微算機系統實習
MICROPROCESSOR SYSTEMS LAB.
SPRING, 2021

Instructor：Yen-Lin Chen(陳彥霖), Ph.D.
Professor
Dept. Computer Science and Information Engineering
National Taipei University of Technology

LECTURE4-
運用CGI技術進行物聯網遠端操控GPIO-COMMON GATEWAY INTERFACE(CGI)

OUTLINE

- 物聯網基本概念
- CGI簡介
- Nodejs簡介
- Nodejs的用途為何？
- npm簡介與安裝使用
- 簡易web server範例

物聯網基本概念

- 物聯網的概念最早出現在比爾蓋茨1995年「未來之路」一書，在書中，比爾蓋茲已經提及Internet of Things的概念
- 1998年，美國麻省理工大學首先提出了當時被稱作「EPC (Electronic Product Code)系統」的物聯網的構想
- 1999年，美國EPCglobal 的Auto-ID中心首先提出物聯網的概念，稱物聯網主要是建立在物品編碼、RFID技術和網際網路的基礎上，實施智慧化識別與管理

物聯網基本概念

- 2005年，ITU發佈了「**ITU網際網路報告2005：物聯網**」，綜合二者內容，正式提出物聯網的概念，包括了所有物品的聯網和應用，宣稱小至牙刷、大至洗衣機、汽車等都可以透過網路相連與通信
- 2009年，歐盟執委會發表了「**物聯網行動計畫**」，提出了包括隱私及個資保護、信任和安全、標準化、研究開發、開放和創新、制度意識、國際對話、污染監測管理及未來發展等I4項行動內容
- 2012年，特斯拉推出Model S時奠定了車連網的標準，並引入汽車軟體無線網路更新功能，整合成類似的物聯網技術

物聯網基本概念

- 2014年，Apple、Google等企業公司陸續發表智慧家庭應用等產品
- 2016年，宏碁、聯發科及研華等國內大廠，成立「**亞洲·矽谷物聯網產業大聯盟**」，推動「物聯網產業創新研發」及「健全創新創業生態系」
- 2025年，依據全球人工智慧研究報告，物聯網全球產值預計將高達11.1兆美元

物聯網基本概念

- 物聯網有許多相似的名稱，例如：大陸稱其為物聯網、台灣稱其為智慧聯網、Cisco公司稱其為Internet of Everything (IoE)、美國加州柏克萊大學稱其為Cyber-physical system(CPS)、IBM公司稱其為 Smarter Planet (2009年)等
- 物聯網的重點是參與連結之物品會適時產生資訊且都須具備有通訊連結能力，因此感測元件與網路通訊技術自然就成為了物聯網的核心，是一項跨領域之各種技術的整合應用

物聯網基本概念

- 物聯網 (The Internet of Things，IoT)定義
  - 顧名思義是指將眾多物體(各種裝置和設備)及其所產生的資訊都能經由各式網路相連結，透過現有的網際網路形成巨大的「**物物相連的網際網路**」，進行資訊收集、發佈、儲存、分析及開發各式可能之應用並進而創造商機
  - 藉由物聯網將使物體具備智慧，使人與人、人和物體、物體和物體之間可以隨時建立相互溝通與對話的環境

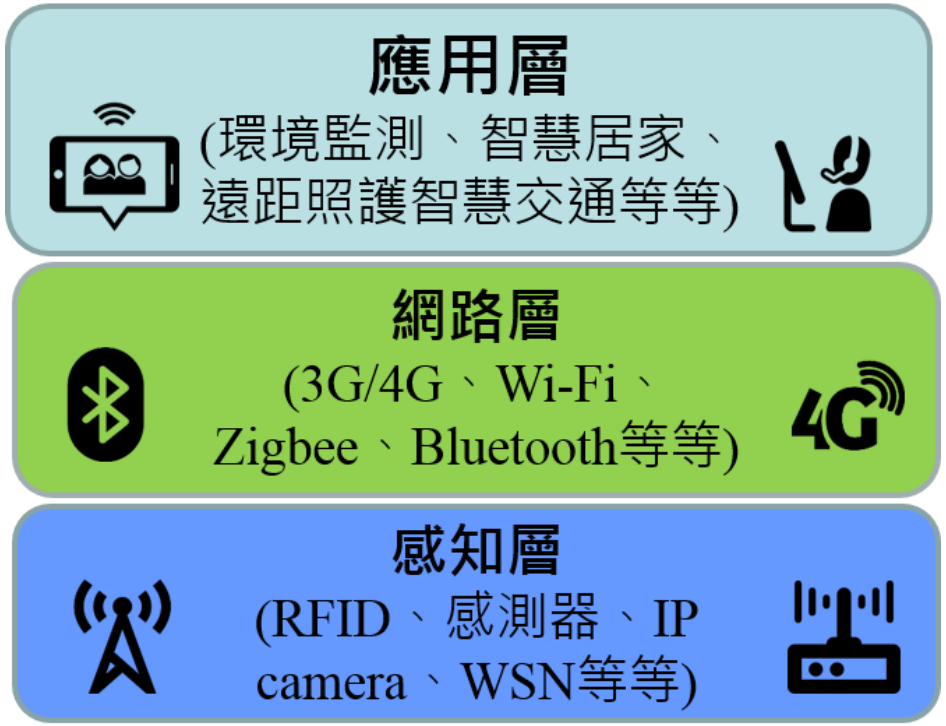
物聯網基本概念

- 協定層架構
  - 感知層 (Perception layer)
  - 網路層 (Network layer)
  - 應用層 (Application layer)
- 第一層為「感知層 (Perception layer)」，由各種資訊擷取、識別的感知元件所組成，包括RFID、標籤技術、感測技術、通訊技術以及人工智慧

物聯網基本概念

- 第二層為「網路層 (Network layer)」，即各類無線傳輸技術，包含無線通訊網路、感測網路、EPCglobal網路架構以及新一代網絡
- 第三層為「應用層 (Application layer)」，即物聯網的各種應用領域，例如：智慧監測、智慧城市、智慧醫療等
- 物聯網三層架構，如下圖所示

