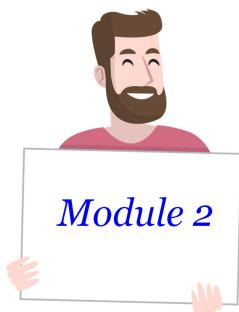




使用 npm 建立專案



designed by freepik

Estimated time:

50 min.

資訊工業策進會 Institute for Information Industry

【Key Points】：

學習目標

- **2-1: 初始化專案**
- **2-2: package.json 檔**
- **2-3: npm start**



2-1

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

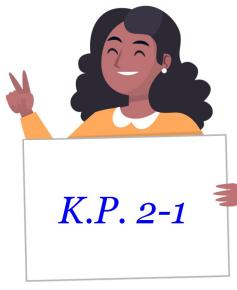
1. 介紹如何使用跟設定npm
2. 介紹如何安裝套件
3. 如何建立主要執行檔（建立node js 程式）
4. 啟動node js 程式
5. 了解package.json檔
6. 學會 npm 的基本使用方式

【Key Points】：

使用跟設定npm
啟動node js 程式
了解package.json檔

2-1：初始化專案

- 什麼是專案資料夾
- 專案初始化語法
- 安裝專案所需的套件
- 常見npm套件指令



designed by freepik



designed by freepik

2-2

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

NPM 為 Node Package Manager 的縮寫，開發者可以透過 Node 隨附的 npm cli，進行套件的安裝、管理與專案程式啟動：

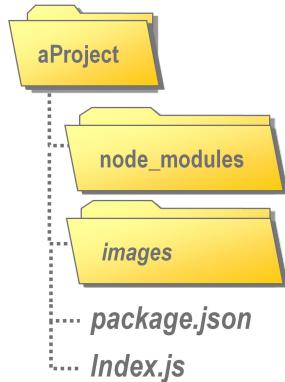
- npm init -f (快速建立專案)
- npm install 套件名稱
- npm uninstall 套件名稱
- npm install 套件名稱 --save
- npm start

【Key Points】：

專案初始化語法
安裝專案所需的套件
常見npm套件指令

什麼是專案資料夾

- 完成系統所需的一組相關檔案與子資料夾
- 工作環境
- 方便備份檔案與轉移工作內容
- 典型的 Node.js 專案資料夾



2-3



軟體系統通常都不只一個檔案，有主程式、畫面、設定檔、測試程式、文件、各式圖檔以及相關模組套件等等。

為了方便開發，往往會將這些檔案放置在一塊，形成一個整合的開發環境。同時，也方便專案的各個檔案互相參照使用。

換句話說，使用資料夾以及子資料夾，來收納專案所需的各類檔案。

有了專案資料夾後，備份以轉移工作內容也會變得容易很多，而且也不會遺漏檔案。

典型的 Node.js 專案會有一個名為 package.json 的檔案，其內容記載著專案的基本資訊、主程式是哪一支(main)、用到哪些模組套件(模組套件都放在 node_modules 資料夾)等資訊。package.json 檔案在專案管理方面，扮演了很重要的角色，舉例來說，檔案內容有 dependencies 屬性，記載著我們的專案用到哪些套件及其版本，因此，我們在拿到一個專案時，就算沒有 node_modules 資料夾，也能透過「npm install」，一口氣重新下載安裝全部所需的模組套件。

【Key Points】：

使用資料夾以及子資料夾，來收納專案所需的各類檔案

有了專案資料夾後，備份以轉移工作內容也會變得容易很多

典型的 Node.js 專案會有一個名為 package.json 的檔案，在專案管理方面，扮演了很重要的角色。

確認NPM版本

- NPM 為 Node Package Manager 的縮寫。開發者可以透過 Node.js 隨附的 npm cli，進行套件的安裝及管理。
- 確認NPM版本：
`npm -v`
- 更新至最新版：
`npm install npm@latest -g`

