第一章

HTML5 Canvas簡介

*HTML5*是時下最夯的*HTML*（超連結文件標示語言）。HTML在1993年首次被標準化之後，從那刻起即點燃了網際網路的戰火。HTML是由一些在 < > 裡的標記（tags）來定義網頁內容的一種方式。

*HTML5 Canvas*可以和JavaScript一起搭配在畫面上立即呈現點陣圖；而畫面上立即呈現的方式就是依據canvas在畫面上所繪製的圖像；HTML5 canvas可以在每個框架（frame）上完整的繪製點陣圖就是靠著在JavaScript裡呼叫Canvas API而成的。對程式設計師而言，你的工作就是在frame要呈現之前將要顯示的畫面設定好，如此就可正確的將圖像顯示出來。

HTML5 Canvas是在保留模式（*retained mode*）下作業的，如此使得它有別於Flash、Silverlight和SVG。此模式下，所有要顯示的物件是被保存在圖形處理器內，而這些物件就是依據程式碼裡的屬性（如：*X*座標的位置、*Y*座標的位置和物件的alpha透明度等）在畫面上顯示出來的。這讓程式程設計師不需使用低階的操作，但是對於最後圖像畫面的呈現就缺乏了全面的控制。

基本的 HTML5 Canvas API包含 2D context它可讓程式設計師繪製各種不同的形狀、繪製文字，以及在瀏覽器上事先定義好的區域內直接顯示圖片。你可以指定顏色、旋轉角度、透明度、像素的控制，以及在canvas上放各種不同型態的線條、曲線、方形和填上色彩的形狀、文字還有圖片。

HTML5 Canvas 2D context是一種用來在繪圖區域做演算繪製的顯示API，但在開發應用程式時很少人會使用這種技術。藉由加入可跨不同瀏覽器的JavaScript功能，如鍵盤與滑鼠的輸入、時間區間、事件、物件、類別、聲音、數學函式…等等，你可以學會使用HTML5 Canvas，並且開發出令人驚艷的動畫、應用程式與遊戲。

從這裡我們將正式進入本書的主題。首先將Canvas API拆解至容易了解的部份，然後再一一的將它組合起來，最後再完整的呈現如何使用它來建立應用程式。在本書中你將會學到很多已經在其他平台上成功被測試過的技術，現在我們就開始學習這令人興奮的新技術吧！

支援 HTML5 Canvas 的瀏覽器

現今大部份的網路瀏覽器皆透過各種不同的方式來支援HTML5 Canvas，會丟出例外訊息的IE8也不例外；在幾乎都相同的基礎功能下對於特殊功能的支援本來就會不斷的持續增加中。對於HTML5 Canvas支援最好的平台似乎是Googld Chrome，再來就是 Safari、Firefox及Opera。我們將使用JavaScript *modernizr.js*函式庫，因為它會協助我們找出哪個瀏覽器支援Canvas的哪些功能。因此，你若擔心曾經允諾過將會支援Canvas的第9版IE 瀏覽器支援上的問題，此刻，你可以先下載使用Google Chrome Frame（*http://code.google.com/chrome/chromeframe/*），它會讓你的IE也可以執行Canvas。

基本的 HTML 網頁

在開始進入Canvas主題之前，需先說明一下我們將會使用到哪些HTML5標準語法來建立我們的網頁。

HTML是一個用來建構瀏覽器網頁上的標準程式語言。在這裡我們不會花太多時間在HTML上，但由於<canvas>是form上的基礎格式，所以我們不能完全不加以說明。

基本的HTML網頁一般被分成<head>與<body>二個sections。新的HTML5規範裡，增加了幾個sections，如<nav>、<article>、<header>和<footer>。

在建立HTML網頁時<head>標籤包含了HTML <body>內將會被用到的一些資訊。在標準的規範裡JavaScript函式要放在<head>裡，之後我們討論<canvas>標籤時就會看到。然而某些特殊的原因我們需將 JavaScript放在<body>裡，但我們會試著讓事情更為簡單點，儘量將JavaScript都放在<head>裡。

基本的HTML網頁應該看起來會像範例1-1的內容。

範例1-1 一個基本的HTML網頁

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>CH1EX1: Basic Hello World HTML Page</title>
</head>
<body>
Hello World!
</body>
</html>
```

<!doctype html>

此標籤是告訴瀏覽器要編排一個標準模式的網頁。依據W3C所訂定的HTML5規範裡，在HTML5文件中這個標籤是必要的。當在不同的瀏覽器上編排HTML時，這個標籤大大的簡化了冗長及一堆奇奇怪怪的現象，要注意的是它一定要出現在HTML的第一行。

<html lang="en">

<html>標籤用來指明網頁使用那種語言；舉例來說"en"=English。還有一些其他較常使用到的值：

