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。

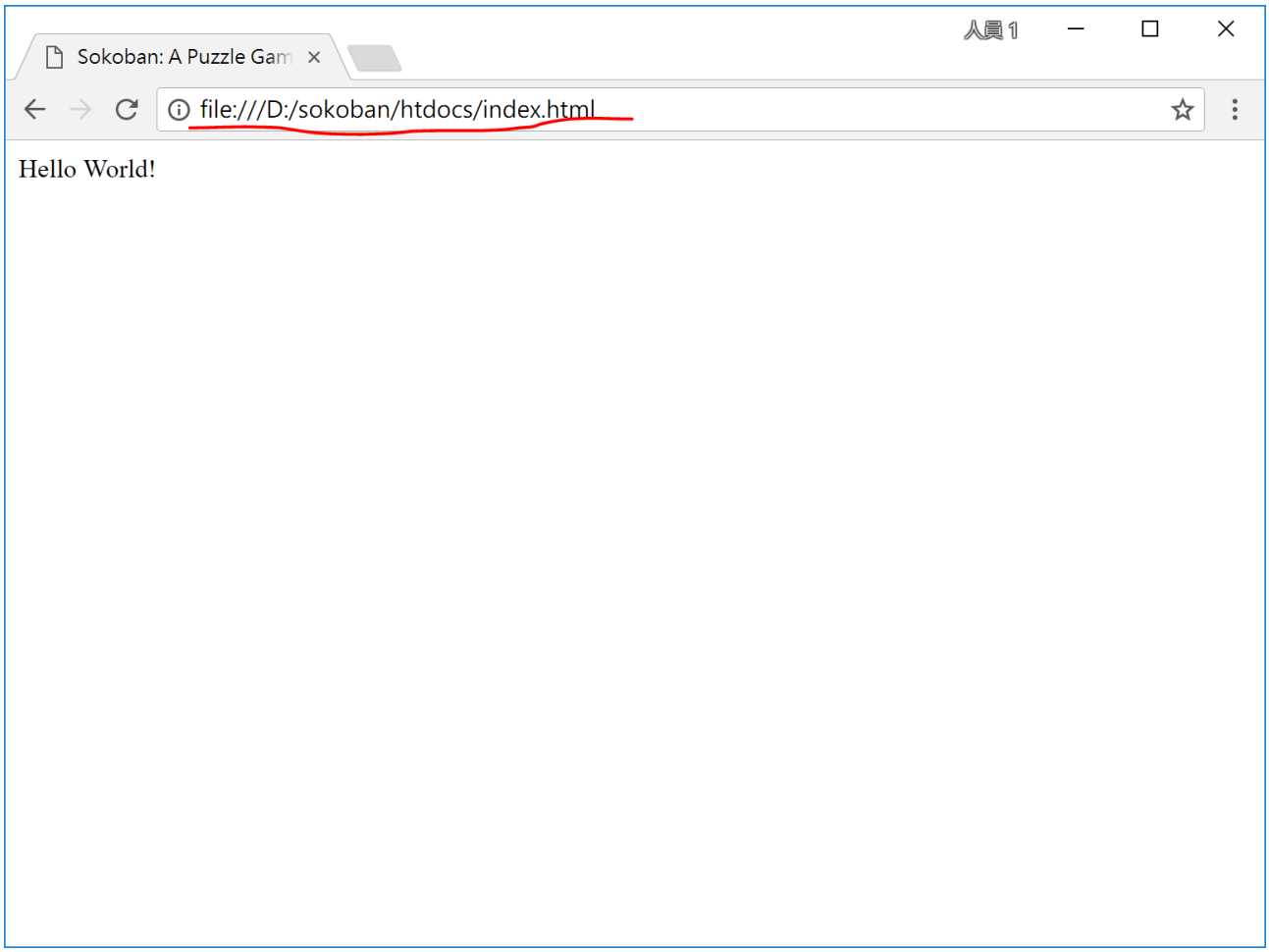

Figure 1: 瀏覽器開啟 index.html

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。

4 簡單的 HTML 版面架構

目前 index.html 看起來看單純的 Hello World! 沒什麼兩樣。因為整個還沒有置入文件結構和版面訊息。為了對 HTML 文件結構和版面的關係有個基本的理解，Figure 2 利用 CSS 將 HTML 不同版塊設定不同的背景顏色，以呈現 HTML 元素之間的關聯。

