

3

單位與值

在本章裡，我們會學到 CSS 裡幾乎每一件事情都會用到的基礎：影響特性所使用的顏色或是距離設定的「單位」(unit)。如果沒有單位，就不可能宣告段落應該是紫色的、或是圖片周圍應該有 10 個像素的空白空間、或是在某個標題應該有特定的大小。瞭解這些概念之後，你就能更快地學習與使用 CSS 的其餘部份。

然而，這是好消息的部份。壞消息是，本章將會包含許多關於瀏覽器的臭蟲與作業系統之間的不協調的警告和討論。然而要記得，CSS 並不能被預設為一個完全精確的版面編排語言，而且，本章裡所討論的許多議題並不是 CSS 的錯，而是一些當你用電腦來處理任何事情時都會遇上的基本問題。所以當你看完本章時，你不但將會對於 CSS 單位的運作有基本的瞭解，可能也會瞭解一些你未曾體會到的基本議題。

然而最重要的是，不管事情看起來多麼糟糕，要繼續努力！你的堅忍不拔會有回報的。

顏色

當然，幾乎每一個網站設計初學者都會想知道的一個問題是「我要如何為我的網頁設定顏色？」在 HTML 裡有兩種選擇：使用數量不多的顏色名稱，例如 red 或 purple，或是採用神秘的 16 進位顏色代碼。這兩種用來描述顏色的方式都可以在 CSS 裡使用，此外還有一些可能更複雜的方式。

具名顏色

假設你樂於採用少數基本的顏色，那麼最簡單的方法就是採用你想要的顏色的名稱，這就是所謂的具名顏色 (named color)。

相對於一些瀏覽器廠商想要你相信的，你是被限制在可用的具名顏色範圍內。舉例來說，珍珠色 (mother-of-pearl) 是不會產生作用的，因為它並不是一個被定義的顏色 (或者說還沒有被定義)。技術上來說，並「沒有」任何被定義的顏色，但是有 16 種顏色是由規格所建議的，並被主要的瀏覽器所認可：

aqua	水藍	gray	灰	navy	深藍	silver	銀
black	黑	green	綠	olive	橄欖綠	teal	藍綠
blue	藍	lime	淡黃綠	purple	紫	white	白
fuchsia	洋紅	maroon	暗紅	red	紅	yellow	黃

如果這些看起來像是古怪的顏色名稱，那是因為它們正是如此，這是我的看法。這些顏色是從哪來的呢？這是最初的 16 種 Windows VGA 顏色，瀏覽器應該要能夠產生接近這 16 種顏色的色彩。它們可能是五顏六色的雜牌軍，但就是我們僅有的。

所以假設我們的第一級標題要使用暗紅色，最好的宣告方式會是：

```
H1 {color: maroon;}
```

簡單、明確、不易忘記。不會有什麼更好的方式了。以下是另外一些例子：

```
H1 {color: gray;}
H2 {color: silver;}
H3 {color: black;}
```

當然，你可能看過（或甚至可能使用過）上面所列出之外的顏色名稱。舉例來說，如果你指定：

```
H1 {color: orange;}
```

你可能會看到所有的 H1 組件都變成橘色，儘管 orange 並沒有列在具名顏色的名單中。這是因為大部分的瀏覽器可以辨識 140 種顏色名稱，其中包括 16 種標準的顏色。然而使用這些例外的顏色會有兩種問題，首先是並不是所有的瀏覽器都認得它們，例如 Opera 就謹守標準的 16 種顏色，至少在 Opera 3.x 系列就是如此。這樣做並不是一種失敗的情形，而是代表一種對於標準的支援的非凡承諾，即使這樣做可能會困擾或激怒網頁設計者。

重現色彩

色彩的重現（reproduction）是一個大問題。我們稍後會看到所有的顏色都可以用一個一致的方法來指派，看起來似乎可以解決讓不同的瀏覽器顯示相同的顏色的問題。但是事實上，情況要複雜得多。首先，人類的感覺是相對的，顯示在同一台顯示器上的同樣顏色，可能會因為照明、亮度、附近的顏色和許多其他因素的改變而改變。你只要改變電腦桌面的背景色，就可以簡易地實驗一下這種效果，你的圖示的顏色似乎會隨著背景色的改變而變化。

為了試著解決這個問題，顯示器通常都會有一個預設的 gamma 值（gamma value），這是一種根據顯示條件修改色彩的係數。gamma 通常是透過作業系統來設定，而一些昂貴的顯示器可能會有自己的 gamma 設定。問題是不同的系統有不同的 gamma 值，因此，如果你設計一個有背景色的網頁，並在同樣的照明環境下同時顯示在 Windows 和 Macintosh 電腦裡，背景色看起來還是會不一樣。網站上所使用的圖片也有同樣的問題，在 Windows 電腦上所設計的圖片對於 Macintosh 使用者來說會偏暗，反之在 Macintosh 電腦上所設計的圖片對於 Windows 使用者來說會偏亮。

列印的時候，這種情況更為嚴重，因為使用的紙張的顏色與質料的不同，甚至列印設備的溫度，都會影響色彩在紙張上重現的結果。

實際上，這又是另一個很好的例子，讓你瞭解要完全控制文件的呈現是幾乎不可能的。在這個例子裡，是因為不一致的作業系統設定和人類感官的難以預測的結合，這是電腦在短期內所無法克服的障礙。

第二個問題是更為基礎的問題：這些名稱並沒有對應的標準顏色值。宣告一個組件的顏色應該成為 `orange`，並不代表不同的瀏覽器（或是不同平台上同樣的瀏覽器）會精確地產生同樣的色彩。如果使用 16 種標準色彩，至少可以期望它們顯示出的顏色濃淡是相同的，因為這 16 種顏色的值已經定義好了。除此之外，所有的情形都不確定，瀏覽器可能會為同樣的顏色名稱採用相似的顏色，也可能不會。其不同之處可能不會為人眼所察覺，或是明顯地令人不悅。

