Sokoban.js 專案設定

喬逸偉 (Yiwei Chiao)

1 HTML5/CSS3

在網頁瀏覽器 (browser) 上，JavaScript (ECMAScript) 控制**程式行為** (behavior)，HTML (Hyper Text Markup Language) 決定文件的**組織結構** (structure)，而 CSS (Cascading Style Sheets) 處理**排版** (style)。三者各司其職。

Sokoban.js 專案既然是一個網頁遊戲專案，自然少不了 HTML 和 CSS。只是專案重心在 JavaScript，所以 HTML，CSS 只會簡單帶過使用到的部份。其餘更全面的介紹或進階的主題，需要去參考其它的資源 (如這裡給的連結：HTML，和 CSS)。

1.1 index.html

首先，在 sokoban/htdocs 資料夾下，建立 index.html 檔案，內容如下：

```
<!DOCTYPE html>
<html lang="zh-TW">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Sokoban: A Puzzle Game</title>
    <meta name="author" content="Yiwei Chiao">
    <meta name="description" content="A web-based Sokoban (倉庫番) game.">
    <meta name="keywords" content="Javascript, game, Sokoban">
  </head>
  <body>
    Hello World!
  </body>
</html>
```

在 index.html 的內容列表中，用 <> 框起的字串稱為**標記** (*tag*)，它們也就是 HTML 標記語言的組成部份。針對 HTML 較詳細的介紹放在這一章的後半，這裡需要注意的只是 <body> 和 </body> 夾起的 Hello World!。

準備好 sokoban/htdocs 資料夾下的 index.html 後，可以開啟瀏覽器，在瀏覽器的網址列內輸入：

- Windows: `file:///d:/sokoban/htdocs/index.html`
- Linux: `file:///home/ywchiao/sokoban/htdocs/index.html`
- MacOs: `file:///Users/ywchiao/sokoban/htdocs/index.html`

其中 Windows 的 d:，Linux/MacOS 裡的 ywchiao 請依個人情況更改。在 Linux/MacOS 系統如果不清楚路徑要怎麼打，可以在 terminal 下利用 cd 指令，切換工作目錄到 sokoban/htdocs 之後，輸入 pwd (Present Working Directory)，依螢幕輸出打就行了；而 Windows 則可以利用檔案總管，切換資料夾到 sokoban/htdocs 後，在檔管總管的瀏覽器列空白處，點一下滑鼠左鍵就可以看到要輸入的內容。

