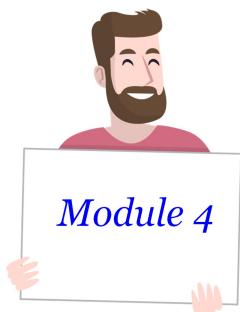




# 使用 npm 專案初始化



designed by freepik

Estimated time:

50 min.

資訊工業策進會 Institute for Information Industry

【Key Points】：

## 學習目標

- 4-1: 基本命令列指令
- 4-2: 建立專案資料夾
- 4-3: 初始化專案



4-1

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

我們即將一起學習：

- 查閱NPM指令用法
- 安裝專案所需的套件
- 查詢套件資訊
- 常用套件模組簡介
- 什麼是專案資料夾
- 建立專案資料夾
- 以 VS Code 開啟專案資料夾
- 確認專案路徑
- 專案初始化語法
- package.json
- 匯入引用套件
- 執行Node.js程式
- npm start

【Key Points】：

- 4-1: 基本命令列指令
- 4-2: 建立專案資料夾
- 4-3: 初始化專案

## 4-1：基本命令列指令

- 查閱NPM指令用法
- 安裝專案所需的套件
- 查詢套件資訊
- 常用套件模組簡介



designed by freepik



designed by freepik

4-2

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

接下來，我們要一起學習：

- 啟動「命名提示字元」，輸入指令。
- 查詢NPM版本: `npm -v`
- 查詢NPM指令用法: `npm help`
- `npm install` 套件名稱
- `npm uninstall` 套件名稱
- `npm install` 套件名稱 `--save`
- 透過 `npm search` 這個指令查詢套件
- 請留意一下 NAME 那欄的內容，後來安裝模組套件時會用到
- 透過 `npm view` 指令查詢該套件的詳細資訊
- 簡介常用套件模組

### 【Key Points】：

查閱NPM指令用法  
安裝專案所需的套件  
查詢套件資訊

# 查閱NPM指令用法

- 啟動「命名提示字元」

- 查詢NPM版本:

**npm -v**

(通常也用來確認是否已安裝NPM)

- 查詢NPM指令用法

**npm help**

- 條列NPM指令詳細用法

**npm -l**

```
PS C:\Users\PC> npm help

Usage: npm <command>

where <command> is one of:
  access, adduser, audit, bin, bugs, c, cache, ci, cit,
  clean-install, clean-install-test, completion, config,
  create, ddp, dedupe, deprecate, dist-tag, docs, doctor,
  edit, explore, get, help, help-search, hook, i, init,
  install, install-ci-test, install-test, it, link, list, ln,
  login, logout, ls, org, outdated, owner, pack, ping, prefix,
  profile, prune, publish, rb, rebuild, repo, restart, root,
  run, run-script, s, se, search, set, shrinkwrap, star,
  stars, start, stop, t, team, test, token, tst, un,
  uninstall, unpublish, unstar, up, update, v, version, view,
  whoami

  npm <command> -h  quick help on <command>
  npm -l      display full usage info
  npm help <term>  search for help on <term>
  npm help npm  involved overview

Specify configs in the ini-formatted file:
  C:\Users\PC\.npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config

npm@6.12.1 C:\Program Files\nodejs\node_modules\npm
```

4-3



啟動「命名提示字元」，輸入指令。

如何啟動「命名提示字元」：

- 按下組合鍵「Windows + R」
- 輸入「cmd」後，點按「確定」按鈕。（啟動「命令提示字元」）
- 上述步驟1+2，也可以：按下組合鍵「Windows + S」，然後鍵盤輸入「cmd」找到「命令提示字元」。

查詢NPM版本: (通常也用來確認是否已安裝NPM)