就讓設計者自己去選擇使用具名顏色所必須面對的問題，但是至少因著有了這 16 種顏色，讓我們對於一致性有了適度的期望。

是的，這似乎是最簡單的指定顏色的方法。另外的四種方法比較複雜，但其好處是你可以指定在 8 位元的顏色光譜中的任何顏色，而不是只有那 16 種顏色。這是利用電腦產生色彩的方式來達成的。

RGB 色彩

利用不同等級的紅色、綠色、藍色來組成所需的色彩，就是為何電腦裡的色彩常被稱為 RGB 色彩 (RGB color) 的原因。事實上，如果你拆開一台電腦顯示器，甚至是一台電視，你就會發現一個映像管，裡面有三個投射「槍」(但是要記得如果真的去找這三個投射槍的話，會讓你的保證期失效)。這些槍分別射出 RGB 三種色彩之一的不同明暗等級的光束，投射到螢幕的每一個點上。每一個光束的亮度集合在一點上，就形成你在螢幕上所看到的每一個色彩。每一個點，也就是所謂的像素 (pixel)，將在本章稍後有詳細介紹。

瞭解這種顯示器產生色彩的方式，就不難理解設定顏色的好方法就是讓你直接採用這些顏色等級，從而決定你自己的光束混合方式。很顯然地這方法是有一點複雜，但是所付出的代價是值得的，因為你就不會受限於那些具名顏色。

色彩表示：比例

實際上，有四種方法可以影響 RGB 色彩，第一種方法可能是最容易理解的，因為它採用比例的概念。這裡有個例子：

```
rgb(100%,100%,100%)
```

這樣色彩宣告設定了紅、藍、綠的最大值，組成起來就產生白色，也就是所有色彩的組合。換言之，為了指定黑色 - - 沒有顏色 - - 所有這三種原色都會被設定為 0%。這裡是另外一些例子：

```
H1 {color: rgb(0%,0%,0%);}          /* 黑色 */
H2 {color: rgb(50%,50%,50%);}      /* 中灰 */
H3 {color: rgb(25%,66%,40%);}
```

這種色彩設定的常用語法是：

```
rgb(color)
```

其中 *color* 有兩種方法以指定色彩。第一種方法是使用比例，第二種方法是採用數字，將在稍後討論。

假設你要讓 H1 組件的顏色是紅色與暗紅色之間的色彩。紅色的值是 `rgb(100%,0%,0%)`，暗紅色大約是 `rgb(50%,0%,0%)`，或許可以試試這樣的值作為這兩個色彩的中間值：

```
H1 {color: rgb(75%,0%,0%);}
```

這可以讓指定色彩的紅色成分比暗紅色明亮些，但是比紅色暗一點。如果你想要設計一種淡紅色，那麼你可能需要提高另外兩個值：

```
H1 {color: rgb(75%,50%,50%);}
```

最簡單的視覺化呈現這些比例所對應的色彩的方法，就是建立一個灰色值的表格。由於本書只能提供灰階印刷的效果，這也是最好的方法：

```
P.one {color: rgb(0%,0%,0%);}
P.two {color: rgb(20%,20%,20%);}
P.three {color: rgb(60%,60%,60%);}
P.four {color: rgb(80%,80%,80%);}
P.five {color: rgb(100%,100%,100%);}
```

圖 3-1 顯示不同比例值的顯示效果。

當然，既然我們可以處理灰色的值，所有的 RGB 數值其實也是用一樣的方式。如果任何一個值不同於其他，就會出現一種不同的色彩。舉例來說，如果 `rgb(50%,50%,50%)` 被改為 `rgb(50%,50%,60%)`，結果將是帶點藍色的中灰色。

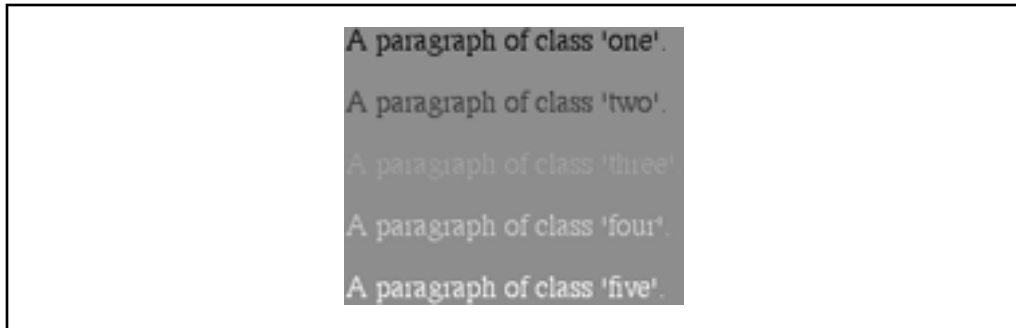


圖 3-1: 灰階值

不同的色彩的組成方式，顯示在表 3-1。

表 3-1: 常見色彩的 RGB 比例

顏色		比例
red	紅	rgb(100%,0%,0%)
orange	橘	rgb(100%,40%,0%)
yellow	黃	rgb(100%,100%,0%)
green	綠	rgb(0%,100%,0%)
blue	藍	rgb(0%,0%,100%)
indigo	青	rgb(20%,0%,100%)
violet	暗紫	rgb(80%,0%,100%)
medium gray	中灰	rgb(50%,50%,50%)
dark gray	暗灰	rgb(20%,20%,20%)
tan	棕褐	rgb(100%,80%,60%)
gold	金	rgb(100%,80%,0%)
purple	紫	rgb(100%,0%,100%)

理論上，使用帶有小數點的值也是可能的。舉例來說，你可能想要一種顏色精確到 25.5% 的紅、40% 的綠、98.6% 的藍，這也是可行的：

```
H2 {color: rgb(25.5%,40%,98.6%);}
```

