

HTTP server (httpd)

喬逸偉 (Yiwei Chiao)

1 httpd server 和檔案服務

之前的 httpd/index.js 檔案可以接受使用者連線，傳回簡單的 Hello World! 訊息；但是 [BreakIt!](#) 的內容是在 htdocs/ 資料夾下的 index.html 和 htdocs/js 資料夾下的 index.js，與 htdocs/assets/css/ 資料夾下的 styles.css 檔案。也就是說，httpd server 必需能在收到使用者要求時傳回相應的檔案內容，而不是簡單的文字而已。

要作到這個目的，需要 Node.js 的 fs 模組。

2 Node.js 的 fs 模組

針對檔案系統的讀寫，Node.js 提供了一個 fs (*File System*) 模組。

先看引入了 fs 模組的 httpd/index.js 的程式碼：

```
1. 'use strict';
2.
3. let http = require('http');
4.
5. http.createServer((request, response) => {
6.   // 取得 node.js 的 fs 模組
7.   let fs = require('fs');
8.
9.   fs.readFile('../htdocs/index.html', (err, data) => {
10.     response.writeHead(200, {
11.       'Content-Type': 'text/html'
12.     });
13.
14.     response.write(data);
15.
16.     response.end();
17.   });
```

```
18. }).listen(8088);
19.
20. // log message to Console
21. console.log(' 伺服器啟動，連線 url:  http://127.0.0.1:8088/');
```

和原來的 `index.js` 內容比較，主要的變化出現在第 6 行到第 17 行這段 `callback` 函數的內容。具體的說是：

- 第 7 行：利用 `require('fs')` 載入了 Node.js 的 `fs` (File System) 模組，並將產生的物件放入同名的 `fs` 變數內。
- 第 9 行：呼叫 `fs` 物件的 `readFile` 方法；讀入 `index.html` 檔案；有趣的在第二個參數的 `callback` 函數。這個 `callback` 函數本身需要兩個參數：
 - `err`：代表 `readFile()` 執行中發生錯誤。
 - `data`：代表讀取成功的資料。目前的 `index.js` 檔案暫時不處理錯誤，所以並沒有對 `err` 進行處理。而讀入的 `data` 就直接準備傳送給客戶端 (瀏覽器)。
- 第 10 到 16 行：和之前一樣，呼叫 `response` 三步走；不一樣的是，現在這幾行變成 `readFile(fname, callback)` 第二個參數：`callback` 函數的內容：
 - 第 10 行，`writeHead(...)`；因為傳回的資料現在是 `html`，所以 `'Content-Type'` (MIME Type) 設為 `'text/html'`。
 - 第 14 行，`write(data)`：呼叫 `response` 的 `[write][responsewrite]` 方法將讀入的資料 (`data`) 傳送給客戶端 (瀏覽器)
 - 第 16 行，`end()`：結束 `response` 物件的工作，確實將資料傳送出去。

2.1 非同步 (asynchronous) 的 `fs.readFile(...)`

如果去查 `index.js` 第 9 行的 `fs.readFile(...)` 說明文件，會注意到文件特別強調它是 *asynchronous* (非同步) 的。這是 Node.js 的一個特點。`[Node.js]``nodejs` 提供的模組裡的 `APIs` (Application Programming Interface: 應用程式介面)，除非特別聲明，或者如 `readFile(...)` 的姊妹函數 `readFileSync(...)` 般，函數名稱裡就帶有 *Sync* (*SYNChronous*)，全部都是**非同步** (*asynchronous*) 的。

所謂**非同步** (*asynchronous*) 指的是，以 `readFile(...)` 方法為例，Node.js 不會等檔案讀取完畢之後才進行下一步驟的執行；Node.js 啟動 I/O 作業，開始讀取檔案後，就去處理程式的下一個指令了；一直到 I/O 系統完成了工作，才會透過 `readFile(...)` 的 `callback` 函數，通知 Node.js 回頭進行讀取資料的後續處理。

這樣設計的好處是，同樣以 `readFile(...)` 為例，如果讀取的檔案很大，Node.js 可以不用傻傻的在那兒等檔案讀完，而可以先去忙其它事情，等到檔案讀完再回頭處理。從而最大化運算核心和記憶體的使用效率。

3 routing (路由)

