附錄 B

呈現方式與樣式一覽

對於Windows來說，使用HTML、CSS及Javascript來建立「原生」應用程式是全新的概念，所以這些語言對你而言就好像外語；而且在遠離Web數年之後，對這些語言甚至會覺得有點陌生。但無論如何，本附錄在此會讓你快速上手並熟悉HTML及CSS，無論你是再次或首次接觸。我們先從瞭解HTML開始，特別是那些導入HTML5的新元素。其中有一些可以讓我們建立稱為是語意標記的東西，也可說是一種能夠更好顯示頁面上元素目的的標記，因此更容易維護。其他的元素可以讓我們以令人興奮的新方法，在應用程式中使用媒體與多媒體。最後，我們將討論HTML在Windows市集應用程式中的本質，並且看看Windows市集應用程式最為簡單而不同尋常的結構。

談到HTML，我們就要把話題將切換到備受爭議的語言，CSS。因為許多開發人員，甚至是網頁開發人員，對於CSS的瞭解及體驗的程度都不相同，所以我們將先從討論CSS本身開始，它是什麼、它如何運作，還有將CSS規則包含在頁面中的多種方法，還有使用它的最佳方式。我也會分享一段簡短初級的CSS規則層疊，這將有助於你在嘗試要弄清楚為什麼有時候某些規則會被覆蓋或是被忽略時，避免落入常見的陷阱。最後，我們將會討論Windows市集應用程式中的CSS，包括如何套用預設樣式以及如何在應用程式中快速地覆蓋這些樣式。

為應用程式內容及結構使用HTML

就如同你現在會滿懷希望的推測，Windows市集應用程式可以使用各種不同的程式語言（Javascript、C#、VB.NET或C++），以及兩種表現語言（XAML或HTML/CSS）其中的一種撰寫。對於後者HTML與CSS的組合來說，HTML是用來定義應用程式的內容及結構，而CSS是用來排版及呈現。現在就讓我們來看看HTML。

HTML是什麼？

HTML是Hypertext Markup Language（超文本標記語言）的縮寫，從一開始，HTML就被視為是在網頁上編寫內容的近乎通用的語言[1]。網際網路工程工作小組（Internet Engineering Task Force, IETF）於1993年發佈了HTML 1的版本，接下來在1993至1999年間從第2版發展到4.01版，最後就由全球資訊網協會（World Wide Web Consortium, W3C）所主管。在撰寫本書的此時，市面上幾乎每一種可取得的瀏覽器，包括Internet Explorer 6及之後的版本，還有Firefox、Chrome、Safari、Opera還有其他瀏覽器的版本，都支援HTML 4.01，該版本也是由W3C管理的目前最佳的版本[2]。

無論是什麼版本，每一個HTML文件都是由單獨的根<html>元素以及兩個子元素，<head>及<body>所組成，看來就像下面這樣：

```
<html>
  <head>
    <title>My Page</title>
  </head>
  <body>
    <p>The coolest HTML page ever</p>
  </body>
</html>
```

1 我說「近乎通用」的原因是其他的標記語言，例如XML，也可以被瀏覽器使用及解析。此外，這些年來行動與手機瀏覽的世界也來來去去地出現了一些標記語言（例如，WML）。

2 HTML 4.01的規格可以在此找到http://w3.org/TR/html401/（http://tinysells.com/227）。

如果你將上述的標記放到檔案中，並將檔案取名為indel.html，然後在任一個過去15年內製造的瀏覽器中載入此頁面，這些標記就會依照所期望的方式顯示出來。它也是有史以來產生的最無聊頁面，所以就不截圖啦！

在HTML被創造出來的多年以前，Web僅是一個帶有少數影像及超連結到其他文字文件的文字文件集合。這種「Web小冊子」定義了那段時間的Web，但是在2000年代早期，Web的發展已經超越了超連結文件的組合，進入到一種線上零售、社交場合甚至應用程式的集合。這都得感謝HTML的簡單，無所不在的網頁瀏覽器，比以往更快的網際網路連線，讓Web快速成長。

隨著不斷地成長，由HTML、Javascript以及CSS可能形成的邊界也持續地延伸。之後，在2000年代中期，HTML 4.01及CSS 2.1需要演化以支持現代的網站及應用程式，已經是明顯不過的事。因此，就導入了HTML5及CSS3[3]。

HTML 5的新內容

這些日子以來，無論你問誰，HTML5幾乎可以代表一切。對於某些人，它就是一個涵蓋性的專有名詞，包含了Web上全新且有趣的每一件事，從HTML和CSS，到新的Javascript API和語言的改進。對於其他人，HTML5看起來就像是百分之百的嘈雜聲和立刻消失。然而，根據定義，HTML5是一個由W3C維護的獨立又冗長的規格。這個規格涵蓋了每件事，從新的元素及屬性、新的輸入型態及新的網頁表單驗證規則，到瀏覽器實作該規格的解析規則所有內容。就讓我們看看HTML5對於Windows市集應用程式開發人員來說特別有趣的三個新面向。

3 事實上，HTML5的歷史是比我在這裏所寫出來的要更加複雜，有兩個標準組織（而不是一個）對規格一開始有衝突，後來有相同的目。HTML5完整的起源已經超過了本附錄的範圍，不過我強烈推薦Mark Pilgrim所撰寫的*HTML5: Up and Running*（O'Reilly, 2010），這本書的第一章，有更深入的說明。

語義標記

首先，我們要看的是語義。語義這個字的簡單意思就是「含義」，而且要在HTML5的背景之下介紹這個概念，就讓我們看看典型的HTML5之前的文件：

```
<html>
  <head><title>My Page</title></head>
  <body>
    <div id="header">
      <h1>Wikipedia</h1>
      <imgsrc="images/Wikipedia-logo.png" alt="Wikipedia Logo" />
    </div>
    <div class="nav">
      <ul class="menu">
        <li>Home</li>
        <li>Articles</li>
       <li>Contact Us</li>
      </ul>
    </div>
    <div id="main">
      <div class="article">
        <p>Article one text</p>
      </div>
      <div class="article">
        <p>Article two text</p>
      </div>
    </div>
    <p>Some text.</p>
      <div>
        <p>Some text.</p>
        <p>Some more text.</p>
        <p>Even more text.</p>
        <div id="level">
          <p>Even more more text</p>
        </div>
        <p>The most of all text.</p>
      </div>
    <div id="footer">
      <p>Copyright &copy; 2012 – Carrot Pants Studios</p>
    </div>
  </body>
</html>
```