**npm -v**

查詢NPM指令用法: **npm help**

條列NPM指令詳細用法: **npm -l**

【Key Points】：

啟動「命名提示字元」，輸入指令。

查詢NPM版本: **npm -v**

查詢NPM指令用法: **npm help**

## 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**

4-4



// 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

# 查詢套件資訊

- 關於套件查詢/查看詳細資料指令

指令	說明	範例
npm search <套件名>	搜尋套件	npm search express
npm view <套件名>	查看套件資訊	npm view express

```
PS D:\程式\node\node-project> npm search express
NAME          | DESCRIPTION           | AUTHOR        | DATE       | VERSION | KEYWORDS
express       | Fast, unopinionated, minimalist web framework
              | ...                   | =dougwilson... | 2019-05-26 | 4.17.1   | express framew
```

```
PS D:\程式\node\node-project> npm view express
express@4.17.1 | MIT | deps: 30 | versions: 264
Fast, unopinionated, minimalist web framework
http://expressjs.com/
keywords: express, framework, sinatra, web, rest, restful, router, app, api
dist
tarball: https://registry.npmjs.org/express/-/express-4.17.1.tgz
shasum: 4491fc38605cf51f8629d39cb5d026f984c134
integrity: sha512-mHJ9079RqlupHrcw2X/GTh3k9tVv8YcoY4Kkh4WDMUYKRZUq8h1o8w2rrrxBqM7VoeUVqgb27x1EMXTnYt4g==
unpackedSize: 208.1 kB
dependencies:
accepts: ~1.3.7      cookie: 0.4.0      finalhandler: ~1.1.2      path-to-regexp: 0.1.7
array-flatten: 1.1.1  debug: 2.6.9      fresh: 0.5.2      proxy-addr: ~2.0.5
body-parser: 1.19.0   dnode: ~1.1.2     merge-descriptors: 1.0.1  qs: 6.7.0
```

4-5



- 當你想安裝套件，卻不確定套件的名稱時，可以透過 npm search 這個指令查詢。  
例如: npm search express
- 此時會條列出「含有你所輸入關鍵字名稱」的套件。
- 請留意一下 NAME 那欄的內容，安裝模組套件時會用到。
- 確定要哪個套件之後，可以透過 npm view 指令查詢該套件的詳細資訊。  
例如: npm view express
- 想看套件的詳細資訊，不是只能夠過指令的方式，NPM也有提供網站做套件資訊的查詢，網址如下: <https://www.npmjs.com>。

## 【Key Points】：

透過 npm search 這個指令查詢套件

請留意一下 NAME 那欄的內容，後來安裝模組套件時會用到

透過 npm view 指令查詢該套件的詳細資訊

# 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

4-6



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)

## 4-2：建立專案資料夾

- 什麼是專案資料夾
- 建立專案資料夾
- 以 VS Code 開啟專案資料夾
- 確認專案路徑



designed by freepik



designed by freepik

4-7

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

接下來，我們要一起學習：

- 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。
- 啟動「命令提示字元」
- mkdir 資料夾名稱
- 在檔案總管，以滑鼠右鍵點按特定資料夾，然後從快捷功能表選擇「新增 | 資料夾」
- 輸入: 「code .」 (code + 空隔 + 句號。code 是程式，句號代表目前資料夾)
- 檔案總管右鍵點按 myapi 資料夾，從快捷功能表選「以Code開啟」
- 啟動 VS Code，功能表: 檔案 | 開啟資料夾。
- 「Ctrl + 撇號」可開啟 terminal 終端機視窗
- 確認路徑是否是我們的專案資料夾
- 確保輸入指令乃至後續開發寫程式時，都作用在正確的位置

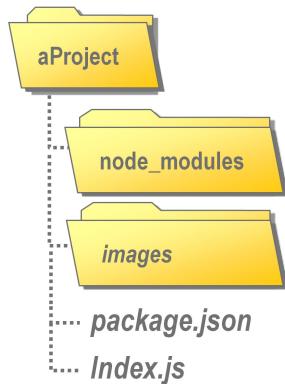
什麼是專案資料夾

建立專案資料夾：

以 VS Code 開啟專案資料夾

# 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



4-8

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

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。

# 建立專案資料夾

- 啟動「命名提示字元」
  1. 輸入：`mkdir myapi` 建立名為 myapi 的資料夾作為專案名稱。
  2. 輸入：`cd myapi` 切換到該目錄
- 另外還有一個作法：( 檔案總管 )
  1. 在檔案總管，以滑鼠右鍵點按特定資料夾(例如 c:\work)。
  2. 從快捷功能表選擇「新增 | 資料夾」
  3. 輸入資料夾名稱。

