



# 建立 HTTP Server



Estimated time:

50 min.

 資訊工業策進會 Institute for Information Industry

【Key Points】：

## 學習目標

- 6-1: http 套件
- 6-2: 官方文件
- 6-3: 建立簡易 Web Server



6-1

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

我們即將一起學習：

- WWW運作的概要，包括 URI簡介、HTML簡介、HTTP簡介、HTTP如何運作、網頁如何運作、常見的 HTTP 請求(POST,GET,PUT,DELETE)等等。
- 建立伺服器
- 關於Callback函式
- 輸入與輸出
- createServer 內的回呼函式
- 啟動 HTTP Server
- 設定port
- 安裝nodemon與執行nodemon

【Key Points】：

WWW運作的概要

建立伺服器

啟動 HTTP Server

## 6-1: HTTP 套件

- WWW運作的概要，包括 URI簡介、HTML簡介、HTTP簡介、HTTP如何運作、網頁如何運作、常見的 HTTP 請求(POST,GET,PUT,DELETE)
- HTTP套件



designed by freepik



designed by freepik

6-2

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

本知識點重要內容：

1. 學習什麼是HTTP套件，獲得關於HTTP套件的介紹
2. Web Server的概念
3. Web Server的特點
4. Web Server的溝通標準
5. 了解HTTP協議的基礎知識

【Key Points】：

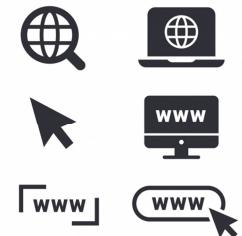
了解HTTP協議的基礎知識

什麼是HTTP套件

Web Server的概念

# WWW簡介

- **World Wide Web (全球資訊網)**
- 本體為互相連結的 **Hypertext** 超文字系統 (文字、圖像、超連結)
- **三項核心技術構成：**
- 全球網路資源識別用 **URI**
- 超文件標記語言 **HTML**
- 超文字傳輸協定 **HTTP**



designed by freepik

6-3



在介紹 HTTP 協定之前我們先來了解一下 “WWW” ，相信這是學員們在網路世界最熟悉的英文名詞之一

1. WWW 是全球資訊網 World Wide Web的縮寫
2. 他是由世界各地互相連結的網頁所組成的
3. 網頁最基本的文本格式為 HTML
4. 互相把網頁連結起來所需要的超連結格式為 URI
5. 而用於傳輸網頁內容的協議為 HTTP

上述三樣核心技術創造了現今你我所不可或缺的網路服務之一。

我們來逐一介紹一下這三項技術的內容

## 【Key Points】：

WWW 是全球資訊網 World Wide Web的縮寫

傳輸網頁內容的協議為 HTTP

文本格式為 HTML

# URI簡介

- **URL**
- 描述了資源主要的訪問機制，用來定位資源，例如：網址。
- 組成格式為 協議:// 目錄A / 目錄 B / 資源.txt

**https :// ccu.edu.tw : 8080 / path / [? query] / [# fragment]**  
協議 :// Host : Port / 路徑 / 資源查詢 / 標記片段

- **URN**
- 資源在全域的唯一名稱，例如：ISBN。

6-4



說到這邊細分的 URL 跟 URN，相信大家對 URL 比較有聽過。

1. URL就是用來描述網路資源的訪問方法。
2. 除了資源的路徑外，還有查詢參數跟標記的功能(用？跟#符號來識別)
3. URL除了http之外還有其他常用的協議，如ftp、file、mailto。
4. URN這個名詞可能大家比較陌生了，他是資源在全域的唯一名稱(代號)
5. 實際很常見比如說國際標準書碼 ISBN
6. 你可以把它想像成是資源的身分證字號

## 【Key Points】：

URL就是用來描述網路資源的訪問方法

URL除了http之外還有其他常用的協議，如ftp、file、mailto

URN是資源在全域的唯一名稱(代號)

# HTML簡介

- **HyperText Markup Language** (超文本標記語言)
- 用來建立網頁的 **標記語言** (不是程式語言ㄛ!!)
- 描述網頁的結構，由瀏覽器解析
- 跟 **CSS**、**JavaScript** 搭配使用，讓內容更豐富

