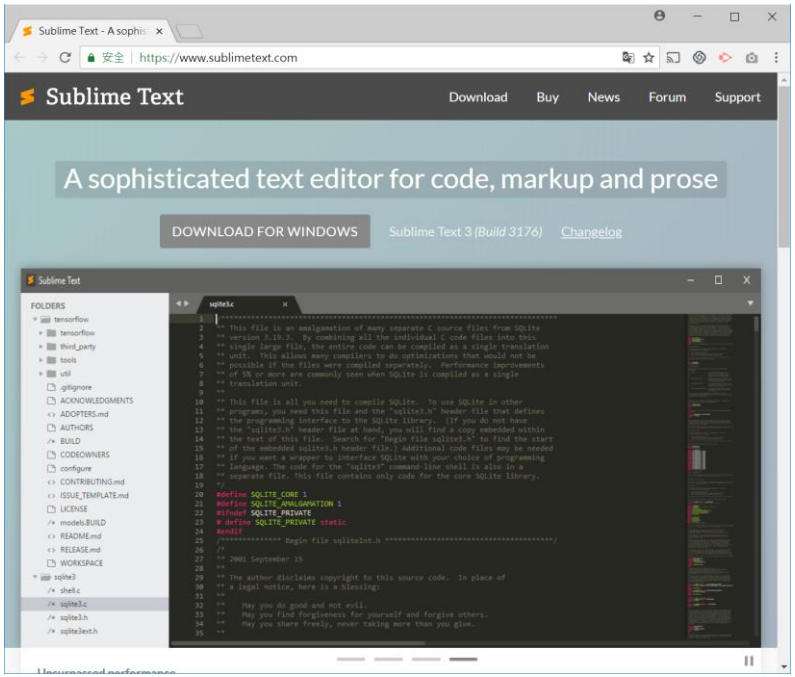
JS程式碼編輯器-開發介面



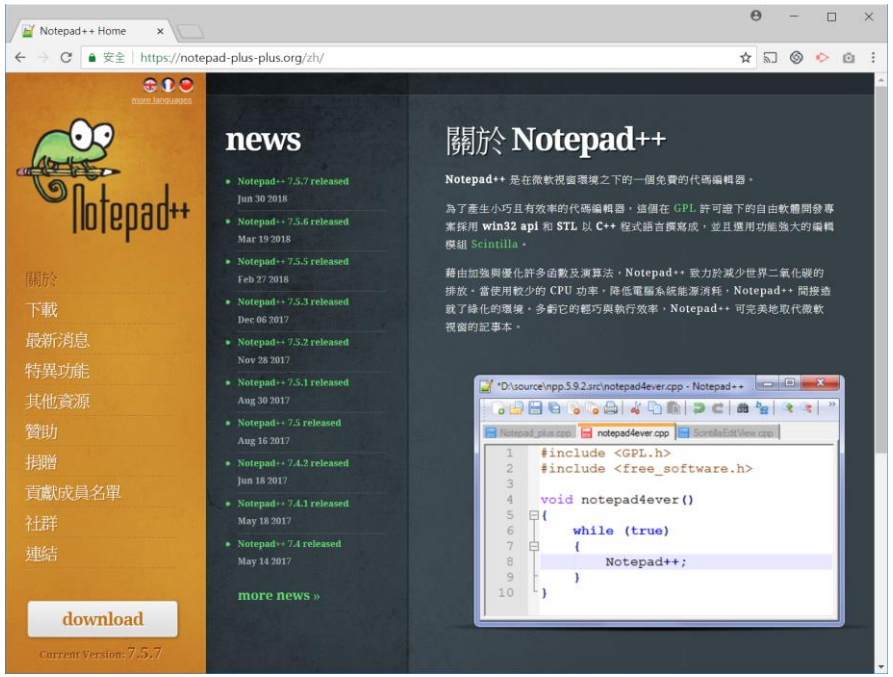


JS程式碼編輯器-其他選擇

- Sublime Text (https://www.sublimetext.com/)

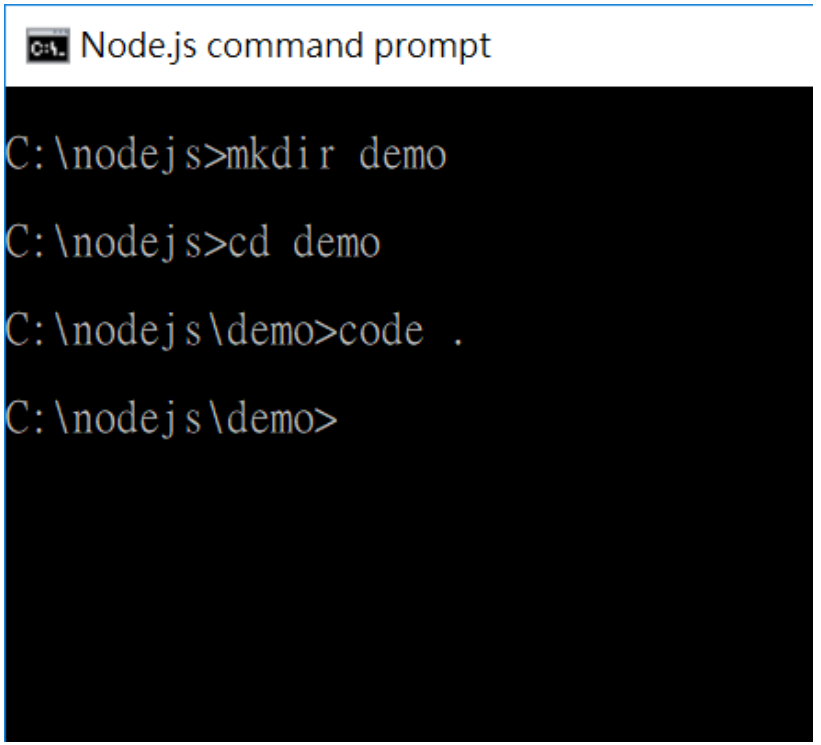


- Nodepad++ (https://notepad-plus-plus.org/zh/)