這份文件在今日經常可在線上的網站找到。結構很清晰，而且主要地是以 為基礎，使用class或id屬性來將意義附加到那些 上，這也使它們能夠很容易地透過CSS來選擇以進行樣式化。在Web的環境下， 是一個語義化的中性元素，也就是說，它自己本身是無法傳達它為什麼會在這份文件中的含義。雖然class和id有助於進行解釋，但除非它們也使用在CSS中，否則他們在增加清晰的同時也帶來了混亂。在這個範例中的其他元素，如<ul>（無順序的項目清單列表），<p>（段落）以及<img>（影像）都被稱之為「語義豐富」，因為它們清楚地傳達了其功能及目的的意義。不需要額外的標記來解釋它們為什麼會在那裏。

HTML5一個主要的特色之一就是導入了新的、語義富豐的HTML5元素，開發人員可以使用它們來編寫清楚無疑義的標記。這裏有一個例子，是在上述的傳統範例中使用了幾個新元素：

```
<!doctype html>
<html>
  <head>
    <title>My Page</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <header>
      <h1>Wikipedia</h1>
      <img src="images/Wikipedia-logo.png" alt="Wikipedia Logo" />
    </header>
    <nav>
      <ul class="menu">
        <li>Home</li>
        <li>Articles</li>
       <li>Contact Us</li>
      </ul>
    </nav>
    <div id=""main"">
      <article>
        <p class="summary">Article one text</p>
      </article>
      <article>
```

```
        <p>Article two text</p>
        <input type="text" id="name" />
      </article>
      <p>Some text.</p>
      <div>
        <p>Some text.</p>
        <p>Some more text.</p>
        <p>Even more text.</p>
        <div id="level">
          <p>Even more more text</p>
        </div>
        <p>The most of all text.</p>
      </div>
    </div>
    <footer>
      <p>Copyright &copy; 2012 - Carrot Pants Studios</p>
    </footer>
  </body>
</html>
```

如你所見，新的元素像<header>、<nav>、<article>和<footer>傳達了為何它們會處在頁面中的清晰意義，同時還能夠針對不需要的class和id屬性進行處理。你也會注意到即使導入了新的HTML元素，並不意謂著已不再需要舊的好的 。但就像之前那樣，你仍會經常使用 這個非常有用的通用分割元素，儘管HTML5中那些導入的新元素會讓你不用像以前那樣經常地使用<dvi>。

在上述範例中所列舉的新的HTML元素只是目前最常使用的一部份，所以你也可以看到這些標記廣泛地使用在Windows市集應用程式，還有線上文件[4]。現在，就讓我們簡要地討論一些媒體標記元素。

媒體標記

除了規格導入新的語義元素之外，HTML5經常與四個類型的媒體元素關聯在一起，它們在各個瀏覽器之間都有廣泛的支援及採用，也同樣被Windows市集應用程式開發所支援。

4 HTML5新元素的完整清單以及它們的預期用法，請參閱：http://msdn.microsoft.com/en-us/library/windows/apps/hh767420.aspx（http://tinysells.com/228）.

前兩種類型的媒體標記是canvas和svg，這兩個媒體元素通常與繪圖及動畫有關。實際上它們兩個存在的歷史比HTML5還要長一而且svg有另一個規範一不過它們一般都被廣泛地認為是HTML5所涵蓋的一部份[5]。canvas和svg是由相對應的HTML元素<canvas>及<svg>來表示，並透過XML標記（svg）或Javascript（Canvas）來管理。我已經在第六章中提過這兩個元素，你可以閱讀那一章來學習關於這兩個元素的更多知識。

而HTML5所導入的其他媒體元素是<audio>及<video>，就像它們的名稱含義一樣，允許開發者整合音訊與視訊軌道到它們的應用程式中，在引入這兩個元素之前，Web應用程式中要使用媒體的話，就需要像Silverlight或Flash這樣的附加元件，所以這些元素加入HTML家族之後非常受歡迎。有了HTML5音訊和視訊，開發人員也可以透過CSS為媒體元素來設定樣式，甚至可以透過Javascript來存取及操作它們。想要更詳細地瞭解HTML音訊及視訊，請參閱第五章的內容。

資料屬性

我們要簡短討論的HTML5第三個面向，肯定在規格上並不具備最讓人興奮的部份，但卻是Windows市集應用程式開發的一個關鍵部份，也正因如此，所以在此更值得關注。這個特性就是資料屬性，經常因為它的語法而被稱之為data-*屬性[6]。

data-*是特殊的HTML5屬性類型，你可以使用它把自定的中繼資料加到標記中，對script來說是有效而且是可利用的。這也是這個功能如此強大並且在網頁上廣泛使用的關鍵原因。假設我要在標記中加入一個特別的控制項屬性：

```
<div data-control="clock" id="myClock">Tick Tock</div>
```

5 想要瞭解更多有關於這些元素的資訊，請參閱http://dev.w3.org/html5/2dcontext/（http://tinysells.com/229）來瞭解Canvas，另外請參閱http://www.w3.org/TR/SVG/（http://tinysells.com/230）來瞭解SVG規範。

6 HTML5規格已經包含了資料屬性，你可以在以下相關網址瞭解相關內容：http://w3.org/TR/html5/global-attributes.html#embedding-custom-nonvisible-data-with-the-data-attributes（http://tinysells.com/231）

在control前使用data-可以讓我使用任何的自訂中繼資料來裝飾我的元素。這樣非常棒，但你可能會想知道為什麼這樣會有價值，或者是為什麼在一開始時是必要的。畢竟，從來沒有什麼可以阻止我在標記中增加任意的屬性：

```
<div control="clock" id="myClock">Tick Tock</div>
```