當然，這還有一個問題。一些瀏覽器可能無法辨識這些帶有小數點的值，甚至有的瀏覽器有可能會忽略了小數點，把前面的數值解譯為 `rgb(255%,40%,986%)`。在這個例子裡，假設瀏覽器的動作正確，超出範圍的值將被改為最接近的合法數值，也就是 100%。因此，忽略了小數點的瀏覽器應該會把宣告的數值看成是 `rgb(100%,40%,100%)`。當然，它是否會這麼做則又是另一個故事。此外，負數的值是不被允許的，所以任何低於 0% 的值都會被改為 0%。舉例來說，以下的值將會變為如圖 3-2 所示的情形：

```
P.one {color: rgb(300%,4200%,110%);}
P.two {color: rgb(0%,-40%,-5000%);}
```

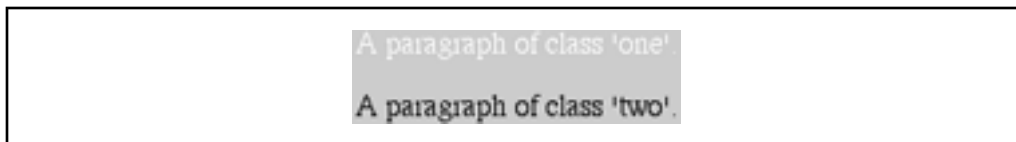


圖 3-2: 超出範圍的值將會被修改

色彩表示：數字

非常接近比例的方法是使用數字來設定色彩。這些數字範圍是從 0 到 255，所以 `rgb(0,0,0)` 代表黑色，`rgb(255,255,255)` 代表白色。你大概可以從其他的來源認出這些數字的範圍：這是八位元的二進位數字。如果你不清楚，只需明白電腦是採用二進位值（開或關）來代表每一件事，包括了數字與顏色，255 是其中一個數字，正好應用在這裡。

不論如何，這幾乎是與採用比例的方式有同樣的精確性：不同的地方僅在於其度量的不同，最大值是 255，而不是 100%。表 3-2 顯示了其所對應的常見色彩的值。

就如你所預料的，所有在 0 到 255 範圍之外的數值都會被省略，就像是採用比例時那樣。當然，在這裡數值會被改為 0 和 255：

```
H1 {color: rgb(0,0,0);}           /* 黑 */
H2 {color: rgb(127,127,127);}    /* 灰 */
H3 {color: rgb(255,255,255);}    /* 白 */
P.one {color: rgb(300,2500,101);} /* 白 */
P.two {color: rgb(-10,-450,-2);} /* 黑 */
```

3-2 常見色彩的 RGB 二進位數值

顏色	比例
red	<code>rgb(255,0,0)</code>
orange	<code>rgb(255,102,0)</code>
yellow	<code>rgb(255,255,0)</code>
green	<code>rgb(0,255,0)</code>
blue	<code>rgb(0,0,255)</code>
indigo	<code>rgb(51,0,255)</code>
violet	<code>rgb(204,0,255)</code>
medium gray	<code>rgb(128,128,128)</code>
dark gray	<code>rgb(51,51,51)</code>
tan	<code>rgb(255,204,153)</code>
gold	<code>rgb(255,204,0)</code>
purple	<code>rgb(255,0,255)</code>

如果你屬意採用比例的方式，就用比例吧。要在比例與數字之間轉換，也是非常簡單的一件事。如果你知道你所要的每一個 RGB 等級的比例，只需要將其乘上 255，就可以得到所需的數值。讓我們假設你有一個顏色的值是 25% 紅、37.5% 綠、60% 藍，將他們各乘上 255，我們就得到 63.75、95.625、153。我們只需要把它的設定改為 `rgb(64,96,153)`，因為當使用數字時，只允許使用整數。比例可以有小數，但這些數字不行。

當然，如果你已經知道比例，並不一定要轉換到數字。這樣的標記方式對於使用像是 Photoshop 這樣的軟體的人來說是很有用的，因為這些軟體也是用這樣的標記方式，或是對於熟悉色彩產生的技術細節的人來說也是很有用的。

不過，這樣的人可能更熟悉採用 16 進位的概念來定義色彩，這正是我們接下來要討論的。

色彩表示：16 進位

如果你過去曾從事過網站設計，並在製作的過程中設定過色彩，那麼這部分對你來說將很輕鬆。你可以採用與網站設計類似的方式，使用 16 進位的概念來設定顏色：

```
H1 {color: #FF0000;} /* 設定 H1 為紅色 */
H2 {color: #903BC0;} /* 設定 H2 為暗紫色 */
H3 {color: #000000;} /* 設定 H3 為黑色 */
H4 {color: #808080;} /* 設定 H4 為中灰色 */
```

如果你不熟悉這種概念，以下有一個快速的入門指引。首先，16 進位 (hexadecimal) 表示以 16 作為基礎的計算，所以基本的單位是以 16 為一組，而不是我們所慣用的 10。在 16 進位裡，有效的數字是從 0 到 9，以及 A 到 F，當你算到 F 時，下一個數字就是 10。因此，一個學習 16 進位的小孩字會用這樣的方法來數數字：

```
00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F,
10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F,
20, 21, 22, 23, ...
```

我瞭解把字母想成數字可能會有一點詭異，但這正是 16 進位的運作方式。數字 A 到 F 實際上只是一種符號，它們可以是任何東西。只是有人決定字母比新發明的符號容易記憶，也不需要再發明新的符號。

這些數字對應到我們慣用的 10 進位數字的方式是非常直接的，05 等於 5，0C 等於 12，0F 就是 15，10 就是 16，1F 等於 31，20 等於 32，依此類推。如以下所示：

```
16 進位    01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10,
10 進位    01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16,

16 進位    11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20,
10 進位    17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,

16 進位    21, 22, 23, ...
10 進位    33, 34, 35, ...
```

電腦採用 16 進位的概念已經有一段時間了，一般的程式設計師若不是受過相關訓練，就是有實際的相關經驗。不論如何，大部分的程式設計師都很習慣 16 進位，甚至以此來思考，所以它就成了 CSS 規格的一部份。為什麼？因為規格是由程式設計師負責撰寫與編輯的，所以將其放進色彩的架構中也很合理。

