程式人 月刊 雜誌

Programmer

捐發票愛心條碼

讀書做善事、寫書做公益 – 歡迎程式人認養專欄或捐出您的網誌

參考價： NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體

羅慧夫顱顏基金會 彰化銀行 (009) 帳號： 5234-01-41778-800

程式人雜誌

2014 年 9 月號

本期焦點：從 CSS, LESS/SASS 到 Bootstrap

程式人雜誌

前言

編輯小語

在本期的「程式人雜誌」中，聚焦的主題是「網頁排版語言」。

雖然網頁的排版並不是「程式」，但在網站技術發達的今天，程式人不懂排版語言恐怕也是不行的！

CSS 是個標準的網頁排版語言，而 LESS 與 SASS 則是從 CSS 延伸而來，功能較強大的排版語言，至於 Bootstrap 則是由 Twitter 所示出的一組網頁排版框架，Bootstrap 包含一組用 LESS 所撰寫的排版語言，以及 jQuery 所建構的互動機制，可以讓您輕易的做出在電腦和手機上都能美觀呈現的網頁。

有了 Bootstrap，程式人就不怕做出來的網頁太醜而難登大雅之堂了！

----（程式人雜誌編輯 - 陳鍾誠）

授權聲明

本期焦點

CSS 網頁排版語言

CSS 是用來與 HTML 搭配，專門指定排版呈現格式的簡易語言，在本文中我們將介紹簡易的 CSS 語言與用法。

舉例而言，假如我們想讓表格的標頭欄以黑底白字的方式顯示，那麼就得靠 CSS 來幫忙了。

CSS 是 Cascading Style Sheets 的簡稱，中文可翻譯為「串接樣式表」，CSS 主要是用來設定 HTML 的顯示版面，包含「字體、大小、顏色、邊框、背景」等等，一個好的 CSS 可以讓您的網頁變得很美觀，而且不需要大幅修改網頁的內容，只要加入一個 CSS 引用即可。

當套用不同的 CSS 時，網頁的風格就可以產生完全不同的感覺，這讓你的網頁可以輕易的更換成不同的版型。事實上、當您在 blogspot 或 wordpress 網誌平台套用不同版型的時候，只不過是更換了 CSS 樣版而已。

在 HTML 當中，CSS 有兩種寫法，一種是內嵌式的寫法，另一種是外連式的寫法。

為了說明 CSS 的語法，先讓我們回顧一下，一個簡單沒有格式化的 HTML 表格，其寫法如下：

```
<html>
<head><meta charset='utf-8'></head>
<body>
<table>
<tr><th></th><th>欄 1</th><th>欄 2</th></tr>
<tr><th>列 1</th><td>1,1</td><td>1,2</td></tr>
<tr><th>列 2</th><td>2,1</td><td>2,2</td></tr>
</table>
</body>
</html>
```

檢視檔案：https://dl.dropboxusercontent.com/u/101584453/web/wp/code/table.htm

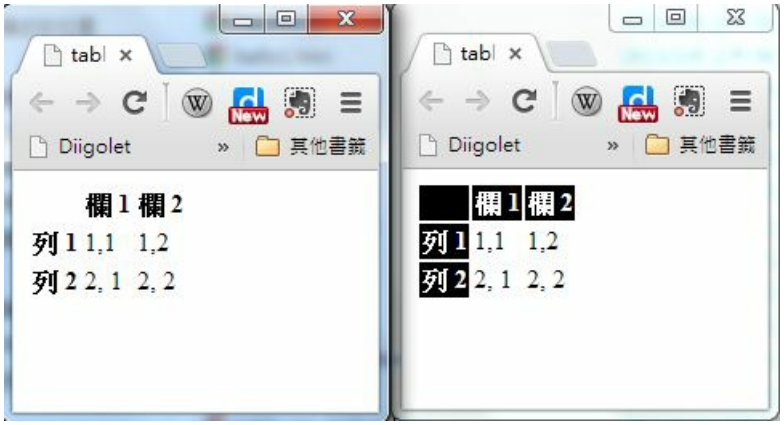
如果我們將上述表格的 th 部分，都加上 style="background-color:black; color:white;" 這樣的 CSS 語法，那麼我們就會得到一個黑底白字的表格。

```
<html>
<head><meta charset='utf-8'></head>
<body>
<table>
<tr>
<th style="background-color:black; color:white;"></th>
<th style="background-color:black; color:white;">欄 1</th>
<th style="background-color:black; color:white;">欄 2</th>
```

```
</tr>
<tr>
<th style="background-color:black; color:white;">列 1</th>
<td>1,1</td>
<td>1,2</td>
</tr>
<tr>
<th style="background-color:black; color:white;">列 2</th>
<td>2, 1</td>
<td>2, 2</td>
</tr>
</table>
</body>
</html>
```