當我在瀏覽器中觀看這個標記時，每個東西看來都呈現得不錯。此外，我可以在Javascript中，透過 屬性上的getAttribute方法從技術上去存取這個屬性。那麼，為什麼我要關心那額外的五個特性呢？

有兩個原因：有效性及瀏覽器支援。資料屬性所提供的最大好處之一是能夠定義自訂且完全任意的HTML屬性，同時也是完全有效的HTML。所以，如果我要透過HTML5驗證器[7]來執行第一個帶有data-controls屬性的例子，會通過驗證。換句話說，如果我直接以control屬性來執行這段程式碼，測試就會失敗。我們非常關心像AML及XML這些標記語言的有效性，儘管它的本質是寬容的。

data-*屬性的另一個好處是關於內建瀏覽器支援。對於開發人員來說，除了可以透過傳統的方式使用之外，在Javascript中，在DOM元素上使用新的data屬性也可以快速地存取data-*屬性：

```
var clock = document.querySelector('myClock');
clock.data["control"]; // "clock"
```

我稍早提到data-*是值得關注的是因為它們已經在Windows市集應用程式中扮演了主要的角色，而如果你從頭開始閱讀本書，到此時你也許應該猜到為什麼了：

```
<div data-win-control="WinJS.UI.ListView"
     data-win-options="{
                        itemDataSource: mySource,
                        itemTemplate: select('#myTmpl') }"></div>
```

7 就像http://html5.validator.nu/（http://tinysells.com/232）的這一個。

為了能夠以宣告的方式使用控制項，WinJS使用特別為Windows市集應用程式的data-*屬性。透過使用這些屬性，開發者可以定義簡單且有效的標記，讓Windows 8可以為你的應用程式點燃「亮點」。當你在Javascript中看到以下的指令時，就能夠做到：

```
WinJS.UI.processAll();
```

遇到這個命令時，Windows 8會解譯你的標記，並且在遇到data-win-*屬性的地方，所有的應用程式控制項及選項會進行處理及初始化。這是非常平順的，而且，這些都是完全有效的HTML5。

使用CSS進行應用程式排版及樣式化

HTML在定義應用程式的結構及內容方面是很有用的，但如果沒有CSS的協助，它就只是個無聊的白底黑字網頁。例如，圖B.1顯示了顯示了我們漂亮、語義標記在最早未套用CSS時的樣子，除了繪製引擎最初提供的預設樣式之外就什麼都沒有了。

CSS對建立使用者所喜愛的應用程式是不可或缺的。即使我們開發人員最討厭CSS，它也是一種可供我們使用既強大又簡易的語言。就讓我們快速地看一下CSS的詳細情況。

什麼是CSS？

層疊樣式表，或者叫CSS，是一種樣式化語言，可以讓你定義排版及呈現規則，並且可以應用到所有它支援的標記格式（CSS也可以套用到XML，如果想要套用的話）。就像HTML，CSS也是由W3C進行標準化。目前CSS的完全標準化版本是2.1，而CSS3是最新的，也是最棒的一個發展中的版本，是目前所有瀏覽器的目標。就像Windows 8中的Internet Explorer 10，Windows市集應用程式完全支援CSS2.1以及CSS3的一些功能。其中大部份都已在本書中涵蓋，第三章、第四章以及第六章都包含了對新CSS功能最深層的涵蓋。

Wikipedia

- Home
- Articles
- Contact Us

Article one text

Article two text

Some text.

Some more text.

Even more text.

Even more more text

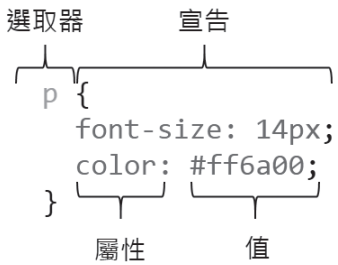
The most of all text.

Copyright © 2012 – Carrot Pants Studios

圖B.1 未套用CSS的HTML網頁。

我們先看一下CSS的基本知識。這個語言由四個部份構成：選取器、宣告、屬性和值，如圖B.2所示。

選取器是一個字串，指定一個給定的樣式應該套用到的一個或多個元素，在大括號裏，或是在選取器之後，會有一個或多個宣告，由分號隔開。每個宣告都包含了由冒號隔開的成對組合，它指定了一個屬性及結果



圖B.2 CSS語言組成結構

值，用於設定選擇器所符合的元素。在圖B.2的範例中，我把頁面上的所有段落（<p>）的font-size屬性設定成14px。我也把段落文字的顏色設定為橘色的RGB值。結果如圖B.3所示。

在CSS中，屬性可用於樣式、位置以及文字及區塊元素的轉換，也可以用於管理高級的文件排版規則。例如，下列這些都是有效的CSS樣式屬性：

```
height: 120px;
width: 20%;
font: italic 16pt 'Lucida Sans', Helvetica, sans-serif;
color: #fff;
background: url('myBG.jpg') no-repeat;
margin: 0px 5px 5px 10px;
padding: 10px;
```

在此我們無法更加深入地討論CSS屬性，所以我會鼓勵你參考Windows發人員中心以取得更多關於CSS屬性資訊，以及如何在Windows市集應用程式中使用。此外，要特別關注本書所使用的CSS屬性，特別是前面以CSS為重點的那幾章。

CSS選取器

從另一方面來看，CSS選取器還需要一些額外的解釋，所以我們現在來看一下。就如我先前所提到的，CSS選取器是一個字串值，它從頁面的DOM中「選取」一個或多個元素來進行樣式化。它們可以像標籤名稱那麼簡單，也可以像同層相鄰選取器、屬性選取器或是偽類選取器那麼複雜。

圖B.3 為應用程式設定CSS樣式

基本CSS選取器：CSS選取器最簡單的別就是通用選取器（*），它可以將樣式套用到文件中所有的元素。在下列的程式碼片段中，我使用了CSS選取器，將文件中所有的文字設成是12像素的Arial字體：