所以，藉由將三個 16 進位數值串接起來，就能設定一種顏色。一般的描述方式是這樣的：

```
#RRGGBB
```

這樣看來，16 進位的方式也很像我們之前所討論的方式 - - 採用 0 到 255 的數字。事實上，10 進位的 255 就等於 16 進位裡的 FF，這就可以解釋這種方式運作的原理了。這與上一種方式實際上是一樣的：只是採用一個不同的數字系統。如果你必須從兩者中擇一，儘管使用你比較習慣的方式。

與使用三個 0 到 255 之間的數字來定義色彩一樣，你可以用三個 16 進位數字來指定色彩。如果你有一個可以轉換 10 進位與 16 進位的計算機，那麼這樣的轉變應該是很簡單的。如果不是，這可能會有一點複雜（當然，你也可以完全不採用這種方法）。表 3-3 是一些顏色的 16 進位數值。不管你信或不信，這是可以少打一些字的色彩設定方式。

表 3-3: 常見色彩的 16 進位數值

顏色	16 進位數值
red	#FF0000
orange	#FF6600
yellow	#FFFF00
green	#00FF00
blue	#0000FF
indigo	#3300FF
violet	#CC00FF
medium gray	#808080
dark gray	#333333
tan	#FFCC99
gold	#FFCC00
purple	#FF00FF

色彩表示：簡短的 16 進位色彩

終於要談最後一個方法了。讓我們先看一個例子，然後再加以解釋：

```
H1 {color: #000;} /* 設定 H1 為黑色 */
H2 {color: #666;} /* 設定 H2 為暗灰色 */
H3 {color: #FFF;} /* 設定 H3 為白色 */
```

就如你所見，每個顏色的值只有三個數字。而既然從 00 到 255 的 16 進位數字各需要兩位數，但我們總共只有三個數字，這是如何運作的呢？

答案在於瀏覽器會取出每個數字，然後再加以複製。因此，#F00 就等於 #FF0000 - - 就這麼簡單。除此之外，這方法就和之前所討論的 #RRGGBB 方法一樣，只是更為簡短些。#6FA 就和 #66FFAA 一樣，而 #FFF 就等於 #FFFFFF，也等於 white。這種方法有時叫做 16 進位速記法 (shorthand hex notation)。

16 進位法有一個不同於數字方法所必須要注意的事，就是 16 進位系統裡並沒有定義修改數值的方法。如果你輸入一個無效的值，瀏覽器的回應是無法預測的。一個完善的瀏覽器會略去超出範圍的值，但是你不能夠依賴這種情況。舉例來說，Netscape Navigator 4.x 不會忽視或省略無效的色彩值，而是會做出某種轉換，以一個無法預期的色彩來取代。

整合運用色彩

表 3-4 顯示對於我們所討論的顏色的綜覽。斜體字標示的色彩名稱是可以合法地作為色彩名稱來使用，其他非斜體字可能不會被瀏覽器所承認，因此應該用 RGB 或 16 進位數值來定義 (以策安全)。此外，有一些 16 進位速記值並沒有顯示出來，因為無法簡化其全長 6 位數的值。舉例來說，#880 可以延伸為 #888800，而不是 #808000 (也就是橄欖色 olive)。因此，#808000 並沒有簡短版本，所以表格中的對應欄位是空白的。

表 3-4: 色彩對應表

顏色	比例	數值	16 進位	16 進位速記法
red	rgb(100%,0%,0%)	rgb(255,0,0)	#FF0000	#F00
orange	rgb(100%,40%,0%)	rgb(255,102,0)	#FF6600	#F60
yellow	rgb(100%,100%,0%)	rgb(255,255,0)	#FFFF00	#FF0
green	rgb(0%,100%,0%)	rgb(0,255,0)	#00FF00	#0F0
blue	rgb(0%,0%,100%)	rgb(0,0,255)	#0000FF	#00F
indigo	rgb(20%,0%,100%)	rgb(51,0,255)	#3300FF	#30F
violet	rgb(80%,0%,100%)	rgb(204,0,255)	#CC00FF	#C0F
aqua	rgb(0%,100%,100%)	rgb(0,255,255)	#00FFFF	#0FF
black	rgb(0%,0%,0%)	rgb(0,0,0)	#000000	#000
fuchsia	rgb(100%,0%,100%)	rgb(255,0,255)	#FF00FF	#F0F
gray	rgb(50%,50%,50%)	rgb(128,128,128)	#808080	
lime	rgb(0%,100%,0%)	rgb(0,255,0)	#00FF00	#0F0
maroon	rgb(50%,0%,0%)	rgb(128,0,0)	#800000	
navy	rgb(0%,0%,50%)	rgb(0,0,128)	#000080	
olive	rgb(50%,50%,0%)	rgb(128,128,0)	#808000	
purple	rgb(50%,0%,50%)	rgb(128,0,128)	#800080	
silver	rgb(75%,75%,75%)	rgb(192,192,192)	#C0C0C0	
teal	rgb(0%,50%,50%)	rgb(0,128,128)	#008080	
white	rgb(100%,100%,100%)	rgb(255,255,255)	#FFFFFF	#FFF
darkgray	rgb(20%,20%,20%)	rgb(51,51,51)	#333333	#333
tan	rgb(100%,80%,60%)	rgb(255,204,153)	#FFCC99	#FC9

安全的網頁色彩

你可能會回想到在先前的討論裡曾提過，在不同的作業系統、瀏覽器中，色彩並不總是能夠一致。有一個方法可以部分地解決這個問題，然而這是代表必須限制你使用的色彩數目。有 216 種顏色被稱為「網站安全色」，意思是它們在所有的電腦與瀏覽器上，應該看起來都是一樣，而不會有任何遞色或色偏的情形。要注意的是雖然我說的是「應該」，但這並無法保證。然而通常都會運作正常。