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>This is a title</title>
5   </head>
6   <body>
7     <p>Hello world!</p>
8   </body>
9 </html>
```

6-5



1. HTML (HyperText Markup Language) · 超文本標記語言
2. 這是用來描述網頁結構的一種標記語言，供瀏覽器解析。如果我們查看網頁原始碼的話就會看到HTML。
3. HTML標籤可以 inline style，來規範元素的樣式 (顏色、大小)  
`<p style="color:red;">紅色的字</p>`
4. 不過為了方便維護通常會寫成CSS文件加載進來。
5. 記得 HTML 不是程式語言ㄛ!! 當有人問你會什麼程式語言時可別說 HTML 鬧笑話了

## 【Key Points】：

HTML (HyperText Markup Language) · 超文本標記語言

描述網頁結構的一種標記語言

HTML 不是程式語言

# HTTP簡介

- **HyperText Transfer Protocol** (超文字傳輸協定)
- 提供**發布、接收** HTML 頁面的協定，標示在 URI (**http**、**https**)
- **RFC7231** 規範了 Client 的請求方式跟 Server 回應的狀態碼

## 請求方法

- GET
- POST
- HEAD
- PUT
- DELETE
- TRACE
- OPTIONS
- CONNECT

## 狀態碼

- 1XX 已被server接收，正在處理
- 2XX 成功請求、處理完畢
- 3XX 需要後續操作才能完成 (跳轉)
- 4XX 有語法錯誤或無法執行請求
- 5XX 伺服器處理正確請求時錯誤

6-6



1. HTTP (HyperText Transfer Protocol)
2. 從名稱便知道它是拿來傳遞超文本的協定
3. 那麼傳遞訊息一定會有客戶端(client)跟伺服器端(server)，我們等等會說明它們實際上怎麼溝通的
4. RFC7231中有規範HTTP的Client的請求方法與Server回應的狀態碼
5. 投影片上有列舉一些常見用法，更詳盡的說明可以翻閱 <https://tools.ietf.org/html/rfc7231>

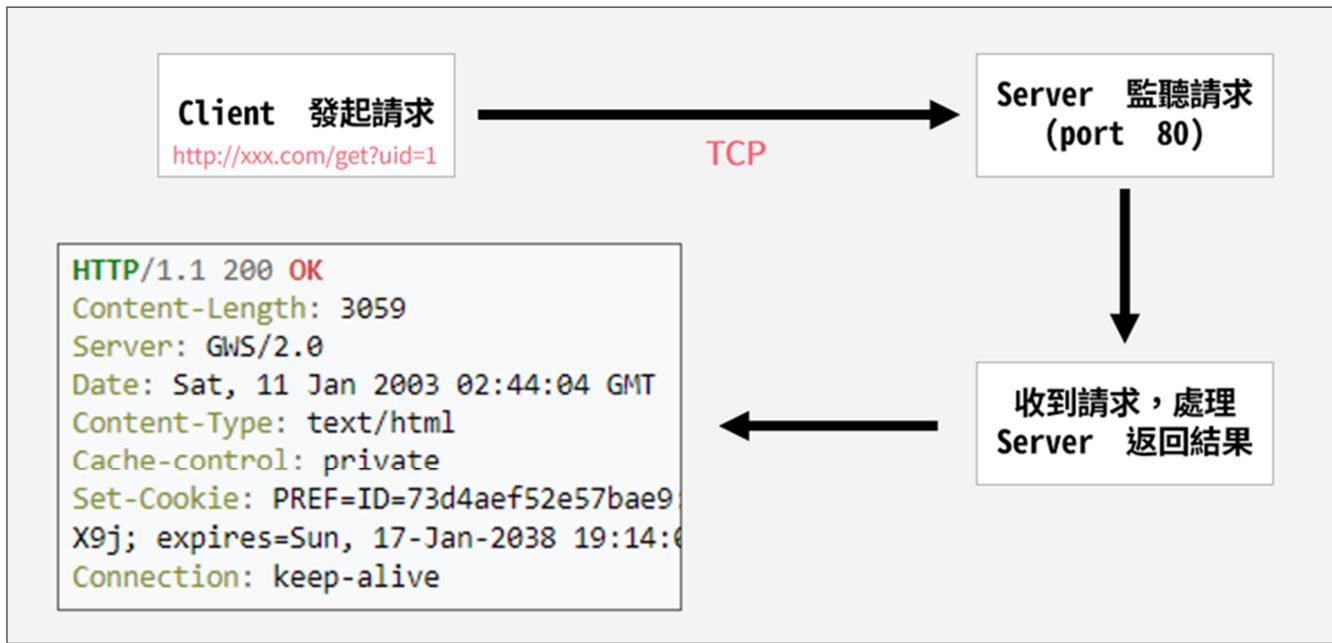
## 【Key Points】：

HTTP (HyperText Transfer Protocol)

傳遞超文本的協定

RFC7231中有規範HTTP的Client的請求方法與Server回應的狀態碼

# HTTP如何運作



6-7

1. 通常每個連線都是由HTTP用戶端發起一個請求
2. 建立一個到伺服器指定埠（預設是80埠）的TCP連接。
3. HTTP伺服器則在那個埠監聽用戶端的請求。
4. 一旦收到請求，伺服器會向用戶端返回一個狀態，比如“**HTTP/1.1 200 OK**”
5. 還有要返回的內容，如請求的檔案、錯誤訊息、或者其它訊息。

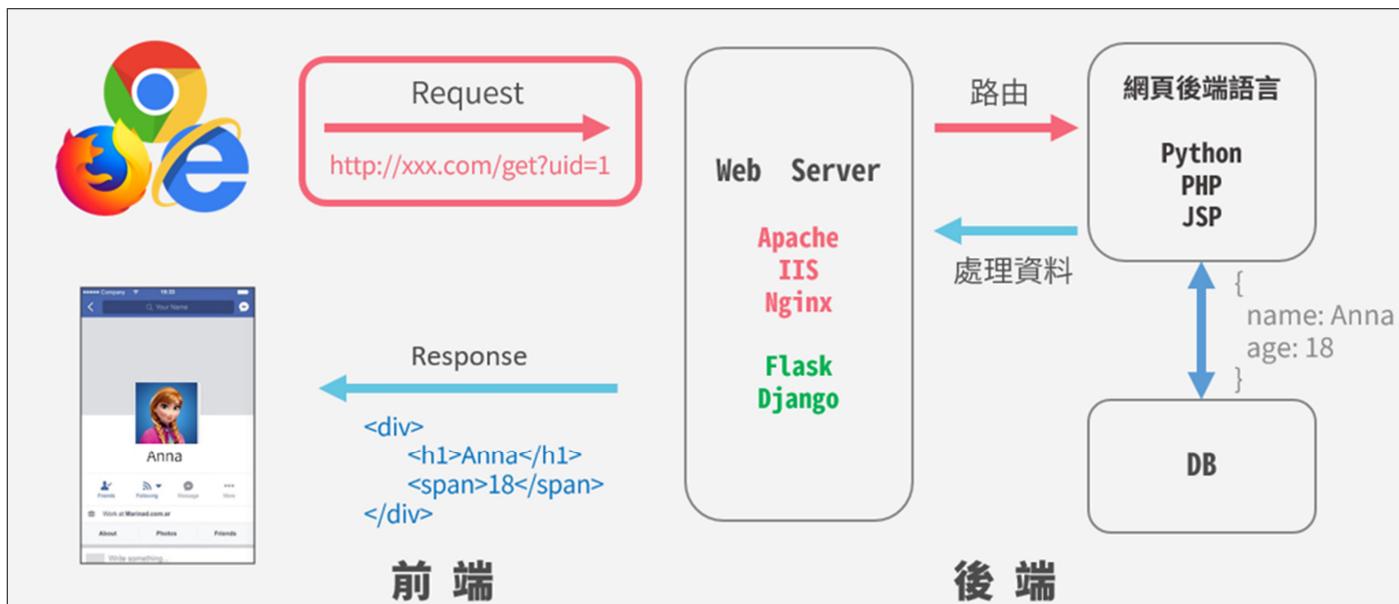
## 【Key Points】：

HTTP伺服器監聽用戶端的請求

由HTTP用戶端發起一個請求

一旦收到請求，伺服器會向用戶端返回一個狀態，比如“**HTTP/1.1 200 OK**”

# 網頁如何運作



6-8

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

1. 這張圖大概講一下**前端**跟**後端**分別在整個網站瀏覽流程的部分
2. 基本上瀏覽器上面呈現的種種畫面、元素都屬於**前端**範疇。
3. 而消息傳遞到**Web server**之後所開始處理的路由選擇、邏輯處理、資料存取等動作皆屬於**後端**。
4. 以**Node.js** 為例，你可以想像留在終端機上跑的那些程式都是屬於**後端**
5. 在瀏覽器上跑的 **HTML**、**JS**、**CSS** 都屬於**前端**範疇

## 【Key Points】：

基本上瀏覽器上面呈現的種種畫面、元素都屬於**前端**範疇  
消息傳遞到**Web server**之後所開始處理的路由選擇、邏輯處理、資料存取等動作皆屬於**後端**  
以**Node.js** 為例，你可以想像留在終端機上跑的那些程式都是屬於**後端**

# 常見的 HTTP 請求(一)

## GET

語義:  
獲取、查詢、檢索

描述:  
給我「<http://xx.com/Anna>」這個資源

大部分上網的請求都是GET，  
他會把表單的參數置於URI上。  
<http://xx.com/hello?name=Anna>

## POST

語義:  
提交、發布

描述:  
提交一份資料表單  