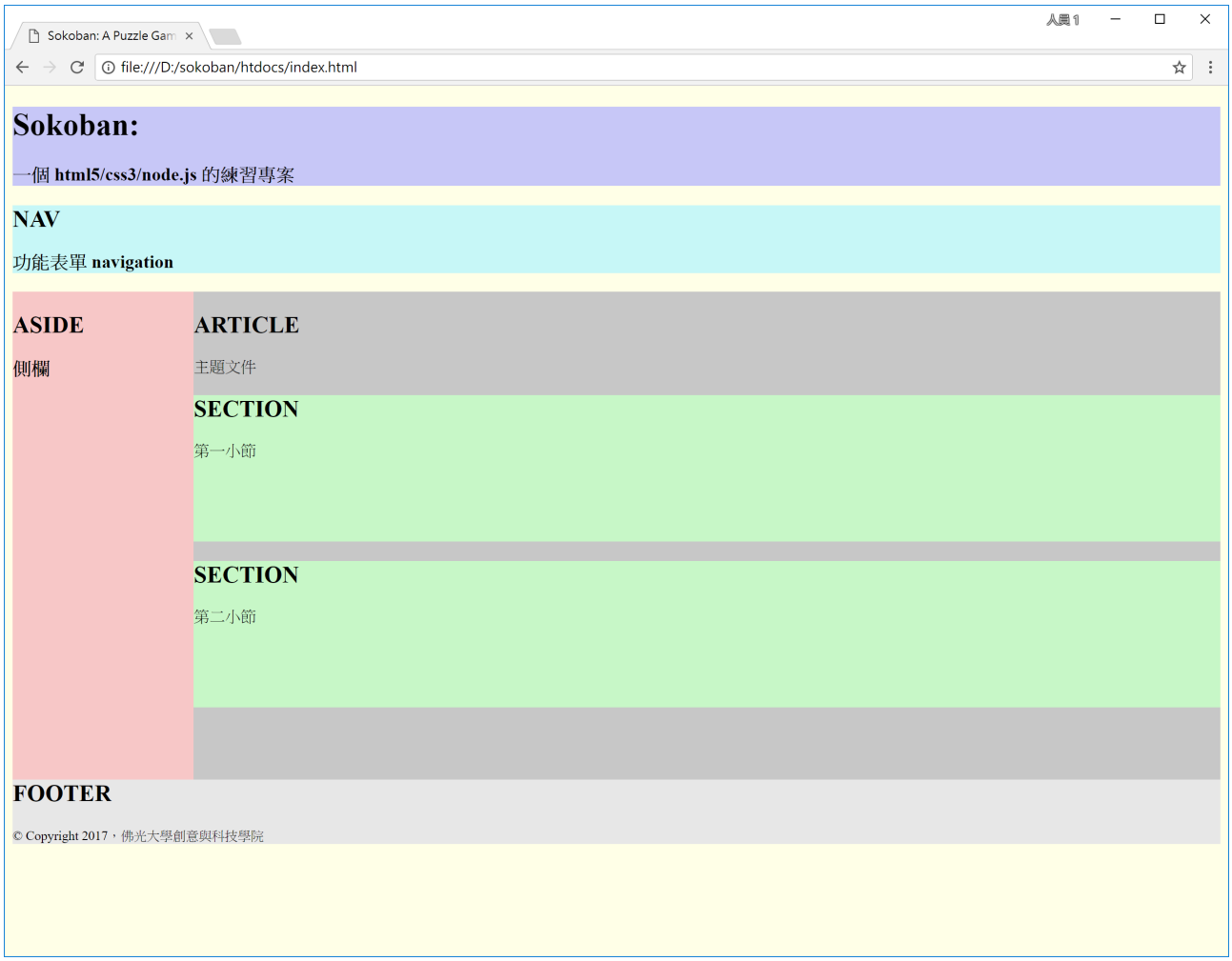

Figure 2: HTML 基本版塊

4.1 HTML 和內嵌的 CSS

Figure 2 的源碼如下：

```
<!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 style="background-color: #ffffe7;">
    <header style="background-color: #c7c7f7;">
      <h1>Sokoban:</h1>
      <h3>一個 html5/css3/node.js 的練習專案</h3>
    </header>
    <nav style="background-color: #c7f7f7;">
      <h2>NAV</h2>
      <h3>功能表單 navigation</h3>
    </nav>
    <aside style="background-color: #f7c7c7; width: 15%;
      height: 500px; float: left;">
      <h2>ASIDE</h2><h3>側欄</h3>
    </aside>
    <article style="background-color: #c7c7c7; width: 85%;
      height: 500px; float: left;">
      <h2>ARTICLE</h2>
      主題文件
      <section style="background-color: #c7f7c7;
        height: 150px;">
        <h2>SECTION</h2>
        第一小節
      </section>
      <section style="background-color: #c7f7c7;
        height: 150px;">
        <h2>SECTION</h2>
        第二小節
      </section>
    </article>
    <footer style="background-color: #e7e7e7;">
```

```
38.         <h2>FOOTER</h2>
            <small>&copy; Copyright 2017，佛光大學創意與科技學院<small>
39.       </footer>
40.     </body>
41. </html>
```

