Canvas 繪圖

喬逸偉 (Yiwei Chiao)

1 前言

前一章嘗試在網頁上呈現了對滑鼠座標變化的追踪。這一章開始進入 *canvas* 2D 繪圖。

1.1 專案準備

和前一章比較，有作更動的檔案只有 `htdocs/js/index.js`；如前所述，需要的 HTML 元素都可以利用 JavaScript 操作 DOM 來建構。因此，這裡就利用這一點，持續利用 JavaScript 來建構專案想像的版面。

目前執行畫面，如下圖 1：

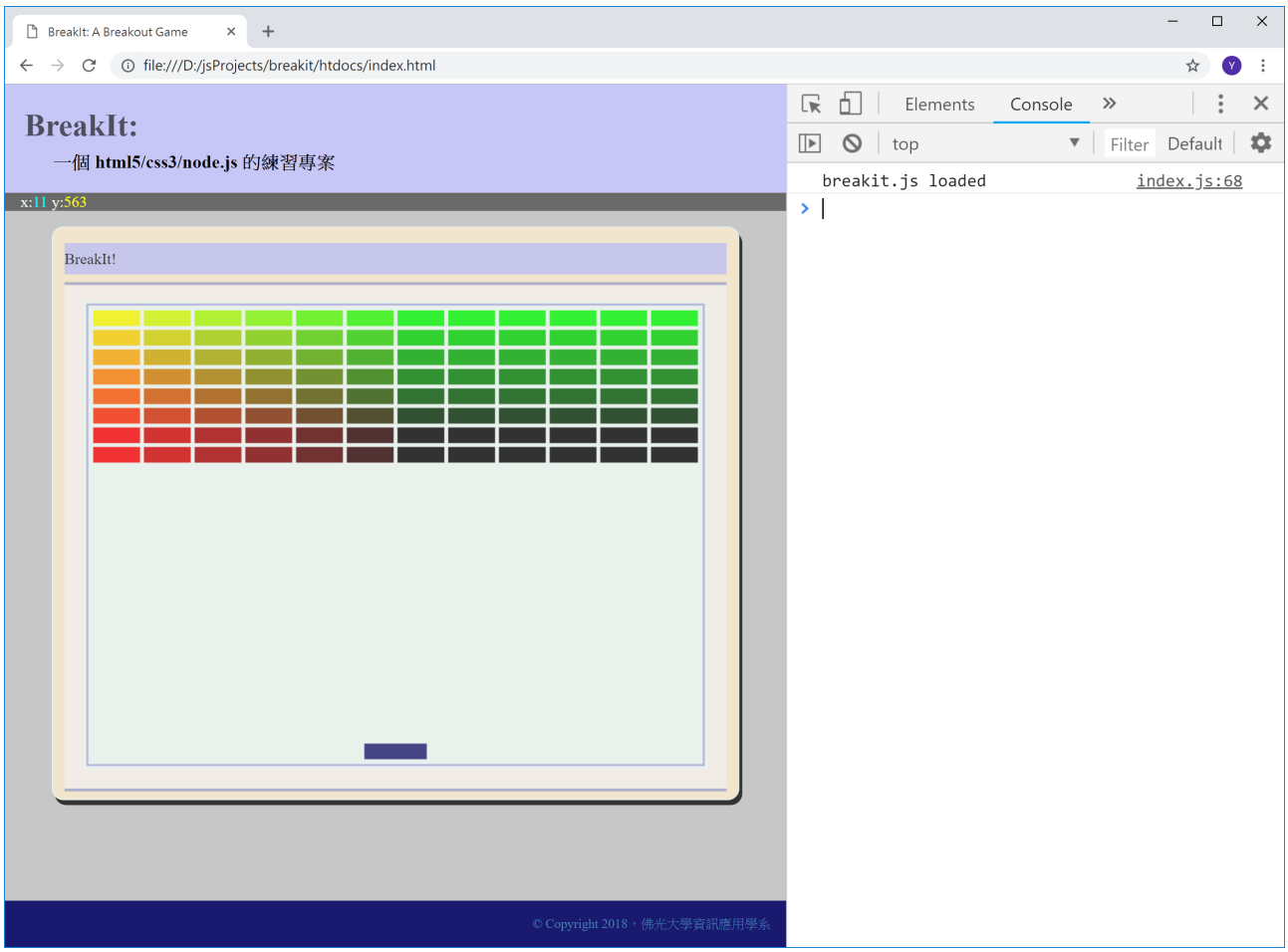

Figure 1: 基本遊戲畫面想像

1.2 DOM canvas 繪圖

htdocs/js/index.js 的內容如下：

```
'use strict';

// 繪出 * 擊球板 * (paddle)
let paintPaddle = function (ctx) {
  ctx.save();

  ctx.fillStyle = 'midnightblue';
  ctx.fillRect(288, 454, 64, 16);

  ctx.restore();
};

// 繪出 * 磚塊 * (paddle)
```

```
14. let paintBricks = function (ctx) {
15.   ctx.save();
16.
17.   let width = 12;
18.   let height = 8;
19.
20.   for (let x = 0; x < width; x ++) {
21.     for (let y = 0; y < height; y++) {
22.       ctx.fillStyle =
23.         `rgb(${Math.floor(255 - 42.5 * x)}, ${Math.floor(255 - 42.5 *
24.
25.       ctx.fillRect((x * 52) + 10, (y * 20) + 10, 48, 16);
26.     }
27.   }
28.
29.   ctx.restore();
30. };
31.
32. // 重繪 * 遊戲盤面 *
33. let paint = function (ctx) {
34.   // 將圖紙填滿背景色
35.   ctx.fillRect(0, 0, 640, 480);
36.
37.   ctx.strokeStyle = 'slateblue';
38.   ctx.strokeRect(4, 4, 632, 472);
39.
40.   // 繪出磚塊
41.   paintBricks(ctx);
42.
43.   // 繪出擊球板
44.   paintPaddle(ctx);
45. };
46.
47. /**
48.  * breakit 程式進入點
49.  *
50.  * @callback
51.  * @param 'load' : DOM 事件名
52.  * @returns {undefined}
53.  */
54. window.addEventListener('load', () => {
55.   console.log("breakit.js loaded");
```

```
56.
57.    // 準備承載 * 遊戲標題 * (title) 的 HTML 元素
58.    let gameTitle = document.createElement('span');
59.    gameTitle.textContent = 'BreakIt!';
60.
61.    // 準備承載 * 遊戲版頭 * (header) 的 HTML 元素