物聯網基本概念



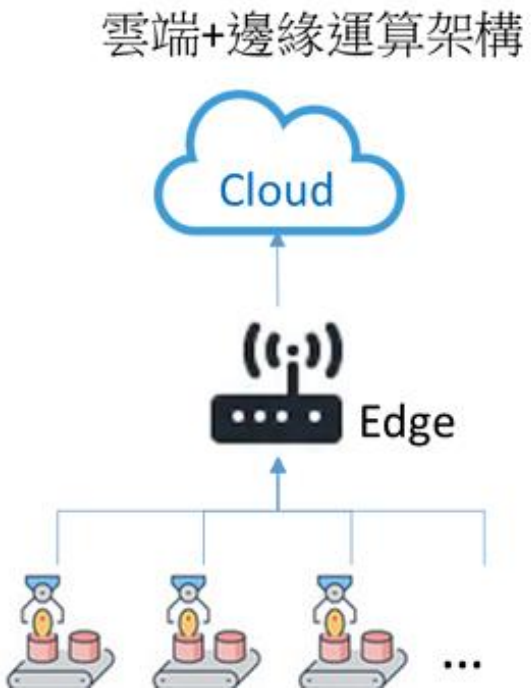
物聯網3層協定架構

物聯網基本概念

- 物聯網相關技術需求
  - 與 Smart Device 的互連
  - 雲端技術的提升
  - 操作簡便的自然人機介面
  - 良好的使用者體驗

邊緣運算(Edge Computing)

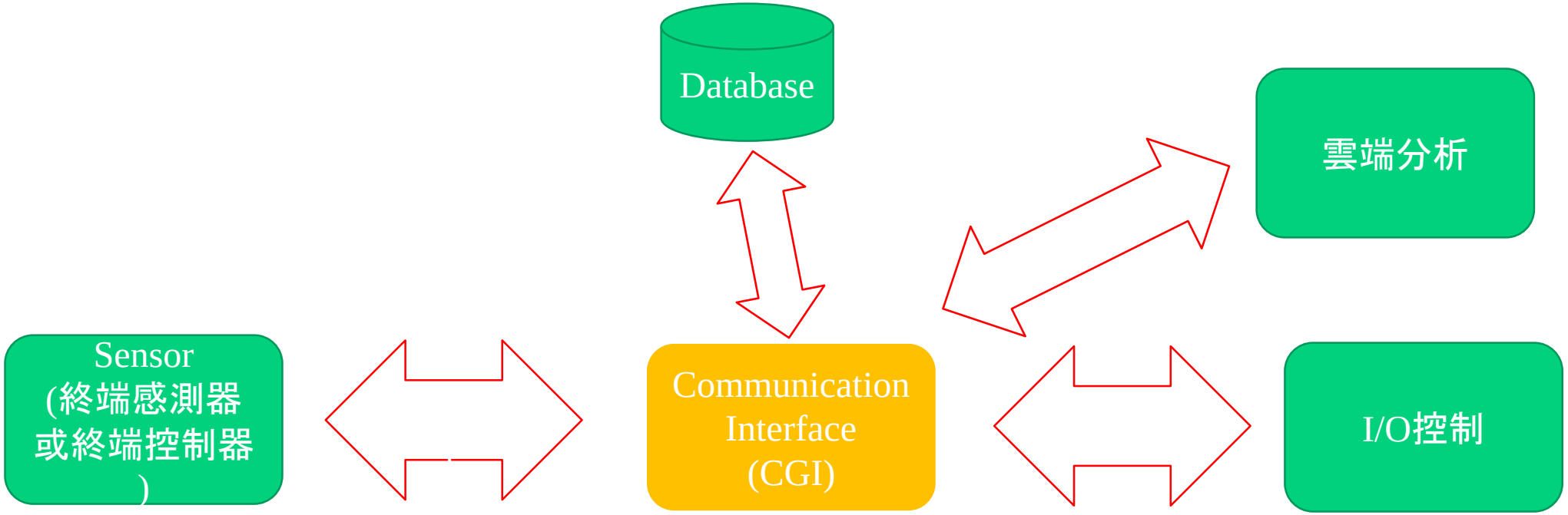
- 為何需要:
  - 網路邊緣的線上設備數量迅速增加，這些會產生大量數據資料(如:像城市的IoT設備)，同時也消耗大量資料頻寬
- 簡介
  - 邊緣運算是一種網路運算架構，運算過程盡可能靠近資料來源以減少延遲和頻寬使用，最大限度地減少異地用戶端和伺服器之間必須發生的通信量。近年來，技術的快速發展使硬體趨向小型化、高密度以及軟體的虛擬化，讓邊緣運算的實用度更加可行
  - 邊緣運算是加速互聯網的主要關鍵



各式感測器與基本原理

- 感測器(Sensor)是由對周圍環境或者人體身上的某些物理化學反應，轉成可量化的電訊號，常見的有溫度感應器、光度感應器、陀螺儀、加速計、影像感測器等、深度感測器.....等
- 大部分的感測器本身沒有辦法直接利用,需要放大訊號以便讀取，所以通常會接到微電腦控制器上面，這讓應用上有極大好處，無論是做有線無線傳輸訊號到遠端機器上，都提高了便利性與增加應用擴充性 ，如下圖所示