檢視檔案：https://dl.dropboxusercontent.com/u/101584453/web/wp/code/table_css_embed.htm

我們使用 CSS 排版前與排版後的結果可以對照如下：



表單的顯示結果

您可以看到在這種寫法當中，我們幾乎都一直在 th 重複撰寫 style="background-color:black; color:white;" 這一行文字，總共重複了 5 次，這顯然是很浪費時間的行為。

如果我們可以改用統一的方式，寫出「對於所有 th 而言，我都要用 background-color:black; color:white; 這樣的方式顯示」那不就簡單多了嗎？

是的、CSS 確實可以讓您這樣做，讓我們將上述範例修改成統一設定格式的版本，如下所示。

```
<html>
<head>
<meta charset='utf-8'>
<style>
```

```
th { background-color:black; color:white; }
</style>
</head>
<body>
<table>
<tr><th></th><th>欄 1</th><th>欄 2</th></tr>
<tr><th>列 1</th><td>1,1</td><td>1,2</td></tr>
<tr><th>列 2</th><td>2,1</td><td>2,2</td></tr>
</table>
</body>
</html>
```

上述統一將 CSS 寫在 <head><style>...</style></head> 裏的這種寫法，比起內嵌式的顯然簡短多了，這種寫法正是 CSS 的精華之所在。

但是、如果我們想要對整個網站的所有檔案，都套用同一種格式的話，那麼在每個 HTML 的頭部都要嵌入一整套 CSS 語法，那就不太方便了，萬一我們想要改變版面格式，不就每個 HTML 檔案都要更改，這顯然還不夠好用。

要解決這個問題，可以將表頭的 CSS 獨立到一個檔案中，然後再用連結的方式引用，這樣只要修改那個 CSS 檔案，整個網站的風格就跟著改變，所以上述檔案就可以改寫如下。

檔案：table_css.htm

```
<!DOCTYPE html>
<html>
<head>
<meta charset='utf-8'>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>
<table>
<tr><th></th><th>欄 1</th><th>欄 2</th></tr>
<tr><th>列 1</th><td>1,1</td><td>1,2</td></tr>
<tr><th>列 2</th><td>2,1</td><td>2,2</td></tr>
</table>
</body>
</html>
```

檔案：mystyle.css

```
th { background-color:black; color:white; }
```

透過這種方式，我們就可以為網頁設計統一的風格，也比較容易更換網頁的排版風格了，這就是 CSS 語法的用途了。

當然、上述的網頁還不夠好看，如果我們將 mystyle.css 修改如下，您就會發現表格變得好看多了。

```
table { border-collapse: collapse; border: 1px solid #373737; }
th { background-color:black; color:white; padding:10px; margin:10px;
}
td { padding:10px; margin:10px; }
```

較優美的表單顯示結果

路徑, tag, id 與 class

即使有了可以統一套用的方式，有時我們還是會感覺到不方便，舉例而言，如果我們想要對某個表格套上 A 格式，然後在對另一個表格套上 B 格式，這時採用上述方法就不太夠用了。

為了解決這種問題，HTML 發展出了 id 與 class 這兩個屬性，我們可以根據 id 或 class 的名稱分別套用不同的 CSS 樣式。

舉例而言，以下網頁中有兩個 div 框，我們可以分別為這兩個框套用不同的格式，其中 classA 前面加了點符號，成為「.classA」，代表要比對的是 class 屬性，如果想要比對的是 id 屬性，那麼就可以加上井字符號 # (像是範例中 #topbar 的情況)。

```
<html>
<head>
<meta charset='utf-8'>
<style>
#topbar { background-color:gray; color:blue; padding:10px; margin:10p
x; }
.classA { background-color:black; color:white; padding:10px; margin:1
0px; }
.classB { background-color:blue; color:yellow; padding:10px; margin:1
```

```
0px; }
</style>
</head>
<body>

<div id="topbar">頂區塊</div>

<div class="classA">A 區塊</div>

<div class="classB">B 區塊</div>

</body>
</html>
```

檢視檔案：https://dl.dropbox.com/u/101584453/wp/code/div_css.htm

兩種 CSS class 的顯示結果

到目前，我們可以看到有三種 css 的指定方式，也就是「標記 tag、代號 id 與類別 class」，可以用 CSS 來定位。事實上、這些指定方式還可以互相串接，以下是一些範例：

```
.classA table { ... }                  // 套用到 class="classA" 內部的 table
 標記
#topbar { ....}                        // 套用到 id="topbar" 的元素上
#topbar table .classA a { .... }       // 套用到 table 內具有 class="classA"
 類別裏的超連結 a 標記上。
```