62.    let gameHeader = document.createElement('header');
63.    gameHeader.className = 'card-header';
64.
65.    // 將 * 遊戲標題 * 放上 * 遊戲版頭 *
66.    gameHeader.appendChild(gameTitle);
67.
68.    // 準備 * 遊戲盤面 * 的繪圖圖紙 (canvas)
69.    let gameCanvas = document.createElement('canvas');
70.
71.    // 取得能在 canvas 上繪圖的 context2d 物件
73.    let ctxPaint = gameCanvas.getContext('2d');
74.
75.    // 設定繪圖圖紙的寬高
76.    gameCanvas.width = 640;
77.    gameCanvas.height = 480;
78.
79.    // 設定圖紙背景色
80.    ctxPaint.fillStyle = 'mintcream';
81.
82.    // 繪出基本遊戲盤面
83.    paint(ctxPaint);
84.
85.    // 準備承載 * 遊戲內容 * 的 HTML 元素
86.    let gameContent = document.createElement('article');
87.    gameContent.className = 'card-content';
88.
89.    // 將 * 遊戲盤面 * 放上 * 遊戲內容 * 容器
90.    gameContent.appendChild(gameCanvas);
91.
92.    // 準備 * 遊戲桌面 * 的 HTML 元素
93.    let gameDesktop = document.createElement('section');
94.    gameDesktop.className = 'card';
95.
96.    // 將 * 遊戲版頭 * 放上 * 遊戲桌面 *
97.    gameDesktop.appendChild(gameHeader);
98.
```

```
 99.    // 將 * 遊戲內容 * 放上 * 遊戲桌面 *
100.    gameDesktop.appendChild(gameContent);
101.
102.    // 將 * 遊戲桌面 * 放上 * 網頁 *
103.    let desktop = document.querySelector('.site-body')
104.    desktop.appendChild(gameDesktop);
105.
106.    /**
107.     * 滑鼠游標移動追踪
108.     *
109.     * @callback
110.     * @param 'canvasmove' : DOM 事件名
111.     * @param e : DOM event 物件
112.     * @returns {undefined}
113.     */
114.    desktop.addEventListener('canvasmove', (e) => {
115.      document.getElementById('cursor-x').textContent = e.clientX;
116.      document.getElementById('cursor-y').textContent = e.clientY;
117.    });
118. });
```