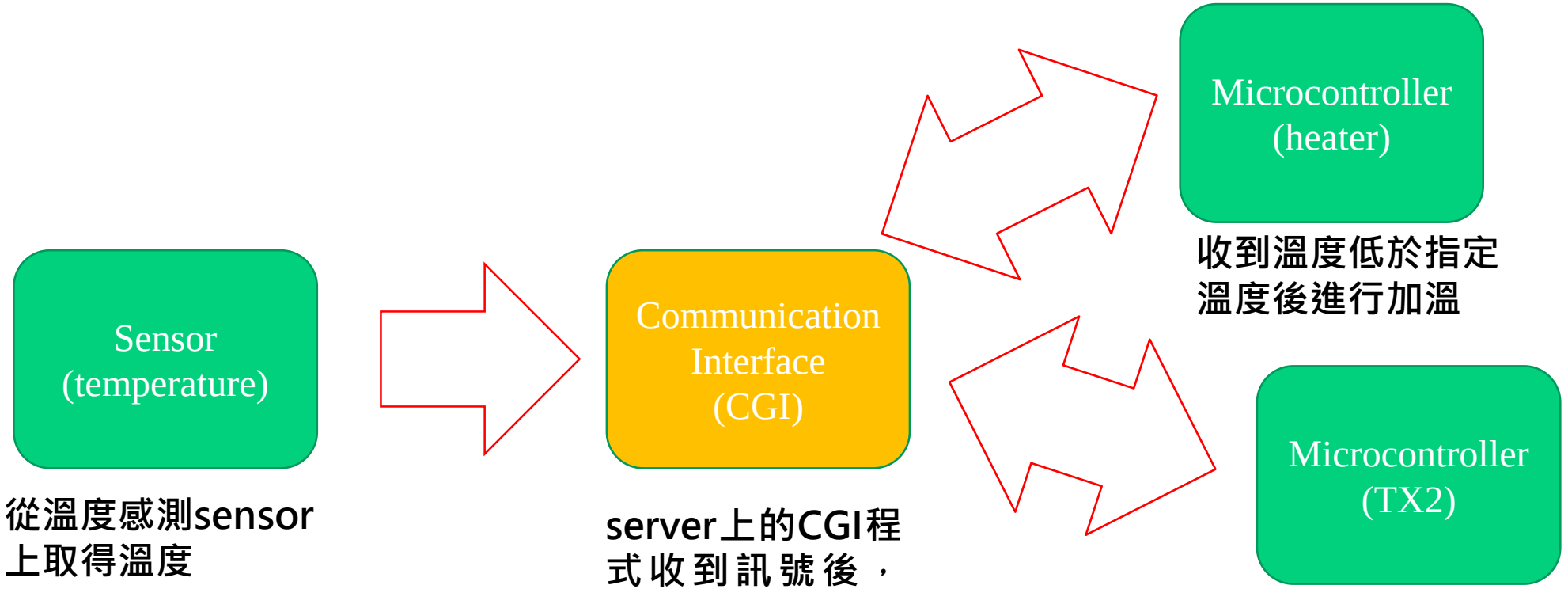
IOT架構範例



server上的CGI程式收到訊號後，將數據進行處理，並將處理結果送至對應的機制(例如:儲存、送雲端分析或呼叫I/O控制設備)

終端機取得訊息後可以依照訊息做對應的處理

IOT與CGI關係範例(魚塭養殖)



從溫度感測sensor上取得溫度

server上的CGI程式收到訊號後，將數據進行分析，並接結果送至對應的處理機制

收到溫度低於指定溫度後進行加溫

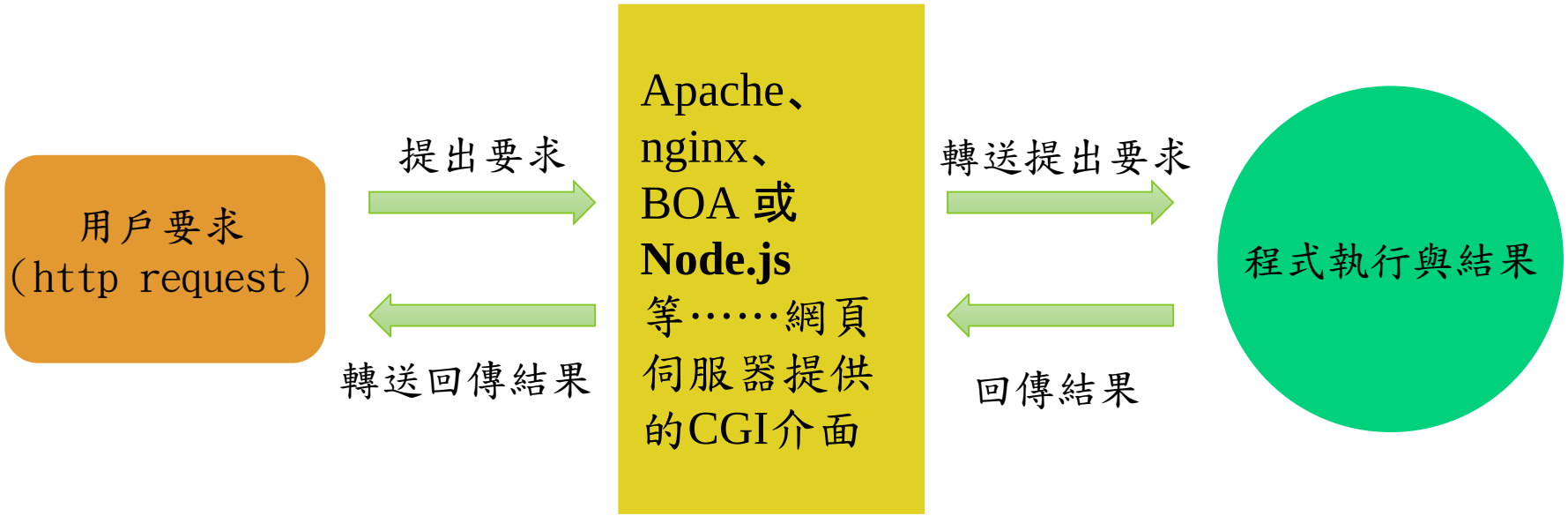
終端機取得訊息後可以依照訊息做對應的處理(例如控制GPIO設備)

CGI簡介

- 通用閘道器介面（Common Gateway Interface, CGI）
- CGI是一種重要的網際網路技術，可以讓一個用戶端，從網頁瀏覽器向執行在網路伺服器上的程式請求資料。CGI描述了伺服器和請求處理程式之間傳輸資料的一種標準
- 早期實現CGI經常使用程式語言Perl，但CGI的一個目的是要獨立於任何語言的，Web伺服器無須在這個問題上對語言有任何了解
- CGI程式基本上可以使用任何程式實現，只要這個語言可以在這個系統上執行
- 本章節CGI實作會以**Node.js**作示範介紹

CGI簡介

除Perl拿來實現CGI外，像Unix shell script、Python、Ruby、PHP、C/C++、Java、JavaScript、Node.JS等……都可以用來編寫CGI程式



CGI簡介

CGI常用的環境變數（僅供參考，不同的伺服器可能提供不同的環境變數）

伺服器名稱
ex.www.test.com

網頁使用port
預設為80

CGI程式所放置的位置

Environment Variable	Description
SERVER_SOFTWARE	The type of Web server running the CGI program
SERVER_NAME	The name of the Web server host
SERVER_PORT	The port address of the Web server
GATEWAY_INTERFAC	The version number of the CGI standard
SERVER_PROTOCOL	The version of HTTP the server is running
REQUEST_METHOD	The method of requesting data specified by the client
QUERY_STRING	Request parameters supplied by the client
SCRIPT_NAME	The resource locator of the CGI program
REMOTE_HOST	The name of the client host
REMOTE_ADDR	The Ipaddress if the client host
AUTH_TYPE	Authorization method,often blank
REMOTE_USER	The name of the user provided by the client
REMOTE_IDENT	An identify fot the client user,not often available
REFERER_URL	How the client got here
HTTP_ACCEPT	The MIME types accepted by the client
HTTP_USER_AGENT	The client browser type
CONTENT_TYPE	The MIME type of data supplied with the request

CGI的用途為何?