約 40 行的 HTML 碼，看起來是有那麼點手足無措。對照 Figure 2，將 index.html 的 <body></body> 結構獨立出來後，其實只有如下約 20 行的內容：

```
13.     <body style="background-color: #ffffe7;">
14.       <header style="background-color: #c7c7f7;">
17.       </header>
18.       <nav style="background-color: #c7f7f7;">
21.       </nav>
22.       <aside style="background-color: #f7c7c7; width: 15%;
            height: 500px; float: left;">
24.       </aside>
25.       <article style="background-color: #c7c7c7; width: 85%;
            height: 500px; float: left;">
28.         <section style="background-color: #c7f7c7;
              height: 150px;">
31.         </section>
32.         <section style="background-color: #c7f7c7;
              height: 150px;">
35.         </section>
36.       </article>
37.       <footer style="background-color: #e7e7e7;">
39.       </footer>
40.     </body>
```

為了讓 HTML 的 <tag> 能更清楚的表達文件的結構，網頁設計者能更明確地將他/她的設計意圖傳達給瀏覽器，HTML5 引入了一些，如上面片段裡的，新的**語義化** (semantic) 的 <tag> 來標示文件結構：

- <header></header>: 用來放網頁橫幅 (banner)，摘要 (abstract) 等表頭 (head) 資訊。
- <footer></footer>: 放置版權聲明，腳註 (footnote) 資料，頁碼等。
- <nav></nav>: 分頁，功能表，等引導使用者在網頁內移動的資訊。
- <aside></aside>: 側欄。置放和主要文章的相關資料或導覽。
- <article></article>: 主題文草。
- <section></section>: 文件段落。

在如 <body style="background-color: #ffffe7;"> 內出現的 style="..." 就是所謂的 CSS。如在上面的 index.html 源碼裡看到的，在 CSS 裡可以利用 background-

color 來設定背景顏色；height, width 屬性等。

像這樣將 CSS 設定直接利用 HTML tag 的 style 屬性寫在 HTML 檔案裡稱為 *embedded CSS*，嵌入式的寫法。這種作法在 Web 早期還沒有目前這麼泛用，或簡單網頁時很方便；但明顯已無法應付現在對網頁的嚴苛要求。因此，目前作法多會將 CSS 獨立在自己的 .css 檔內，再利用 <head> 區塊裡的 <link> tag 來結合兩者。

5 簡單的 CSS 檔案設定

直接在 HTML *tag* 裡插入各式各樣的屬性設定，除了極少數的情況外，實在不是個好主意。所以才有 CSS 的出現，也就是 **外部的** (*external*) 獨立 .css 檔案。

原來的 index.html 可以改寫如下：

```
 1. <!DOCTYPE html>
 2. <html lang="zh-TW">
 3.   <head>
11.
12.     <link rel="stylesheet" href="assets/css/styles.css">
13.   </head>
14.   <body>
15.     <header id="page_top">
16.       <h1 class="irontext">Sokoban:</h1>
17.       <h3>一個 html5/css3/node.js 的練習專案</h3>
18.     </header>
19.     <nav>
20.       <h2>NAV</h2>
21.       <h3>功能表單 navigation</h3>
47.     </nav>
48.     <div class="flexbox">
49.       <aside>
50.         <h2 class="irontext">ASIDE</h2>
51.         <h3>側欄</h3>
52.       </aside>
53.       <article>
54.         <h2 class="bluetext">ARTICLE</h2>
55.         主題文件
56.         <section class="browntext" id="sec1">
57.           <h2 class="redtext">SECTION</h2>
58.           第一小節
59.           <img src="assets/png/SokobanClone_byVellidragon.png"
                alt="sokoban icons" width="480" height="480">
```

```
60.           <button type="button">
61.             <a href="#page_top">回上方</a>
62.           </button>
63.         </section>
64.         <section class="browntext milky" id="sec2">
65.           <h2 class="redtext">SECTION</h2>
66.           第二小節
67.         </section>
68.         <section class="browntext milky" id="sec3">
69.           <h2 class="redtext">SECTION</h2>
70.           第三小節
71.         </section>
72.       </article>
73.     </div>
74.     <footer class="lighttext">
75.       <h2 class="blacktext">FOOTER</h2>
76.       <small>&copy; Copyright 2017，佛光大學創意與科技學院<small>
77.     </footer>
78.   </body>
79. </html>
```