如果瀏覽器的網址列輸入正確，應該會看見如 Figuer 1 的畫面。

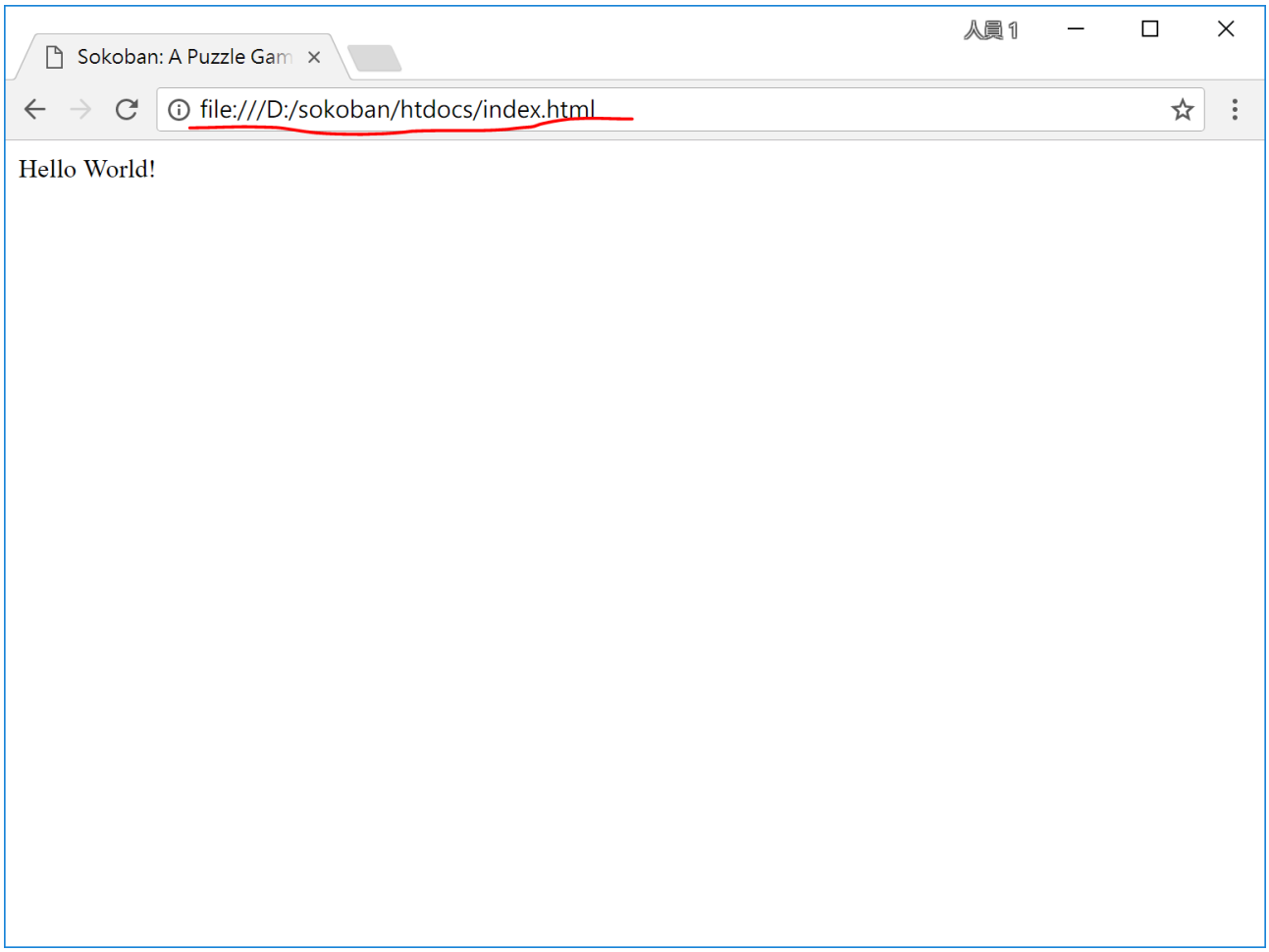

Figure 1: 瀏覽器開啟 index.html

1.1.1 HTML 標題 <h1> ... <h6>

Figure 1 看起來沒什麼不同？的確如此，因為前面提過，HTML 的用途在決定文件**結構** (structure)，而非呈現。不過，一些簡單的效果還是有的。修改:

```
html  <body>    Hello World!  </body>
```

成為：

```
html  <body>    <h1>Hello World!</h1>  </body>
```

存檔後，重新整理網頁，可以發現 Hello World! 的字型大小變了。這是因為 <h1></h1> 是 HTML 用來標記**標題** (Heading) 的 *tag*；其中，<h1> 標記標題的開始，而 </h1> 則標記標題的結束。排版習慣上，標題的字體通常會比內文大一些。所以，HTML 的 heading tags，標記的文字也會大一些。

HTML 總共定義了六 (6) 級的 heading 大小，分別以 <h1>. <h2>。一直到 <h6> 標記。可以逐一試試效果。

2 Node.js 的 fs 系統

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；htdocs/index.html 則是 HTML 版的 Hello World!。如果將兩者結合，也就是當伺服器收到使用者要求時，它會回傳 index.html 的內容；這樣的 index.js 就有點真正的網頁伺服器的樣子了。

修改過的 httpd/index.js 內容如下：

```
 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*) 指得是，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；**Node.js 啟動 I/O 作業，開始讀取檔案後，就去處理程式下一步指令了；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，等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 HTML 簡介

HTML 是 HyperText Markup Language (超文本標記語言) 的縮寫。**標記語言** (markup language) 和程式語言 (programming language) 有本質的不同。如 HTML 這樣的標記語言設計上是為**文本** (text) 的不同部份加上**標記** (tag)，方便工作人員或處理工具理解原始設計者／創作者的創作意圖，進而依據這些預先定義好的**標記**意義進行後製 (post-production) 加工。

在 Web 相關領域，目前常見的標記語言有 HTML，Markdown，XML，YAML 等。每個標記語言都有它想解決的問題和想達成的目的。

3.1 HTML 結構

HTML 採用的**標記**，稱為 HTML *tag*，都以**成對**的角括號 <...> 包夾，成 <tag> 型式；如 <h1>，<h2> 等。

之前提過，HTML 是設計來規範文件的結構。而一個最簡單的 HTML 結構大概如下所示：

```
<html>
  <head>
    <title>HTML 簡介</title>
  </head>
  <body>
    Hello HTML。
  </body>
</html>
```

由上面的 HTML 內容可以注意到幾件事情：

- HTML 檔案開頭與結束分別是 <html> 與 </html> 的 *tag* 其中 <tag> 稱為 *tag* **開始**標記，而 </tag> 則稱為 *tag* **結束**標記。
- HTML 的內容可以分為 <head></head> 和 <body></body> 兩大區塊：
  - <head></head>: 放置.html 作者想讓瀏覽器知道，除了文件結構之外，一些額外的處理**注意事項**，相關檔案，和被稱為 *meta-data* 的文件描述。在 GitHub 上有一份整理的很好的文可以參考：*HEAD*
  - <body></body>: HTML 真正要呈現的內容。
- HTML *tag* **不**區分大小寫，不過 HTML5 建議採用**全**小寫。

3.2 index.html 的 <head></head>

htdocs/index.html 裡的 <head></head> 內容如下：

```
1.  <head>
2.    <meta charset="utf-8">
3.    <meta name="viewport" content="width=device-width, initial-scale=1.0"
4.
5.    <title>Sokoban: A Puzzle Game</title>
6.    <meta name="author" content="Yiwei Chiao">
7.    <meta name="description" content="A web-based Sokoban (倉庫番) game.">
8.    <meta name="keywords" content="Javascript, game, Sokoban">
9.  </head>
```

- 第 2 行：通知瀏覽器，index.html 採用的內容編碼是 utf-8。
- 第 3 行：預設使用設備的全螢幕顯示；放大倍率是 1.0
- 第 5 行：網頁的標題 (title)；這個值會被用作網址列的內容，我的最愛，或搜尋引擎。
- 第 6~8 行：網頁基本資訊，提供給搜尋引擎或網路爬蟲處理。