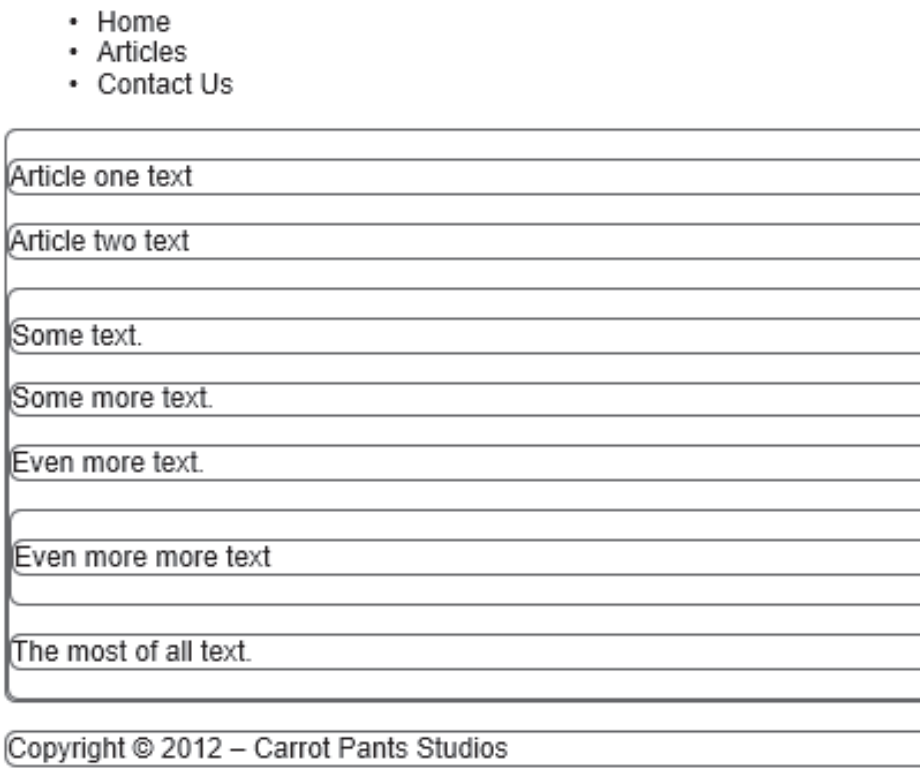
```
* {
  font-family: 'Arial';
  font-size: 12px;
}
```

另一種簡單的選取器是類型選取器，它透過元素的標籤來選取元素。例如，我可以透過參考各自的標籤來選取頁面上所有的 及<p>：

```
p, div {
  border: 1px solid #0094ff;
  border-radius: 5px;
}
```

這些樣式套用的結果如圖B.4所示。

在這個範例中你也會注意到我使用了逗號來分隔多個選取器。這是非常有用的方式，可以在多個元素或元素群組上套用通用樣式，而不必在宣告中重覆屬性。



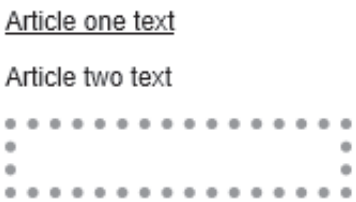
圖B.4 為div及段落套用邊框

另外兩個常用的選取器是class及id選取器，分別使用點（.）及井號（#）表示。在HTML中，id表示是頁面上元素的唯一標識，可透過CSS或Javascript來操作，而出於相同的目的，class可以用來將元素指定至通用的群組中。當頁面元素使用class或id修飾時，可以使用這些選取器類型來為它們進行樣式化：

```
<!-- foo.html -->
<article>
  <p class="summary">Article one text</p>
</article>
<article>
  <p>Article two text</p>
  <input type="text" id="name" />
</article>

/* foo.css */
#name {
  border: 5px dotted #ff6a00;
  width: 150px;
  height: 25px;
}

.summary {
  text-decoration: underline;
}
```

這些樣式套用的結果如圖B.5所示。在這一節中，我們要看的最後兩個基本的CSS選取器是子選取器與後代選取器，它們可以讓我們以文件中元素的父—子關係基礎來捉取元素。子選取器（>）可以讓我們選取一個元素的所有直屬的子元素或是與父代選取器相符的元素。

```
<!-- foo.html -->
<div id="#myText">
  <p>Some text.</p>
  <p>Some more text.</p>
  <p>Even more text.</p>
  <div id="level">
    <p>Even more more text</p>
  </div>
  <p>The most of all text.</p>
</div>

/* foo.css */
div#myText > p {
  color: blue;
}
```

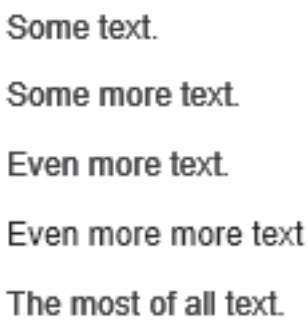


圖B.5 透過class及id來套用樣式

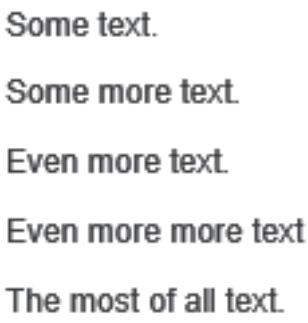
在這個範例中，我結合了其他的基本選取器，`type`及`id`，透過使用`mytext`的`id`來參考`<div>`，以啟動我的選取器。接著，我限縮了我的選取器，透過使用大於符號（`>`）並隨著段落類型選取器，只選取目前直屬的段落子元素。這個範例的結果是，只有在`<div>`的前三個及最後一個段落有套用藍色文字的樣式，如圖B.6所示。

後代選取器的作用也非常類似，不過它不是尋找父代元素的直屬子代元素，這個類型會選取所有父代的後代，即便它們不是直屬的子代。在本質上，這種選取器只不過是用空格隔開的選取器清單，可以用在「遍及DOM樹」，並且能夠以任意深度找到存在於父元素的一個或多個特定元素。

```
/* foo.css */
div#myText p {
    color: blue;
}
```