和原來的 index.html 重要的差別有四 (4): 1. 第 *12* 行，加入了 <link> *tag*，連結到外部皂 .css 檔案，styles.css。1. 第 *59* 行，加入了 <img> *tag*，連結到外部皂 .png 檔案。SokobanClone_byVellidragon.png。1. 所有 *tag* 裡有關**背景顏色** (*background-color*) 的描述都拿掉了。1. 有些 *tag* (如第 *15* 行的 <header>) 多了 id 的設定；而另有些 *tag* (如第 *16* 行的 <h1>) 多了 class 的設定。

5.1 <link> *tag*

HTML 利用 <link> *tag* 來標明和外部 (external) 資源 (resource) 的聯繫。最**常用**的情況就是用來標明使用的 .css 檔。

index.html 的第 *12* 行內容如下：

```
12.     <link rel="stylesheet" href="assets/css/styles.css">
```

其中 rel 代表 *relation* (關係)，就是 <link> 的外部資源和目前的 .html 檔的**關係** (relationship)，這裡填入 stylesheet 代表是相關的 .css 檔案；後面 href (hyper-reference) 則是 url 指出如何取得這個外部 .css 檔案。這裡利用以 index.html 為**參考點**的**相對路徑** (relative path) 去取得置於在 assets/css 資料夾下的 styles.css 檔案。

5.2 <img> *tag*

<img> *tag* 是 HTML 標示要插入圖檔 (*image*) 的 *tag*。

```
59.             <img src="assets/png/SokobanClone_byVellidragon.png"
                  alt="sokoban icons" width="480" height="480">
```

src (source) 是圖檔的 url，這裡一樣利用以 index.html 位置作為參考點的**相對路徑**來取得置放在 assets/png 資料夾下的 SokobanClone_byVellidragon.png 檔。alt (alternative) 是標明當圖片無法顯示或在**無障礙**瀏覽模式，讓瀏覽器可以顯示/語音念出映圖片內容的說明文字。

而 SokobanClone_byVellidragon.png 檔本身則是將用來作為 Sokoban 遊戲內的顯示元件使用。可以在 opengameart 取得。

5.3 styles.css 檔

assets/css 資料夾的 styles.css 檔案內容如下：

```
 1. html {
 2.     height: 100%;
 3. }
 4.
 5. body {
 6.     background-color: #ffffe7;
 7.     height: 100%;
 8. }
 9.
10. header {
11.     background-color: #c7c7f7;
12. }
13.
14. nav {
15.     background-color: #c7f7f7;
16. }
17.
46. aside {
47.     background-color: #f7c7c7;
48.     width: 15%;
49.     float: left;
50. }
51.
52. article {
```

```
53.     background-color: #c7c7c7;
54.     width: 85%;
55.     float: left;
56. }
57.
58. section {
59.     background-color: #c7f7c7;
60. }
61.
62. footer {
63.     background-color: #e7e7e7;
64. }
65.
66. .flexbox {
67.     display: flex;
68. }
69.
70. .blacktext{
71.     color: #0f0f0f;
72. }
73.
74. .redtext {
75.     color: #ff3333;
76. }
77.
78. .bluetext {
79.     color: #3333ff;
80. }
81.
82. .irontext {
83.     color: #efefaf;
84. }
85.
86. .browntext {
87.     color: #9f5f5f;
88. }
89.
90. .sectext {
91.     color: #7f7fef;
92. }
93.
94. .lighttext {
```

```
 95.     color: #9f9f7f;
 96. }
 97.
138. #about {
139.     float: right;
140.     margin: 0px 2em 0px 0px;
141. }
142.
151. .milky {
152.     background-color: #ffffea;
153. }
```

如上所示，.css 內是以一個名稱，稱作**選擇器** (*selector*) 開始，後面跟著用 {...} 標示的**區塊** (block)，在**區塊**內就是 css 的屬性設定。

屬性 (atttibute) 設定遵循 attribute: value; 的格式；要設定的 **屬性**名稱和**值** (value) 由 : 隔開；而**屬性**和**屬性**之間則以 ; 分隔。

5.3.1 選擇器 (selector)