修改過後的 `index.js` 執行結果如圖 Figure 1，

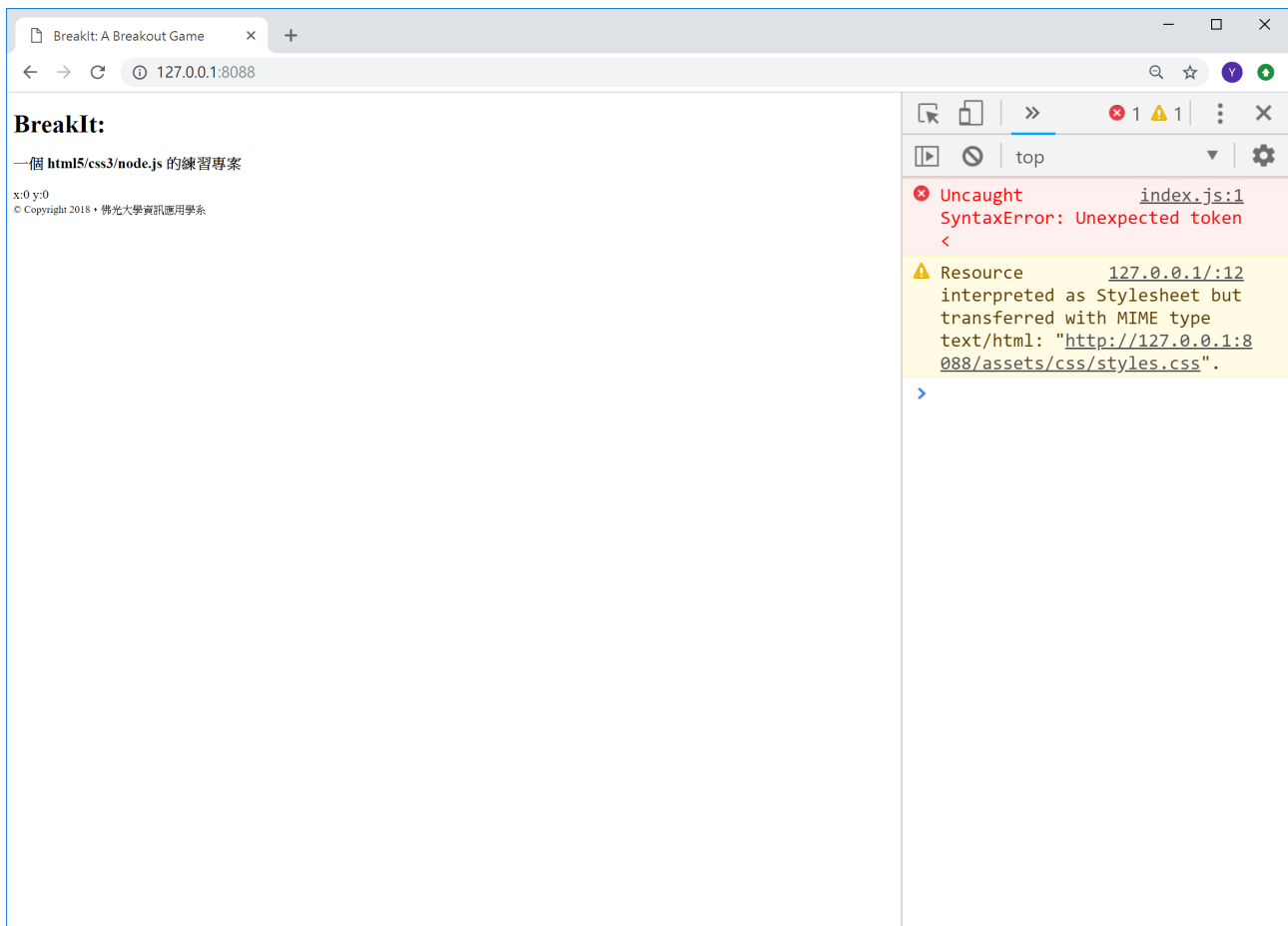


Figure 1: 修改後的 `index.js` 客戶端結果

這是因為目前 `BreakIt` 專案需要回傳給瀏覽器的資料分成三 (3) 個檔案:

- `index.html` 在 `breakit/htdocs` 資料夾下，網頁客戶端的 `HTML` 框架。
- `styles.css` 在 `breakit/htdocs/assets/css` 資料夾下，負責網頁客戶端的 `CSS` styling。
- `breakit.js` 在 `breakit/htdocs/js` 資料夾下，`BreakIt` 遊戲程式。