新增 or 建立資源 (上傳檔案)

僅次於GET，最常見的用法，  
主要參數置於Message body上。  
Demo <http://live.ccu.edu.tw/admin/>

6-9



•回到 Client發起請求的部分，由上上一篇投影片有提到HTTP有幾種請求方法，而我們取其中最常見的四種：GET、POST、PUT、DELETE做介紹。

- GET:

GET是最常見的HTTP方法，用途是Client向伺服器請求某目標的資源，然後Server會回傳該資源的 representation，比如說網頁(HTML)。

因此 GET是資訊檢索最主要的機制，也是效能優化的重點 (對搜尋引擎尤其重要)。

GET請求時如果有需提供參數的話，會以 key-value的方式置於 URI query的部分。也就是說你的參數值會出現在瀏覽器網址列的上頭，而且是明碼的！

- POST:

POST是僅次於GET常見的用法，跟GET不同的是它會把參數置放在Message body中，由於不會赤裸裸地出現在網址列上頭，因此是相對安全的。

但其實還是不安全，如果沒有使用HTTPS，Message body依然可以被中間人(intermediaries)輕易的察看到、甚至修改內容。

\* GET、POST的用法與更詳細的介紹請查看 <https://notfalse.net/44/http-get-vs-post>

### 【Key Points】：

GET是最常見的HTTP方法，用途是Client向伺服器請求某目標的資源

POST跟GET不同的是它會把參數置放在Message body中

如果沒有使用HTTPS，Message body依然可以被中間人(intermediaries)輕易的察看到、甚至修改內容

## 常見的 HTTP 請求(二)

PUT

DELETE

語義:

置換、更新

描述:

以請求payload更新特定資源

語義:

刪除

描述:

移除 目標中所匹配到的資源

6-10



### 1. PUT:

PUT，用途是替換、更新目標內容，跟POST一樣它把要更新的參數與內容放置於Message body中，如果當前要更新的內容不存在，即做”新增”操作。

### 2. DELETE:

DELETE，如其名就是刪除目標內容，跟GET一樣它把要查詢的參數置於URI的query中。

如果DELETE操作成功那伺服器會回應 200 (OK) 的狀態

如果請求成功但伺服器沒執行操作會以 202(Accepted)回應

若請求成功但無進一步資訊回傳會以 204(No content)回應

\* PUT、DELETE可參考 <https://notfalse.net/45/http-head-put-delete>

### 【Key Points】：

PUT，用途是替換、更新目標內容

DELETE，如其名就是刪除目標內容

HTTP 回應 200 表示一切正常

## 關於HTTP套件

- HTTP 套件是內建於Node.js的套件
- Node.js專案不需另外 npm install 安裝套件便可以使用 http 模組
- 以 var http = require("http") 語法，匯入模組
- 此為基礎的HTTP Server，通常我們會使用如express套件來協助開發。
- 可以用簡單的幾行程式碼建立一個Web Server
- 監聽某個port並建立網站伺服器

6-11



1. HTTP 套件是內建於Node.js的套件
2. Node.js專案中不須另外下npm install 安裝此套件便可以使用
3. 此為基礎的HTTP Server，通常我們會使用如express套件來協助開發。
4. 可以用簡單的幾行程式碼建立一個Web Server
5. 監聽某個port並建立網站伺服器

### 【Key Points】：

HTTP 套件是內建於Node.js的套件

用簡單的幾行程式碼建立一個Web Server

監聽某個port並建立網站伺服器

## 6-2：官方文件

- HTTP 伺服器程式
- 建立伺服器
- 關於Callback函式
- 輸入與輸出



designed by freepik



designed by freepik

6-12

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

本知識點重要內容：

1. 透過簡單的程式碼建立HTTP Server
2. 透過瀏覽器訪問自己建立的HTTP Server
3. 了解程式碼的內容
4. 更改HTTP Server的port並透過瀏覽器訪問
5. 更改HTTP Server回應給瀏覽器的訊息

【Key Points】：

建立HTTP Server

透過瀏覽器訪問自己建立的HTTP Server

更改HTTP Server回應給瀏覽器的訊息

# HTTP 伺服器程式

- 稍後我們要逐一介紹下列程式中的各個模組、物件與方法：