為了知道 .css 檔內設定的屬性要應用在 .html 檔內的那個 tag 上，CSS 設定了**三** (3*) 的層級的**選擇器**，如下：

- HTML tag: 每個 HTML 的 tag 都是**第一階**的選擇器。在這裡設定的 style，如果沒有被其它選擇器覆蓋掉，會應用在 .html 檔裡**所有**相同的 tag 上。如 styles.css 裡的第 1, 5, 10 行等。
- class: 在 HTML 的 tag 裡設定的 class；如 index.html 的第 *16* 行 <h1 class="irontext" 裡的 irontext 就對應上面 styles.css 檔裡的第 *82* 行。注意，在 CSS 裡，class 選擇器以 . (句點) 開頭。和 HTML tag 選擇器類似，如果沒有另外被覆蓋，設定的 style 會應用在**所用**相同 *class* 的 *tag* 上。
- id: 在 HTML 的 tag 裡設定的 id；如 index.html 裡的第 *15* 行和 styles.css 裡的第 *138* 行；和前兩 (*2*) 者不同的是，id 是**唯一** (*unique*) 的。

CSS 除了上述三類選擇器，還設計了輔助用的運算子，如 >，可以更精確的選出需要排版的元素，可以參照 CSS 的說明。

5.4 範例：以 CSS 搭配 HTML 製作下拉功能表

上面 index.html 和 styles.css 裡刪除的部份其實是個簡易的下拉式功能表單。程式表列出如下。將它們放入原來的檔案後，index.html 就有了一個簡單的下拉表單。

```
22.       <ul>
23.         <li id="sec1">第一頁
```

```
24.           <ul class="drop_box" id="drop_1">
25.             <li><a href="#sec1">Section 1</a></li>
26.           </ul>
27.         </li>
28.         <li>第二頁
29.           <ul class="drop_box" id="drop_2">
30.             <li><a href="#sec2">Section 2</a></li>
31.           </ul>
32.         </li>
33.         <li>第三頁
34.           <ul class="drop_box" id="drop_3">
35.             <li><a href="#sec3">Section 3</a></li>
36.           </ul>
37.         </li>
38.         <li>第四頁</li>
39.         <li>第五頁</li>
40.         <li id="about">關　於
41.           <ul class="drop_box" id="drop_about">
42.             <li>小組成員</li>
43.             <li>工作分工</li>
44.           </ul>
45.         </li>
46.       </ul>
```

```
18. /*
19.  * 選擇 nav 下面的 ul 元素下的 * 所有 * li 元素
20.  */
21. nav > ul > li {
22.     display: inline-block;
23.     height: 2em;
24.     min-width: 4em;
25.     line-height: 2em;
26.     text-align: center;
27.     // 上下 pading: 0; 左右 padding: 0.5 em;
28.     padding: 0px .5em;
29.     // 上下 pading: 0; 左右 padding: 0.25 em;
30.     margin: 0px .25em;
31.     border: none;
32.     background-color: bisque;
33. }
34.
35. /*
```

```
36.  * 當滑鼠游標移到 nav > ul > li 上時，更改 * 背景顏色 *
37.  */
38. nav > ul > li:hover {
39.     background-color: #dfdfdf;
40. }
41.
42. nav > ul > li:hover [id*=drop_] {
43.     display: block;
44. }
45.
98. /*
99.  * 下拉式 (drop-down) 功能表
100.  */
101. .drop_box {
102.     // 預設不顯示;
103.     display: none;
104.     // 去除 ul.li 的項目標示;
105.     list-style-type: none;
107.     // 位置跟隨它的父元素;
108.     position: absolute;
109.     background-color: #f9f9f9;
110.     padding: 0px;
111.     margin: 0px .25em;
112. }
113.
114. /*
115.  * 下拉式功能表裡的 *li* (list item)
116.  * 選單項目屬性設定
117.  */
118. .drop_box ul li {
119.     min-width: 4em;
120.     text-align: center;
121.     padding: 0px .5em;
122.     margin: .25em 0px;
123.     border: none;
124.     background-color: bisque;
125. }
126.
127. /*
128.  * 當滑鼠游標移至
129.  * 下拉式功能表裡的 *li* (list item)
130.  * 選單項目時，
```

```
131.  * 更改屬性設定
132.  */
133. .drop_box li:hover {
134.     color: DodgerBlue;
135.     background-color: Salmon;
136. }
137.
143. /*
144.  * 當滑鼠游標移至 * 關於 * (about) 時，將
145.  * 對應的 drop_box 顯示出來
146.  */
147. #about:hover #drop_about {
148.     display: block;
149. }
150.
```