更棒的是、這些選取方式之間還可以共用，只要利用逗點符號「,」分隔就可以了，以下是一些範例：

```
.classA, #topbar { ... }               // 套用到 class="classA" 或 id="topba
r" 的元素上
#topbar a, .classA a { ....}           // 套用到 id="topbar" 或 class=".clas
```

```
sA″ 裏的 a 元素上
```

一個實用的 CSS 的範例

如果您想進一步學習 CSS 的各個屬性與用法，請參考下列網址，我們將不一一說明這些屬性的用法。

- http://www.w3schools.com/css/default.asp

在此、筆者將自己常用的一個 CSS 檔案列出，讀者觀察後可以自行複製修改使用。

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
  margin: 0;
  padding: 0;
  border: 0;
  font: inherit;
  vertical-align: baseline;
  line-height:160%;
}

h1, h2, h3, h4, h5, h6 {
  color:#333333;
  margin: 0;
  font-family: '標楷體', 'Times New Roman';
  font-weight: bold;
}

p {
```

```
  margin: 10px 0 15px 0;
  font-size:100%;
}

li {
  font-size:100%;
}

footer p {
  color: #f2f2f2;
}

a {
  text-decoration: none;
  color: #007edf;
  text-shadow: none;

  transition: color 0.5s ease;
  transition: text-shadow 0.5s ease;
  -webkit-transition: color 0.5s ease;
  -webkit-transition: text-shadow 0.5s ease;
  -moz-transition: color 0.5s ease;
  -moz-transition: text-shadow 0.5s ease;
  -o-transition: color 0.5s ease;
  -o-transition: text-shadow 0.5s ease;
  -ms-transition: color 0.5s ease;
  -ms-transition: text-shadow 0.5s ease;
}

table {
  border-collapse: collapse;
  border-spacing: 0;
  border: 1px solid #373737;
  margin-bottom: 20px;
  text-align: left;
  margin-left:auto;
  margin-right:auto;
```

```
}

th {
    padding: 10px;
    background-color:black;
    color:white;
}

td {
  padding: 10px;
  border: 1px solid #373737;
}

em { font-weight:bold; }

#topbar {
  margin: 0;
  padding: 1px;
  border: 0;
  font: inherit;
  vertical-align: baseline;
  background-color:black;
  color:white;
  color:white;
  width:95%;
  text-align:right;
  font-weight:bold;
}

#content {
  margin:10px;
  padding:10px;
}

pre {
  border: 1px solid #373737;
  background-color:#dddddd;
```

```
  color:#333333;
  font-size:medium;
  width:95%;
  padding:10px;
}

img {
  border: 1px solid #373737;
  margin-left: auto;
  margin-right: auto;
  display: block;
}

.figure .caption {
  text-align:center;
}

#footer {
  text-align:center;
  font-size:small;
  color:#666666;
  margin: 10px;
  padding: 10px;
}
```

參考文獻

- http://www.w3schools.com/css/default.asp
- http://css.maxdesign.com.au/listutorial/

CSS 的變種： LESS 與 SASS

瀏覽器所認識的語言，不外乎是 HTML/CSS/JavaScript 這些 W3C 所規定的官方語言。因此像 LESS 與 SASS 這些語言，當然就屬於「非官方語言」，也就是「變種」。

LESS 與 SASS 都是 CSS 的變種，語法和 CSS 有點像，但卻又不太一樣。這類變種語言通常提供了更強大的功能，更簡潔的語法，而且最後會被轉換成 CSS 語言之後，才能放到瀏覽器上使用。

變種 1 ： LESS

LESS 是由 Alexis Sellier 所設計的工具，原本是用 Ruby 實作，用來將 LESS 語法轉換成 CSS 語言。後來改用 JavaScript 設計後，就能內嵌到瀏覽器中，動態的將 LESS 轉為 CSS 後立刻呈現。

舉例而言、 LESS 擁有變數的概念，例如以下範例中的 @color 就是一個變數。

```
@color: #4D926F;

#header {
  color: @color;
}
h2 {
  color: @color;
}
```

上述 LESS 文件被轉換成 CSS 之後，會變成下列形式。

```
#header {
  color: #4D926F;
}
h2 {
  color: #4D926F;
}
```

當然、LESS 還支援更強大的語法，例如 Mixin 就是個經典的功能，以下是一個簡單的範例。

```
.rounded-corners (@radius: 5px) {
  border-radius: @radius;
  -webkit-border-radius: @radius;
  -moz-border-radius: @radius;
}

#header {
  .rounded-corners;
}
#footer {
  .rounded-corners(10px);
}
```

上述文件被轉換成 CSS 之後會變成下列形式。

```
#header {
```

```
  border-radius: 5px;
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
}
#footer {
  border-radius: 10px;
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
}
```

LESS 可以提供層次結構的套嵌語法，以下是一個範例。