問題在目前伺服端的 `index.js` 檔案執行時只是簡單地讀入 `index.html` 檔案內容，並將它傳給客戶端瀏覽器；而實際上它還需要 `styles.css`, `breakit.js` 等不同形式的檔案，`index.js` 應該在 **何時**，**如何**讀取它們的內容並傳給客戶端，就是現在要處理的挑戰。

3.1 `http.IncomingMessage`

原始的 `index.js` 內容如下：

```

1. 'use strict';
2.
3. let http = require('http');
4.
5. http.createServer((request, response) => {
6.   // 傳送 HTTP header
7.   // HTTP Status: 200 : OK
8.   // Content Type: text/plain
9.   response.writeHead(200, {
10.    'Content-Type': 'text/plain'
11.   });
12.
13.   // 傳送回應內容。
14.   response.end('Hello World!\n');
15.
16.   console.log('request.headers: \n', request.headers)
17. }).listen(8088);
18.
19. // log message to Console
20. console.log(' 伺服器啟動，連線 url:  http://127.0.0.1:8088/');

```

目前關注的是第 5 行的 `http.createServer((request, response) => {`。這裡，`index.js` 建立了真正的 HTTP 伺服器物件；而 `request` 參數就是客戶端送來的請求。

據 `Node.js` 文件，`request` 物件的型別是 `http.IncomingMessage`。由 `Node.js` 文件裡對 `http.IncomingMessage` 的說明，可以找到兩個重要的資料欄位：

- `message.method`: 客戶端要求使用的方法，如：GET，POST 等；
- `message.url`: 客戶端提出要求使用的 URL (Uniform Resource Locator)，也就是一般習稱的網址。

要理解這兩個欄位的意義，可以修改 `index.js` 如下：

```

1. 'use strict';
2.
3. let http = require('http');
4.
5. http.createServer((request, response) => {
6.   request.on('end', () => {
7.     console.log(`Request method: ${request.method}`);
8.     console.log(`Request url: ${request.url}`);
9.   });
10.
11.   // 傳送 HTTP header

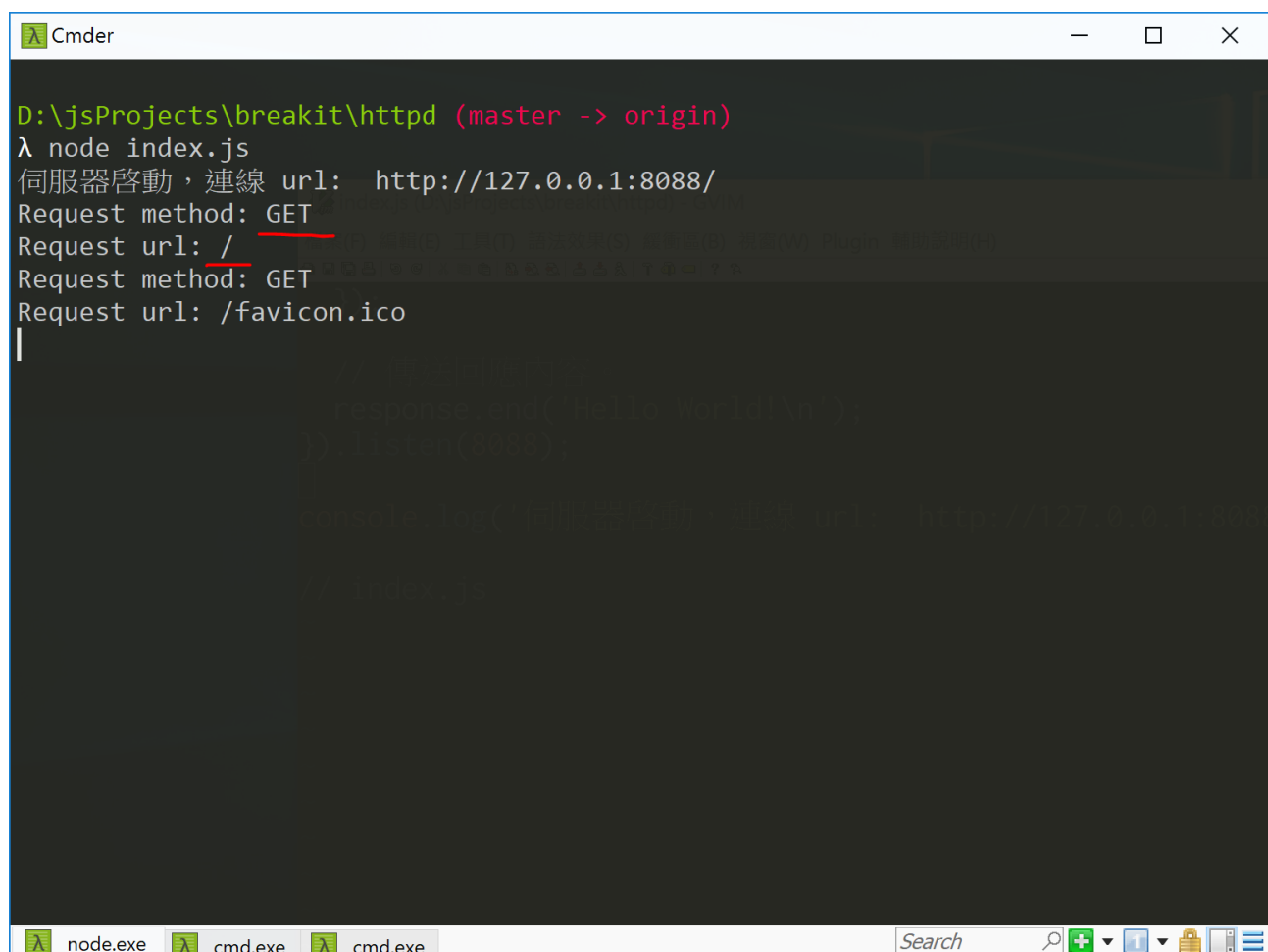
```

```

12. // HTTP Status: 200 : OK
13. // Content Type: text/plain
14. response.writeHead(200, {
15.   'Content-Type': 'text/plain'
16. });
17.
18. // 傳送回應內容。
19. response.end('Hello World!\n');
20. }).listen(8088);
21.
22. // log message to Console
23. console.log('Server running at http://127.0.0.1:8088/');