2-4



1. Node.js安裝之時，預設已安裝 NPM 。
2. NPM 為 Node Package Manager 的縮寫，
3. 開發者可以透過 Node 隨附的 [npm cli](#)，進行套件的安裝及管理。
4. 所謂 npm cli 是 npm command line，是一個指令列工具，可以下一些命令來執行套件管理，
例如: `npm init`（後面會介紹）
5. 執行下列指令可確認NPM版本：
`npm -v`
6. 更新至最新版：
`npm install npm@latest -g`

【Key Points】：

NPM 為 Node Package Manager 的縮寫
開發者可以透過 Node 隨附的 [npm cli](#)，進行套件的安裝及管理。
`npm -v`，確認NPM版本

專案初始化語法

- **npm init**

專案初始化 & 建立 package.json

- **npm init -f**

-f 表示快速建立

```
npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (module-2 使用 npm) module2
version: (1.0.0) 0.0.1
description: module 2 using npm
entry point: (index.js) [
```

2-5



1. 想快速建立專案，各選項都照預設值的話，在 npm init 後面加上-f 即可。
2. init 是 initialize 「初始化」的意思
3. -f 表示快速建立 (f 是 force 的意思)，相當於 npm init -y 代替
4. 實務上，我們會在資料夾內新增一個名為 .gitignore 的檔案，並在該檔案內記錄 node_modules 是不需 git 列管的資料夾。
5. .gitignore 檔主要是在接下來 npm 在安裝套件後，避免 node_modules 資料夾的套件檔案一同被備份到 git repository 內 (git: 版本控制工具) 。

【Key Points】：

npm init

npm init -f 表示快速建立

為什麼 node_modules 的內容不必備份呢？利用 npm install 可全部重新下載。

安裝專案所需的套件

- 在 VS Code 按下「Ctrl + 滑鼠右鍵」可開啟 terminal 終端機視窗。
- 在終端機視窗，依下列語法安裝套件：
npm install 套件名稱
- 請將「套件名稱」換入你要安裝的套件名稱，例如：
npm install express
- 解除安裝的語法：
npm uninstall 套件名稱

2-6



// i 為 install 縮寫

1. npm i axios 或 npm install axios

// 可以指定版本號(安裝某一個版本的套件)

2. npm i lodash@4.17

// --save 會把套件加入 dependencies

3. npm i axios --save

// 更新套件

4. npm update 套件名

// 解除安裝

5. npm uninstall 套件名稱 或 npm remove 套件名稱

【Key Points】：

npm install 套件名稱

npm uninstall 套件名稱

npm install 套件名稱 --save

常見npm套件指令

安裝全域套件

npm install -g [PACKAGE]

安裝特定版本

npm install -g [PACKAGE]@[VERSION]

移除全域套件

npm uninstall -g [PACKAGE]

更新全域套件

npm update -g

安裝專案套件

npm install [PACKAGE]

若專案裡已定義 package.json 則可直接下指令

npm install

移除專案套件

npm uninstall [PACKAGE]

列出全域套件

npm ls -g



1. 全域套件與專案套件的差別在於：全域套件安裝後，可以在任何一個Node.js專案執行，只要是在同一台電腦/伺服器上，不需另外安裝。
2. 不同套件會依需求而有不同的安裝方式。比方說nodemon通常會安裝全域套件，因為幾乎所有的Node.js專案都會用到。
3. 不同的參數會有不同的效果。
4. 若是使用別人的專案，例如從github.com上clone回來的程式，可以利用npm install一次安裝缺乏的套件。
5. 本課程用到的套件，屆時各章節會適時提出說明。