```
#header {
  h1 {
    font-size: 26px;
    font-weight: bold;
  }
  p { font-size: 12px;
    a { text-decoration: none;
      &:hover { border-width: 1px }
    }
  }
}
```

上述範例被轉換成 CSS 之後如下所示。

```
#header h1 {
  font-size: 26px;
  font-weight: bold;
}
#header p {
  font-size: 12px;
}
#header p a {
  text-decoration: none;
}
#header p a:hover {
  border-width: 1px;
}
```

另外、 LESS 還提供了「運算式與函數呼叫」等功能，請看以下範例。

```
@the-border: 1px;
@base-color: #111;
@red:        #842210;

#header {
  color: @base-color * 3;
  border-left: @the-border;
  border-right: @the-border * 2;
}
#footer {
  color: @base-color + #003300;
  border-color: desaturate(@red, 10%);
}
```

上述範例轉換成 CSS 之後，應該會像下面文件這樣。

```
#header {
  color: #333;
  border-left: 1px;
  border-right: 2px;
}
#footer {
  color: #114411;
  border-color: #7d2717;
}
```

變種 2 : SASS

SASS (Syntactically Awesome Stylesheets) 是由 Hampton Catlin 設計，然後由 Natalie Weizenbaum 實作的一種樣式語言。

如果您瞭解 JavaScript 與 Python 的差別，或許就能輕易理解 LESS 與 SASS 的差別了。

LESS 與 JavaScript 一樣，有用 {...} 的大括號框起來，而 SASS 則像 Python 一樣，沒有用大括號，而是透過換行與縮排來表現這種層次結構。

事實上，SASS 還有一個變種，就是加上大括號的版本，稱為 SCSS，這兩種風格可以互相轉換，您可以依照習慣挑選使用。

舉例而言，以下是一個 SASS 與 SCSS 的對照範例。

```
<table><tr><th>SASS</th><th>SCSS</th></tr>
<tr><td>
$blue: #3bbfce
$margin: 16px

.content-navigation
  border-color: $blue
  color: darken($blue, 9%)

.border
  padding: $margin/2
  margin:  $margin/2
  border-color: $blue
</td><td>
$blue: #3bbfce;
$margin: 16px;

.content-navigation {
  border-color: $blue;
  color: darken($blue, 20%);
}

.border {
  padding: $margin / 2;
  margin: $margin / 2;
  border-color: $blue;
}
</td></tr>
</table>
```

SASS 與 LESS 都提供了「變數、mixin、函數」等功能，但是 SASS 還提供了「繼承」的功能，以下是一個範例。

```
.error
  border: 1px #f00;
  background: #fdd;

.error.intrusion
```

```
  font-size: 1.3em;
  font-weight: bold;

.badError
  @extend .error;
  border-width: 3px;
```

上述範例轉換成 CSS 之後，將會產生下列文件。

```
.error, .badError {
  border: 1px #f00;
  background: #fdd;
}

.error.intrusion,
.badError.intrusion {
  font-size: 1.3em;
  font-weight: bold;
}