```

主要差別在增加了第 6 ~ 9 行的程式碼。其中第 6 行設定當 HTTP 伺服器完成接收 request 物件時執行；而第 7, 8 行則分別在 console 印出 request.method 和 request.url 的內容。執行結果應該有點像圖 Figure 2，



```

D:\jsProjects\breakit\httpd (master -> origin)
λ node index.js
伺服器啟動，連線 url: http://127.0.0.1:8088/
Request method: GET
Request url: /
Request method: GET
Request url: /favicon.ico

```

Figure 2: request.method 和 request.url

伺服器可以利用這兩個欄位來達成回傳不同檔案的目的。

3.2 index.js 修正

依之前對 request.method 和 request.url 的了解，index.js 可以修改如下：

```
1. 'use strict';
2.
3. let http = require('http');
4.
5. http.createServer((request, response) => {
6.   let fs = require('fs');
7.   let postData = ''; // POST 資料
8.
9.   // 利用 'data' event 消耗掉 data chunk;
10.  // 'end' event 才會被 fired
11.  request.on('data', (chunk) => {
12.    postData += chunk;
13.
14.    console.log(
15.      `接收的 POST data 片段: [${chunk}].`
16.    );
17.  });
18.
19.  request.on('end', () => {
20.    switch (request.url) {
21.      case '/':