```
var http = require("http");
var host = "127.0.0.1";
var port = 80;
var server = http.createServer( function (request, response) {
    console.log("Got a request: " + request.url);
    response.writeHead(200, {"Content-type": "text/plain"});
    response.end("Hello! I'm a Node.JS server.");
});
server.listen(port, host, function () {
    console.log("Server started");
});
```

6-13



請先看一下程式。

稍後我們要逐一介紹下列程式中的各個模組、物件與方法：

```
var http = require("http");
var host = "127.0.0.1";
var port = 80;
var server = http.createServer( function (request, response) {
    console.log("Got a request: " + request.url);
    response.writeHead(200, {"Content-type": "text/plain"});
    response.end("Hello! I'm a Node.JS server.");
});
server.listen(port, host, function () {
    console.log("Server started");
});
```

主要有兩個重點：

首先，呼叫 `http.createServer()` 並指定每次有用戶端連線提出請求時，要執行的程式。

其次，呼叫 `listen(埠號)` 開始監聽並接受用戶端連線

## 【Key Points】：

以 `require()` 引用 `http` 模組

呼叫 `http.createServer()` 並指定每次有用戶端連線提出請求時，要執行的程式

呼叫 `listen(埠號)` 開始監聽並接受用戶端連線

# 建立伺服器

1. 以 `require()` 引用 `http` 模組
2. 呼叫 `http.createServer()` 並指定每次有用戶端連線提出請求時，要執行的程式
3. 呼叫 `listen(埠號)` 開始監聽並接受用戶端連線