將以上的內容放到 htdocs/js/index.js 檔案內之後，在瀏覽器內就可以看到如圖 1 的畫面。

和前一章相比，程式碼由 12 行暴增至 118 行；不過這是因為裡面加了大量註解 (comments) 的原因。對照瀏覽器開發人員視窗裡顯示的 HTML 結構，配合程式註解，其實並不難懂。

和之前相同，可以依程式列表自行輸入，測試，除錯；或者到 github 取得專案源碼。

git clone https://github.com/ywchiao/breakit.git

1.3 APIs 說明

和前一章相比，這一章裡大量使用了 DOM 裡的 建構 APIs；另一部份則是 canvas 的 2D 繪圖 APIs。

以下分別討論。

1.3.1 DOM APIs

程式碼 57 行到 104 行的作用是在建立畫面呈現需要的 HTML 結構，也就是圖 2 裡紅線框起來的部份。

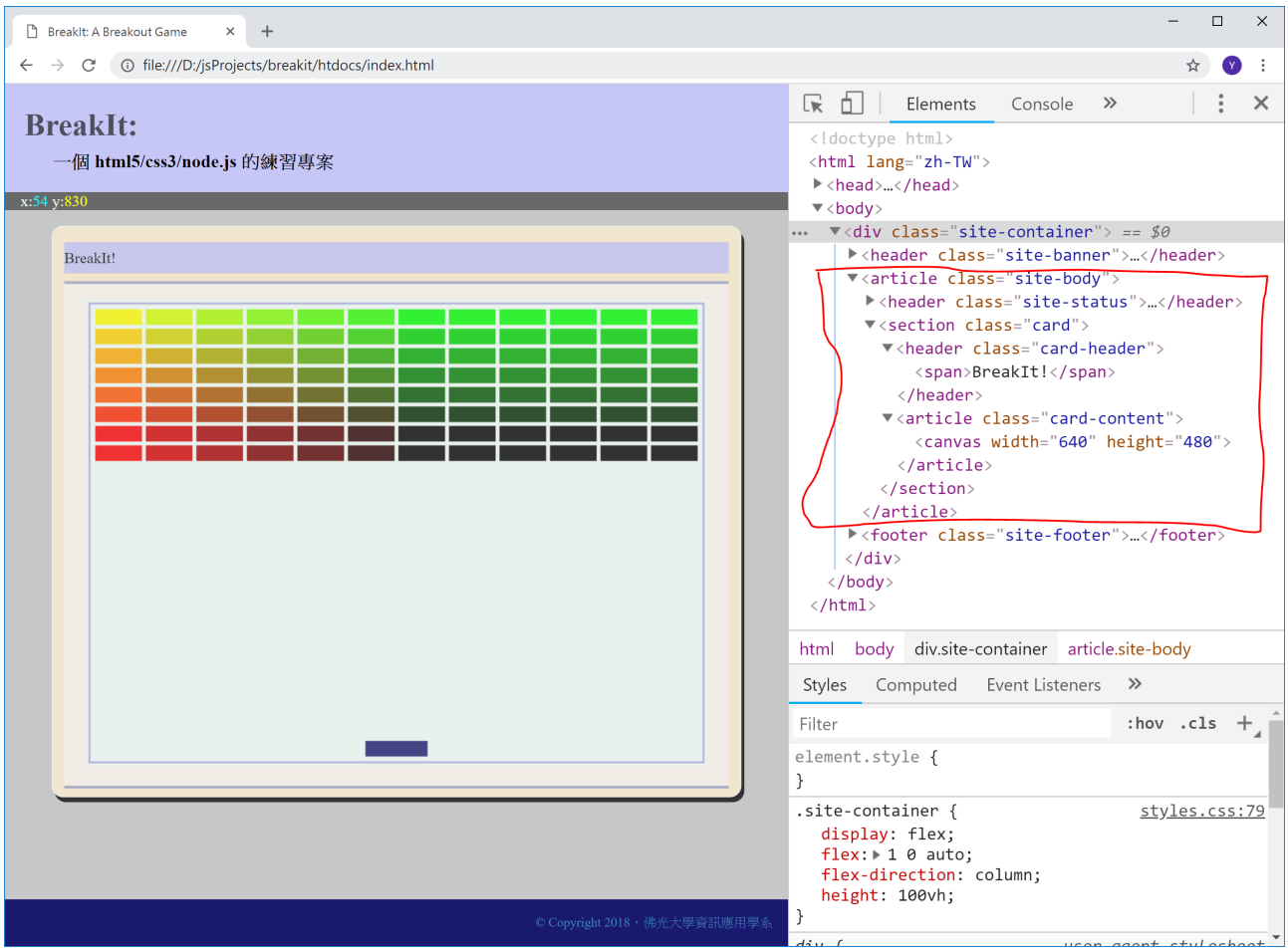

Figure 2: 遊戲畫面的 HTML 結構

程式碼看起來很長，其實就只是兩三個步驟的重覆而已：

1. 呼叫 `document.createElement(string)` 建構想要的 DOM 元素，其中 *string* 是 HTML 的 *tag* 名稱。如 62 行，建立一個 `<header>` 元素，同時將它放進 `gameHeader` 變數保存。
2. 設定新建立的元素的 CSS 屬性；一般是設定它的 CSS `class`。如 63 行，設定 `gameHeader` 的 CSS `class` 是 `card-header`。
3. 呼叫 `parent.appendChild(element)` 將建構好的 (child) *element* 插入 DOM 樹裡，當作 parent 的一個子節點。如 97 行，將 `gameHeader` 元素插入 DOM 樹內，作為 `gameDesktop` 的子節點 (child node)。

程式碼的流程，對照圖 2 裡的紅圈部分，是先產生最內層的子元素，再往外 (上) 產生親 (parent) 元素，再利用 `appendChild()` 將親/子元素結合，重覆到結構完成。

1.3.2 [canvas][mdnCanvas] APIs