【Key Points】：

安裝全域套件

npm install

npm ls -g

2-2: package.json 檔

- package.json 檔案內容
- 關於package-lock.json
- 匯入引用套件
- 執行Node.js程式



designed by freepik



designed by freepik

2-8

III 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

套件模組分為兩種：core 跟 npm modules。

core 是 Node.js 內建的; npm modules 則是根據需求另外以 npm 安裝。

說明 package.json 的內容，例如：

dependencies: 本專案使用到的套件

start : 專案一開始要執行的程式檔檔名

不論是使用 Core 或 NPM modules，都需要載入才能使用。require() 回傳的可能是物件，也可能函式。

package-lock.json 記錄了專案實際安裝的各個 npm 套件的來源和版本。

【Key Points】：

不論是使用 Core 或 NPM modules，都需要載入才能使用

說明 package.json 的內容

package-lock.json 記錄了專案實際安裝的各個 npm 套件的來源和版本

package.json 檔案內容

```
"name": "專案名稱",
"version": "專案版號",
"description": "描述",
"scripts": {
    // npm script (可以將各種指令組合一起輸出)
"start": "node app.js" // 簡化指令
},
"repository": {
"type": "git",
"url": "上GitHub / GitLab建立專案的網址"
},
"author": "作者名",
"license": "專案版權", // ISC MIT
"devDependencies": {
// 專案開發時使用的套件，專案輸出時不會加入
"eslint": "^5.16.0",
},
"dependencies": {
// 專案上線時相依的套件，專案輸出時會加入
"axios": "^0.19.0"
}
```

- 左邊為 Node.js 專案常見的 package.json 內容，內含很多細節：
 - repository
 - license
 - dependencies
 - version
 - script.start

2-9



針對 package.json 幾個比較重要的內容，說明如下：

- repository：程式碼版本控制設定，例如 git repository 的網址
- license：程式碼著作權 (例如 ISC / MIT)
- dependencies：本專案使用到的套件清單
- version：版本號，首個數字代表主版本，第二個代表次版本 (新功能)，第三個：代表更新 (Patch,Bug Fix)
- script.start：npm start 時，要執行什麼程式，例如: "start": "node app.js"。

package.json 版號標記符號：

^ 表更新至最新的次版本 (不會更新主版本)，舉例來說，^1.2.0 表示可以接受 1.2.0 ~ 1.999.999 版，2.0.0 以後的不行。

~ 表更新至最新的更新 (Patch)，不會更動次版本，舉例來說 ~1.2.3 表示可以接受 1.2.3 ~ 1.2.999 版，1.3.0 以後的不行。

【Key Points】：

說明 package.json 的內容

dependencies：本專案使用到的套件

start：專案一開始要執行的程式檔檔名

關於 package-lock.json

- 執行 `npm install` 的時候生成/更新的檔案
- 記錄了專案實際安裝的各個 `npm` 套件的來源和版本
- 確保之後使用專案的人，可利用 `npm install` 指令，安裝的套件都跟你用的是相同的版本

```
"dependencies": {  
  "@types/node": "^8.0.33",  
},
```

2-10



`package-lock.json` 是執行 `npm install` 的時候生成/更新的檔案。

檔案記錄了專案實際安裝的各個套件的來源和版本

確保之後使用專案的人，可利用 `npm install` 指令，安裝的套件都跟你用的是相同或相容的版本。

舉例來說：

```
"dependencies": {  
  "@types/node": "^8.0.33",  
}
```

`^` 表更新至最新的次版本 (不會更新主版本)，舉例來說，`^1.2.0` 表示可以接受 `1.2.0 ~ 1.999.999` 版，`2.0.0` 以後的不行。

`~` 表更新至最新的更新 (Patch)，不會更動次版本，舉例來說 `~1.2.3` 表示可以接受 `1.2.3 ~ 1.2.999` 版，`1.3.0` 以後的不行。

【Key Points】：

執行 `npm install` 的時候生成/更新的檔案

記錄了專案實際安裝的各個 `npm` 套件的來源和版本

確保之後使用專案的人，可利用 `npm install` 指令，安裝的套件都跟你用的是相同的版本

匯入引用套件

假設我們的專案需要安裝一名為 ip 的套件模組，協助我們處理有關 ip 的相關工作。

執行下列指令可下載安裝 ip 套件
npm install ip

安裝之後，Node.js 程式透過
var ip = require("ip");
引用 ip 套件模組

```
npm install ip
npm WARN module2@0.0.1 No repository field.

+ ip@1.1.5
added 1 package from 1 contributor and audited 127 packages in 1.909s
found 0 vulnerabilities
```