```
var http = require("http");

var server = http.createServer( function (req, res) {
    // 每次有用戶端連線提出請求時，要執行的程式
});

server.listen(port, host, function () {
    console.log("Server started");
});
```

6-14



1. 以 `require()` 引用 `http` 模組
  2. 呼叫 `http.createServer()` 並指定每次有用戶端連線提出請求時，要執行的程式
  3. 呼叫 `listen(埠號)` 開始監聽並接受用戶端連線
- 傳入 `http.createServer()` 的回呼函數，必須有兩個參數，依序是記錄 HTTP 請求內容的 `request`，經常看到將此參數命名為 `req` 用來回應訊息給瀏覽器的 `response`，經常看到將此參數命名為 `res`

`require()` 在Node.js官方網站的說明文件:

[https://nodejs.org/api/modules.html#modules\\_require\\_id](https://nodejs.org/api/modules.html#modules_require_id)

`createServer()` 在Node.js官方網站的說明文件::

[https://nodejs.org/api/http.html#http\\_http\\_createserver\\_options\\_requestlistener](https://nodejs.org/api/http.html#http_http_createserver_options_requestlistener)

`listen()` 在Node.js官方網站的說明文件::

[https://nodejs.org/api/http.html#http\\_server\\_listen](https://nodejs.org/api/http.html#http_server_listen)

【Key Points】：

以 `require()` 引用 `http` 模組

呼叫 `http.createServer()` 並指定每次有用戶端連線提出請求時，要執行的程式

呼叫 `listen(埠號)` 開始監聽並接受用戶端連線

# 關於Callback函式

- 同伺服器回應的根據:callback 函式
- **callback function 定義:在一個函數中調用另外一個函數就是callback**
- 透過Node.js 建立Web Server後，瀏覽器端會傳需求(request)給我們的Web Server，而這時候Web Server就要根據瀏覽器傳來的訊息回覆(response)給瀏覽器。
- 在實做的過程中使用的就是**callback function**，透過**callback function**針對瀏覽器的需求做處理。

6-15



1. 同伺服器回應的根據: callback 函式
2. **callback function 定義:在一個函數中調用另外一個函數就是callback**
3. 透過Node.js 建立Web Server後，瀏覽器端會傳需求(request)給我們的Web Server，而這時候Web Server就要根據瀏覽器傳來的訊息回覆(response)給瀏覽器。
4. 在實做的過程中使用的就是**callback function**，透過**callback function**針對瀏覽器的需求做處理。
5. Callback 函式就是在函式中使用另一個函式

## 【Key Points】：

在一個函數中調用另外一個函數就是**callback**

Web Server就要根據瀏覽器傳來的訊息回覆(response)給瀏覽器

透過**callback function**針對瀏覽器的需求做處理

# 輸入與輸出

- `createServer( function (req, res) { ... } );`
- `Req` 記錄 HTTP 請求內容，例如 `req.url` 記載了當次請求的網址
- `res` 則是用來回應訊息給瀏覽器
  - `res.writeHead("...")` 用來輸出 HTTP 標頭 (headers)
  - `res.write("...")` 用來輸出內容到瀏覽器
  - `res.end("...")` `res.write("...") + 結束當次請求。`

```
var server = http.createServer( function (req, res) {
    console.log("Got a request: " + req.url);
    res.writeHead(200, {"Content-type": "text/plain"});
    res.end("Hello! I'm a Node.JS server.");});
```

6-16



拿人類舉例來說，有耳朵可聽（輸入），有嘴巴可說（輸出）。我們的伺服器程式，`req` 記錄 HTTP 請求內容，例如 `req.url` 記載了當次請求的網址。`res` 則是用來回應訊息給瀏覽器。