```
Chinese - lang = "zh"
French - lang = "fr"
German - lang = "de"
Italian - lang = "it"
Japanese - lang = "ja"
Korean - lang = "ko"
Polish - lang = "pl"
Russian - lang = "ru"
Spanish (Castilian) - lang = "es"
```

<meta charset="UTF-8">

此標籤是在告訴瀏覽器此網頁所使用的字元編碼為何。一般的情況下是不需要更動它的值，除非你明確的知道自己要做什麼。這個標籤在HTML5的網頁中也是必要的。

<title>…</title>

它會將HTML網頁的標題名稱顯示在瀏覽器視窗的標題欄內。這是個非常重要的一個標籤，它有一部份功能是提供搜尋引擎更容易的搜索到你的網頁。

一個簡單的HTML5網頁

現在讓我們在瀏覽器上看一看這頁的樣子（這是你將所有的工具整合在一起，開始寫程式最好的時機）。打開你要使用的文字編輯器，同時準備好要使用的瀏覽器：Safari、Firefox、Opera、Chrome或IE。

1. 在文字編輯器上，輸入範例1-1裡的程式碼。
2. 選好你要存入檔案的路徑，並將它取名為*CH1EX1.html*。
3. 在Chrome、Safari或Firefox瀏覽器的File選單裡，找到Open File這個選項並點選它，你應該會看到一個開啟檔案的視窗。（在Windows中使用Chrome瀏覽器，要按Ctrl+O來將檔案開啟。）
4. 選擇你剛剛建立好的*CH1EX1.html*檔案。
5. 點選開啟。

你應該可以看到如圖1-1的樣子。

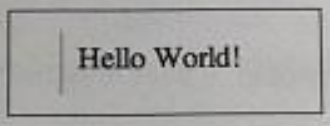

圖1-1 HTML Hello World!

這是本書中兩個可以在IE8或更早之前版本的瀏覽器上執行成功的程式之一。

本書將會使用到基本的 HTML

很多的HTML標籤可以被用來建立HTML網頁，在過去的HTML版本裡，用標籤來規範瀏覽器該如何編排HTML網頁（例如：<font>與 ）是非常普遍的用法。然而，對瀏覽器來說，在過去的十年裡標準反而變成了一種限制，這些標籤被排擠，使用CSS（Cascading Style Sheets）型態的HTML文件內容反而變成了主流。因為本書並不是在介紹如何建立HTML網頁（也就是沒有使用Canvas的網頁），所以，我們將不會對CSS做進一步的討論。

我們只會將重點放在最基本的兩個標籤： 及<canvas>。

這是書中會使用到最主要的一個HTML標籤。我們將在HTML網頁中用它來為<canvas>定位。

範例 1-2 使用了 將"Hello Word!"呈現在螢幕上，如圖1-2的樣子。

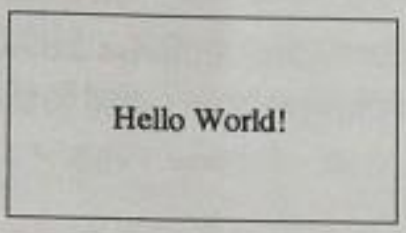

圖 1-2 使用 來顯示 HTML5 Hello World!

範例 1-2 HTML5 Hello World!

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>CH1EX2: Hello World HTML Page With A DIV </title>
</head>
<body>
<div style="position: absolute; top: 50px; left: 50px;">
Hello World!
</div>
</body>
</html>
```

`style="position: absolute; top: 50px; left: 50px;"`是在HTML網頁內CSS型式裡的一行程式碼例子。它是在告訴瀏覽器以網頁的左上角為基準點開始往下數50個像素，並從最左邊開始算起第50個像素的絕對位置上開始編排內容。

<canvas>

使用 裡的絕對位置方法，將使我們用<canvas>更有效益。將<canvas>放入 標籤裡，它會幫助我們取得一些資訊，例如：當滑鼠出現在canvas上方時，可得知滑鼠指標的相對位置。

文件物件模組（DOM）與Canvas

DOM（Document Object Model）可以用來代表在HTML網頁內容裡的所有物件。它與程式語言及平台是無關的，它也允許你在瀏覽器編排好網頁後，重新的修改網頁上的內容與型式。可以透過JavaScript來取用DOM，自90年代後期起它也變成了JavaScript、DHTML與CSS基本的功能。

Canvas元素在瀏覽器上可以透過DOM並經由Canvas 2D context來存取，但在Canvas上所建立的個別圖形化元素是不可被DOM所取用的，這是因為Canvas是在立即的模式下運作，它並沒有屬於自己的物件，只有在單一的frame下繪製才可被建立。

我們的第一個範例將會在HTML5網頁上使用DOM來放置<canvas>標籤，如此我們就可以使用JavaScript來操作它。在我們要開始使用<canvas>之前還有兩個特別的DOM物件必須先了解：window與document。

Window物件是DOM的最高階層。在開始我們的Canvas應用程式前，要先測試這個物件以確保所有的程式碼與需用到的檔案等都已載入。

Document物件包含了在HTML網頁上所有的HTML標籤。我們要來看看此物件透過使用JavaScript以找出<canvas> instance。

JavaScript與Canvas

JavaScript是一種程式語言，我們將用它來建立Canvas應用程式，它可被執行於目前已存在的網頁瀏覽器內。如果你需要複習這部份的話可以閱讀Douglas Crockford著的*JavaScript：The Good Parts*（O'Reilly）一書，這是一本關於JavaScript的書籍，寫得非常棒且相當受歡迎。

JavaScript開發平台與函式庫

現今有很多受歡迎的JavaScript開發平台，開發者用它們來從無到有的產生JavaScript程式碼，包括函式庫，例如：jQuery、Processing.js…等，我們希望這些開發平台在接下來的6-12個月裡可以強力的支援Canvas。目前我們會將重點放在直接的使用JavaScript控制Canvas上。但在適當的情況下，我們也會介紹開發平台與JavaScript函式庫，如此將會大大的加速Canvas 的開發（如：Modernizr、JSColor與WebGL）。

JavaScript要寫在哪裡？為什麼？

因為我們要在JavaScript裡建立Canvas的程式邏輯，於是有一個問題便產生了：在我們已建立好的網頁中，JavaScript要放在哪裡呢？

將JavaScript放在HTML網頁的<head>內似乎是個不錯的想法，因為這樣會非常容易的被找到。但是，將JavaScript放在那裡就表示，JavaScript與HTML執行前，所有HTML網頁裡的東西都必須要先被載入。也就是說，在所有的網頁內容被載入前JavaScript程式碼就要開始執行了。因此，你需要測試看看在執行JavaScript程式之前是否HTML網頁有被載入。

將JavaScript搬移到HTML文件最下方</body>之前，也可確保在JavaScript執行前所有的網頁都被載入。不過，因為我們將要測試在執行<canvas>程式之前，網頁是否有載入JavaScript，所以必須將JavaScript放在以往的<head>位置內。如果你對這樣感到不習慣的話，可以將程式碼改成你自己習慣的型式。

不管你將該行程式碼放在那裡，放在HTML網頁內或以載入*external.js*檔案的方式都是可行的，這個需另外載入的JavaScript檔案看起會是這樣：

```
<script type="text/javascript" src="canvasapp.js"></script>
```

為了要讓事情更加的簡單，我們會將JavaScript寫在HTML網頁裡。如果你知道你在幹什麼的話，將它存成外部的檔案並將其載入也是可行的方案。

在 HTML5 裡你將不再需要指定Script的型態。

HTML5 Canvas "Hello World!"