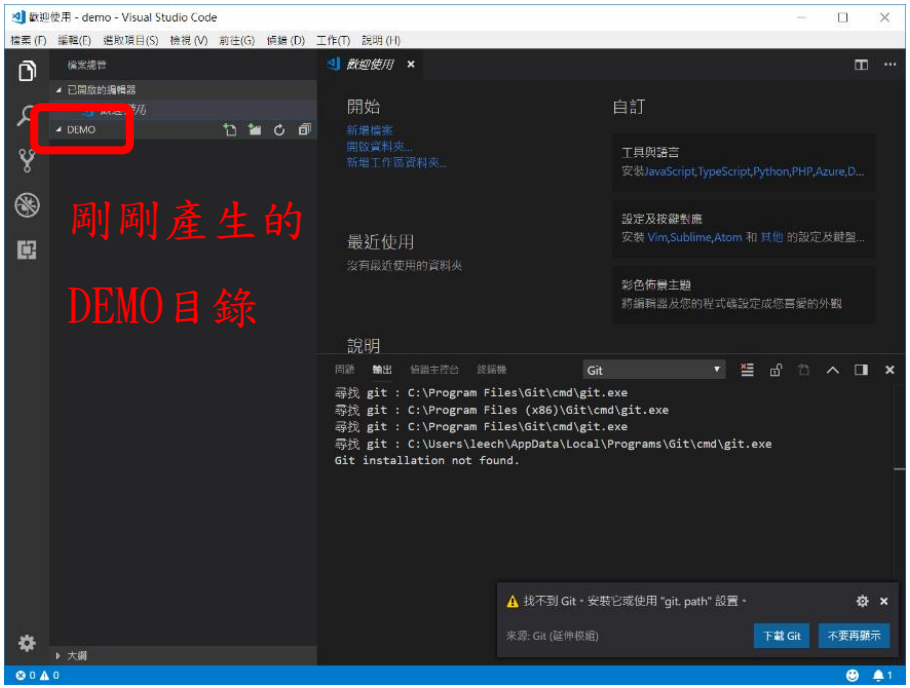
Node.JS-操作範例

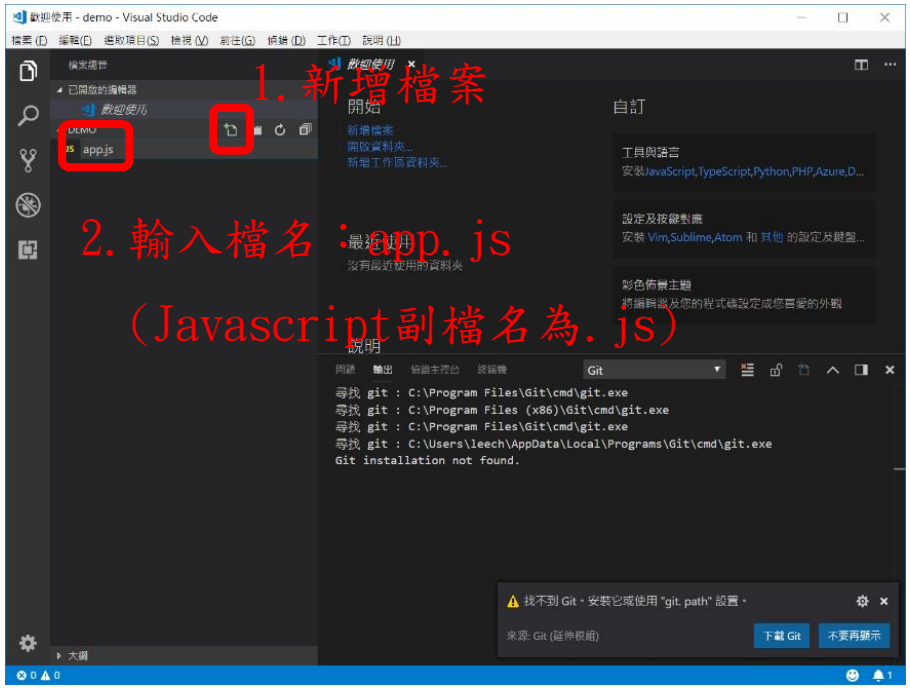
- 使用指令
  - 新增目錄：mkdir <目錄名>
  - 進入目錄：cd <目錄名>
  - 開啟VS Code：code .
    - “.”代表在此目錄底下打開