網站安全色是那些由 RGB 值為 20%、51，以及其所對應的 16 進位值 33，還有其倍數所組成的色彩。此外，0% 或 0 也是安全的數值。所以如果你採用 RGB 比例，就採用 0% 或可以用 20 除盡的數值，例如 `rgb(40%,100%,80%)`、`rgb(60%,0%,0%)`。如果你採用 0 到 255 的 RGB 數值，就應該用 0 或可以被 51 除盡的數字，例如 `rgb(0,204,153)`、`rgb(255,0,102)`。

至於 16 進位表示法，恰當的數值是 00、33、66、99、CC、FF，任何採用這些數值所組成的代碼都是網站安全色，例如 #669933、#00CC66、#FF00FF。這表示 16 進位速記值得安全色是使用 0、3、6、9、C 與 F，因此 #693、#0C6 與 #F0F 都是網站安全色。

天知道怎麼會有這麼多種定義色彩的方法，我敢打賭你以後看到彩虹時心裡一定有許多不同的想法。現在，可以移動到下一個度量的單位了。

長度單位

許多 CSS 特性 - - 例如邊界 - - 都依賴長度的度量單位，用來正確地顯示不同的網頁組件。也許你不會感到驚訝，因為在 CSS 裡也同樣有許多不同的方式來設定長度。

所有長度單位的值都可以使用正數或負數，再加上一個單位記號，但是有些特性只接受正數。長度單位的值可以是整數，也可以採用小數，例如 10.5、4.561。所有的長度單位後面都有兩個字母的縮寫，代表長度所指定的實際單位，例如 in (inch, 英吋) 或 pt (point, 點)。唯一的例外是長度為 0 (零) 的情形，並不需要加上單位名稱。

這些長度單位可分為兩種類型：絕對單位 (absolute unit) 與相對單位 (relative unit)。

絕對長度單位

這是最容易瞭解的長度單位，儘管很少使用在網站設計裡。絕對單位共有五種：

in (inch 英吋)

就是美國人所用的度量單位。儘管全世界幾乎都在使用公制系統，這單位之所以出現在規格裡的原因，是因為美國人對於網際網路的高度興趣。但這是社會政治學的議題，在此不多討論。

cm (centimeter 公分)

公分是世界上許多地方的尺上面所使用的刻度，一英吋等於 2.54 公分，一分等於 0.394 英吋。

mm (millimeter 公厘)

就是公分的十分之一。對於美國人來說，10 公厘等於一分，25.4 公厘等於一英吋，一公厘等於 0.0394 英吋。

pt (point 點)

現在開始有趣了。點是排版印刷上的標準度量單位，已經為打字機與印表機使用了數十年之久，文書處理軟體也採用了好幾年。從定義上來看，一英吋有 72 點，所以點實際上是在公制系統的使用之前就被定義的。一個 12 點的大寫字母應該是 1/6 英吋高。例如 `P {font-size: 18pt;}` 就等於 `P {font-size: 0.25in;}`。

pc (pica)

這是另一個印刷業的專業術語。一個 pica 等於 12 點，也就是說 6 pica 等於一英吋。因此，一個 1 pica 的大寫字母應該等於 1/6 英吋高。舉例來說，`P {font-size: 1.5pc;}` 會把文字設定為與前一個例子相同的大小 ($1.5pc = 18pt = 0.25in$)。

當然，只有在你的瀏覽器瞭解顯示你的頁面的顯示器、用來列印複本的印表機或任何你可能使用的代理程式的所有細節，這些單位才有實際的用處。

舉例來說，會影響瀏覽器的顯示結果的因素，可能是顯示器的尺寸大小與所設定的解析度。當然，身為一個網頁設計者，你並沒有辦法做些什麼事，但至少所設定的度量單位將是一致的，也就是說 `1.0in` 將會是 `0.5in` 的兩倍寬，如圖 3-3 所示。

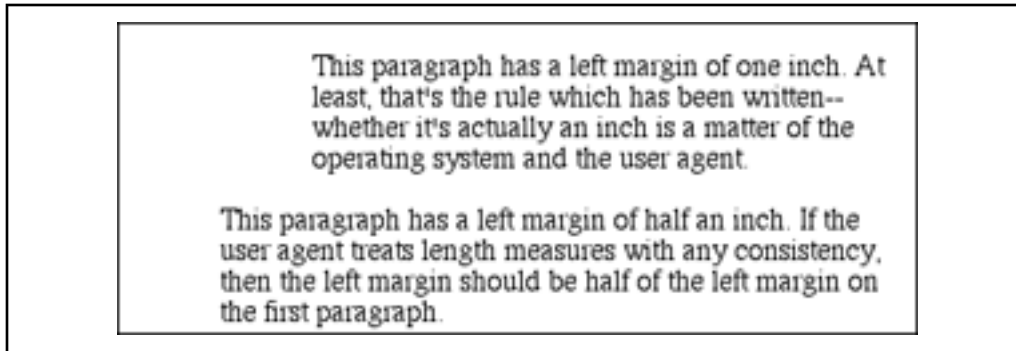


圖 3-3: 1.0 英吋寬的邊界（上）和 0.5 英吋寬的邊界（下）之比較

絕對長度的運用

身為一個 Windows 的使用者，你可能可以設定你的顯示器驅動程式採用實際的度量單位。試著點選【開始】按鈕，選擇【設定】，然後點選【控制台】，再點選【顯示器】。在顯示器的對話框裡，應該還有一個【設定】標籤，點選它，然後再點選【進階】按鈕，就會出現另一個對話框，其內容可能會因為不同的電腦而有所不同。你應該可以看到有一個地方標示著【字型大小】，請選擇【其他】，然後拿把尺出來比對，並拖曳畫面上的尺規直到兩者一致。然後只要按幾下【確定】，就完成設定了。

如果你是 Macintosh 使用者，並沒有地方可以讓你設定這些資訊，但是沒有關係，因為 Mac 作業系統早已預設了螢幕上的像素與絕對的度量單位之間的關係，由顯示器上的 72 個像素組成一英吋。這樣的假設其實是不對的，但是已經內建在作業系統裡，因此至少到目前為止是無可避免的（如果幸運的話，未來的瀏覽器可以包括一個偏好設定，用來定義你自己想要的每英吋有多少點的值，就可以避免任何作業系統所產生的限制）。