22.        fs.readFile('../htdocs/index.html', (err, data) => {
23.          if (err) {
24.            console.log('檔案讀取錯誤');
25.          }
26.          else {
27.            response.writeHead(200, {
28.              'Content-Type': 'text/html'
29.            });
30.
31.            response.write(data);
32.            response.end();
33.          }
34.        });
35.
36.        break;
37.
38.        default:
```

```

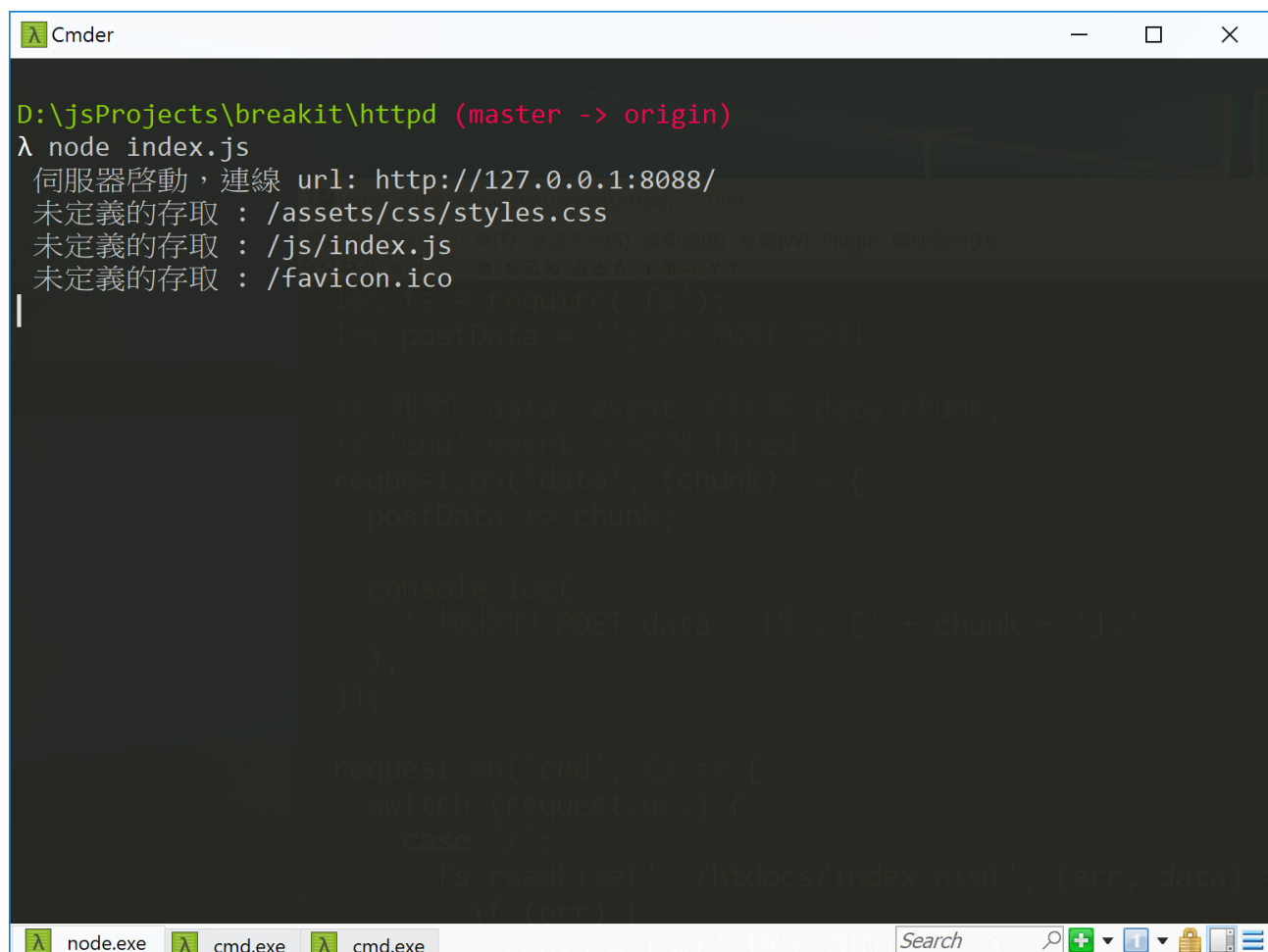
39.         console.log(`未定義的存取: ${request.url}`);
40.
41.         response.end();
42.
43.         break;
44.     }
45. });
46. }).listen(8088);
47.
48. // log message to Console
49. console.log(' 伺服器啟動，連線 url: http://127.0.0.1:8088/');

```

3.3 問題與練習

修改過後的 index.js 執行結果如圖 Figure 1，

而伺服端的輸出如圖 Figure 3，



```

D:\jsProjects\breakit\httpd (master -> origin)
λ node index.js
伺服器啟動，連線 url: http://127.0.0.1:8088/
未定義的存取 : /assets/css/styles.css
未定義的存取 : /js/index.js
未定義的存取 : /favicon.ico

```

Figure 3: 修改後的 index.js 伺服端結果

嘗試解決這個問題。