Node.JS-操作範例

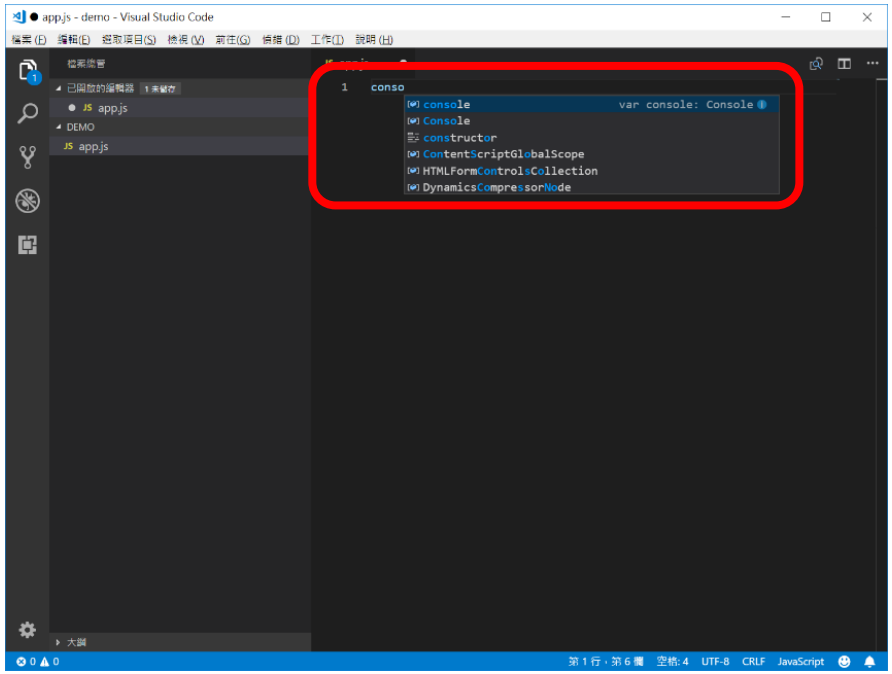
- 使用VS Code新增並編輯Javascript程式

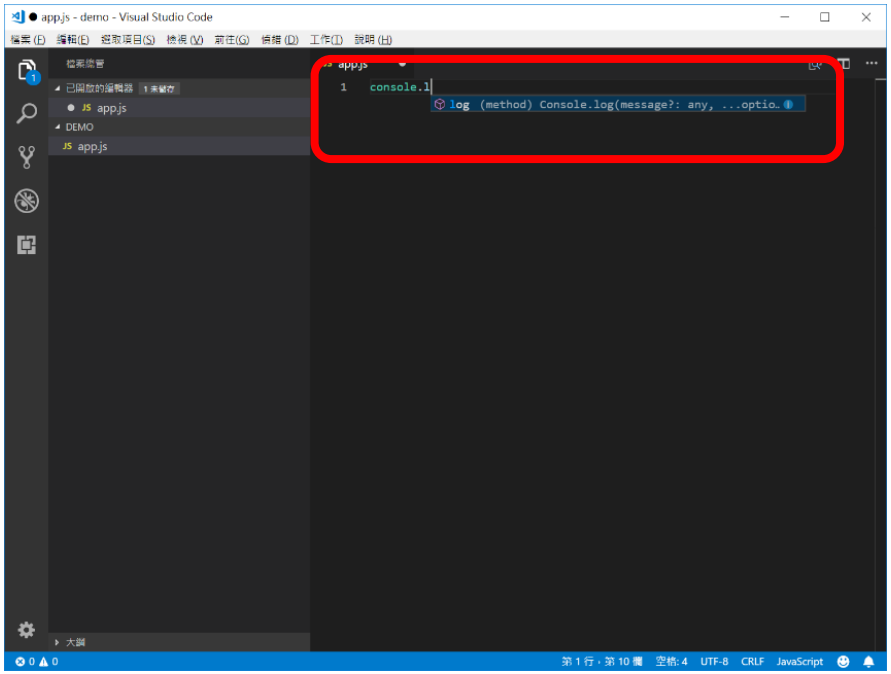




Node.JS-操作範例

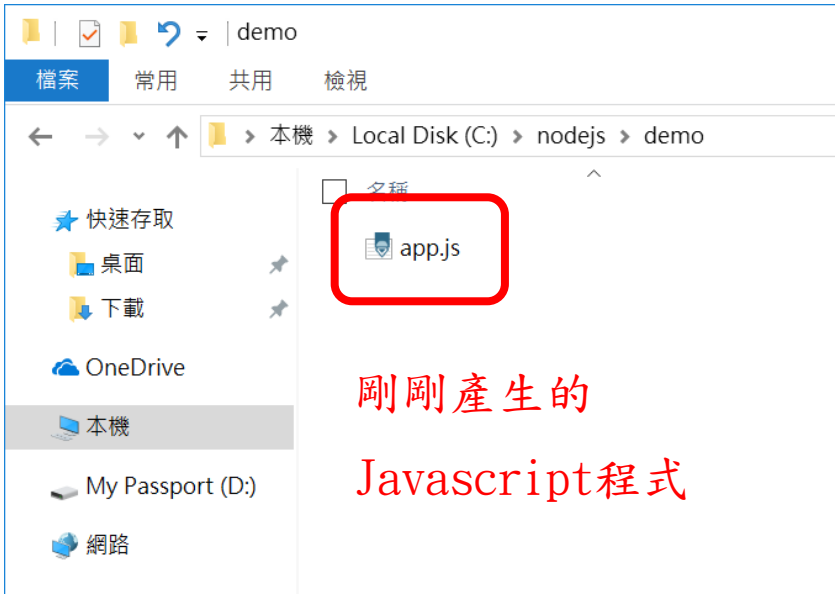
- VS Code支援智慧型輸入(提示相關保留字與關鍵字)
  - console.log(“Hello World!”);