因此，在許多 Macintosh 瀏覽器上，例如 Netscape 4.x、Internet Explorer 3.x 與 4.x，任何點的值都會與像素的值相等：24pt 的文字與 24 像素的文字一樣高，而 8pt 的文字則和 8 像素一樣高。很不幸的，這樣的字體太小而不易閱讀，儘管它們在 Windows 裡的瀏覽器上看起來很清楚。圖 3-4 顯示了其中的問題。

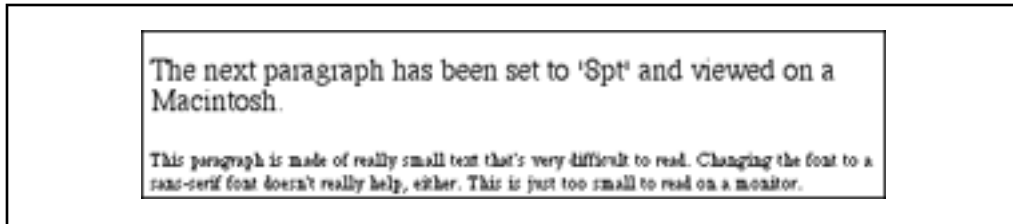


圖 3-4: Macintosh 顯示的文字較小，不易閱讀

這是一個很好的例子，可以說明為何在設計網站時應該避免點的使用。對於瀏覽器的顯示來說，`em`、比例、甚至像素都是比較適合的選擇。

儘管有這些問題，讓我們假設你的電腦已經瞭解它的顯示系統，並且可以精確地再生真實世界裡的度量方式。這時，你可以用 `P {margin-top: 0.5in;}` 來確認讓每一個段落擁有半英吋高的上邊界。不論週遭的狀況是如何，每個段落都會有一個半英吋高的上邊界，不管其字型大小等任何變數。

絕對單位在定義紙本文件的樣規時顯得有用多了，因為這時用英吋、點、`pica` 作為度量單位是比較普遍的事。就如我們所見，嘗試在網站設計裡使用絕對單位可會有許多危險，所以讓我們轉向其他有用的度量單位吧。

相對長度單位

這與愛因斯坦毫無關係，之所以稱為相對單位，是因為它們是藉由其他事物來決定其長度。它們實際的（或絕對的）距離會因著它們無法控制的因素而產生變化，例如螢幕的解析度、可視範圍的寬度、使用者的偏好設定，還有許多其他的事項。此外，對於一些相對單位來說，它們的長短總是與使用它們的組件相關，因此會隨著組件而有所不同。

有三種相對長度單位：`em`、`ex`、`px`。前面兩個分別代表 `em-height` 與 `x-height`，它們是很常見的印刷度量單位。然而在 CSS 裡，它們的意義可大不相同。最後一個長度單位是 `px`，代表像素（`pixel`），一個像素是電腦顯示器上的一個點，如果你靠得夠近的話。這個值之所以是相對的，是因為它會依賴顯示設備的解析度，稍後我們會仔細討論。

em 與 ex

首先，讓我們來討論 em 與 ex。在 CSS 裡，一個 em 就是某個字型的 font-size 值，所以如果該組件的 font-size 是 14 點，那麼對於該組件來說，一個 em 就等於 14 點。換句話說，不論你所設定的字型大小是多少，都等於該 em 的值。對於一個文字是 18 點的段落來說，em 的長度就是 18 點。

很明顯的，這個值會隨著組件的不同而改變。例如一個字型是 24 像素大小的 H1 組件、一個字型大小是 18 像素的 H2 組件、以及一個字型為 12 像素段落，會有如圖 3-5 的結果：

```
H1 {font-size: 24px;}
H2 {font-size: 18px;}
P {font-size: 12px;}
H1, H2, P {margin-left: 1em;}

<H1>Left margin = 24 pixels</H1>
<H2>Left margin = 18 pixels</H1>
<P>Left margin = 12 pixels</P>
```

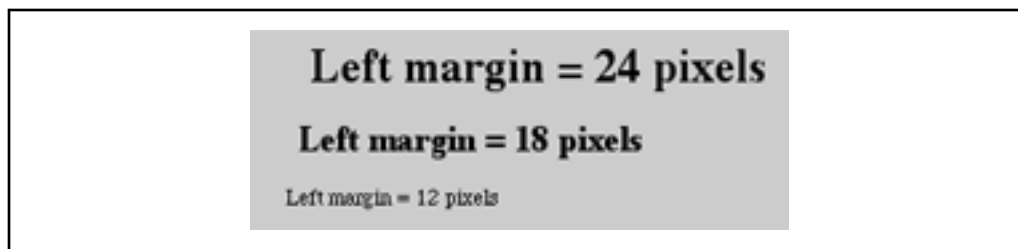


圖 3-5: em 會根據字型大小而改變

這個規則的唯一例外是字型大小的設定，這時 em 的值是相對於母組件（如圖 3-6 所示）：

```
SMALL {font-size: 0.8em;}

<P>Although this text is the default size for the user agent, the
<SMALL>small-text element</SMALL> within the paragraph has a font
size which is 80% the rest of the text in the paragraph.</P>
```

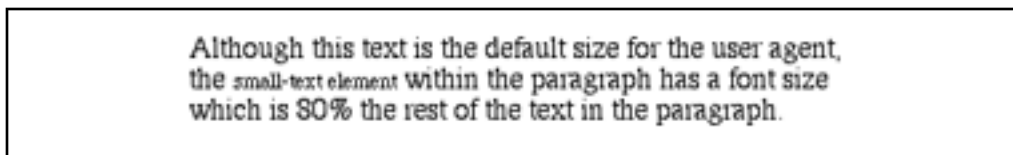


圖 3-6: 字型大小的 em 值是相對於母組件

另一方面，ex 代表的則是所使用字型的小寫字母 x 的高度。因此如果你有兩段文字，其字型大小為 24 點，但是每一個段落是使用不同的字型，那麼每個段落的 ex 值可能會有不同。這是因為不同的字型裡的 x 有不同的高度，如圖 3-7 所示。