正如剛剛我們才提到的，當我們將Canvas放入HTML5網頁中第一件要做的事是，在我們開始要執行任何的動作之前測試看看是否所有的網頁內容都被載入並呈現出來。當我們開始要在Canvas上使用圖像與聲音時，這件事是一定要做的。

要這麼做，需在JavaScript中使用事件（*events*）。當一個已被定義的event發生時，events會被物件（objects）所調用。其他的物件也會去監聽events，如此它們才可以依不同的event做不同的事情。在JavaScript裡一些物件常見的events有按下鍵盤的event，滑鼠移動的event，以及東西何時載入完成的event。

第一個我們要監聽的是window物件的load事件，這是發生在當網頁載入完成時。

要為事件加入監聽器（*listener*），需使用addEventListener()方法，它是屬於DOM物件的一部份。因為window代表了HTML網頁，它是DOM的最頂層。

addEventListener()函式可接受3個參數：

*Event:*load

它是我們要增加監聽器的event名稱。對於像window這樣已存在的物件而言，events是已經被定義好的。

Event處理程序函式：eventWindowLoader()

當一個event發生時會呼叫此函式。在我們的程式碼裡，接著會去呼叫canvasApp()函式，它會啟動我們的主程式並執行之。

useCapture:ture或false

這個函式是在物件的DOM樹較低層級傳播之前擷取event目前的型態。我們將這個值設定為false。

我們將用以下的最終程式碼來測試看看window是否有被載入：

```
window.addEventListener("load", eventWindowLoaded, false);
function eventWindowLoaded () {
   canvasApp();
}
```

此外，你也可以用其他的方法來為load設定event監聽器：

```
window.onload = function()
  {
     canvasApp();
  }
```

或

```
window.onload = canvasApp();
```

在本書中，我們將使用第一種方式。

為Canvas封裝你的JavaScript程式碼

我們已經建立好一個方法來測試HTML網頁是否有被載入，現在可以開始準備寫JavaScript應用程式了。因為JavaScript是在HTML網頁中執行的，它也可以同時和其他的JavaScript

應用程式與程式碼一起執行，一般來說，這樣是不會有問題的。但，在HTML網頁上你的程式碼可能會與其他的JavaScript程式碼在變數上或函數上會有所衝突。

Canvas應用程式與其他在瀏覽器上執行的應用程式有些許的不同，因為Canvas是在畫面上所定義好的區域內執行它的顯示功能，它的功能只有它自己能夠獨用而已，所以它和其他的網頁之間不會互相干擾，反之亦然。你也可能會在同一個網頁上放置多個Canvas應用程式，所以，當你在定義程式碼時必須為你的JavaScript做些區隔。

為了避免這樣的問題發生，你可以將變數與函數封裝起來，並放在另一個函數裡頭。在JavaScript裡的函數是屬於物件它們自己的，而物件們都有兩個屬性（Properties）與方法（methods）。藉由將一個函數放在另一個函數裡，於是在第一個函數的範圍裡你便建立了第二個函數。

在我們的範例裡，會有一個被window load事件所呼叫的canvasApp()函數，裡頭包含了全部的Canvas應用程式。而這個"Hello World!"範例有一個名為drawScreen()的函數；只要當canvasApp()被呼叫時，就會立刻呼叫drawScreen()來繪製我們的"Hello World!"字樣。

drawScreen()函數現在是放置於canvasApp()裡，任何我們所要建立的變數與函數都將放置於drawScreen()裡，但對其餘的HTML網頁或JavaScript應用程式來說是不可執行的。

這裡是Canvas應用程式該如何封裝函數與程式碼的範例程式：

```
function canvasApp() {
   drawScreen();

   ...

   function drawScreen() {

      ...

   }
}
```

在HTML網頁中加入Canvas

請使用以下的程式碼，在HTML網頁裡的<body>段落加入<canvas>標籤：

```
<canvas id="canvasOne" width="500" height="300">
Your browser does not support HTML5 Canvas.
</canvas>
```

讓我們來仔細的了解一下這段程式碼在做什麼。<canvas>標籤有三個主要的屬性，在HTML裡，屬性是被放在HTML標籤的尖括號裡的，我們需要設定的三個屬性分別是：

id

在JavaScript程式碼裡，我們就是透過此id來找到這個<canvas>標籤的，也就是說<canvas>標籤的名字就是用id來設定的；在這裡我們把它取名為canvasOne。

width

canvas的寬度，以像素（pixels）來計算；在此我們設定為500個像素。

height

canvas的高度，以像素（pixels）來計算；在此我們設定為300個像素。

包括canvas在內的所有HTML5元素，都有許多的屬性：tabindex、title、class、accesskey、dir、draggable、hidden...等。

如果在沒有支援Canvas的瀏覽器上執行HTML網頁時，你可以將想要顯示的文字放在<canvas>與</canvas>標籤之間。在我們的Canvas應用程式裡，將會使用"Your browser does not support HTML5 Canvas"這串文字，當然，你也可以改成任何你想要改成的字串。

在JavaScript裡使用document來參考canvas元素

現在我們要使用DOM來參考在HTML上所定義的<canvas>。應該還記得當網頁被載入後，document物件代表著HTML網頁上的每一個元素。

我們要參考到Canvas物件，如此才會知道JavaScript裡所呼叫的Canvas API要在哪兒顯示。

首先，需先定義一個名為theCanvas的新變數，它將用來控制Canvas物件。

其次，藉由呼叫document裡的getElementById()函數對canvasOne做存取，並傳入我們在HTML網頁中為<canvas>標籤所定義的名字canvasOne：

```
var theCanvas = document.getElementById("canvasOne");
```