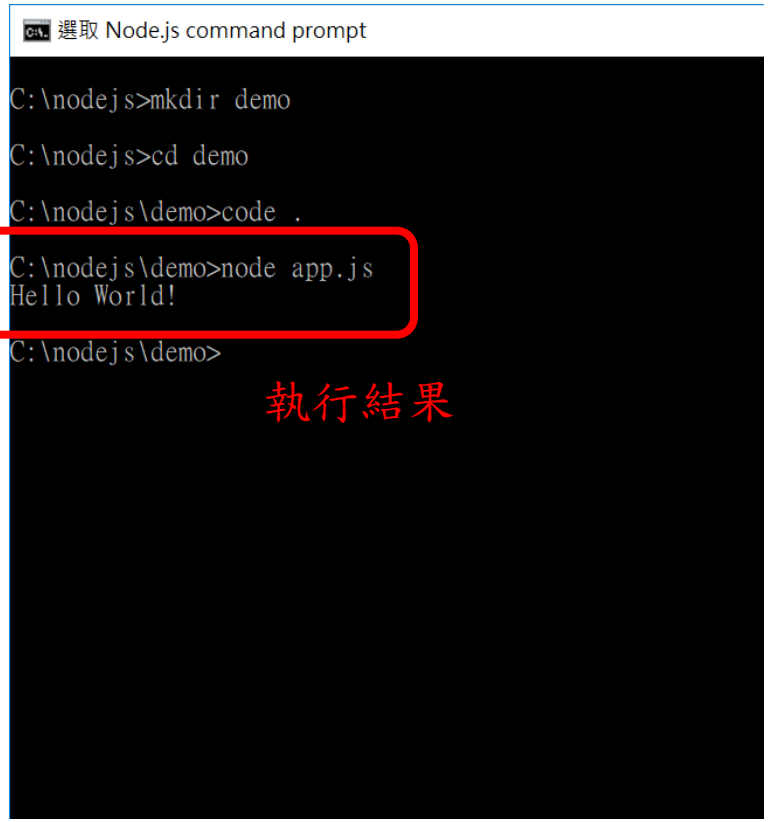




Node.JS-操作範例

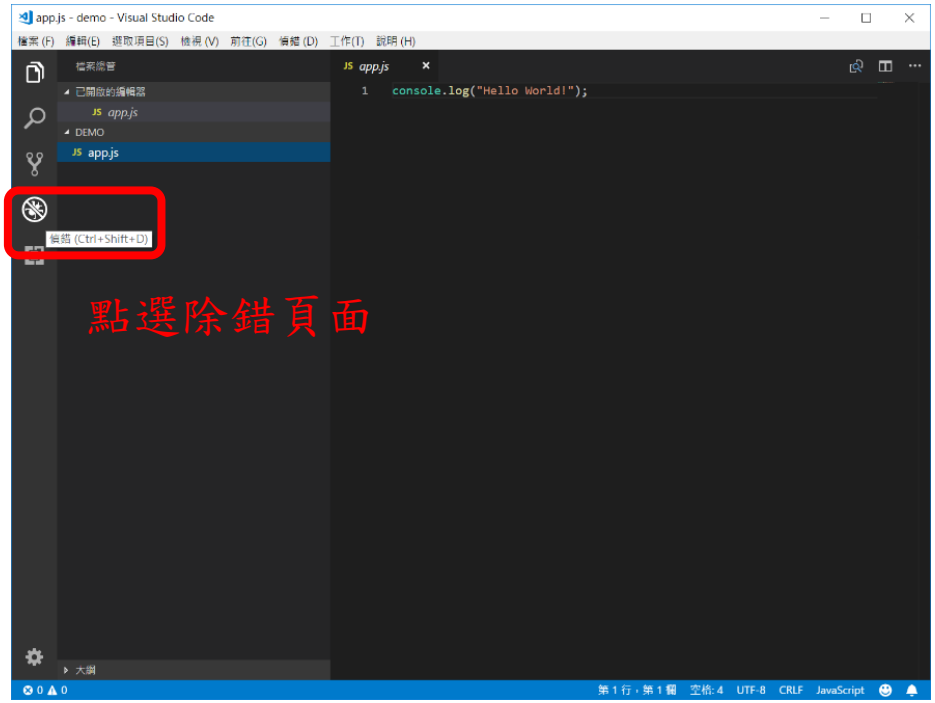
- 以指令執行剛剛撰寫的程式碼
  - node <檔名>

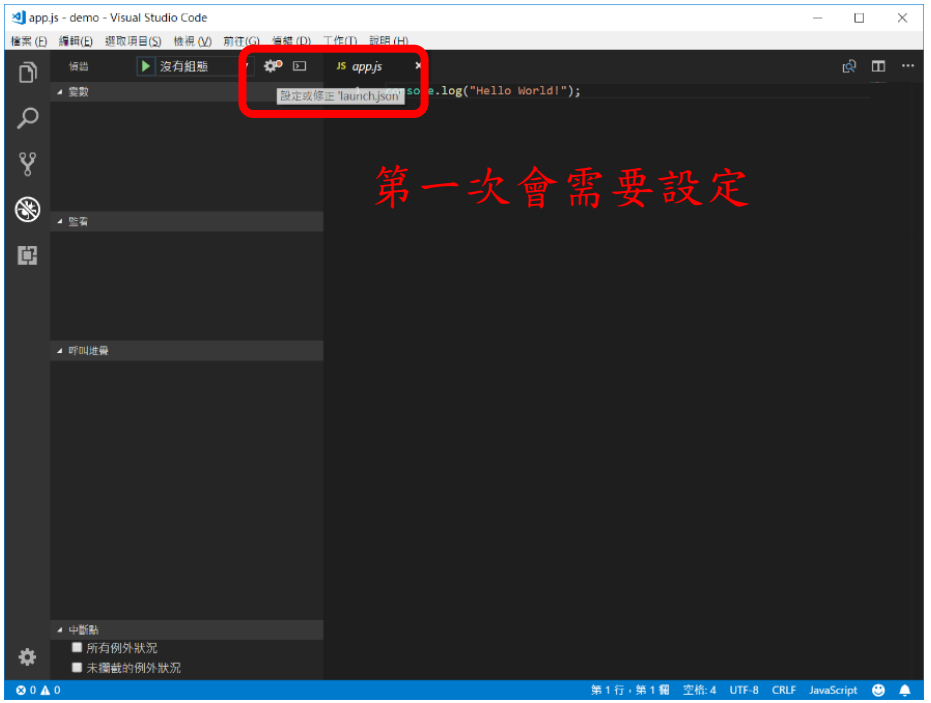




Node.JS-操作範例

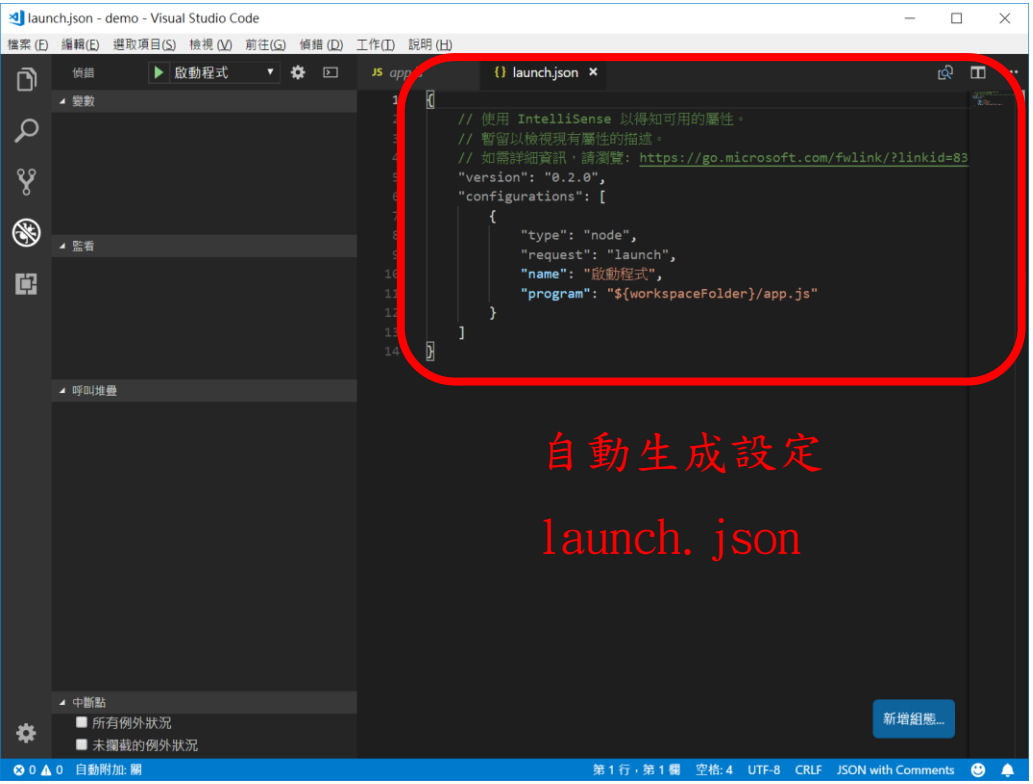
- 以VS Code執行剛剛撰寫的程式碼

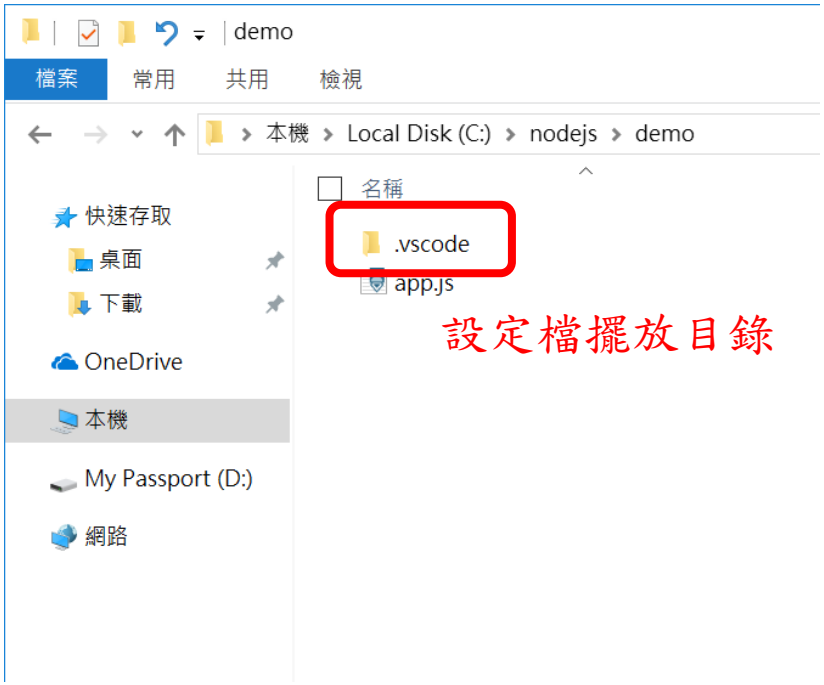




Node.JS-操作範例