.badError {
  border-width: 3px;
}
```

除此之外，SASS 還提供了多重繼承的功能。

由於 SASS 主要是用 Ruby 實作的，因此不像 LESS 一樣有提供立即在瀏覽器內用 JavaScript 轉換成 CSS 的功能，不過開放原始碼的領域無奇不有，或許也有人用 JavaScript 實作出 SASS 轉 CSS 的功能也說不定。

參考文獻

- 聊聊主流框架，Less/Sass/Compass/Bootstrap/H5bp
- 为您详细比较三个 CSS 预处理器（框架）：Sass、LESS 和 Stylus
- http://lesscss.org/
- http://sass-lang.com/
- Wikipedia:LESS層疊樣式表
- http://sass-lang.com/guide

Bootstrap 網頁排版框架

Bootstrap 是由 Twitter 釋出的「高彈性」網頁呈現框架，其程式碼主要是用 LESS 和 jQuery 所架構出來的，LESS 負責網頁排版的部份，而 jQuery 則負責動態的程式部份。

以下是一個使用 bootstrap 的最簡單網頁框架，

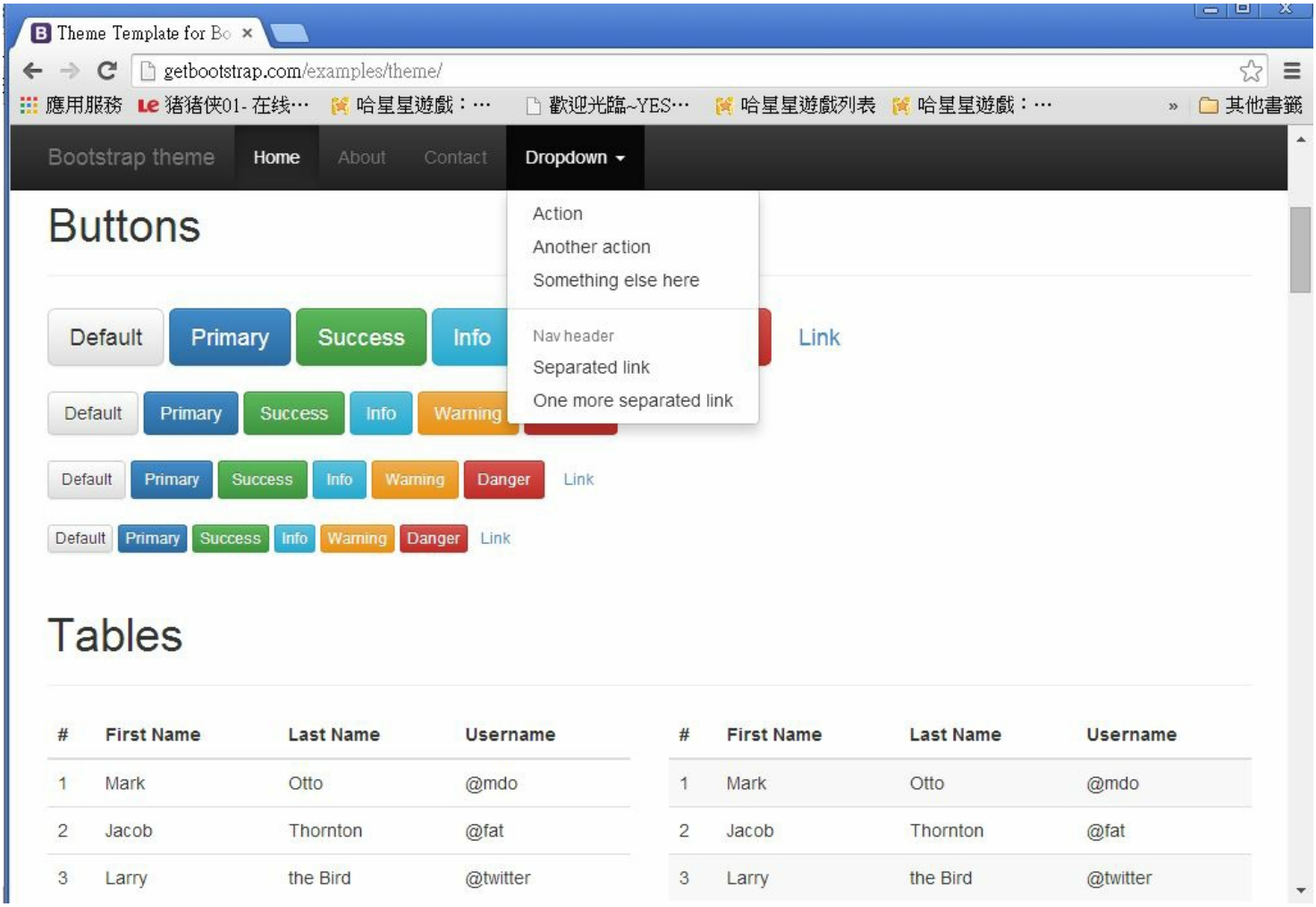
```
<!DOCTYPE html>
<html lang="zh-tw">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap 簡單範例! </title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <h1>Hello ! 您好! </h1>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

在上面 bootstrap 的 HTML 檔案中，您可以注意到 CSS 被放在前面，但是 javascript 被放在最後，這是為了讓網頁顯示速度能夠最快所採用的布局，因為 javascript 若先被載入的話，就會立刻被執行，反而容易讓瀏覽器多花時間在 javascript 的載入與執行上，浪費了 CPU 與網路時間。

```
...
    <link href="css/bootstrap.min.css" rel="stylesheet">
  </head>
    ...
    <script src="js/bootstrap.min.js"></script>
  </body>
...
```