- 依照不同身分執行不同權限的動作
  - 可以依照不同登入的身分給予不同的操作權限
- 用於IoT
  - 透過網頁控制硬體
  - 透過行動裝置監控硬體資源
  - 可以設計對應的程式控制嵌入式平台上的GPIO
  - 可以設計對應的程式控制RS-232介面
- 跨平台的操作介面
  - 主要介面為網頁，可以運行在任何有瀏覽器的裝置
  - 可以設計對應的API透過Http Request來讓其他程式呼叫使用

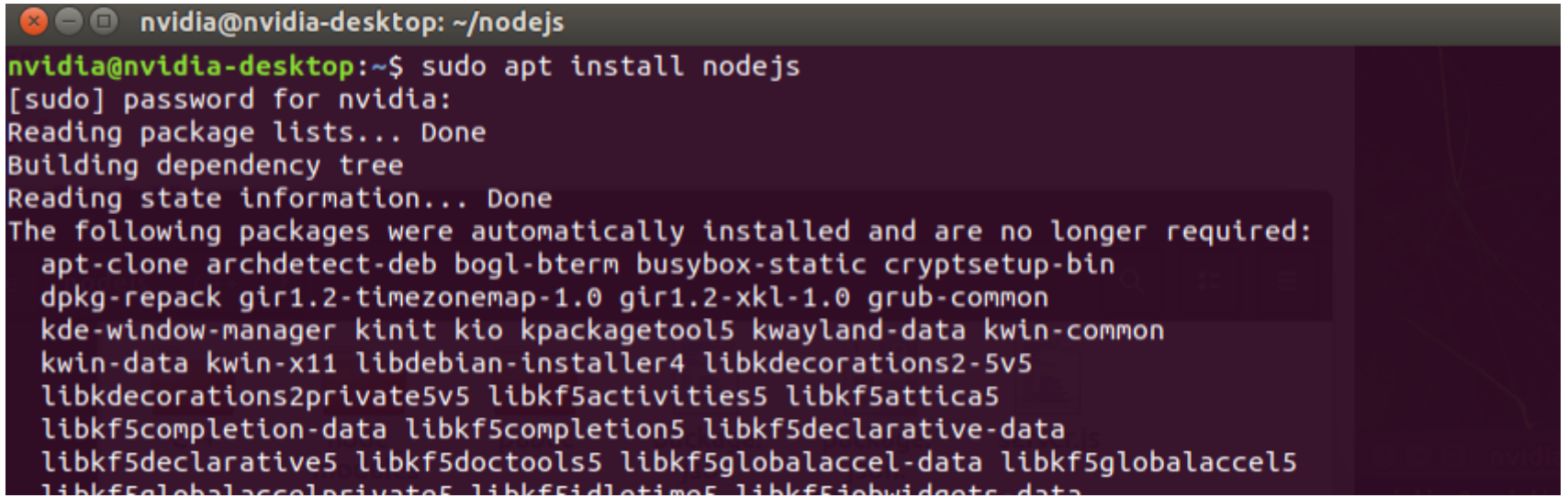
Node.js簡介

Node.js以JavaScript語言為基礎，是Ryan Dahl基於Google Chrome V8引擎於2009年釋出的JavaScript 開發平台，是一個高效能、易擴充的網站應用程式開發框架(Web Application Framework)