測試瀏覽器是否支援Canvas

現在我們已經可以參考到HTML網頁中的canvas元素，我們還需要測試看看它是否包含*context*。Canvas *context*是指定義在瀏覽器上用來支援Canvas的繪圖面。簡單的說就是，如果context不存在，canvas也不會存在。有很多種方法可以做測試，第一個要測試的是getContext method是否存在，在我們使用Canvas來呼叫它之前，就如同已經在HTML網頁上所定義好的這樣：

```
if (!theCanvas || !theCanvas.getContext) {
   return;
}
```

事實上，這可以測出兩個東西：第一，theCanvas是否不為false（如果id的名字不存在，document.getElementById()就會傳回false）；第二，getContext()函數是否存在。

如果測試結果失敗，return語句就會讓它跳出並停止執行程式。

還有另一個測試的方法－發表於Mark Pilgrim的HTML5網站上，*http://diveintohtml5.org*－在函數裡用一個不存在的canvas來測試看看瀏覽器是否有支援canvas：

```
function canvasSupport () {
    return !!document.createElement('testcanvas').getContext;
}
function canvasApp() {
    if (!canvasSupport) {
       return;
    }
}
```

*modernizr.js*是我們最喜歡使用的程式庫，你可以在這個網站中找到：*http://www.modernizr.com/.Modernizr*－一個簡易且小巧的程式庫，可用來測試各種不同的網頁技術－你可以針對是否有支援Canvas來建立一些靜態的布林值。

要在網頁中包括（include）*modernizr.js*，必須先在*http://www.modernizr.com/*網頁中將它下載並於你的HTML網頁中將*external.js*檔案include進來。

```
<script src="modernizr-1.6.min.js"></script>
```

為了要測試Canvas，必須先將canvasSupport()函數修改成如下：

```
function canvasSupport () {
   return Modernizr.canvas;
}
```

現在我們準備開始要使用*modernizr.js* method了，因為它為測試網頁是否支援Canvas提供了最好的方式。

取用2D Context

最後，我們還要取得2D context，如此才可進一步的對它操作。HTML5 Canvas設計成可多個context互相運作，包括了3D context在內。但本書在此的目的，只需要使用到2D context即可：

```
var context = theCanvas.getContext("2d");
```

drawScreen()函數

現在要開始建立真正的Canvas API程式碼了。我們將透過context object來操作顯示在Canvas上的每一個動作，就如同我們在HTML網頁中參考object般。

在稍後的章節裡對於在HTML5 Canvas上寫字，繪製圖形與圖像將會有深入的探討，現在，我們將花一點點的時間在drawScreen()程式碼上。

"screen"在這裡是定義為canvas的繪圖區域，並非全部的瀏覽器視窗，我們在這裡提到它是因為你將來在寫遊戲或應用程式的context裡會用到，在canvas中使用"window"或"screen"可更有效的控制它的顯示。

清楚的界定繪圖區域是我們第一件要做的事。接下來的兩行程式碼是在畫面上畫一個和canvas相同大小的黃色方形。fillStyle()用來設定它的顏色，fillRect()則是在建立一個方形並將它放在畫面上：

```
context.fillStyle = "#ffffaa";
context.fillRect(0, 0, 500, 300);
```

提醒您，我們現在所使用的是context函數。並不是screen物件，color物件或什麼其他的物件。而它就是我們之前所提到的立即模式（immediate mode）的一個範例。

在下一章裡我們將會討論Canvas的文字函數，但在這裡只是很簡要的看一下在畫面上會出現"Hello World!"的程式碼。

首先，用之前設定方形顏色的方式來設定文字的顏色：

```
context.fillStyle = "#000000";
```

然後，再設定文字的大小和寬度：

```
context.font = "20px _sans";
```

再來，設定文字垂直對齊的方式：

```
context.textBaseline = "top";
```

最後，我們藉由呼叫context物件的fillText()函數在畫面上呈現出要測試的結果，這個函數的三個參數分別為要顯示的文字字串、x座標及y座標：

```
context.fillText ("Hello World!", 195, 80);
```

我們再來為"Hello World!"文字加些圖片：首先，先將圖像載入並將它顯示出來。在第四章裡我們將深入的研究圖像與操作圖像，但現在，我們只需要將圖片置於畫面即可。為了要在canvas上顯示圖片，你必須建立一個Image()物件的實例（instance），並將Image.src屬性設定成載入圖像的檔名。

你也可以將其他的canvas或video當做image一樣來顯示。這些都將在第四章與第六章中討論到。

在要顯示它之前，必須要寫一段將圖片載入的程式碼，藉由設定Image物件的onload函數，為Image load event建立一個callback()函數。當onload event發生時就會執行callback()，而圖像被載入後，就要去呼叫context.drawImage()，並同時傳入三個參數：Image物件、x座標與y座標：

```
var helloWorldImage = new Image();
helloWorldImage.src = "helloworld.gif";
helloWorldImage.onload = function () {
   context.drawImage(helloWorldImage, 160, 130);
}
```

最後，再讓我們在文字與圖片的周圍畫一個方形吧。要畫一個沒有填上色彩的方框，只需用context.StrokeStyle() method來設定筆的顏色（方形的邊框），然後再呼叫context.strokeRect() method來繪製矩形的邊框。strokeRect() method的四個參數分別為左上角的x與y座標，以及右下角的x與y座標：

```
context.strokeStyle = "#000000";
context.strokeRect(5, 5, 490, 290);
```

範例1-3就是完整的HTML5 Hello World!程式碼，它的結果請參考圖1-3。

範例1-3 HTML5 Canvas Hello World!