圖B.6 使用子選取器樣式化CSS



圖B.7 使用後代選取器來樣式化CSS

在這個範例中，我只使用了空格來取代大於符號，它就會把樣式套用到所有的段落中，而不僅是前三個跟最後一個段落。所以，如果我像上一個例子一樣，在相同的標記上使用了修改後的選取器，我就會得到如圖B.7的結果。

進階CSS選取器：我們討論過的選取器也許很基礎，不過你會發現到它們可以滿足大部份的樣式化需求。然而，這些選取器也有它們的限制，所以CSS提供了一大堆額外的基本與進階選取器，在有需要的時候就可以使用。我們無法在本附錄中涵蓋所有的高階選取器，但會著重強調幾個常在Windows市集應用程式中看到的進階選取器[8]。

第一個介紹的選取器是屬性選取器，就像它的名稱一樣，可以讓你尋找給定的屬性是否存在，以及（或者）它的值為何，然後從DOM中選取這個屬性：

```
<!-- foo.html -->
<input type="tel" id="phoneNum" data-foo="A made-up attribute" />
<input type="text" id="name" />
<div data-win-control="WinJS.UI.ListView" id="list"></div>

/* foo.css */
input[type="text"] {…} //Selects name
[data-win-control="WinJS.UI.ListView"] {…} //Selects list
```

8 想要獲得更多的進階選取器資訊，請參考以下的網址：http://css-tricks.com/category/advanced/（http://tinysells.com/234）

屬性選取器會出現在中括號內，要不放在元素的旁邊來檢查是否符合條件，就是單獨用來檢查DOM中的所有元素的屬性。你可以使用這些選取器來尋找帶有給定值的屬性，或是尋找屬性是否存在，或者甚至你可以使用正規表示式來選取具有部份屬性值的元素，等下我們會看到這種用法。

另外值得提及的兩種類型選取器，是因為它們在網頁及Windows 8中的突出位置，它們分別是虛擬類別及虛擬元素選取器。不同於一般的CSS選取器對實的DOM元素進行選取及操作，這兩種選取器是比較獨特的，它們不是針對DOM元素進行操作，就是在DOM外面進行操作。虛擬類別及虛擬元素選取器兩種都可以用分號來定義，後面接著虛擬類別識別子。在這兩種選擇器中，某些虛擬類別選擇器對你來說應該比較熟悉了。

```
<!-- foo.html -->
<a href="http://www.microsoft.com">Microsoft</a>
<input type="text" id="name" disabled />

/* foo.css */
a:hover { font-size: 22px; }
input[type="text"]:disabled { color: grey; }
```

這些樣式的結果如圖B.8所示。

圖B.8 使用虛擬類別選取器對元素套用樣式

許多的虛擬類別選取器，如hover、active、visited、enabled、disabled、checked及其他類似的選取器，可以讓你利用文件中聯結輸入元素的各種狀態，來調整這些狀態被使用者觸發時所使用的樣式。其他的選取器可以用來精簡已選取元素的數目，或者甚至是精簡元素所選擇的範圍。假設我們使用前面例子中的mytext div，在其中加入一些樣式：

```
/* foo.css */
p:not(.first) { font-weight: bold; }
p:first-letter { text-transform: uppercase; font-size: 24pt; }
```

在這個範例中，我使用了not虛擬類別只選取不包含first類別的段落，之後，我使用了first-letter虛擬類別來捉取所有段落中的第一個字母，並且使用text-transform屬性將字母變成大寫[9]。結果如圖B.9所示。

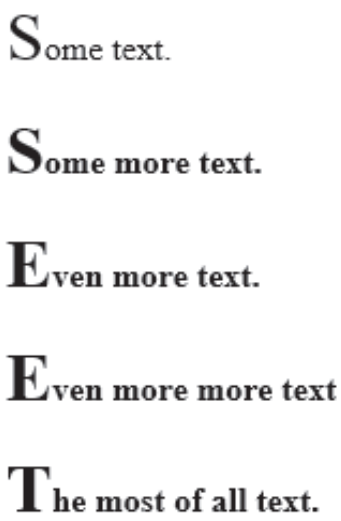
其他可用的虛擬選取器是虛擬元素選取器。和虛擬類別選取器對元素及其狀態進行操作不同，虛擬元素是存在於DOM之外的「幽靈」元素，靠近它們的父元素。

```
<!-- foo.html -->
<button data-win-control="back" id="back"></button>
<p>This is some text </p>

/* foo.css */
button::before { … }
p::after { … }
```

before及after是目前唯一可用的虛擬元素，最常用於加入影像或者對父元素進行樣式化，而不用在頁面本身增加額外的標記。在Windows 8的Windows市集應用程式中，許多視覺化的控制項及AppBar按鈕都使用了before虛擬元素，在第四章我已經討論過這些內容了。

CSS3選取器：除了在CSS中已經存在很久的基本及進階的選取器之外，CSS3導入了一組新的選取器，也被Windows 8的Windows市集應用程式所支援。其中有一些你可能在前面幾章已經瞭解了。就像在這裏所提及的其他類型選取器一樣，新的CSS3選取器的功能涵蓋了從基本到進階的，還有幾個介於兩者之間。有一些選取器我覺得很有趣，也希望你能夠經常使用，

9 要更深入的探索CSS2.1及CSS3中的虛擬類別選擇取器，可以參考Chris Coyier的這篇文章：http://css-tricks.com/pseudo-class-selectors/（http://tinysells.com/234）。



圖B.9 額外的虛擬類別樣式

它們分別是子字串屬性選取器及新的位置選取器。

子字串屬性選取器擴充了我們在前面討論過的屬性選取器的能力，而且可以讓你使用類似正規表示式的語法來選取元素。

```
<!-- foo.html -->
<div id="cal" data-win-control="WinJS.UI.DateTimePicker"></div>
<div id="list" data-win-control="WinJS.UI.ListView"></div>

/* foo.css */
div[data-win-control^="WinJS"] {...} // 與cal和list符合
div[data-win-control$="ListView"] {…} // 與list符合
div[data-win-control*="UI"] {…} // 與cal和list符合
```