4-9



讓我們從頭開始創建一個 Node.js 專案。

在 c:\Work ( 例如 ) 建立一個叫做 myapi 的資料夾作為專案名稱 ( 也可改為你想要的名字 ) 。

1. 按下組合鍵「Windows + R」
2. 輸入「cmd」後，點按「確定」按鈕。 ( 啟動「命令提示字元」 )
3. 上述步驟1+2, 也可以: 按下組合鍵「Windows + S」，然後鍵盤輸入「cmd」找到「命令提示字元」。
4. 輸入：`mkdir myapi` 建立名為 myapi 的資料夾作為專案名稱。
5. 輸入：`cd myapi` 切換到該目錄

另外還有一個作法: 在檔案總管，以滑鼠右鍵點按特定資料夾(例如 c:\work)，然後從快捷功能表選擇「新增 | 資料夾」，接下來，輸入資料夾名稱。

## 【Key Points】：

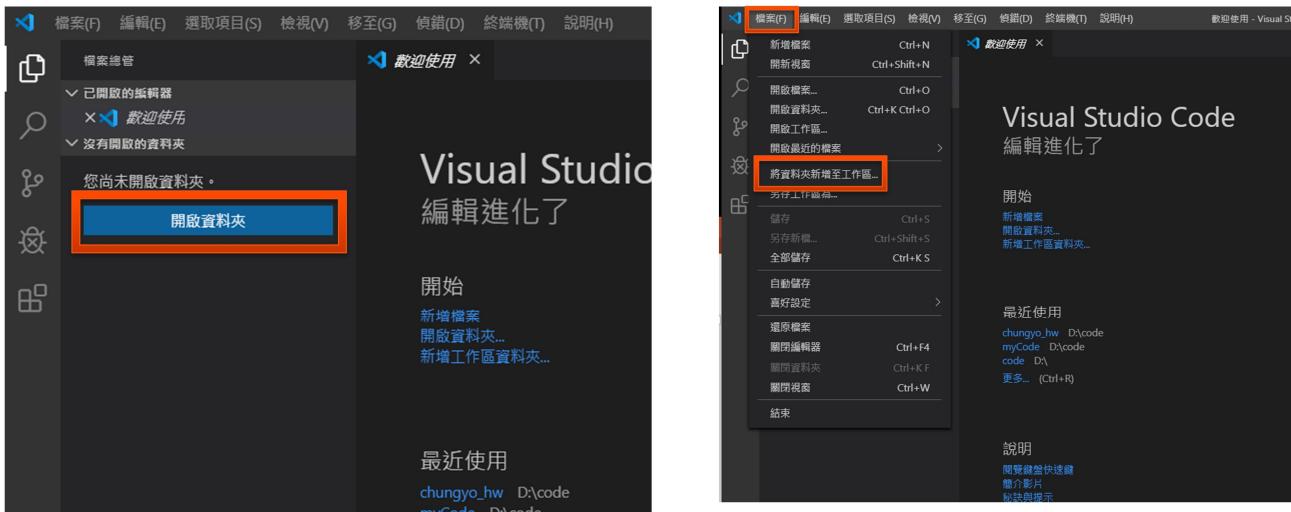
啟動「命令提示字元」

`mkdir` 資料夾名稱

在檔案總管，以滑鼠右鍵點按特定資料夾，然後從快捷功能表選擇「新增 | 資料夾」

# 以 VS Code 開啟專案資料夾

- 啟動 VS Code，功能表：檔案 | 開啟資料夾。
- 啟動 VS Code，功能表：檔案 | 將資料夾新增至工作區。



4-10

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

有許多方式可以 Visual Studio Code 開啟專案：

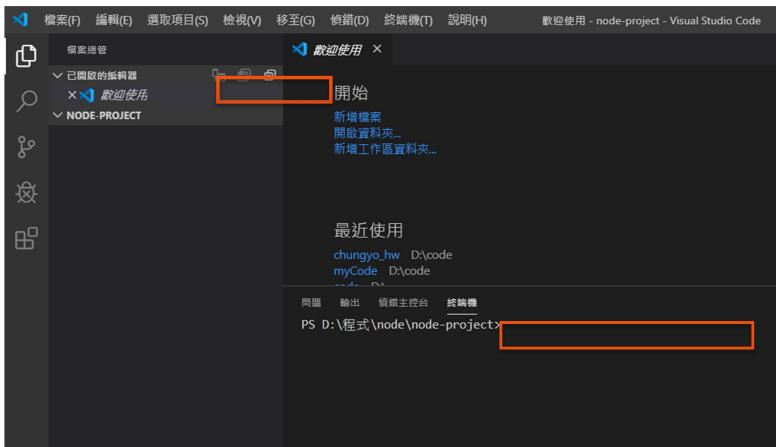
- 承接上一頁的內容，cd 到 myapi 資料夾後，輸入：「code .」（code + 空隔 + 句號。code 是程式，句號代表目前資料夾）
- 也是承接上一頁的內容，在檔案總管右鍵點按 myapi 資料夾，從快捷功能表選「以Code開啟」
- 第一次啟動 VS Code (或上次結束時有關閉資料夾)，操作畫面的左方會有「開啟資料夾」按鈕
- 啟動 VS Code，功能表：檔案 | 開啟資料夾。
- 啟動 VS Code，功能表：檔案 | 將資料夾新增至工作區。

## 【Key Points】：

輸入：「code .」（code + 空隔 + 句號。code 是程式，句號代表目前資料夾）  