```
var server = http.createServer( function (req, res) {
    console.log("Got a request: " + req.url);
    res.writeHead(200, {"Content-type": "text/plain"});
    res.end("Hello! I'm a Node.JS server.");
});
```

`res.writeHead("...")` 第一個參數是回應碼；第二個則是物件，格式：{ "標頭名稱": "標頭內容", ... }  
`res.write("...")` 用來輸出內容到瀏覽器  
`res.end("...")` 與`res.write("...")`類似，但是在輸出內容後，結束這一回合。

`res.write()` 在Node.js官方網站的說明文件

[https://nodejs.org/api/http.html#http\\_response\\_write\\_chunk\\_encoding\\_callback](https://nodejs.org/api/http.html#http_response_write_chunk_encoding_callback)

`res.writeHead()` 在Node.js官方網站的說明文件

[https://nodejs.org/api/http.html#http\\_response\\_writehead\\_statuscode\\_statusmessage\\_headers](https://nodejs.org/api/http.html#http_response_writehead_statuscode_statusmessage_headers)

【Key Points】：

`req` 記錄 HTTP 請求內容，  
`res` 則是用來回應訊息給瀏覽器  
以 `res.write("...")` 輸出內容到瀏覽器

## 6-3: 建立簡易 Web Server

- 建立HTTP Server
- createServer 內的回呼函式
- 設定port、啟動 HTTP Server
- 安裝nodemon與執行nodemon



designed by freepik



designed by freepik

6-17

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

本知識點重要內容:

1. 透過簡單的程式碼建立HTTP Server
2. 透過瀏覽器訪問自己建立的HTTP Server
3. 了解程式碼的內容
4. 更改HTTP Server的port並透過瀏覽器訪問
5. 更改HTTP Server回應給瀏覽器的訊息

【Key Points】：

建立HTTP Server

透過瀏覽器訪問自己建立的HTTP Server

更改HTTP Server回應給瀏覽器的訊息

# 建立HTTP Server

- 新增一個名為 app.js 的檔案，內容：
  - `var http = require('HTTP');`
  - `var server = http.createServer(function(req, res) {  
 res.writeHead(200, {'Content-Type': 'text/html'});  
 res.write('Node.js');  
 res.end('Hello World');`
  - `});`
  - `server.listen(80);`

6-18



1. `var http = require("http"); // 引用HTTP 套件`
2. `http.createServer`是HTTP套件提供的函式(function)
3. 傳給 `createServer()` 的回呼函式，稍後會另外介紹
4. 此段程式碼主要是利用Node.js建立一個簡單的Web Server
5. 最後，監聽埠號 80，等待用戶端連線

【Key Points】：

```
var http = require("http"); // 引用HTTP 套件  
var server = http.createServer();  
server.listen()
```

# createServer 內的回呼函式

- 右邊這段程式碼翻譯成白話大概是：
- 每當有用戶端造訪時...
- 發送 HTTP header 紿瀏覽器  
    // HTTP status code 200 : OK  
    // 內容類型:text/html
- 發送回應內容：  
"Node.js" 跟 "Hello World"

```
var http = require('HTTP');
http.createServer(function (req, res) {
  res.writeHead(200, {
    'Content-Type': 'text/html'
  });
  res.write('Node.js');
  res.end('Hello World');
}).listen(3000);
console.log("HTTP
Server is listening at port 3000.");
```

6-19



這段程式碼翻譯成白話文大概是：

每當有用戶端造訪時 –

發送 HTTP header 跟瀏覽器說我已收到你的訊息，處理之後的回覆代碼: 200  
HTTP status code 200 : OK

請瀏覽器準備接收並處理內容類型為 text/html 的資料

接下來，回應內容: "Node.js" 跟 "Hello World" 到瀏覽器

<Note>

使用埠號 3000 接聽用戶端的 request (請求)

【Key Points】：

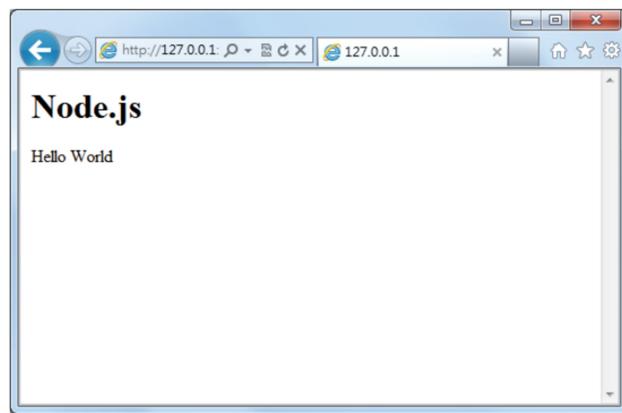
createServer(function( req, res) { })

HTTP header

res.write()

# 啟動 HTTP Server

- 執行node app.js
- 打開任一瀏覽器，前往網址<http://127.0.0.1:3000>



6-20

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

- 也可搭配前一章教學的 npm scripts 來執行 node app.js
- 127.0.0.1是指本台的 ip，又稱 localhost。
- 也可以將127.0.0.1 改成 localhost
- 3000 是我們的port
- Port 類似於銀行櫃檯服務窗口的編號，Port號的範圍從0到65535，比如用於流覽網頁服務的 80 Port，用於FTP服務的21 Port等等。

【Key Points】：

node app.js

127.0.0.1 等於 localhost

Port 類似於銀行櫃檯服務窗口的編號

# 設定 port

- 可將程式碼 `.listen(3000)` 中的數字更改成別的編號，調整接聽窗口的編號，例如：**16888**
- 此時，用戶端的網址就必須相應改成**http://127.0.0.1:16888**