就像這裏所展示的一樣，你可以在現有的屬性選取器內部，使用三種子字串符號（^、$或是*）中的任何一個，根據是否有值相對於存在於屬性的開始、結束或是任一位置來選取元素。

另一個值得關注的新的CSS類型選取器是位置虛擬類別選取器，它可以更容易地基於元素在兄弟元素之間的位置來選擇。這些選擇器可以根據元素相對於其他兄弟元素、父元素或子元素的位置，或者相對於相同類型子元素的位置，來迅速選取一個特定的元素。

```
<!-- foo.html -->
<ul>
  <li>First</li>
  <li>Second</li>
  <li>Third</li>
  <p>Fourth</p>
  <li>Fifth</li>
  <li>Sixth</li>
</ul>

/* foo.css */
ul:first-child {..} //選取第一個
ul:last-child {..} //選取第六個
ul:nth-child(2) {..} //選取第二個
li:first-of-type {..} //選取第一個t
li:nth-of-type(4) {..} //選取第五個
```

相較於其他的選取器，這些選取器是有些囉唆，但他們的威力十分強大，可以讓你在給定類別的兄弟元素或子元素之中選取第一個或最後一個元素，或甚至根據類型來選擇子元素或是元素。Nth-of-type對於常見的清單或表格資料樣式化，也支援一種有用的快捷處理方式。

```
<!-- foo.html -->
<ul>
  <li>First</li>
  <li>Second</li>
  <li>Third</li>
  <li>Fourth</li>
  <li>Fifth</li>
</ul>

/* foo.css */
li:nth-of-type(odd) {…} // 選擇第一個、第三個、第五個
li:nth-of-type(even) {…} // 選擇第二個、第四個、第六個
```

odd及even關鍵字可以在有需求的時候，很容易地在兄弟元素之間選取奇數及偶數元素。對於清單及表格資料，這些選取器在套用斑馬條紋或其他非連貫的元素進行樣式化的時候是非常方便的，不需要使用單獨為樣式化目的的類別來裝飾標記。如果你需要選取的元素不僅是偶數或奇數的話，還有其他有用的位置選取器供你處理[10]。

10 新CSS3選取器的完整清單，請參考：http://tools.css3.info/selectors-test/test.html。

應該在何處定義CSS？

現在我已經用旋風式導覽帶你瞭解CSS選取器，接下來就讓我們來談談你應該在哪裏為Windows市集應用程式定義CSS。CSS樣式可以在元素的程式碼中定義，或是在頁面的<style>區塊進行定義，但是有一種被廣泛認可的最佳實務是在外部檔案定義樣式，然後在頁面的標頭處參考它。要解釋為什麼要這樣做，就讓我們先看一個用各種方法定義頁面樣式的範例。

在有CSS之前，網頁開發人員可以使用style屬性在單獨的元素上，為它們的標記添加樣式。這種實務做法被認為是建立行內樣式，以下是範例：。

```
<p style="color: blue">This is Blue</p>
<p style="color: green">This is Green</p>
<p style="color: blue">This is also blue</p>
```

定義CSS行內樣式是十分容易的，特別是在你編寫標記時就可以同時加入，不過這有兩個問題。首先，這種寫法很冗長，而且會把你的標記弄亂—而標記應該專注在內容及結構上—但現在又夾雜著外觀與樣式，會讓標記更難以理解。這個實作方式會讓應用程式變得更難以維護。請注意我們有兩個元素具有相同的樣式，所以，如果你需要加入一個額外的樣式，將所有藍色的文字也變成粗體的時候，會發生什麼事呢？

```
<p style="color: blue; font-weight: bold;">This is blue</p>
<p style="color: green">This is green</p>
<p style="color: blue; font-weight: bold;">This is also blue</p>
```

在兩個元素的情況下，看來還不太糟，但是如果有五個、十個或更多的元素呢？這還不夠的話，如果頁面上的每一個元素散落著這些屬性呢？這本來是一種快速加入樣式的快捷方式，但卻變成了一場維護的惡夢。幸好，我們還有一種更好的方法：

```
<html>
  <head>
    <style>
p {
  color: green;
}
```

```
p.blue {
  color: blue;
  font-weight: bold;
}

    </style>
  </head>
  <body>
    <p class="blue">This is blue</p>
    <p>This is green</p>
    <p class="blue">This is also blue</p>
  </body>
</html>
```

透過在頁面的標頭加入樣式區塊，我們不但可以把關注外觀的標記從內容及結構中分離出來，而且還可以透過建立套用多個元素上的規則來減少重覆。

的確，「標頭」方法是對行內樣式的改善，但如果我們在多個頁面有兩個或兩個以上的元素需要套用相同的樣式時，就像網站經常會遇到的情況，在每個頁面都有通用的排版或外殼？對於這些情況，我們可以使用一個指向CSS擴充檔案的連結來取代標頭的樣式宣告，在頁面之間共享樣式。然後，我們可以將樣式移到外部樣式表，在需要的地方就可以使用。在這個範則中，`styles.css`檔案包含了與上述程式碼中的樣式區塊一樣的規則：

```
<html>
  <head>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <p>This is blue</p>
    <p>This is green</p>
    <p>This is also blue</p>
  </body>
</html>
```

多少跟Javascript一樣，此時你要使用頁面層級的標頭元素來定義樣式，如果你正在建立的範例或甚至是應用程式，永遠都只有一個單獨的HTML文件，我想應該都沒問題。但對於所有超出這種例子的情況，我建議你最好總是利用外部的樣式表。我也建議你永遠不要使用行內樣式，即使在最簡

單的情況下。建立CSS是為了用來減輕剝離行內樣式的痛苦，所以請不要再重覆過去的錯誤。永遠請將你的樣式放在樣式區塊或是外部樣式表中。

到目前為止，我們在本節中討論了什麼是CSS、它的語法以及如何在應用程式中包含CSS。接下來，就讓我們看看CSS如何評估及套用，還有當兩個衝突的規則定義同一個元素時，會發生什麼事。