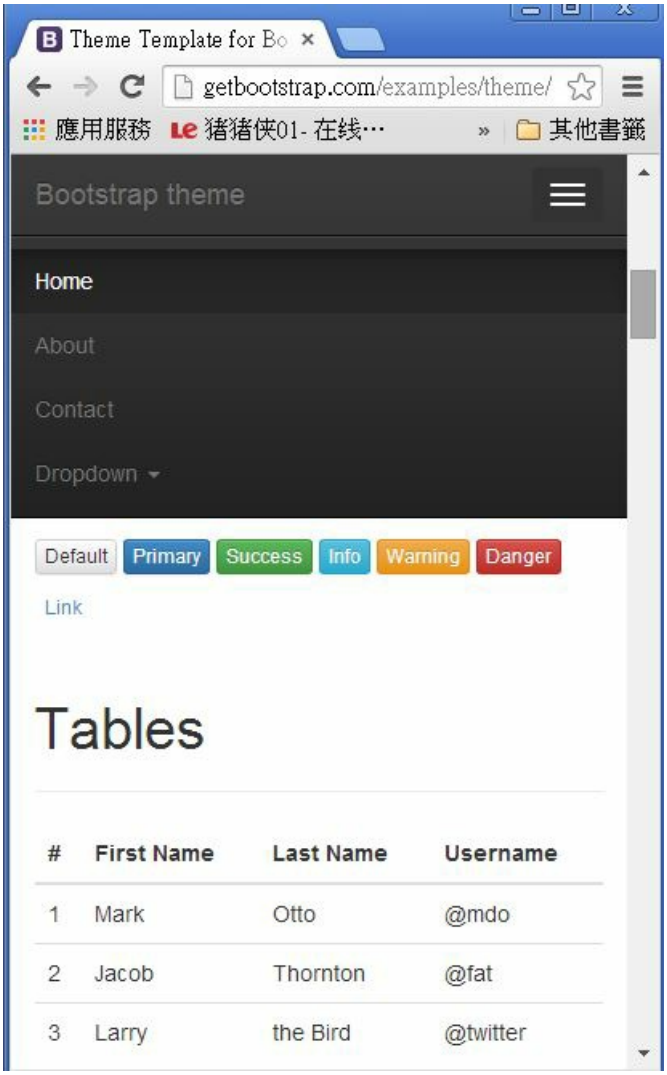
那麼、Bootstrap 到底能做甚麼呢？讓我們看看以下的呈現範例。

範例來源： http://getbootstrap.com/examples/theme/



您可以看到 bootstrap 可以輕易的做出美麗有質感的「標頭列、功能表、按鈕、表格」等等功能，這簡直就是那些「對美工有障礙的程式人」所夢寐以求的神器啊！

更棒的是，如果您將上述網頁變窄一點，小一點的話，整個畫面就會變成下列樣子，您可以看到功能表和表格都還是「自動」排得很好，並沒有因為畫面變窄而亂掉，這種特性在「手機」上特別有用，因此 bootstrap 還是讓你能夠開發出「跨平台」網頁的利器喔！



以下是上述範例的「標頭列與功能表」部分的程式碼，您可以看到要寫 bootstrap 網頁通常只要用 div 元素加上 bootstrap 專有的 class 屬性就可以搞定一切了。

```
...
    <!-- Fixed navbar -->
    <div class="navbar navbar-inverse navbar-fixed-top" role="navigation
">
      <div class="container">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle" data-toggle="colla
pse" data-target=".navbar-collapse">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="#">Bootstrap theme</a>
        </div>
```

```
        <div class="navbar-collapse collapse">
          <ul class="nav navbar-nav">
            <li class="active"><a href="#">Home</a></li>
            <li><a href="#about">About</a></li>
            <li><a href="#contact">Contact</a></li>
            <li class="dropdown">
              <a href="#" class="dropdown-toggle" data-toggle="dropdown"
>Dropdown <span class="caret"></span></a>
              <ul class="dropdown-menu" role="menu">
                <li><a href="#">Action</a></li>
                <li><a href="#">Another action</a></li>
                <li><a href="#">Something else here</a></li>
                <li class="divider"></li>
                <li class="dropdown-header">Nav header</li>
                <li><a href="#">Separated link</a></li>
                <li><a href="#">One more separated link</a></li>
              </ul>
            </li>
          </ul>
        </div><!--/.nav-collapse -->
      </div>
    </div>
...
```