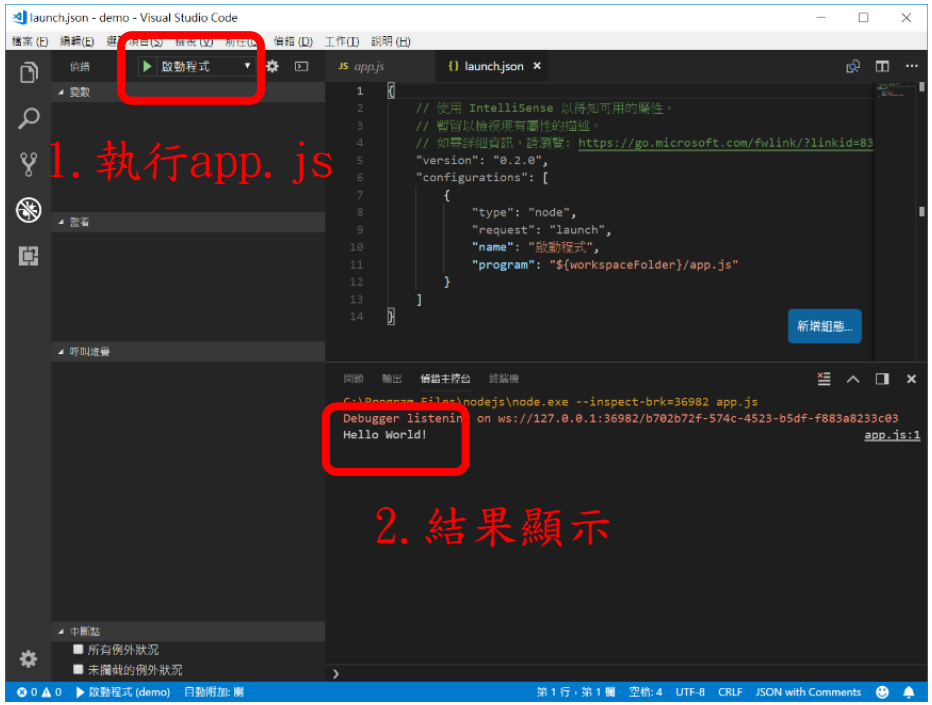
- 以VS Code執行剛剛撰寫的程式碼

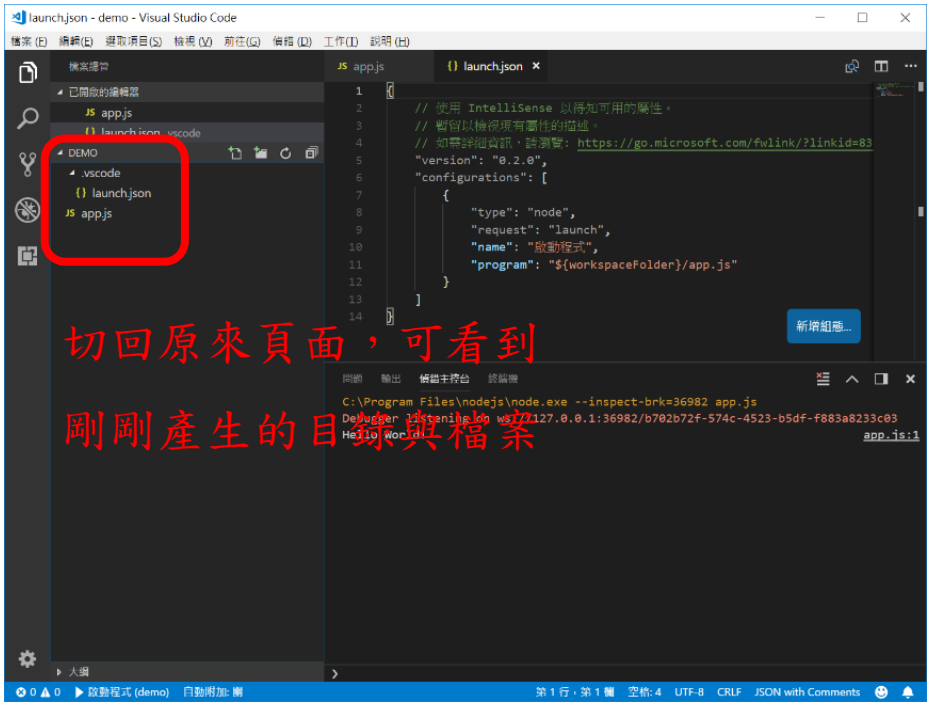




Node.JS-操作範例

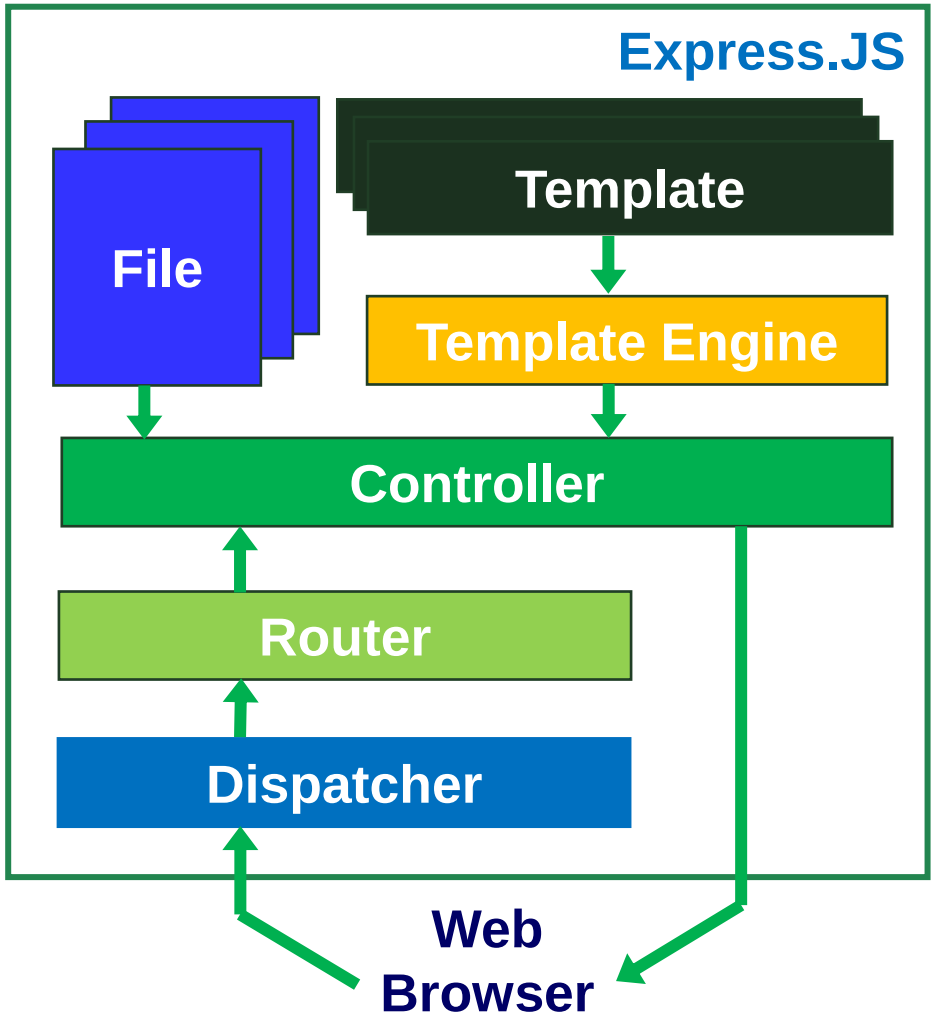
- 以VS Code執行剛剛撰寫的程式碼





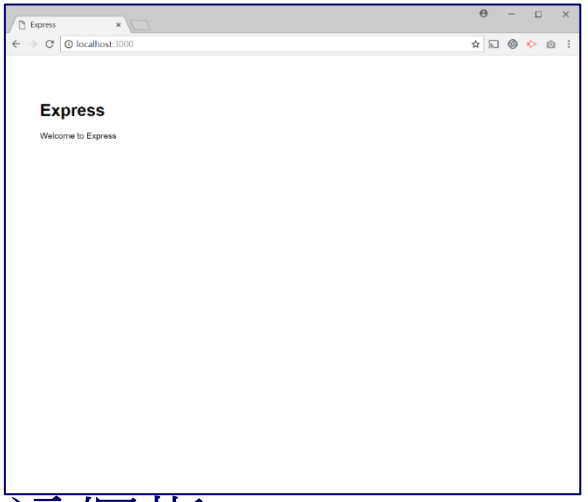
Express

- Express是最小又靈活的Node.js Web應用程式架構，為Web與行動式應用程式提供一組健全的特性。
  - http://expressjs.com/
- 大量的HTTP公用程式方法與中介軟體供您支配，能夠快速又輕鬆的建立完整的API。