CSS規則如何層疊

當CSS被套用到文件時，它的規則會依序被評估及處理，就像HTML元素和script檔案一樣。不管你是用樣式區塊或是外部樣式，CSS都是使用同一種評估演算法，稱之為層疊（cascade），是用來確認哪一條規則要套用在哪一個元素上。對我們來說，大多數的時間只要知道層疊「有起作用」可以了，不過當元素的樣式定義發生衝突時，精確地知道CSS如何套用樣式就非常重要了。

最後規則

CSS的第一個評估方式是最後規則評估，意思是當兩個規則一起指定時，在標頭及外部樣式表中所有樣式中最後定義的規則，將會成為套用到元素上的規則。

```
p { color: green; }
p { color: blue; } // 最後的規則，贏家

<p>Some text</p>
<p>Some more text</p>
```

在這個範例中，我定義了兩個樣式，要用來改變應用程式中所有段落文字的顏色。在這種情況下，勝出的是最後一個規則，因為它定義在最後，所以我的文字會變成藍色的。

明確性

另一個在套用樣式上會起作用的因素是明確性，它是一個測量一個給定的選取器在文件中對一個元素的明確程度。例如，使用子選取器來尋找

在頁面主體中的 裏去尋找所有的段落，比僅僅使用p來選取要更加地明確。透過類別或屬性值來選取段落又會更加明確，而通過ID來選取元素是最為明確的方式。

```
p#myPara { color: red; }
p.blue { color: blue; }
div.colorText p { color: pink; }

<p>Some text.</p> <!-- Not selected -->
<div class="colorText">
  <p class="blue">Some text.</p> <!-- 藍色-->
  <p>Some more text.</p> <!-- 粉紅色 -->
  <p>Even more text.</p> <!-- 粉紅色 -->
  <p class="blue" id="myPara">The most of all text.</p> <!-- 紅色 -->
</div>
```

在這個例子中，我已經定義了套用到段落上的三條規則，一條會套用到所有的段落上，一條會套用到類型為blue的類別，而一條會套用到id為myPara的段落。請注意，在這種情況下，樣式的順序並不會有影響。在CSS中，明確性勝過最後規則，而且因為類別選取器比類型選取器更加明確，就會套用我的類別規則，即便類型選取器放在最後。

CSS的明確性是一種變化的範圍，某些元素比其他的元素會更加明確。ID是最為明確的，正因為這樣，在上述的例子中它的值就套用了，即便它先被定義，即便這個特定的段落也包含了blue類別的參考。

明確性也可以被用來限定樣式的範圍，所以套用到DOM元素上的規則並不會滲透到應用程式中你不想套用的區域。當實作頁面控制項或其他自我包含的樣式時，它們就會套用到範圍內特定類型的所有元素，而不是應用程式中的所有元素。在先前的範例中，div.colorText規則的套用範圍就限定在容器內的那些段落，這樣可以確保它們被正確地套用。

重要性

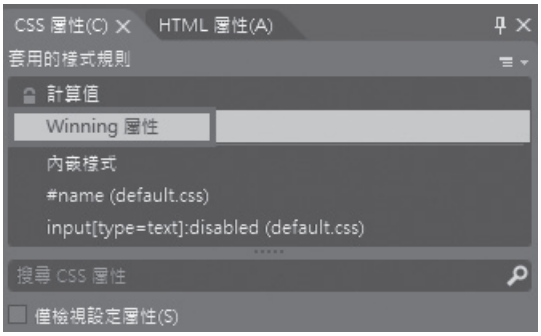
CSS中層疊拼圖的最後一片就是!important規則，它可以用來告訴繪製引擎，應該永遠套用一條特定的規則，不管其順序或明確性為何。

```
p#myPara { color: red; }
p.blue { color: blue; }
div p { color: pink !important; }
```

```
<p>Some text.</p> // Not selected
<div>
  <p class="blue">Some text.</p> // 粉紅色
  <p>Some more text.</p> // 粉紅色
  <p>Even more text.</p> // 粉紅色
  <p class="blue" id="myPara">The most of all text.</p> // 粉紅色
</div>
```

在這個例子中，`!important`修飾將會導致在段落中所有的東西都被樣式化成粉紅色。這是一個很有威力的修飾子，當使用它時要非常機敏。我發現當我在對標記的整體控制稍弱的地方，它是無價的—就像在內容管理系統（CMS）—還有需要經常覆寫樣式。但是，我還是會盡量少用它，即使有的話，在Windows市集應用程式中還是很少有會推薦使用`!important`的地方，大多數的樣式問題都可以透過增加明確性或是改變規則的順序來解決。

在我講完Windows市集應用程式的CSS以結束本附錄之前，我要分享一個定義在應用程式中的樣式的視覺化快速提示。你可能已經猜到了，有時候要搞清楚套用到應用程式的贏家規則可能會令人有些頭痛，所以微軟在Blend提供了一個非常棒的「除錯」及推理樣式的方法。當你在Blend的主要工作區中選擇了一個DOM元素之後，右邊的CSS屬性面版就會產生關於目前套用到這個元素的樣式的一些資訊，包括了「套用樣式規則清單」，如圖B.10所示。



圖B.10 在Blend中的「套用的樣式規則」視窗

這個視窗顯示了目前套用到被選取元素上的所有樣式清單，這是一種有助於快速查看所有已套用樣式來自何處的方法。此外，Blend提供了Winning屬性的視覺化介面，哪一條規則何時被選中，就會在「CSS屬性」面板的下方進行更新，以顯示所有已套用的屬性，以及它們在何處定義，如圖B.11所示。