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

console.log(ip.address()) // my ip address
```



- 套件模組分為兩種：core 跟 npm modules (packages)
- core 是 Node.js 內建的，例如：fs、http。
- npm modules 則是根據需求另外以 npm 安裝，例如：express。
- 不論是使用 Core 或 NPM modules，都需要用 require() 載入才能使用。例如：
`var module = require("express"); // express 可以換成其他名稱`
- 不同的套件模組，基於功能與不同的設計考量，require() 回傳的可能是物件，也可能是函式

【Key Points】：

套件模組分為兩種：core 跟 npm modules

不論是使用 Core 或 NPM modules，都需要載入才能使用

require() 回傳的可能是物件，也可能是函式

執行Node.js程式

在『終端機』或『命令提示字元』輸入
node <檔案名稱.js>

例如：

node index.js

按照前一張投影片的程式：

若沒有安裝前面所說的 ip 套件，會顯示錯誤訊息（右上方的圖）。

一切順利，則會顯示電腦的IP位址：

```
node index.js
internal/modules/cjs/loader.js:797
    throw err;
^

Error: Cannot find module 'ip'
```

```
node index.js
192.168.43.105
```



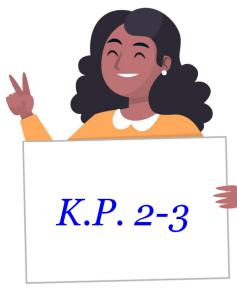
- 在『終端機』或『命令提示字元』模式啟動 Node.js 以執行 JavaScript 程式。例如：
node server.js
- 程式有錯，程式自然無法順利執行（例如沒有 npm install 就想 require 套件）
- 錯誤訊息會顯示在終端機視窗
- 根據錯誤訊息進行除錯
- 每台電腦的 IP 位址都不同，上述投影片顯示的 IP 很可能與你的電腦的IP不同

【Key Points】：

Node.js 程式會在『終端機』或『命令提示字元』模式底下進行執行，例如：node index.js
沒有 npm install 就想 require 套件，程式無法執行
根據錯誤訊息進行除錯

2-3: npm start

- **npm start**
- **npm scripts** 進階使用方式
- 常用套件模組簡介
- 學習資源與網站



designed by freepik



designed by freepik

2-13

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

- 在『終端機』或『命令提示字元』啟動 Node.js 以執行 JavaScript 程式。例如:
`node server.js`
- 在 package.json 文件裡面，可使用 scripts 定義腳本命令，再以下列指令啟動
 - `npm start`
 - `npm run`
- 通常 `npm run` 或 `npm start` 已足夠使用，加上不同的旗標參數來檢視輸出訊息
- 常用套件模組簡介
- 學習資源與網站

【Key Points】：

在『終端機』或『命令提示字元』啟動 Node.js 以執行 JavaScript 程式
`npm start`
學習資源與網站

npm start

- 以 **npm start** 來執行 **scripts.starts** 定義的腳本命令
- 在 **package.json** 文件裡面，可使用 **scripts** 詳定腳本命令。
- “**build": "node build.js"** 右下圖例的內容是 **package.json** 文件的片段，其中的 **scripts** 記載著 **build** 命令對應的腳本是 **node build.js**
- 在終端機使用 **npm run** 命令，就可以執行 **build** 這段腳本：
\$ **npm run build**
等同執行
\$ **node build.js**
- 以下是兩個常用的腳本指令示範：
 - **"start": "node server.js"** ,
 - **"install": "node-gyp rebuild"**

```
{  
  // ...  
  "scripts": {  
    "build": "node build.js"  
  }  
}
```

2-14



在 **package.json** 文件裡面，可使用 **scripts** 定義腳本命令。

1. **"build": "node build.js"**

投影影片的畫有 **package.json** 文件的一個片段，裡面的 **scripts** 記載著 **build** 命令對應的腳本是 **node build.js**

2. 在終端機使用 **npm run** 命令，就可以執行 **build** 這段腳本：

```
$ npm run build  
# 等同執行  
$ node build.js
```

以下是兩個常用的腳本指令示範：

- **"start": "node server.js"** ,
- **"install": "node-gyp rebuild"**

如果想平行處理，使用**&**符號。

```
$ npm run script1.js & npm run script2.js
```

如果是同步執行（前一個動作完成之後，才開始執行下一個動作），則使用**&&**符號：