檔案總管右鍵點按 myapi 資料夾，從快捷功能表選「以Code開啟」  
啟動 VS Code，功能表：檔案 | 開啟資料夾。

## 確認專案路徑

- 打在 Visual Studio Code 按下「Ctrl + 滑鼠」開啟終端機視窗
- 確認路徑是否是我們的專案資料夾



4-11



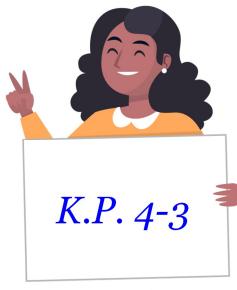
- 在 Visual Studio Code 按下「Ctrl + 滑鼠」開啟 terminal 終端機視窗
- 確認路徑是否是我們的專案資料夾
- 需要查看路徑是因為：我們要確保輸入指令乃至後續開發寫程式時，都作用在正確的位置。
- 例如，在我們輸入 `npm install express` 時，不會將檔案錯放到別的 `node_modules` 資料夾。
- 又例如：假設我們專案的主程式是 `index.js`，在我們輸入 `node index.js` 時，也才能順利載入 `index.js` 執行。

### 【Key Points】：

「Ctrl + 滑鼠」可開啟 terminal 終端機視窗  
確認路徑是否是我們的專案資料夾  
確保輸入指令乃至後續開發寫程式時，都作用在正確的位置

## 4-3：初始化專案

- 專案初始化語法
- **package.json**
- 匯入引用套件
- **npm start**



designed by freepik



designed by freepik

4-12

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

接下來，我們要一起學習：

- 專案初始化語法
- 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。
- 初始專案後會產生一個名為 package.json 的檔案
- ^ 表更新至最新的次版本（不會更新主版本）
- ~ 表更新至最新的更新（Patch），不會更動次版本
- 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
- 在 package.json 文件裡面，可使用 scripts 定義腳本命令。
- npm start
- npm run