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>CH1EX3: Your First Canvas Application </title>
<script src="modernizr-1.6.min.js"></script>
<script type="text/javascript">
window.addEventListener("load", eventWindowLoaded, false);
var Debugger = function () { };
Debugger.log = function (message) {
   try {
      console.log(message);
   } catch (exception) {
      return;
   }
}

function eventWindowLoaded () {
   canvasApp();
}

function canvasSupport () {
    return Modernizr.canvas;
}

function canvasApp () {

        if (!canvasSupport()) {
          return;
        }

     var theCanvas = document.getElementById("canvasOne");
     var context = theCanvas.getContext("2d");

     Debugger.log("Drawing Canvas");

       function drawScreen() {
       //背景
       context.fillStyle = "#ffffaa";
       context.fillRect(0, 0, 500, 300);

       //文字
       context.fillStyle  = "#000000";
       context.font = "20px _sans";
       context.textBaseline = "top";
       context.fillText  ("Hello World!", 195, 80 );

       //圖像
       var helloWorldImage = new Image();
       helloWorldImage.src = "helloworld.gif";
       helloWorldImage.onload = function () {
```

```
            context.drawImage(helloWorldImage, 160, 130);
        }

        //方形
        context.strokeStyle = "#000000";
        context.strokeRect(5,  5, 490, 290);

    }

    drawScreen();
}

</script>
</head>
<body>
<div style="position: absolute; top: 50px; left: 50px;">
<canvas id="canvasOne" width="500" height="300">
 Your browser does not support HTML5 Canvas.
</canvas>
</div>
</body>
</html>
```

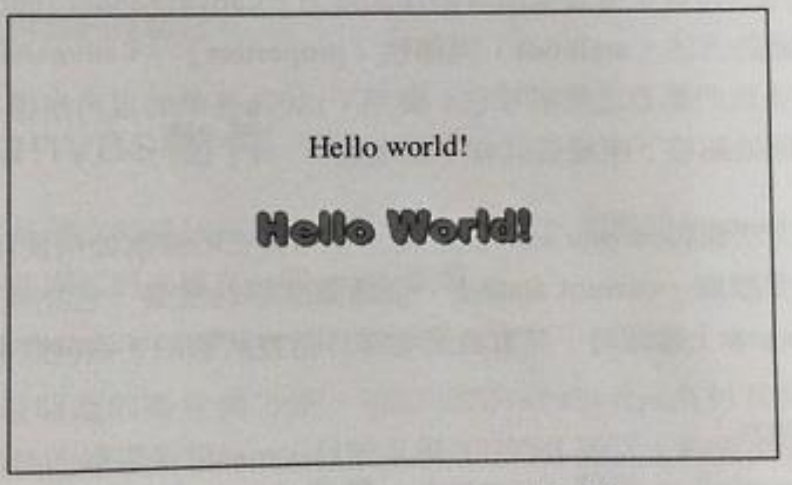

圖1-3 HTML5 Canvas Hello World!

在Console.log裡偵錯

在我們要更深入的探討本書之前，還有一件事情必須要加以說明，我們使用了瀏覽器上的一個非常簡單的除錯方法console.log功能，這個函數可以讓你在JavaScript的控制台（console）裡記錄文字訊息，以協助您找出程式碼出問題的地方。任何一個有JavaScript控制台的瀏覽器（Chrome、Opera、Safari，有安裝Firebug的Firefox），都可以使用console.log；沒有console.log的瀏覽器就不會丟出錯誤的訊息。

要處理這樣的錯誤，我們使用了wrapper將console.log給包裝起來，只提供呼叫的動作，如果有支援該函數的話。wrapper建立了一個名為Debugger的類別（class），然後建立一個可以在程式碼中任意呼叫的靜態函數Debugger.log，如下：

```
Debugger.log("Drawing Canvas");
```

這是console.log()功能的程式碼：

```
var Debugger = function () { };
Debugger.log = function (message) {
   try {
      console.log(message);
   } catch (exception) {
      return;
   }
}
```

2D Context與目前的狀態

HTML5 2D context（CanvasRenderingContext2D物件），可被Canvas物件的getContext() method所呼叫取用，也就是會發生所有動作的地方。CanvasRenderingContext2D包含了所有在canvas上繪圖的方法（methods）與屬性（properties）。CanvasRenderingContext2D（或context，之後我們都要這麼稱呼它）使用了以0為基準的直角座標系統，0在canvas的左上角，從左邊開始越往下座標值就會一直增加。

所有這些屬性與方法都與*current state*做連接，在你真正的瞭解如何使用HTML5 Canvas之前這個觀念必須要瞭解。current state是一個繪圖狀態的堆疊，它完完全全適用於全部的canvas。所以在canvas上繪圖時，其實就是在操作這些狀態的。這些狀態有：

Transformation matrix

等比例縮放（scale）、旋轉（rotate）、轉換（transform）與平移（translate）的methods。

Clipping region

由clip()方法所建立。

Properties of the context

這些屬性有strokeStyle、fillStyle、globalAlpha、lineWidth、lineCap、lineJoin、miterLimit、shadowOffsetX、shadowOffsetY、shadowBlur、shadowColor、globalCompositeOperation、font、textAlign，以及textBaseline。

別擔心，這些看起來還蠻親切的，在接下來的三章會對這些屬性做深入的討論。

還記得之前討論過的立即模式（immediate mode）與保留模式（retained mode）嗎？canvas是一個立即的繪圖模式，也就是說每當圖上的某部份要改變時，所有的東西都需要重新再繪一次。這樣其實也有些好處：例如，全域型的屬性可很容易的在畫面上看到結果。一但你知道了，每次要修改畫面上的圖像時，就可以用這個簡捷的方式在canvas上做修改了。

換句話說，保留模式是這些物件何時會被drawing surface儲存起來，並以列表的方式提供操作使用。Flash及Silverlight就是屬於此種模式。保留模式對於要建立多個物件且各自擁有獨立狀態的應用程式來說是非常有用的。許多完全使用canvas的應用程式（遊戲、動畫）很容易的就可以在保留模式的繪圖面（drawing surface）上寫出程式，特別是對初學者而言。

我們要挑戰的是利用立即模式繪圖面來繪圖，同時在程式碼裡加入各種的功能使得圖畫動起來就像在保留模式下一樣。這整本書中我們要討論的就是在立即模式下操作的方法，以及透過程式碼來更容易的控制它。

HTML5 Canvas物件

還記得Canvas物件被建立時是放在HTML網頁<body>這一部份的<canvas>標籤下的嗎！你也可以寫一行如下的程式碼來建立一個canvas實體：