```
$ npm run script1.js && npm run script2.js
```

【Key Points】：

在 **package.json** 文件裡面，可使用 **scripts** 定義腳本命令。

npm start

npm run

npm scripts 進階使用方式

- 在 npm start 指令後面，可以加上不同的旗標 (flags):
-s, --silent: --loglevel silent
-q, --quiet: --loglevel warn
-d: --loglevel info
-dd, --verbose: --loglevel verbose
-ddd: --loglevel silly

2-15



- 通常 npm run 或 npm start 已足夠使用，但 npm 並非只限於此
- 可加上不同的旗標參數來檢視輸出訊息
- 實務上可以拿來除錯或是監控程式執行狀態
例如: npm start -dd
- 建立指令方式很多元，指令不一定要寫死
- 參考資料：<https://docs.npmjs.com/misc/scripts>

【Key Points】：

通常 npm run 或 npm start 已足夠使用
加上不同的旗標參數來檢視輸出訊息
參考資料：<https://docs.npmjs.com/misc/scripts>

常用套件模組簡介

| 套件模組名稱 | 說明 |
|-------------|---|
| express | 建立 Web 伺服器最常見的套件 |
| body-parser | 協助 Express 解析表單與JSON資料 |
| cors | 協助進行跨來源資源共用 (CORS) 在 HTTP 回應的標頭區加入 Access-Control-Allow-Origin 聲明 |
| mysql | 連線 MySQL 資料庫伺服器存取資料的套件 |
| ejs | 可搭配 Express , 一套UI畫面處理引擎 |

2-16



express 建立 Web 伺服器最常見的套件
body-parser 協助 Express 解析表單與JSON資料
cors 協助進行跨來源資源共用 (CORS)
mysql 連線 MySQL 資料庫伺服器存取資料的套件
ejs 可搭配 Express , UI畫面處理引擎

以上都是本課程會用到的套件。各章節會適時提出說明。

Express (建立 Web 伺服器最常見的套件)
Mysql (連線 MySQL 資料庫伺服器存取資料的套件)
EJS (UI畫面處理引擎)

【Key Points】：

express (建立 Web 伺服器最常見的套件)
cors (協助跨來源資源共用CORS)
ejs (可搭配 Express , 一套 UI 畫面處理引擎)

常用套件模組簡介

- <https://www.npmjs.com/>
1. 在這個網站搜尋套件
2. 透過 npm install 來安裝
- Stackoverflow
<https://stackoverflow.com/>
- W3Schools
<https://www.w3schools.com/>
- GitHub:
<https://github.com>



2-17



除了npm官網之外，網路上還有很多地方學習資源與網站：

Stackoverflow: <https://stackoverflow.com/>，程式疑難的問答討論。

GitHub: <https://github.com>，有很多別人寫過的程式可供參考。

W3Schools: <https://www.w3schools.com/>，不錯的教學與資料查詢網站。

臉書：有多Nodejs相關社團（例如Nodejs台灣, backend tw）可以發問nodejs相關問題

【Key Points】：

Stackoverflow (程式疑難的問答討論)
GitHub (有很多別人寫過的程式可供參考)
W3Schools (教學與資料查詢網站)

Summary〈精華回顧〉

- 初始化專案
- 安裝專案所需的套件
- `package.json` 檔
- 匯入引用套件
- 執行 Node.js 程式
- `npm start`
- 學習資源與網站



2-18

 財團法人資訊工業策進會
INSTITUTE FOR INFORMATION INDUSTRY

開發者可透過 NPM 來初始與執行專案，並且進行套件的安裝管理：

`npm init -f`

`npm install 套件名稱 --save`

`npm start`

套件模組分為兩種：core 跟 npm modules。

core 是 Node.js 內建的；npm modules 則是根據需求另外以 npm 安裝。

不論是使用 Core 或 NPM modules，都需要載入才能使用。

`package.json` 檔案內的 `dependencies` 記錄了專案所需的套件清單

`package.json` 檔案內的 `start` 記載了專案要執行的腳本程式，以 `npm start` 或 `npm run` 啟動。

【Key Points】：

透過 NPM 來初始與執行專案，並且進行套件的安裝管理

`package.json` 檔案內的 `dependencies` 記錄了專案所需的套件清單

`package.json` 檔案內的 `start` 記載了專案要執行的腳本程式，以 `npm start` 或 `npm run` 啟動

線上程式題

- **2-1 初始化 Node.js 專案**

想要將資料夾轉換成 Node.js 專案資料夾，應該執行什麼指令？

- **2-2 相依套件版本控管**

依據 package.json 的下列內容