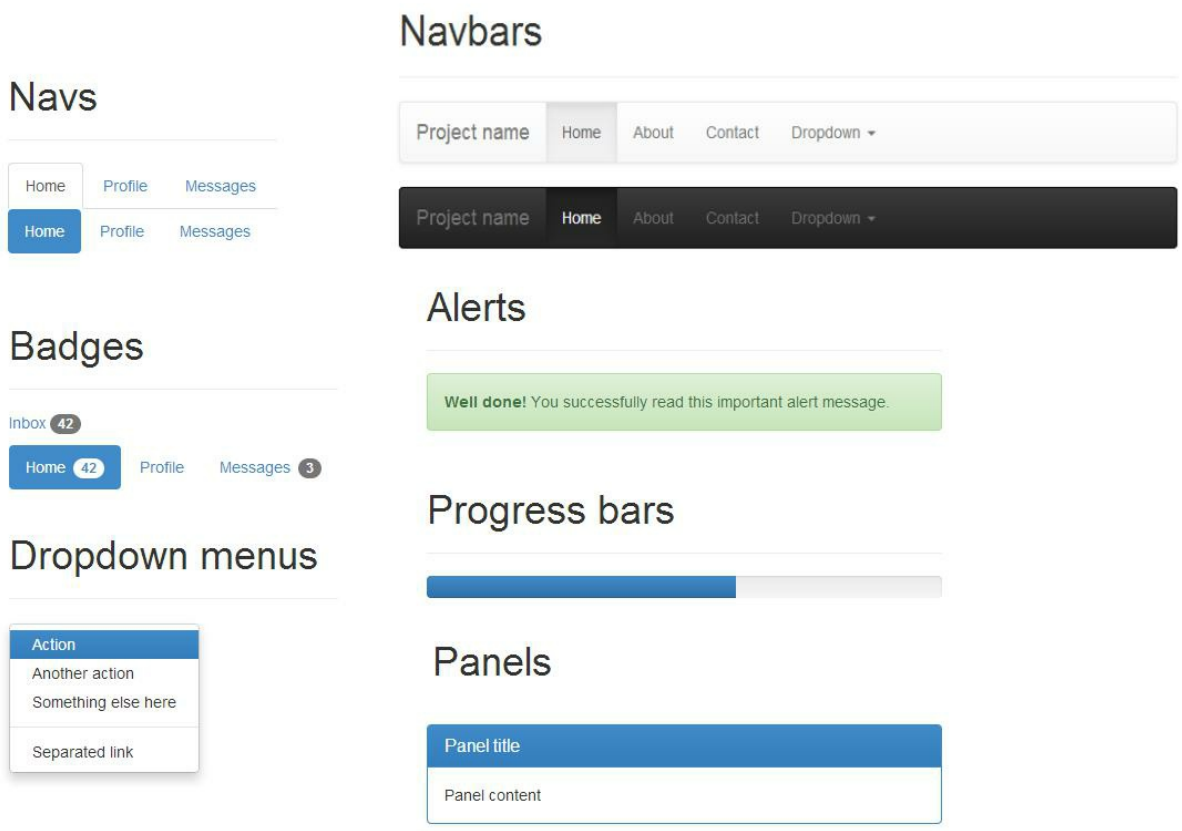
而下列的碼則是第一個表格的描述，由於 bootstrap 的設計理念是將版面寬度設定為 12 欄，因此 col-md-6 代表佔了六欄，因此在寬版的畫面中，會呈現出兩個表格並列呈現的狀況，但是在窄版的畫面中，就只會呈現出一個表格。

```
      <div class="row">
        <div class="col-md-6">
          <table class="table">
            <thead><tr><th>#</th><th>First Name</th><th>Last Name</th>
<th>Username</th></tr></thead>
            <tbody>
              <tr><td>1</td><td>Mark</td><td>Otto</td><td>@mdo</td></
tr>
              <tr><td>2</td><td>Jacob</td><td>Thornton</td><td>@fat</t
d></tr>
```

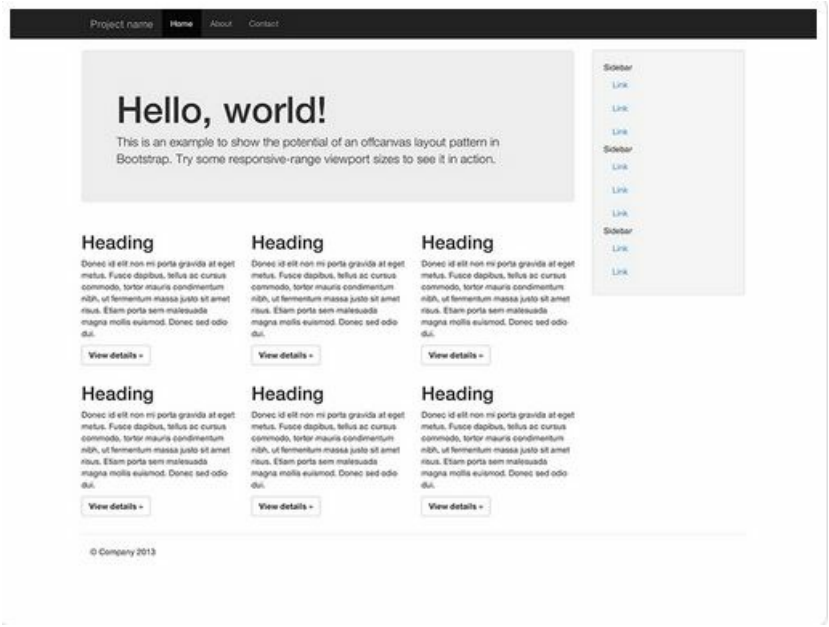
```
            <tr><td>3</td><td>Larry</td><td>the Bird</td><td>@twitte
r</td></tr>
          </tbody>
        </table>
      </div>
```