```
var theCanvas = document.createElement("canvas");
```

Canvas物件有兩個關連的屬性與方法，可以使用JavaScript來對其做存取：width和height；這兩個屬性的意思是說canvas目前呈現在HTML網頁上的寬度與長度。在這裡要注意一下它們並非只可以讀取而已，它們是可以在程式碼中做修改並在HTML網頁中改變的。這是什麼意思呢？它是說你可以在HTML網頁上不做重新載入的動作便可以動態的改變canvas的大小。

你也可以使用CSS樣式來改變canvas的縮放比例。和重新改變大小是不一樣的，縮放比例是將目前canvas的圖像區域重新採樣將它的大小符合於CSS樣式裡指定的寬度與長度。舉例來說，若要將canvas的比例改成400*400的區域，CSS樣式就可以這樣寫：

```
style="width: 400px; height:400px"
```

在第三章裡我們將會有一個使用變換矩陣來縮放canvas的範例。

Canvas還有另外兩個公用的方法（public method）：第一個是我們稍早就使用過的getContext()，本書中我們會不斷地使用到它，就連Canvas 2D context也會使用到，這樣我們才可以在canvas上繪圖；第二個屬性是toDataURL()，它會回傳一個資料字串，該資料字串是目前呈現在Canvas物件上的點陣圖像資訊，就有點像螢幕快照（snapshot）一樣。藉由參數來提供另一種不同的MIME型態，以取得不同型式的資料。基本的型式是*image/png*，而*image/jpeg*與其他種型式都是可以被存取的。在下一個程式裡我們會使用toDataURL()來把canvas上的圖片輸出到另一個瀏覽器的視窗上。

另一個範例：猜字遊戲

我們要快速的看一下這個有"Hello World!"程式的"猜字"遊戲。我們要用這個範例來說明使用JavaScript比使用Canvas API來完成程式快多少。

圖1-4是這個遊戲看起來的樣子，遊戲玩家的工作就是猜出由電腦隨機選出的字母到底是那一個。遊戲會記錄著玩家已經猜過哪些字，並將這些字列出，同時告訴玩家他需要猜更高（往Z的方向）或更底（往A的方向）一點的字。

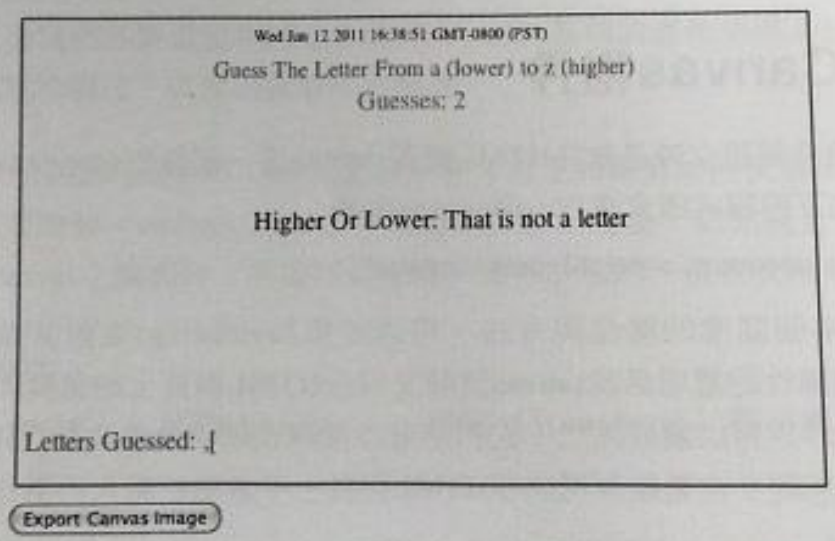

圖1-4 HTML5 Canvas "猜字"遊戲

遊戲如何運作

這個遊戲是從"Hello World!"這個基本架構所衍生而出，canvasApp()是主要的函數，其他的函數都是定義在canvasApp()裡，drawScreen()函數負責在canvas上繪製文字；接下來還會提到程式裡其他的函數。

"猜字"遊戲的變數

在這裡我們會一連串的介紹遊戲中使用到的變數；它們在canvasApp()裡頭被定義及初始化，所以這些變數的範圍就是在我們所定義的封裝函數區域內。

`guesses`

這個變數記錄著玩家按了幾次；數字愈小愈好。

`message`

提供給玩家一些訊息內容的變數，讓玩家知道該猜哪個字。

`letters`

記錄著每一個字母的陣列。就是要使用這個陣列來隨機選出一個神祕的字母，並找出這個字在字母表裡的相對位置。

`today`

這個變數記錄著目前的時間。單純的顯示在螢幕上並沒有其他的意義。

`letterToGuess`

這個變數記錄著正在執行的遊戲是要猜哪個神祕的字母。

`higherOrLower`

這個變數記錄著"Higher"或"Lower"文字，它會依據最後一次猜的字母和神祕字母之前的關係而決定。如果神祕字母比較接近"a"，我們會給"Lower"的提示；如果神祕字母比較接近"z"，我們就會給"Higher"的提示。

`letterGuessed`

這個陣列記錄著玩家已經猜過的字母。這些都會顯示在螢幕上以提示玩家哪些字母他們已經猜過了。

`gameOver`

在玩家獲勝之前這個變數都是設成`false`。程式就是用這個變數來判斷何時該將"You Win"的訊息顯示在螢幕上，而在獲勝之後，玩家就不用再猜了。

這程式碼是：

```
var guesses = 0;
var message = "Guess The Letter From a (lower) to z (higher)";
var letters = [
            "a","b","c","d","e","f","g","h","i","j","k","l","m","n","o",
            "p","q","r","s","t","u","v","w","x","y","z"
            ];
var today = new Date();
var letterToGuess = "";
var higherOrLower = "";
var lettersGuessed;
var gameOver = false;
```

initGame()函數

initGame()函數裡為遊戲的玩家做了一些基本的設定。最重要的兩個程式區塊如下；這段程式碼會在字母陣列中隨機找出一個字母並將它放置於letterToGuess變數中：

```
var letterIndex = Math.floor(Math.random() * letters.length);
letterToGuess = letters[letterIndex];
```

此程式碼為DOM的window物件加入了一個事件監聽者（event listener），用來監控鍵盤keyup的事件。當有一個按鍵被按下時，eventKeyPressed事件控制器（event handler）就會去檢查是哪個按鍵被按下的：