目的是為了讓開發者能夠更容易開發高延展性的網路服務，不需要經過太多複雜的調校、效能調整及程式修改，就能滿足網路服務在不同發展階段對效能的要求

Node.js安裝

- 開啟Terminal連線到TX2後輸入以下指令安裝Node.js
  - $ sudo apt-get install nodejs

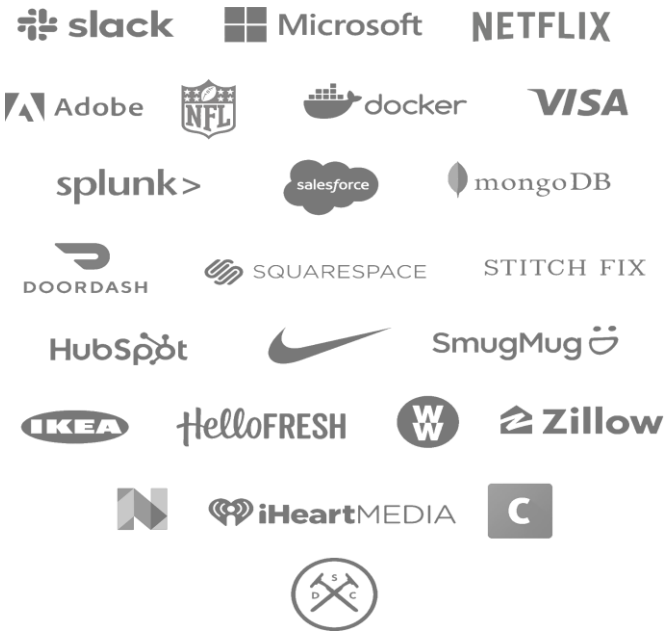


Node.js安裝

- 官方也有提供各種版本的安裝包供使用者下載
  - https://nodejs.org/zh-tw/download/

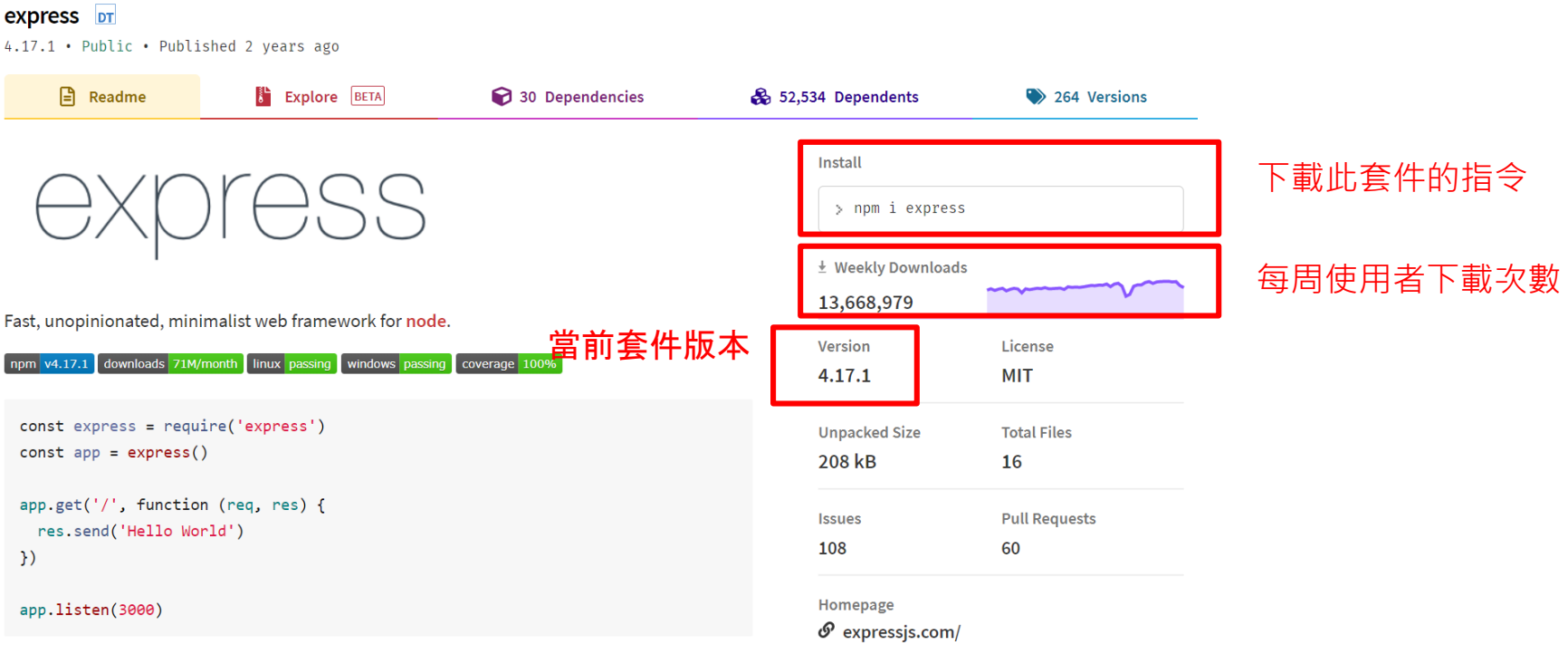
npm簡介

npm（全稱 **Node Package Manager**）是Node.js預設的、用JavaScript編寫的軟體套件管理系統。npm目前累計共有數十萬的開源模組來讓開發人員下載。其中有許多跨國企業將npm作為開發工具。



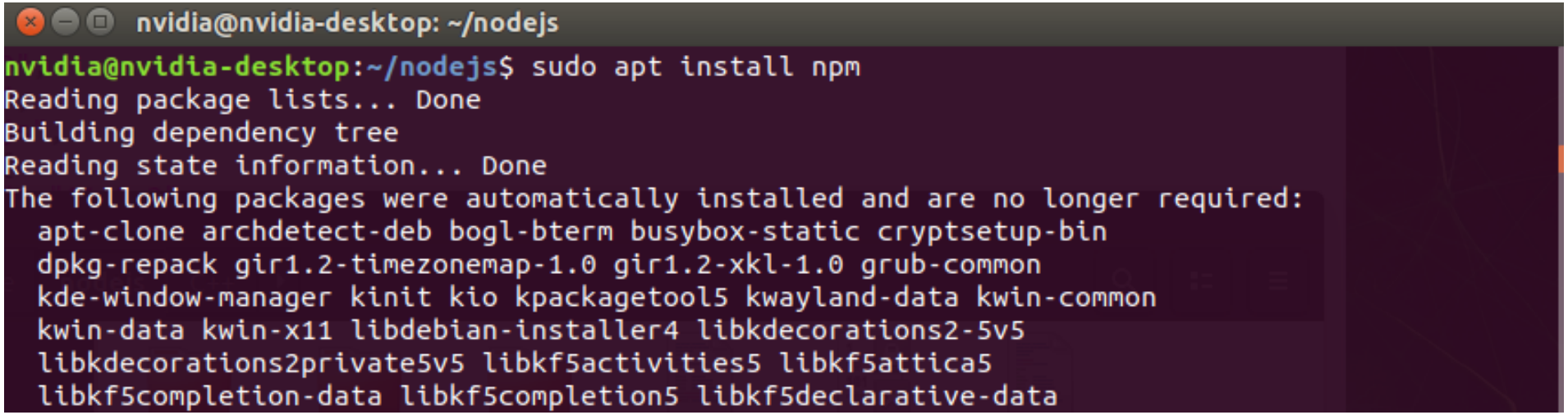
npm模組

- npm包含多種模組，可以到以下網址搜尋適合的模組安裝到電腦裡
  - https://www.npmjs.com/package/package
- 這裡以express為例