圖 3-7: 不同的字型有不同的 ex 高度

雖然範例裡使用的都是 24 點的文字，每一個的 em 值都是 24 點，但是 ex 則會有不同。

em 與 ex 的實際運用

這聽起來已經很複雜了，但實際上還會更困難，因為這只是理論上會發生的情況，實際上，許多瀏覽器會把 em 值的一半拿來當作 ex 值。為什麼呢？很顯然地大部分的字型並沒有內建其 x 的高度值，而這是很難去計算的一件事。既然大部分的字型的小寫字母是大寫字母的一半高度，所以很容易就會假設 1ex 等於 0.5em。當然，我們希望這情形能隨著時間而改變，瀏覽器能開始使用真正的 ex 值。

像素

現在輪到像素了，一個更令人困惑的狀況。表面上看來，像素應該是最直接的，但是如果你很近地看顯示器，你可以看出它是由許多小方塊所組成的，每一個小方塊就是一個像素。然而，一英吋有幾個像素呢？如果顯示器的解析度是 1024 像素寬、768 像素高，而顯示器的螢幕是 14.22 英吋寬、10.67 英吋高，並且顯示區域填滿了整個螢幕，那麼每一個像素的長與寬應各是 1/72 英吋。然而就如你所想的，這是非常稀少的情況，尤其你現在很難找到這樣大小的顯示器。因此，現在大部分的顯示器裡，每英吋像素 (pixels per inch, ppi) 要高於 72，有時甚至高達 120ppi。

更令人困惑的是，CSS 規格建議一個「參考像素」為 90ppi - - 這是一個很少有作業系統會使用的數值（如果有的話）。當規格假設 90px 應該等於 1in 時，卻沒有人真的這樣做，所以這無濟於事。一般來說，如果你假設 `font-size: 18px` 這樣的規則，然後你只是讓瀏覽器去決定一個像素的實際大小。大部分的瀏覽器都會使用你的顯示器所設定的實際像素 - - 畢竟它們早就在那裡等著了。但是其他的顯示設備，例如印表機，所發生的狀況就有點不確定。



在 CSS1 的早期導入階段有一個關於像素度量的問題範例。在 Internet Explorer 3.x 中，當列印文件時，IE3 會假設 18px 等於 18 個點，可能是 18/300、6/100、1 英吋，甚至是 0.06in 也可以。這是非常小的文字！

但另一方面，像素的度量單位非常適合用來表現圖片的大小，因為圖片本身就是以像素來指定長寬。實際上，你唯一應該用其他度量單位來表示圖片大小的狀況，就是如果你要讓圖片隨著文字而縮放。這是一個聰明但偶爾才會派上用場的方法，並且如果是採用向量圖片而非像素式圖片，這樣做才合理。（隨著可延展式向量圖片 (SVG) 的採用，這將有可能實現。）

要怎麼做？

瞭解以上所牽涉的問題之後，最好的度量方法可能是相對式的做法，尤其是 `em`，或是適合 `px` 的狀況。因為目前 `ex` 基本上是由 `em` 而來的虛構數值，所以並不是非常有用。如果瀏覽器可以支援真實的小寫字母高度，那麼 `ex` 才會有一席之地。

因此，我們可以結束關於長度度量單位的討論。本章的其餘部分就簡單多了，不會這麼悲觀。

比例值

與長度單位相比，比例單位就簡單多了，它們就是你所想像的 - - 只是正數或負數的值，接著一個比例的符號（%）。舉例來說：

```
H1 {font-size: 18pt;}
H1.tall {line-height: 150%;}
```

這個例子把所有有 `tall` 類別的 `H1` 組件的 `line-height`，設定為比正常高出一半，如圖 3-8 所示。

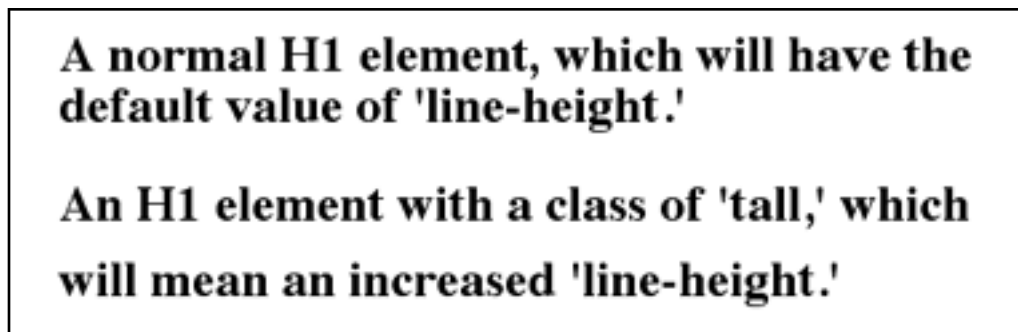


圖 3-8: 100% 行高 (上) 與 150% 行高 (下) 的比較

比例值總是相對於另一個值來計算，通常是一個長度單位。在這個例子裡，`line-height` 實際上是 27 點（18pt 的 1.5 倍）。這與把 `line-height` 設定為 `1.5em` 是同樣的，但這兩種方法並沒有哪一個是比較建議使用的。

一般來說，比例可以是正數或負數。然而有些特性雖然接受比例值，但不允許使用負數（任何形式的，包括比例）；這將會在以後的內容中討論到特性時提到。

URLs

如果你曾經設計過網頁，那麼就應該很熟悉 URL（Uniform Resource Locator，統一資源定位碼）。URL 並不常在樣規中使用，但是如果你真的需要，例如在 `@import` 敘述裡用來匯入一個外部的樣規，那麼一般的格式會是：

```
url(http://server/pathname)
```