```
var http = require('HTTP');
http.createServer(function (req, res) {
  res.writeHead(200, {
    'Content-Type': 'text/html'
  });
  res.write('Node.js');
  res.end('Hello Taiwan');
}).listen(16888);
console.log("HTTP
Server is listening at port 3000.");
```

6-21



1. 設定 port 的時候，請特別注意不能與現有的服務衝突，一個port只能由一個服務佔用，沒辦法同時一起。
2. 業界開發應用程式時，不會將port設定在1~1023之間，這些Port號已保留給傳統的網路服務。比如21 Port分配給FTP服務，25 Port分配給SMTP（簡單郵件傳輸協定）服務，80 Port分配給HTTP服務，135 Port分配給RPC（遠端程序呼叫）服務等等。
3. 通常會將port設定在1024~65535之間
4. 不同作業系統可以透過不同方式查看現在有哪些port是由什麼服務使用的，以 Windows 來說，執行 `netstat` 指令。
5. 200是HTTP狀態碼（HTTP status code）（可參考:[HTTPs://developer.mozilla.org/zh-TW/docs/Web/HTTP>Status/200](https://developer.mozilla.org/zh-TW/docs/Web/HTTP>Status/200)）

## 【Key Points】：

設定 port 的時候，請特別注意不能與現有的服務衝突  
通常會將port設定在1024~65535之間  
用戶端的網址也必須相應改變

# 什麼是nodemon

- nodemon介紹
- node \*.js 與 nodemon \*.js的差別
- nodemon常見指令
  - Debug 模式 (nodemon --debug)
  - 查看指令(nodemon -h)

6-22



1.一般在開發階段時，啟動 node 服務(程式)都是使用 node 檔名.js。然而，如果該服務出現錯誤而中斷，或者是在開發新功能時，都要一再地重新執行 node 檔名.js 指令。此時，nodemon 就能幫解決問題，只要設定好，nodemon 會盯著看看是否需要自動重啟服務，遇到錯誤時也不會再中斷服務。換句話說，寫好新程式碼後，只需要儲存程式檔案，系統即可重新執行。

2. node \*.js 不會自動重啟，nodemon可以。
3. debug 模式：nodemon --debug ./server.js
4. 查看指令用法：nodemon -h
5. 官方原始碼 [HTTPs://github.com/remy/nodemon#nodemon](https://github.com/remy/nodemon#nodemon)

## 【Key Points】：

nodemon會盯著看看是否需要自動重啟服務  
node \*.js 不會自動重啟，nodemon可以  
Nodemon 你的Node檔案.js

# 安裝nodemon與執行nodemon

- 安裝於全域  
`npm install nodemon -g`
- 安裝於專案資料夾中  
`npm install --save-dev nodemon`
- 執行:  
nodemon 你的NodeJS程式檔名.js
- 安裝於全域與專案資料夾的差別:安裝於全域，該電腦的所有 Node.js 專案都可用。

6-23



- Nodemon建議安裝以全域的方式安裝。
- 全域安裝指令: `npm install nodemon -g`
- 安裝於個別專案資料夾:  
`npm install nodemon --save-dev`
- 執行 nodemon 檔名.js
- 安裝於全域與專案資料夾的差別: 安裝於全域，該電腦的所有 Node.js 專案都可用，如果是很常用到，建議安裝於全域。

## 【Key Points】:

Nodemon建議安裝以全域的方式安裝

`npm install nodemon -g`

nodemon 檔名.js

# 執行畫面

這是透過nodemon執行程式，CLI會顯示的畫面