npm安裝

- 接續剛剛安裝完的Node.js，一樣在TX2的Terminal上輸入以下指令
  - $ sudo apt install npm



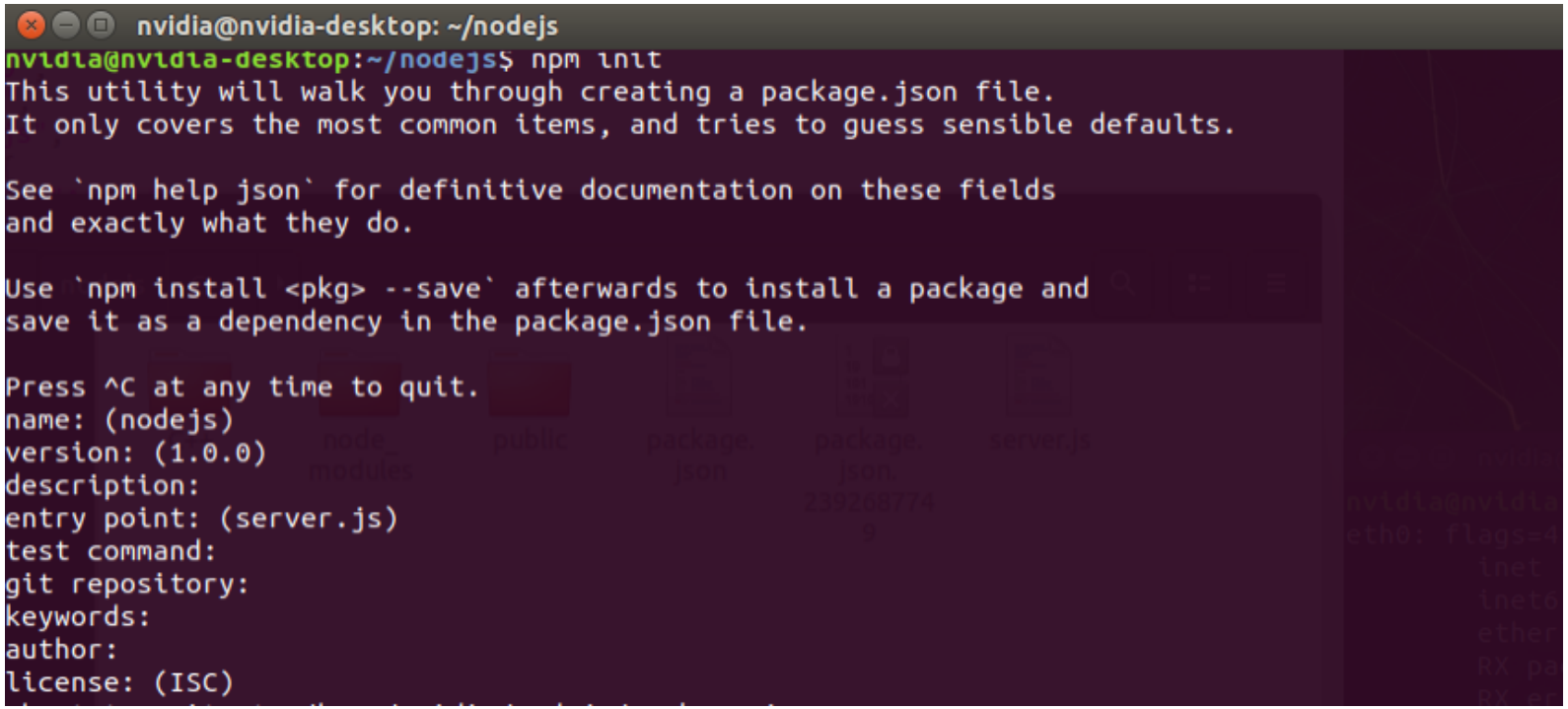
確認Nodejs及npm安裝完成及版本資訊

- 安裝完nodejs及npm後，可輸入以下指令，確認完成安裝及版本資訊
  - $ nodejs --version
  - $ npm --version

```
nvidia@ubuntu:~/Desktop$ nodejs --version
v10.19.0
nvidia@ubuntu:~/Desktop$ npm --version
6.14.4
```

npm 專案初始化

- 安裝完後進到指定的資料夾(範例資料夾名稱為nodejs)，進行初始化，並且產生package.json檔案
  - $ npm init



專案內的package.json

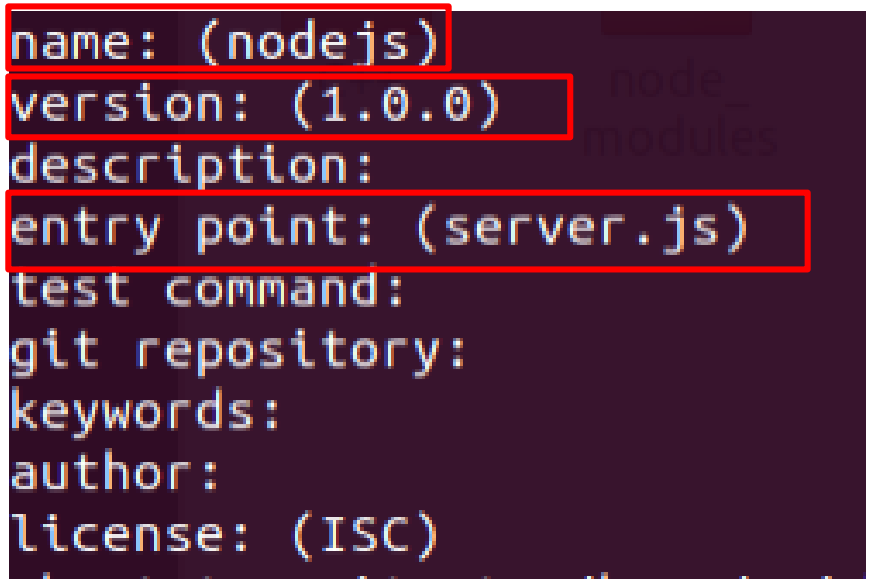
- package.json檔案：

  以下為package.json格式，使用**npm init**生成時，輸入 yes 可自動從當前目錄提取資訊生成預設值，也可由同學們自行定義專案版本等資訊

當前模組名稱

當前模組版本，初始預設為1.0.0

指定了專案載入的入口檔案



express模組介紹

express

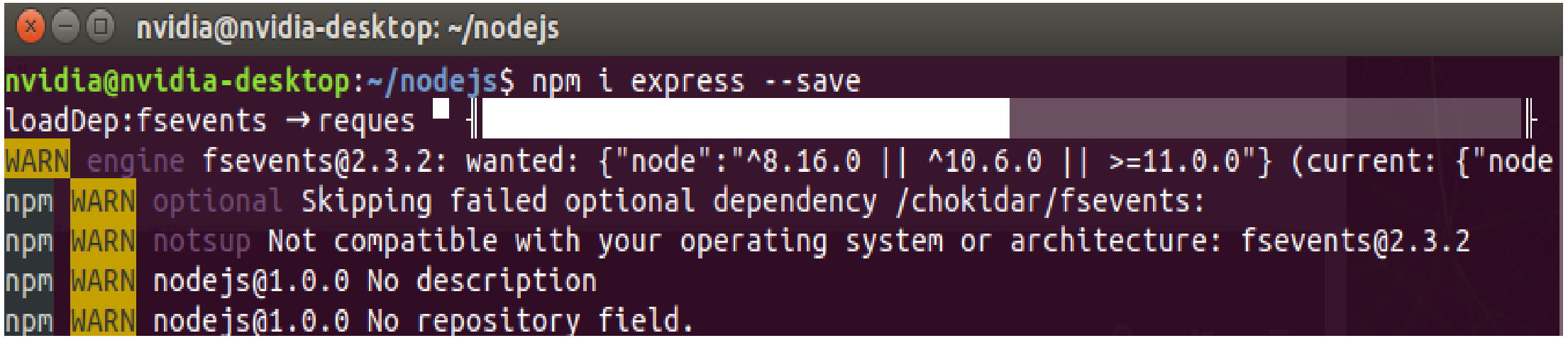
- **express** 能快速架設web server，是npm社群上受歡迎的Node web框架函式庫，該模組提供以下功能：
  - 替不同HTTP Method、不同URL路徑的requests編寫不同的處理方法
  - 透過整合「畫面」的渲染引擎來達到插入資料到樣板中產生response
  - 設定常見的web應用設定，例如：連線用的port和產生response的樣板位置
  - 在request的處理流程中增加額外的「中間層」進行處理