這是一個定義絕對 URL（absolute URL）的例子。「絕對」的意思是不論網頁所處的位置為何，這個 URL 都能起作用，因為它所定義的是一個在網際空間裡的絕對位置。假設有一個伺服器叫做 `www.waffles.org`，在這台伺服器上，有一個目錄叫做 `pix`，在這目錄裡有一個圖片叫做 `waffle22.gif`。在這個例子裡，圖片的絕對 URL 會是 `http://www.waffles.org/pix/waffle22.gif`。這個 URL 是有效的，不論它是在哪裡出現，也不論它是出現在 `www.waffles.org` 或 `web.pancakes.com`。

另一種 URL 是相對 URL（relative URL），因為這種 URL 所指的位址是相對於使用它的文件而定。如果你使用的是相對路徑，例如與你的網頁同一個目錄的一個檔案，一般的格式會是：

```
url(pathname)
```

如果圖片是在與包含該 URL 的網頁的同一台伺服器上，這個 URL 才會有效。舉例來說，我們假設有一個網頁位在 `http://www.waffles.org/syrup.htm`，我們想要圖片 `waffle22.gif` 顯示在該網頁上，它的相對 URL 應該是 `pix/waffle22.gif`。

這可以運作的原因，是因為瀏覽器知道它應該在該網頁的同一路徑加上相對 URL。在這個例子裡，伺服器名稱 `http://www.waffles.org` 再加上 `pix/waffle22.gif`，就等於 `http://www.waffles.org/pix/waffle22.gif`。

以下是另外兩個例子：

```
@import url(http://css1.style.org/example.css);
```

```
BODY {background-image: url(hatch.gif);}
```

不論你使用絕對或相對 URL，其實都無關緊要，只要是以有效的位址來定義它們。你可以使用對於你和你的專案來說比較簡單的方法。

此外，你需要注意的是相對 URL 是相對於樣規，而不是 HTML 文件。舉例來說，你可能有外部樣規匯入另一個樣規，如果你使用相對 URL 匯入第二個樣規，它就必須是相對於第一個樣規。例如一個位在 `http://www.waffles.org/toppings/tips.html` 的 HTML 文件，有一個 LINK 組件連到樣規 `http://www.waffles.org/styles/basic.css`：

```
<LINK REL="stylesheet" TYPE="text/css"
      HREF="http://www.waffles.org/styles/basic.css">
```

在檔案 `basic.css` 裡，有一個 `@import` 敘述指向另一個樣規：

```
@import url(special/toppings.css);
```

這個 `@import` 將會載入 `http://www.waffles.org/styles/special/toppings.css` 檔案，而不是 `http://www.waffles.org/toppings/special/toppings.css`。如果你有一個樣規是放在後者的位置裡，那麼在 `basic.css` 裡的 `@import` 應該是：

```
@import url(http://www.waffles.org/toppings/special/toppings.css);
```

換句話說，就是採用絕對 URL。

這可能是一個適用於任何例子的好辦法。例如 Navigator 4 會把相對 URL 根據 HTML 文件來解譯，而不是樣規。這樣是不對的，但這卻是 Navigator 4 的方式。因此，把所有的 URL 用絕對方式來表示是最簡單的辦法，因為 Navigator 至少可以正確地處理絕對 URL。

要注意的是，在 `url` 和括號之間不能有空格，因此：

```
BODY {background: url(http://www.pix.web/picture1.jpg);} /* 正確 */
BODY {background: url (images/picture2.jpg);} /* 錯誤 */
```

如果加上了空格，整條規則就變為無效，會被瀏覽器所忽略。

CSS2 的單位

除了以上所述之外，CSS2 也新增了一些單位，幾乎都是跟聽覺樣規（讓提供語音功能的瀏覽器使用）相關，我們在此做簡單的說明：

角度值（angle value）

這是用來定義產生聲音時的來源位置。有三種角度單位：`deg`（degree，度）、`grad`（grad，等級）、`rad`（radian，弧度）。舉例來說，直角可以用 `90deg`、`100grad` 或 `1.57rad` 來表示。這些值都會被解譯成為 0 到 360 之間的數值，即使是負數也是可以容許的。`-90deg` 和 `270deg` 的意義是一樣的。

時間值（time value）

這是用來指定聲音組件之間的延遲時間，可以用 `ms`（millisecond，毫秒）或 `s`（second，秒）來表示。因此，`100ms` 和 `0.1s` 是相同的。時間值不可以是負數。

頻率值（frequency value）

用來表示瀏覽器所產生的聲音的頻率。頻率值可以用 `Hz`（hertz，赫茲）或 `mHz`（megahertz，兆赫）來表示，不能使用負數。值的單位沒有大小寫之分，所以 `10mHz` 和 `10mhz` 是一樣的。

除了這些值之外，還有一個改頭換面的老朋友。URI 是統一資源辨識碼（Uniform Resource Identifier）的意思，其實就是 URL 的另一個名稱。不同之處只在於文字上的不同，但是現在有許多設計者開始採用 URI 來指涉網路位址，而不用 URL。CSS 規格仍然要求 URI 採用 `url(...)` 這樣的形式，所以實際上實在很難理解 CSS2 如何使用 URI 而不使用 URL。

總結

單位與值涵蓋的範圍很廣，從長度單位、顏色單位，到檔案（例如圖片）的位置。大部分的時候瀏覽器都可以正確地處理單位，然而你還是會遇到些小問題。舉例來說，錯誤地解譯相對 URL 就令許多設計者抓狂，導致對於絕對 URL 的過度依賴。顏色是另一個瀏覽器在大部分的時候可以正確處理的例子，除了極少數例外。然而長度單位的變換無常並不是一個錯誤，反而是許多設計者有興趣解決的問題。

這些單位都有其優點和缺點，根據它們被使用的環境而定。我們已經檢視其中部分狀況，其餘的部分會在本書後續加以討論，就先從 CSS 特性如何改變文字的顯示方式開始吧。