```
➜ nodemon index.js
[nodemon] 2.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
192.168.0.104
[nodemon] clean exit - waiting for changes before restart
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
192.168.0.104
[nodemon] clean exit - waiting for changes before restart
```

6-24



1. 可以隨時輸入 rs 重啟程式
2. 偵測到程式有變時nodemon會自動重新啟動
3. 可以監控目錄底下所有的檔案變化
4. 有任何錯誤會顯示在這邊
5. 可以透過這邊顯示的錯誤訊息來對程式除錯(debug)

## 【Key Points】：

- 可以隨時輸入 rs 重啟程式
- 偵測到程式有變時nodemon會自動重新啟動
- 可以監控目錄底下所有的檔案變化

# Summary 〈精華回顧〉

- WWW運作的概要
- HTTP套件
- 建立HTTP Server
- 啟動 HTTP Server
- 設定port
- 什麼是nodemon
- 安裝nodemon
- 執行nodemon



6-25

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

我們一起學習了：

- WWW運作的概要，包括 URI簡介、HTML簡介、HTTP簡介、HTTP如何運作、網頁如何運作、常見的 HTTP 請求(POST,GET,PUT,DELETE)
- HTTP套件
- Callback函式
- 建立HTTP Server
- createServer 內的回呼函式
- 啟動 HTTP Server
- 設定port
- 什麼是nodemon
- 安裝nodemon
- 執行nodemon

【Key Points】：

WWW運作的概要  
建立HTTP Server  
執行nodemon

# 線上程式題

- 6-1 建立 HTTP Server

某位程式設計師想用 Node.js 內建的 HTTP 模組，建立一套簡易的 Web 伺服器。請問程式漏寫了什麼？

- 6-2 輸出圖片

如果 HTTP Request 的結果是要傳一張 gif 圖檔給用戶端，請問如何指定 HTTP Header 的內容？

- 6-3 依據網址進行路由分流

<http://localhost/user1>

<http://localhost/user2>

想分別看到不同的執行結果，請問程式該怎麼寫？

Estimated time:

20 min.

6-26



題目名稱: 6-1 建立 HTTP Server

內容說明:

某位程式設計師想用 Node.js 內建的 HTTP 模組，建立一套簡易的 Web 伺服器。請問程式漏寫了什麼？

題目名稱: 6-2 輸出圖片

內容說明:

如果 HTTP Request 的結果是要傳一張 gif 圖檔給用戶端，請問如何指定 HTTP Header 的內容？

題目名稱: 6-3 依據網址進行路由分流

內容說明:

<http://localhost/user1>

<http://localhost/user2>

想分別看到不同的執行結果，請問程式該怎麼寫？

請閱讀教學系統的程式與說明，然後，到「// 作答區」填入答案。答案有區分大寫小寫。

【Key Points】：

6-1 建立 HTTP Server

6-2 輸出圖片

6-3 依據網址進行路由分流

# 課後練習題(Lab)

- **情節描述:**  
本Lab 是假設在已經安裝Node.js和Nodemon環境的Windows 電腦下，如何透過內建HTTP模組和撰寫一些程式去架好一個簡易的HTTP Server。
- **預設目標:**
  - 在本機架好一個簡易的HTTP Server
  - Server會根據用戶的Request URL去回應不同內容
- **Lab01: 安裝Node.js 和 Nodemon**
- **Lab02: 建立主要執行檔 index.js**
- **Lab03: 根據不同的Request回應內容**

Estimated time:  
**20 minutes**

6-27



## 【情節描述】

本Lab 是假設在已經安裝Node.js和Nodemon環境的Windows 電腦下，如何透過內建HTTP模組和撰寫一些程式去架好一個簡易的HTTP Server。

## 【預設目標】

- 在本機架好一個簡易的HTTP Server。
- Server會根據用戶的Request URL去回應不同內容。

情節描述:

預設目標:

Lab01: 安裝Node.js 和 Nodemon

Lab02: 建立主要執行檔 index.js

Lab03: 根據不同的Request回應內容

完成後的程式與檔案，請參考 Example 資料夾的內容。

## 【Key Points】:

Lab01: 安裝Node.js 和 Nodemon

Lab02: 建立主要執行檔 index.js

Lab03: 根據不同的Request回應內容

# 範例程式使用說明

- 範例程式資料夾: Module\_06\_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 node index.js
  6. 啟動瀏覽器，連接 http://localhost:3000

6-28



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 node index.js
6. 啟動瀏覽器，連接 http://localhost:3000

【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」