使用npm安裝express模組

- 安裝模組
  - $ npm i express --save

    * i表示install，也可將此行指令換成 $ npm install express --save

    * --save 是指該模組資訊會記錄在package.json檔裡



Nodejs引入模組-require()

```
// Importing a local module
// working directory. (On Windows, this would resolve to .\path\myLocalModule.)
const myLocalModule = require('./path/myLocalModule');
```

local module

```
// Importing a JSON file:
const jsonData = require('./path/filename.json');
```

JSON File

```
// Importing a module from node_modules or Node.js built-in module:
const child_process = require('child_process');
```

npm module
Node.js built-in module

啟動網頁伺服器

建立**server.js**檔，利用express套件快速架設web server並在前端顯示Hello World!!

server.js

```
1   //include necessary npm package
2   const express = require("express");
3   const app = express();
4
5   //simple routes
6   app.get("/", (req,res)=>{
7     res.send("Hello World!!")
8   })
9
10  // set port, listen for requests
11  const PORT = process.env.PORT || 8080;
12  app.listen(PORT, () => {
13    console.log(`Server is running on port ${PORT}.`);
14  });
```

引入npm模組

建立**GET** / 路由
回傳Hello World

伺服器指定port

啟動網頁伺服器

程式碼撰寫完後，開啟終端機，切換到專案目錄

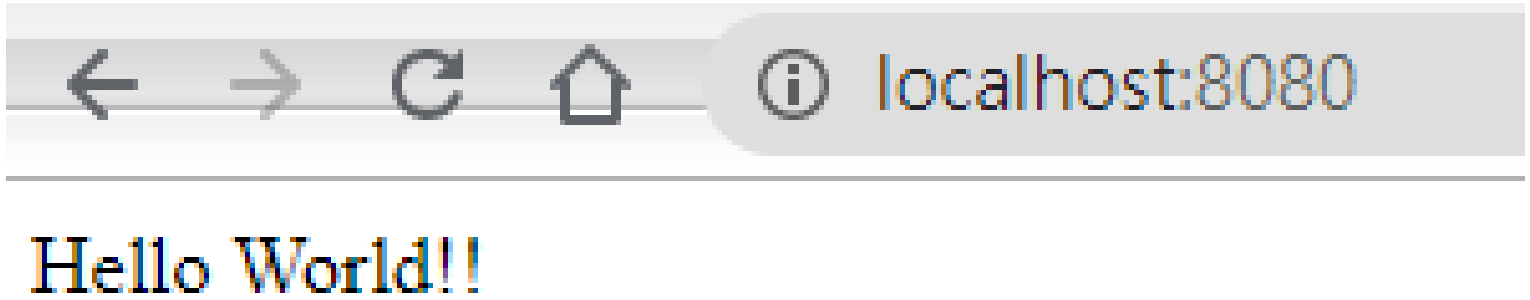
輸入**node .\server.js** 即可啟動網頁伺服器

```
PS C:\Users\Lucas\Desktop\nodejs> node .\server.js
Server is running on port 8080.
```

在本地端查看網頁

- 在伺服器本地端瀏覽器上，網址輸入localhost:8080
- 可以看到畫面顯示Hello World!!

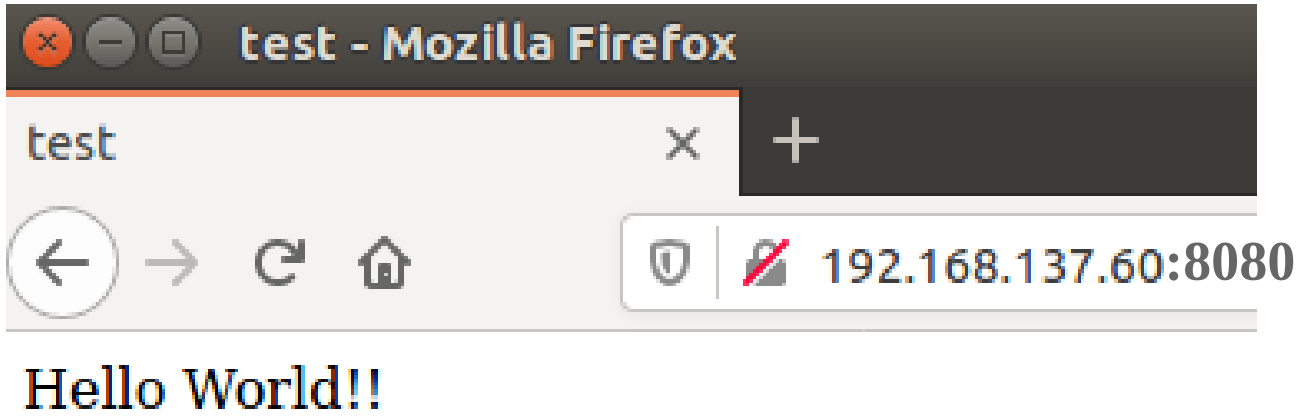
*port為啟動伺服器指定的埠號



在遠端查看網頁

- 在遠端使用瀏覽器連進嵌入式平台伺服器

  (網址為TX2嵌入式平台的ip及port)



簡易web server範例-客戶端傳送參數

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=\, initial-scale=1.0">
    <title>Nodejs</title>
</head>
<body>
    <p>
        <h1>Nodejs</h1>
    </p>
    <hr>
    <form action="/index" method="get">
        <h2>SEND REQUEST</h2>
            <label>姓名<input type="text" name="name"></label>
            <br>
            <label>學號<input type="text" name="id"></label>
            <hr>
            <input type="submit" value="Submit">
    </form>
</body>
</html>
```

index.html

將表單以GET的方式傳送/index 路由
姓名的參數為name
學號的參數為id

簡易web server範例-伺服器端接收參數