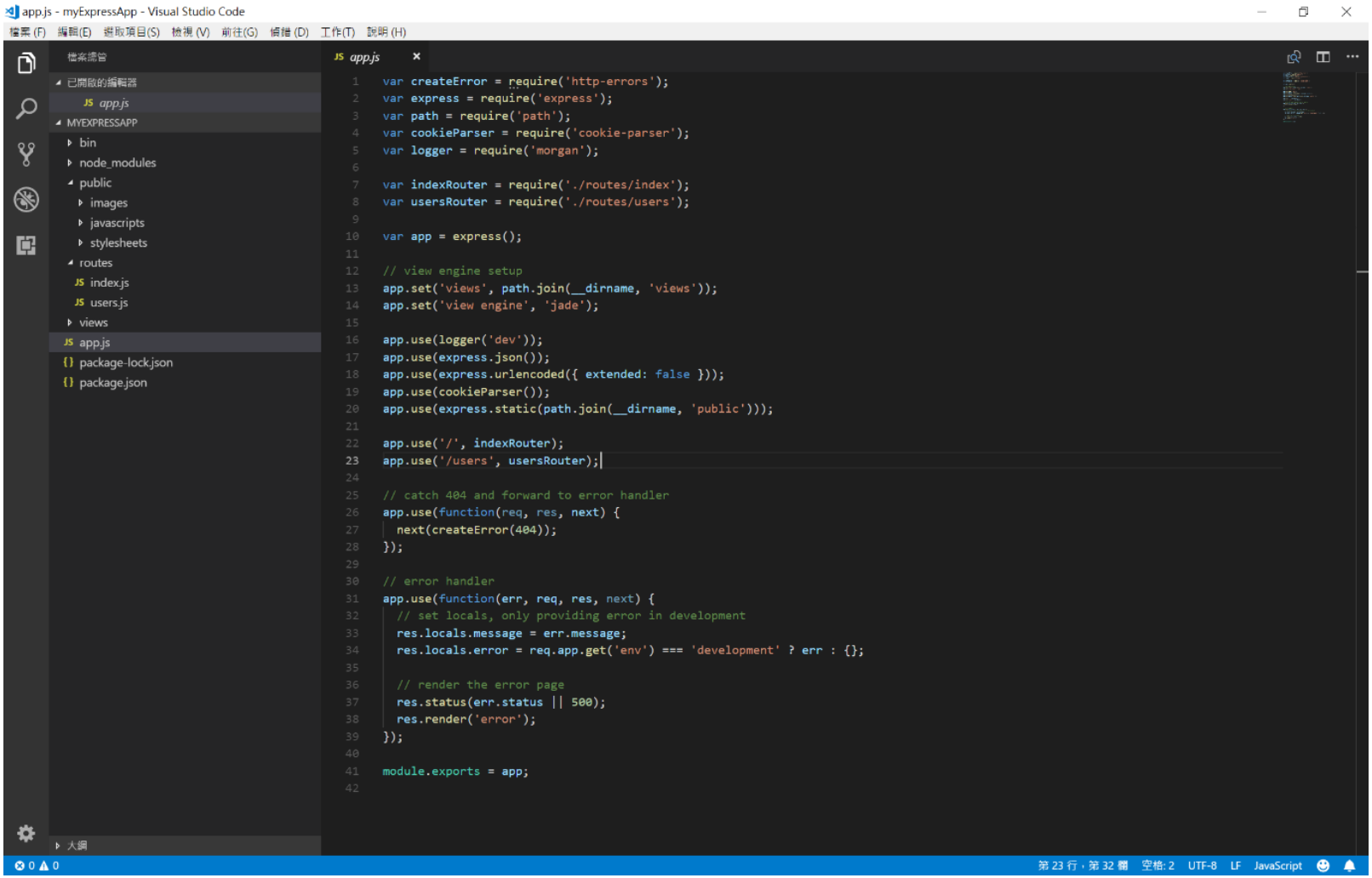
Express-Generator

- Express提供快速建立應用程式架構的工具



1. npm install express-generator -g
2. express myExpressApp
3. cd myExpressApp
4. npm install
5. npm start　(若以上都安裝了，以後只需要在VScode terminal 打這個指令就可以啟動)
6. 用瀏覽器打開http://localhost:3000
7. Ctrl+C可以中斷指令執行

Express-範例



Express-範例

- 基礎路由(Basic Routing)
  - http://expressjs.com/en/starter/basic-routing.html
  - 基礎語法：app.METHOD(PATH, HANDLER)
    - app是express的實例(instance)。
    - METHOD是HTTP要求方法，如：get、post、put、delete。
    - PATH是伺服器上的路徑。
    - HANDLER是當路由相符時要執行的函數。
  - 示範：

```
app.get('/', function (req, res)
{
  res.send('Hello World!');
});
```