```
window.addEventListener("keyup",eventKeyPressed,true);
```

以下是此函數的完整程式碼：

```
function initGame() {
   var letterIndex = Math.floor(Math.random() * letters.length);
   letterToGuess = letters[letterIndex];
   guesses = 0;
   lettersGuessed = [];
   gameOver = false;
   window.addEventListener("keyup",eventKeyPressed,true);
   drawScreen();
}
```

eventKeyPressed()函數

當玩家按了一個按鍵後，這個函數就會被呼叫，在這個遊戲裡這樣的動作佔了大部份。在JavaScript裡每一個事件控制器（event handler）函數會傳入一個關於發生該事件的event物件。在此我們使用e參數來記錄此物件。

第一個要測試的是gameOver變數看它是否為false；如果是的話，我們便可以繼續檢查玩家所按下的鍵盤按鍵；下兩行程式碼就是要表達這個意思。第一行程式碼是要從event中取得按鍵按下的值，並將它轉換成英文字母並儲存於letterToGuess中以供檢查：

```
var letterPressed = String.fromCharCode(e.keyCode);
```

下一行的程式碼是將字母轉換成小寫的字母，如果玩家不小心將大寫鎖定時，我們還是可以繼續檢查：

```
letterPressed = letterPressed.toLowerCase();
```

再來，我們將要用來顯示猜了幾次的guesses數值加1，並使用Array.push()函數將字母加入lettersGuessed陣列中：

```
guesses++;
lettersGuessed.push(letterPressed);
```

現在，要來檢查遊戲目前的狀態並將結果告知玩家。首先，需測試看letterPressed是否和letterToGuess相同，如果是，玩家就贏得了這場遊戲：

```
if (letterPressed == letterToGuess) {
   gameOver = true;
```

如果玩家沒有獲勝，就必須取得letterToGuess的索引值（index），以及在letters陣列中letterPressed的索引值。我們將用這些值來判斷該顯示"Higher"，"Lower" 或 "That is not a letter"。要這麼做，程式中需使用indexOf()陣列method來取得每一個字母相對應的索引值。因為我們在陣列中有按照字母的順序排列，所以就會很容易的測試出到底要顯示哪一個訊息：

```
} else {
   letterIndex = letters.indexOf(letterToGuess);
   guessIndex = letters.indexOf(letterPressed);
```

現在又要做另一個檢查了。首先，如果guessIndex小於0，就表示呼叫indexOf()時，將會傳回-1，也就是玩家按了一個不是字母的按鍵，此時，就可以在螢幕上顯示錯誤訊息：

```
if (guessIndex < 0) {
   higherOrLower = "That is not a letter";
```

剩下來的測試就簡單多了。如果guessIndex大於letterIndex，就將higherOrLower設定成"Lower"；反過來說，如果guessIndex小於letterIndex，higherOrLower就需設定成"Higher"。

```
   } else if (guessIndex > letterIndex) {
      higherOrLower = "Lower";
   } else {
```

```
        higherOrLower = "Higher";
    }
}
```

最後再呼叫drawScreen()在螢幕上畫圖：

```
drawScreen();
```

以下是此函數的完整程式碼：

```
function eventKeyPressed(e) {
        if (!gameOver) {
            var letterPressed = String.fromCharCode(e.keyCode);
            letterPressed = letterPressed.toLowerCase();
            guesses++;
            lettersGuessed.push(letterPressed);

            if (letterPressed == letterToGuess) {
                gameOver = true;
            } else {

                letterIndex = letters.indexOf(letterToGuess);
                guessIndex = letters.indexOf(letterPressed);
                Debugger.log(guessIndex);
            if (guessIndex < 0) {
                higherOrLower = "That is not a letter";
            } else if (guessIndex > letterIndex) {
                higherOrlower = "Lower";
            } else {
                higherOrLower = "Higher";
            }

        }
        drawScreen();
    }
}
```

drawScreen()函數

現在要進入drawScreen()囉，好消息是，幾乎所有的程式碼在之前我們都看過了－只有一點點的地方和"Hello World!"不一樣，例如，我們使用Canvas Text API在畫面上繪了很多個變數。一但所有的文字都要顯示出來的話，我們只需設定context.text Baseline = 'top'即可。還有，我們使用了context.fillStyle來改變顏色，以及context.font來改變字體。

在這裡最有趣的一件事是lettersGuessed陣列的內容，在canvas裡，陣列的值是要用逗號來做分隔的，就像這樣：

```
Letters Guessed: p,h,a,d
```

為了要將值給印出來，就是使用lettersGuessed陣列裡的toString()函數，這樣就會印出陣列裡的值，值和值之間會用逗號做區隔：

```
context.fillText ("Letters Guessed: " + lettersGuessed.toString(), 10, 260);
```

我們還要測試一個gameOver變數，如果它的值為true，就用40px這麼大的字體將*You Got it!*放在螢幕上，以便讓使用者知道他贏了。

以下是此函數的完整程式碼：

```
function drawScreen() {
      //背景
      context.fillStyle = "#ffffaa";
      context.fillRect(0, 0, 500, 300);
      //方形
      context.strokeStyle = "#000000";
      context.strokeRect(5,  5, 490, 290);

      context.textBaseline = "top";
      //日期
      context.fillStyle = "#000000";
      context.font = "10px _sans";
      context.fillText  (today, 150 ,10);
      //訊息
      context.fillStyle = "#FF0000";
      context.font = "14px _sans";
      context.fillText  (message, 125, 30);
      //猜了幾次
      context.fillStyle = "#109910";
      context.font = "16px _sans";
      context.fillText  ('Guesses: ' + guesses, 215, 50);
      //高或低
      context.fillStyle = "#000000";
      context.font = "16px _sans";
      context.fillText  ("Higher Or Lower: " + higherOrLower, 150,125);
      //猜過的字母
      context.fillStyle = "#FF0000";
      context.font = "16px _sans";
      context.fillText  ("Letters Guessed: " + lettersGuessed.toString(), 10, :
      if (gameOver) {
         context.fillStyle = "#FF0000";
         context.font = "40px _sans";
         context.fillText  ("You Got It!", 150, 180);
      }
}
```

在Canvas上輸入圖片

前面，我們有稍微的討論到Canvas物件的toDataUrl()屬性；我們準備要用這個屬性來讓使用者在任何時刻都可以為遊戲畫面建立圖片；這樣的行為就像是一個在Canvas上的畫面捕捉器一樣。

在HTML網頁上我們要建立一個新的按鈕，這樣使用者按下按鈕時就可以截取該畫面了；這個按鈕我們把它放在<form>裡，並將它的id設為createImageData：