```
1   //include necessary npm package
2   const express = require("express");
3   const app = express();
4
5   //static resources
6   app.use(express.static('./public'));
7
8   //simple routes
9   app.get("/index", (req,res)=>{
10    var response = {
11      "my name": req.query.name,
12      "my id": req.query.id
13    }
14    res.send(response)
15  })
16
17  // set port, listen for requests
18  const PORT = process.env.PORT || 8080;
19  app.listen(PORT, () => {
20    console.log(`Server is running on port ${PORT}.`);
21  });
```

server.js

指定靜態檔案的存放路徑

建立簡易路由:

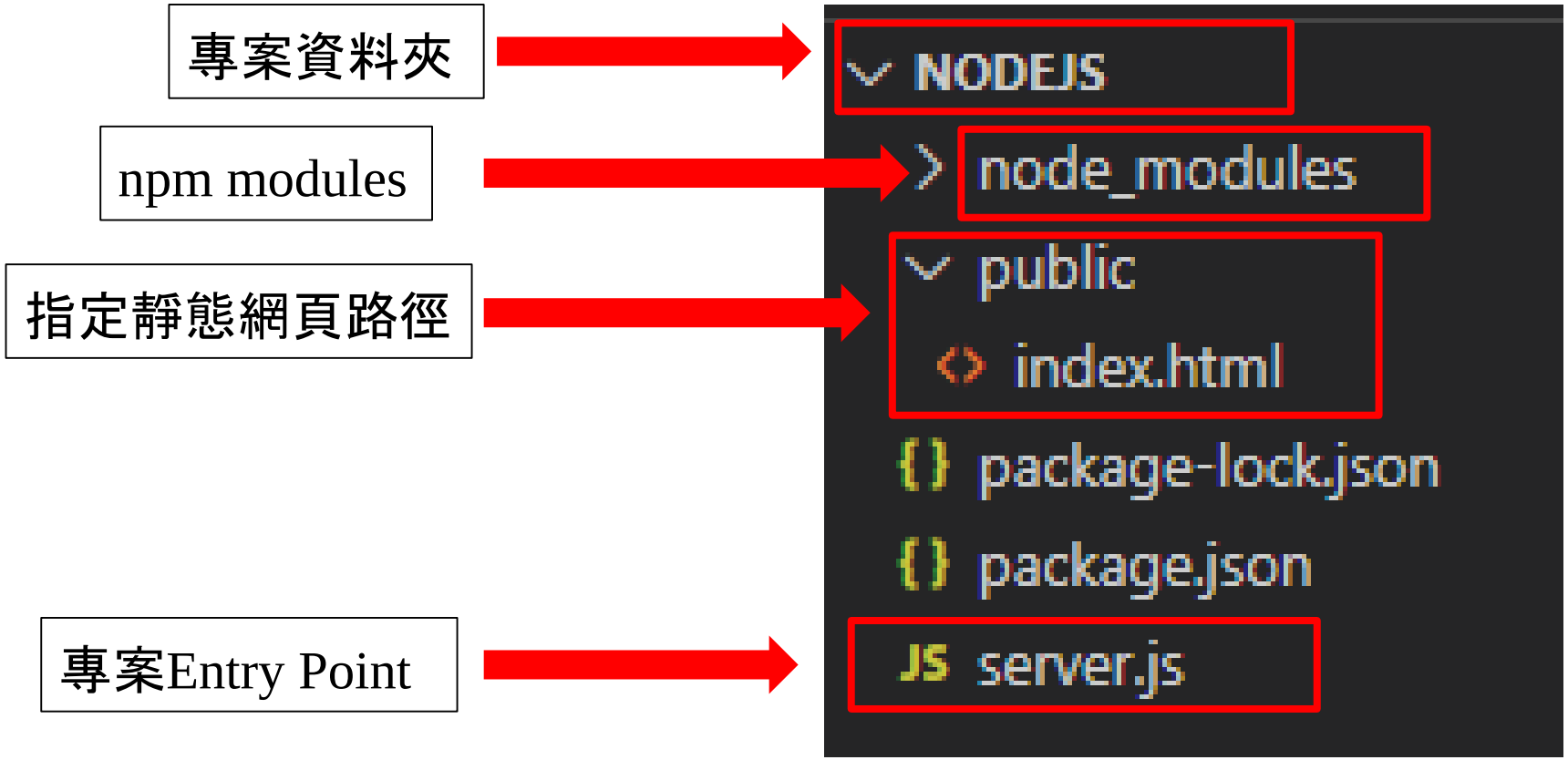
此例是以GET接收/index路由的參數

req.query.name 接收客戶端傳來name

req.query.id 接收客戶端傳來id

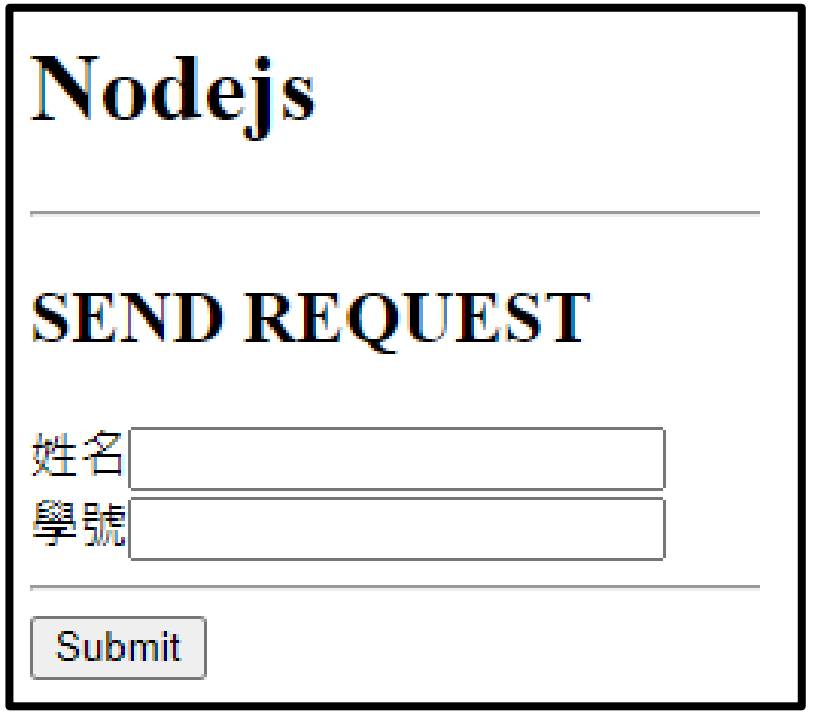
res.send()則是將資訊回傳至客戶端

簡易web server範例-專案架構



網頁顯示畫面

範例輸入:
姓名 Lucas
學號 109598999



Send Request