```
"dependencies": {  
    "express": "^4.17.1"  
}
```

express 這個模組可以接受怎樣的版本範圍？

- **2-3 npm start**

開發人員想在終端機視窗輸入 npm start 來啟動主程式，請問 package.json 該怎麼改？

2-19



題目名稱: 2-1 初始化 Node.js 專案

想要將資料夾轉換成 Node.js 專案資料夾，應該執行什麼指令？

題目名稱: 2-2 相依套件版本控管

依據 package.json 的下列內容

```
"dependencies": {  
    "express": "^4.17.1"  
}
```

express 這個模組可以接受怎樣的版本範圍？

題目名稱: 2-3 npm start

開發人員想在終端機視窗輸入 npm start 來啟動主程式，請問 package.json 該怎麼改？

```
"scripts": {  
    "start": "node index.js"  
},
```

【Key Points】：

2-1 初始化 Node.js 專案

2-2 相依套件版本控管

2-3 npm start

課後練習題(Lab)

- **情節描述:**

NPM 全名為Node Package Manager，是附屬在 Node.js 中的套件管理工具，我們在安裝 Node.js 時，就已安裝到電腦中，可在命令列 / 終端機模式中使用。

- **預設目標:**

1. 初始化一個NPM的資料夾
2. 修改package.json加入開發者的資料
3. 取得其他開發者的程式
4. 查詢目前安裝哪些套件
5. 移除已安裝的套件

Estimated time:

20 minutes

2-20



【情節描述】

NPM 全名為Node Package Manager，是附屬在 Node.js 中的套件管理工具，我們在安裝 Node.js 時，就已安裝到電腦中，可在命令列 / 終端機模式中使用。

【預設目標】

預設目標:

1. 初始化一個NPM的資料夾
2. 修改package.json加入開發者的資料
3. 取得其他開發者的程式
4. 查詢目前安裝哪些套件

Lab01: 安裝Node.js 和 NPM

Lab02: 修改package.json

Lab03: 取得其他開發者的程式

Lab04: 查詢目前安裝哪些套件

Lab05: 移除已安裝的套件

【Key Points】:

Lab01: 安裝Node.js 和 NPM

Lab02: 修改package.json

Lab03: 取得其他開發者的程式

範例程式使用說明

- 範例程式資料夾: Module_02_example
- 使用步驟:
 1. 安裝 Node.js
 2. 安裝 Visual Studio Code
 3. 以 Visual Studio Code 開啟本模組的範例資料夾
 4. 在 Visual Studio Code 按下「Ctrl + 滑鼠右鍵」開啟 terminal 終端機視窗。
 5. 在終端機視窗，輸入下列指令，安裝必要的模組套件:
npm install
 6. 在終端機視窗，輸入 node index.js

2-21



使用步驟:

1. 安裝 Node.js
<https://nodejs.org/en/>
2. 安裝 Visual Studio Code
<https://code.visualstudio.com/>
3. 以 Visual Studio Code 開啟範例資料夾
在檔案總管，滑鼠右鍵點按「本範例資料夾」，從快捷功能表點選「以Code開啟」
或者，啟動 Visual Studio Code 之後，功能表 File | Open Folder，選擇「本範例資料夾」
4. 在 Visual Studio Code 按下「Ctrl + 滑鼠右鍵」開啟 terminal 終端機視窗。
5. 在終端機視窗，輸入下列指令，安裝必要的模組套件:
npm install
6. 在終端機視窗，輸入 node index.js

【Key Points】：

程式執行環境 Node.js

程式開發編輯環境: Visual Studio Code

在 Visual Studio Code 按下「Ctrl + 滑鼠右鍵」開啟 terminal 終端機視窗，輸入：「node 主程式.js」