你會發現這是一個處理CSS的無價工具，所以我鼓勵你去好好的看一下。現在，就讓我們看看Windows市集應用程式中的CSS，以結束本附錄。

圖B.11 在Blend中檢視CSS Winning屬性

Windows市集應用程式中的CSS

如果你正使用HTML及Javascript為Windows 8建立應用程式—我希望你閱讀這本書時可以這樣做—你會發現在Windows SDK中已經為應用程式定義了大量的CSS。你也許會注意到全域/SDK的`css`目錄提供了兩個樣式表，`ui-dark.css`和`ui-light.css`。你的應用程式只會參考這幾個樣式表的其中之一。在撰寫本書時，預設情況下每個內建的範本都會使用`ui-dark.css`樣式表。應用程式的CSS被定義在頁面的`<head>`區塊中：

```
<!-- WinJS 參考-->
<link href="//Microsoft.WinJS.0.6/css/ui-dark.css" rel="stylesheet">
<script src="//Microsoft.WinJS.0.6/js/base.js"></script>
<script src="//Microsoft.WinJS.0.6/js/ui.js"></script>

<!-- MyApp 參考 -->
<link href="/css/default.css" rel="stylesheet">
<script src="/js/default.js"></script>
```

請注意到第一個`<link>`標籤參考了`ui-dark`樣式表。如果你正在使用空白應用程式範本，當你執行應用程式時，會看到與圖B.12相同的畫面。

如果你將`<link>`標籤的參考改為`ui-light.css`檔案，你就會得到一個有白底黑字的範本：

```
<link href="//Microsoft.WinJS.0.6/css/ui-light.css" rel="stylesheet">
```

圖B.12 使用ui-dark SDK樣式表的空白應用程式

當然，因為這只是CSS，你可以自由地選擇這些檔案的其中一個，將其做為起點並進一步地使用在本地default.css檔案中定義的樣式，或者其他套用至應用程式的本地CSS檔案，來客製化你的應用程式。你已經在整本書中看過了許多例子，但我們還是快速地看一下在應用程式中覆寫預設樣式是多麼簡單的一件事。

覆寫預設Windows市集應用程式樣式

雖然這是不言而喻，但無論如何我還是要說：強烈建議永遠不要嘗試修改Windows SDK樣式或是其中的Javascript檔案。這些檔案存放在於本地安裝的Program Files\Microsoft SDKs\Windows\目錄之中，同樣地，當你提交應用程式到市集時，也千萬不要嘗試。永遠不要嘗試修改其中的架構組合，應該以相同的方式對待Windows SDK資源。

你可以建立ui-dark或ui-light的本地實體，然後對本地的版本進行修改。為了讓它起作用，明顯地你需要更新<link>元素的路徑，讓它指向本地的檔案以取代SDK，不過這對任何應用程式來說都是小題大作。透過合適、層疊的規則，完全可以達成你所需要的本地客製化，而不需要在應用程式中複製SDK樣式。

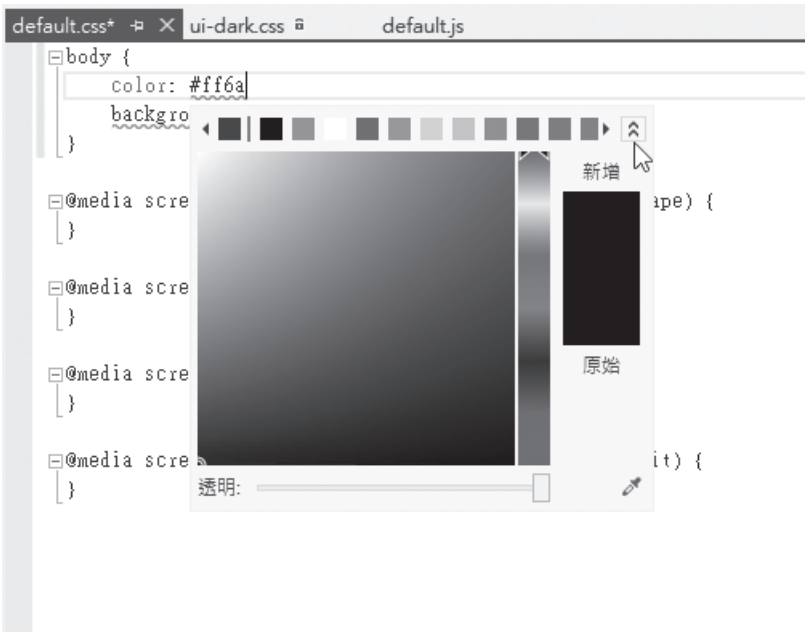
就讓我們使用空白應用程式範本，來建立一個新的應用程式以做為範例。在建立了新的應用程式之後，開啟ui-dark.css檔案，並跳到1443行（譯註：在更新版本的ui-dark.css檔案中，此段CSS規則在1709行），可以找到以下的規則：

```
body {
  color: rgb(255, 255, 255); /* White */
  background-color: rgb(29, 29, 29); /* (mostly) Black */
}
```

現在，將此段CSS規則複製到剪貼簿上，並在本地的CSS資料夾中開啟`default.css`。你會注意到這裏已經定義了一個`body`規則，但它是空的。你可以貼上由`ui-dark.css`複製過來的規則。完成之後，就可以把規則改成你想要的內容。

```
body {
    color: #ff6a00; /* 橘色 */
    background-color: rgb(215, 211, 211); /* 灰色 */
}
```

你可以依照你要的方式來修改這些顏色宣告。其實如果你把滑鼠游標擺在其中一個`rgb`規則內，Visual Studio 2012的內建顏色選取器就會出現（如圖B.13所示）。你就可以從頁面中選取顏色值，適合的顏色（以十六進位制或RGB的形式）就會插入所在的程式行中。



圖B.13 內建於Visual Studio 2012的顏色選取器

在儲存了修改的內容後，執行應用程式時你就會注意到已經成功地修改了基本樣式。我嘗試重新定義的Windows市集應用程式樣式已於圖B.14所示。

圖B.14 使用本地樣式修改預設的ui-dark主題

所以，為什麼我們可以這樣做？當然是因為CSS層疊的關係。如果你重新開啟`default.html`，就會注意到`default.css`是在`ui-dark.css`之後被定義，這意謂著我們在本地為應用程式定義的任何規則—有相同的選取器或明確性的話—都會覆寫`ui-dark.css`所定義的任何規則。這就是為什麼層疊那麼強大的原因。你可以在SDK樣式能夠起作用的地方，隨意地使用它們；而在那些它們無法作用的地方，可以為你的應用程式定義本地樣式，並且修改它們。