[canvas][mdnCanvas] 是 HTML 為了處理 Web 繪圖所提供的數個解決方案之一，主要針對

2D 繪圖；其它還有為了向量圖 (vector graph) 的 SVG 與 3D 繪圖 (支援繪圖卡加速) 的 webGL。Breakit 專案只使用 [canvas][mdnCanvas] 的 2D 繪圖。

[canvas][mdnCanvas] 繪圖的流程很簡單：

1. 建立一個 <canvas> 物件
2. 由這個 <canvas> 物件取得綁定的 ConvasRenderingContext2D 繪圖物件，以下簡稱 'ctx2d'。
3. 呼叫 ctx2d 提供的 2D 繪圖 APIs，在綁定的 <canvas> 物件上作畫。

目前程式碼就遵照上面的邏輯運作。

目前使用的 APIs 有：

. canvas.getContext('2d'): 取得 CanvasRenderingContext2D 物件

. 繪圖

- fillRect(x, y, width, height): 以 (x, y) 作為左上角座標，填滿一個指定寬 (width)，高 (height) 的色塊。如 35 行。
- strokeRect(x, y, width, height): 同上；但只繪出色塊的邊框；而 不會填滿它。如 38 行。

. 屬性

- save(): 保留目前這個 ctx2d 物件的屬性；供之後還原 (restore)。
- restore(): 還原目前這個 ctx2d 物件的屬性到之前 save() 的狀態
- fillStyle: 字串；填滿 (fill) 色塊時使用的 RGBA 顏色。如第 7 行。
- strokeStyle: 字串；繪線 (stroke) 時使用的 RGBA 顏色。如第 37 行。

搭配 JavaScript 的迴圈 (loop) 控制，得到目前的結果。

1.4 思考與練習

- 4 和 14 行是將函數 (function) 指定 (assign) 給一個變數；這樣作的目的，除了因為文法許可，所以就這麼寫，之外，還有什麼考量？為什麼 JavaScript 允許這樣的寫法？其設計目的是？嘗試蒐集資料 (網路，書本，源碼) 了解一下。
- 20, 21 行使用的是 C 語言風格的 for 迴圈；其實，JavaScript 還支援多種不同的 for 迴圈，因應不同的使用情境；嘗試蒐集資料 (網路，書本，源碼) 了解一下。