【Key Points】：

專案初始化語法

package.json

npm start

# 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) [
```

4-13



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】：

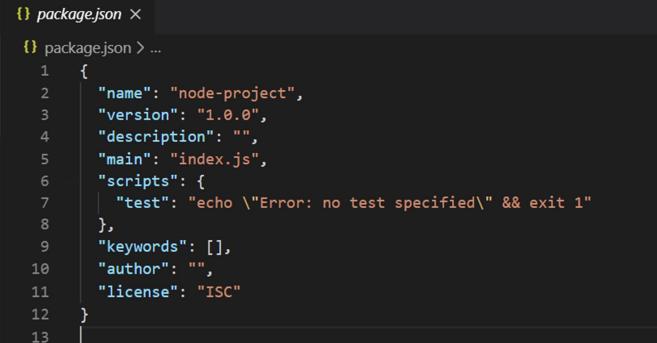
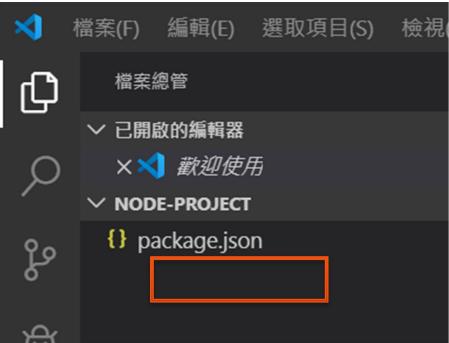
npm init

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。

# package.json

- 初始化之後會產生一個 package.json。
- package.json 是一個 json 格式的說明文件。



```
package.json
{
  "name": "node-project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

4-14

- package.json 內包含了應用程式名稱、版本、描述、關鍵字、授權、貢獻者、維護者、腳本等。
- 其中一個最重要的就是相依套件。
- 之後我們所安裝的套件都會放在 dependencies。
- dependencies 是以套件名稱和版本號組成的。
- 想要查看 package.json 更詳細的介紹可以到 NPM 網站中的 npm-package.json，以下是網址 <https://docs.npmjs.com/files/package.json>。

<Note> 關於 package.json 之 dependencies 所使用的版號標記符號，說明如下：

^ 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】：

初始化專案後會產生一個名為 package.json 的檔案

^ 表更新至最新的次版本 (不會更新主版本)

~ 表更新至最新的更新 (Patch)，不會更動次版本

# 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

## 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"  
  }  
}
```

4-17



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

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

投影片的畫有 **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**

# summary 〈精華回顧〉

- 基本NPM指令
- 什麼是專案資料夾
- 建立專案資料夾
- 以 VS Code 開啟專案資料夾
- 專案初始化
- package.json



4-18

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

剛才，我們一起學習：

- 啟動「命名提示字元」，輸入 NPM 的基本指令：
- 查詢NPM版本: `npm -v`
- 查詢NPM指令用法: `npm help`
- `npm install` 套件名稱
- `npm uninstall` 套件名稱
- 使用資料夾以及子資料夾，來收納專案所需的各類檔案
- 以`mkdir` 資料夾名稱或在檔案總管，以滑鼠右鍵點按特定資料夾，然後從快捷功能表選擇「新增 | 資料夾」
- 檔案總管右鍵點按 `myapi` 資料夾，從快捷功能表選「以Code開啟」來開啟專案資料夾，或者，輸入：「`code .`」 (`code` + 空隔 + 句號。`code` 是程式，句號代表目前資料夾)
- 專案初始化語法 (`npm init`)
- 學習 `package.json` 的檔案內容
- 知道了套件版本的`^` 表更新至最新的次版本 (不會更新主版本) · `~` 表更新至最新的更新 (Patch)，不會更動次版本
- 也學習如何在 `package.json` 文件裡面，使用 `scripts` 定義腳本命令。

【Key Points】：

基本命令列指令  
建立專案資料夾  
初始化專案

# 線上程式題

- 4-1 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？
- 4-2 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？
- 4-3 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？

4-19



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

4-20



## 【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\_04\_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

4-21



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」