Express-範例

- 路由器層次中介軟體(Router-level Middleware)
  - http://expressjs.com/en/guide/using-middleware.html
  - Express中介軟體(Middleware)
    - 代表有權存取(1)要求物件(req)、(2)回應物件(res)以及(3)呼叫下一個中介軟體函數的函數。

```
var app = express();
var router = express.Router();
// a middleware function with no mount path.
// This code is executed for every request to the router
router.use(function (req, res, next) {
      console.log('Time:', Date.now());
      next();
});
```

Express-範例

- 補充：Node.JS File System
  - https://www.w3schools.com/nodejs/nodejs_filesystem.asp
  - 使用內建的fs模組

```
var fs = require('fs');
```

  - 讀檔

```
fs.readFile(path, callback-func);
```

  - 寫檔

```
fs.writeFile(path, data, callback-func);
```

Express練習 (Checkpoint 1)

- 請修正Express範例達成以下要求
  1. 請建立http://localhost:3000/query可讀取特定檔案之內容
  2. 請建立http://localhost:3000/notify可寫入特定檔案之內容

Express-範例

- 靜態檔案(Static Files)
  - http://expressjs.com/en/starter/static-files.html
  - 使用內建函式：express.static(root, [options])
    - 示範：
```
app.use(express.static('public'));
```
    - 代表可以載入位於public目錄中的檔案
      - http://localhost:3000/images/kitten.jpg
      - http://localhost:3000/css/style.css
      - http://localhost:3000/js/app.js
      - http://localhost:3000/images/bg.png
      - http://localhost:3000/hello.html

Express練習 (Checkpoint 2)

- 請修正Express範例達成以下要求
  1. 請建立一HTML檔案，內容為”Hello World!”，並放入到正確位置，讓使用者可透過HTTP存取。
  2. 請建立一圖檔，並放入到正確位置，讓使用者可透過HTTP存取。

參數處理

- https://flaviocopes.com/express-request-parameters/
- 若使用GET，參數夾帶在URL中，則使用req.query物件

```
GET /test?name=fred&tel=0926xxx572

app.get('/test', function(req, res) {
        console.log(req.query.name);
        console.log(req.query.tel);
});
```

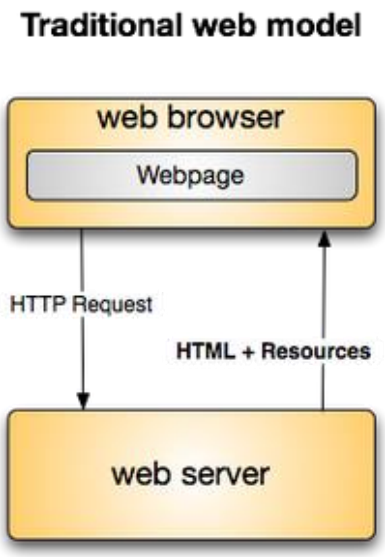
參數處理

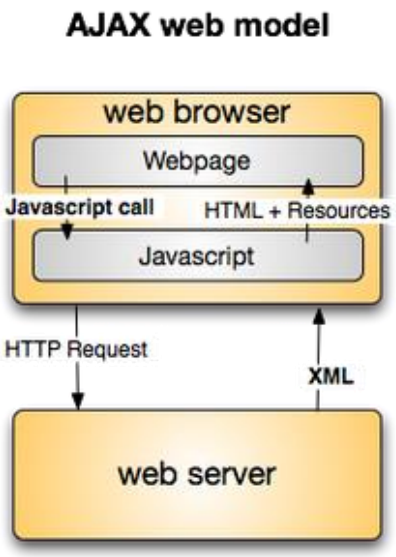
- https://flaviocopes.com/express-request-parameters/
- 若使用POST，參數夾帶在HTTP封包中，則使用req.body物件

```
<form action='/test' method='post’>
    <input type='text' name='name' value='fred’>
    <input type='text' name='tel' value='0926xxx572’>
    <input type='submit' value='Submit’>
</form>

app.post('/test', function(req, res) {
    console.log(req.query.id);
    console.log(req.body.name);
    console.log(req.body.tel);
});
```

Asynchronous JavaScript and XML (Ajax)

- 能在不更新整個頁面的前提下維護資料。
  - 使得Web應用程式更為迅捷地回應用戶動作
  - 避免在網路上傳送那些沒有改變的資訊





Ajax-範例

- Javascript使用XMLHttpRequest物件進行Ajax訊息交換
  - https://www.w3schools.com/js/js_ajax_intro.asp

- 產生要求(request)訊息
  - open(method, url, async)
  - send() → GET方法使用
  - send(string) → PUT方法使用

```
var xhttp = new XMLHttpRequest();
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Ajax-範例

- 處理回覆(response)訊息
  - onreadystatechange屬性：定義Callback函式
  - readyState屬性：XMLHttpRequest物件執行狀態
    - 4: request finished and response is ready
  - status屬性：HTTP回覆狀態碼
    - 200：OK、4XX：客戶端錯誤、5XX：伺服器端錯誤...等

```
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        document.getElementById("demo").innerHTML =
        this.responseText;
   }
};
```

Ajax練習 (Checkpoint 3)

- 請依照以下要求建立Web Service
  1. 請建立一HTML檔案，內含一表單，表單上有一個按鈕，可觸發Ajax事件至Express
  2. 請利用Express範例，自訂一Web API (http://localhost:3000/query)，可回傳一數值
  3. 請將Web API回傳之數值顯示於HTML頁面上

總結

- Web後端(Back-end)開發
  - Node.js + Express
    - 利用URL建構Web API
    - 靜態檔案存取
    - 動態回應
  - [未來]資料庫連接
    - 關聯式資料庫，如：MySQL、MS SQL Server…等
    - NoSQL資料庫，如：MongoDB、HBase…等
  - [未來]HTTP替代方案
    - HTTP/2.0、WebSocket、WebRTC、QUIC…等