```
<form>
<input type="button" id="createImageData" value="Export Canvas Image">
</form>
```

在init()函數裡，藉由document物件的getElementById() method以取得form裡的元素，然後為按鈕設定"click" event的事件處理器為createImageDataPress()函數：

```
var formElement = document.getElementById("createImageData");
formElement.addEventListener('click', createImageDataPressed, false);
```

在canvasApp()裡，我們定義了createImageDataPressed()函數為一個事件處理器，此函數會呼叫window.open()，並傳入Canvas.toDataUrl()函數的回傳值做為視窗的來源值；因為資料格式是一個合法的*PNG*，所以影像就會被顯示在另一個新的視窗裡。

```
function createImageDataPressed(e) {

 window.open(theCanvas.toDataURL(),"canvasImage","left=0,top=0,width=" +
 theCanvas.width + ",height=" + theCanvas.height +",toolbar=0,resizable=0");
 }
```

在第三章我們將會做更深入的探討。

遊戲的最終程式碼

範例1-4完整的列出猜字遊戲的程式碼：

範例1-4 猜字遊戲

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>CH1EX4: Guesss The Letter Game</title>
<script src="modernizr-1.6.min.js"></script>
```

```
<script type="text/javascript">

window.addEventListener('load', eventWindowLoaded, false);

var Debugger = function () { };
Debugger.log = function (message) {
   try {
      console.log(message);
   } catch (exception) {
      return;
   }
}

function eventWindowLoaded() {

   canvasApp();
}

function canvasSupport () {
     return Modernizr.canvas;
}

function eventWindowLoaded() {

   canvasApp();
}

function canvasApp() {
   var guesses = 0;
   var message = "Guess The Letter From a (lower) to z (higher)";
   var letters = [
              "a","b","c","d","e","f","g","h","i","j","k","l","m","n","o",
              "p","q","r","s","t","u","v","w","x","y","z"
              ];
   var today = new Date();
   var letterToGuess = "";
   var higherOrLower = "";
   var lettersGuessed;
   var gameOver = false;

   if (!canvasSupport()) {
         return;
      }

   var theCanvas = document.getElementById("canvasOne");
   var context = theCanvas.getContext("2d");

   initGame();

   function initGame() {
      var letterIndex = Math.floor(Math.random() * letters.length);
      letterToGuess = letters[letterIndex];
```

```
        guesses = 0;
        lettersGuessed = [];
        gameOver = false;
        window.addEventListener("keyup",eventKeyPressed,true);
        var formElement = document.getElementById("createImageData");
        formElement.addEventListener('click', createImageDataPressed, false);
        drawScreen();
    }

    function eventKeyPressed(e) {
        if (!gameOver) {
            var letterPressed = String.fromCharCode(e.keyCode);
            letterPressed = letterPressed.toLowerCase();
            guesses++;
            lettersGuessed.push(letterPressed);

            if (letterPressed == letterToGuess) {
                gameOver = true;
            } else {

                letterIndex = letters.indexOf(letterToGuess);
                guessIndex = letters.indexOf(letterPressed);
                Debugger.log(guessIndex);
                if (guessIndex < 0) {
                    higherOrLower = "That is not a letter";
                } else if (guessIndex > letterIndex) {
                    higherOrLower = "Lower";
                } else {
                    higherOrLower = "Higher";
                }

            }
            drawScreen();
        }
}

function drawScreen() {
    //背景
    context.fillStyle = "#ffffaa";
    context.fillRect(0, 0, 500, 300);
    //方形
    context.strokeStyle = "#000000";
    context.strokeRect(5,  5, 490, 290);
    context.textBaseline = "top";
    //日期
    context.fillStyle = "#000000";
    context.font = "10px _sans";
    context.fillText  (today, 150 ,10);
    //訊息
    context.fillStyle = "#FF0000";
    context.font = "14px _sans";
    context.fillText  (message, 125, 30);
    //猜了幾次
```

```
        context.fillStyle = "#109910";
        context.font = "16px _sans";
        context.fillText  ('Guesses: ' + guesses, 215, 50);
        //高或低
        context.fillStyle = "#000000";
        context.font = "16px _sans";
        context.fillText  ("Higher Or Lower: " + higherOrLower, 150,125);
        //猜過的字母
        context.fillStyle = "#FF0000";
        context.font = "16px _sans";
        context.fillText  ("Letters Guessed: " + lettersGuessed.toString(), 10, 260);
        if (gameOver) {
           context.fillStyle = "#FF0000";
           context.font = "40px _sans";
           context.fillText  ("You Got It!", 150, 180);
        }
    }

    function createImageDataPressed(e) {

        window.open(theCanvas.toDataURL(),"canvasImage","left=0,top=0,width=" +
        theCanvas.width + ",height=" + theCanvas.height +",toolbar=0,resizable=0");
    }

}

</script>
</head>
<body>
<div style="position: absolute; top: 50px; left: 50px;">
<canvas id="canvasOne" width="500" height="300">
 Your browser does not support HTML5 Canvas.
</canvas>
<form>
<input type="button" id="createImageData" value="Export Canvas Image">
</form>
</div>
</body>
</html>
```

接下來呢?

到目前為止你應該對於在網頁上使用HTML與JavaScript來繪圖並控制HTML5 Canvas有了基本的認識。在接下來的一章裡，我們會在這樣的基礎下將程式再擴大些，並建立一個互動式的應用程式，在螢幕上用canvas呈現出所有的資訊。