當然、以上描述並不能完整的呈現 bootstrap 的功能，只是一個簡單的入門範例而已。Bootstrap 雖然以排版呈現為主，但是由於搭配了 jQuery，因此也有一些進階的函數可以使用。

另外、還有以下這些常見的控制項，像是 Nav, Navbar, Badge, Dropdown Menu, Progress Bar, Panel 等等，這些控制項的美工都設計得很好，可以「程式人」輕鬆的設計出美觀個網頁。



而且 bootstrap 在提供多欄排版上也很方便，例如下圖就是一個採用三欄排版的範例。



不過看到上述功能，我就已經受益良多了，因為我就是那種對美工毫無概念，又不想去處理「手機排版」的懶惰程式人，有了 bootstrap 之後，我想我也可以輕鬆的設計出看起來很漂亮的網站了。

參考文獻

- http://getbootstrap.com/
- Mokoversity:Bootstrap 3 & HTML5 入門 (教學影片)
- Robin Notes: 使用 Bootstrap 建置雛型網站
- 網設必備－《Bootstrap》視覺、程式都能快速上手的網頁模組
- 做網站非學不可的Twitter Bootstrap
- 黑暗執行緒:Bootstrap!
- Bootstrap V2中文教學 - KKBruce
- Bootstrap 中文文档
- Wikipedia:Bootstrap (front-end framework)

從 Arduino 到 AVR 晶片(4) -- Blink with Timer（作者：Cooper Maa）

實驗目的

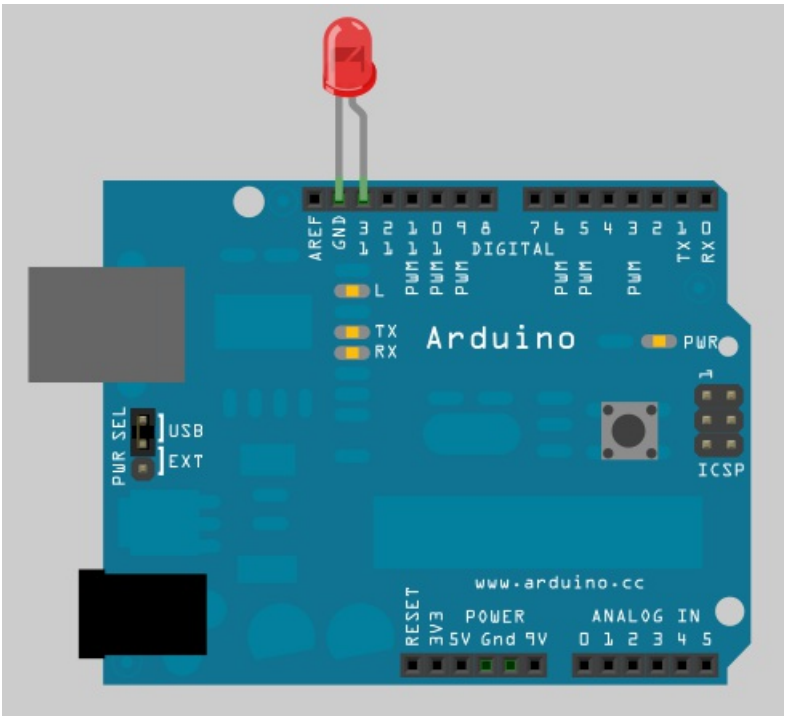
使用 Timer 計算時間，讓一顆燈號閃爍，每隔一秒切換一次燈號。

材料

- Arduino 主板 x 1
- LED x 1

接線

把 LED 接到 Arduino 板子上，LED 長腳 (陽極) 接到 pin13，短腳 (陰極) 接到 GND，如下圖：



程式碼

先來看 Arduino 版本的 Blink 程式:

```
/*
 * Blink.pde: 讓一顆燈號閃爍&#65292;每隔一秒切換一次燈號
 */

const int ledPin =  13;          // LED pin
```

```
void setup() {
  pinMode(ledPin, OUTPUT);      // 把 ledPin 設置成 output pin
}

void loop() {
  digitalWrite(ledPin, HIGH);   // 打開 LED 燈號
  delay(1000);                  // 延遲一秒鐘
  digitalWrite(ledPin, LOW);    // 關閉 LED 燈號
  delay(1000);                  // 延遲一秒鐘
}
```

這支程式是 Arduino 的入門程式，相信你應該很熟悉。

接下來我們改用 Timer 改寫程式，利用 Timer1 計算一秒鐘的時間:

```


/*
 * BlinkWithTimer1.pde: 讓一顆燈號閃爍&#65292;每隔一秒切換一次燈號&#65292;使用 Timer 算時間
 */

const int ledPin =  13;          // LED pin

void setup() {
  pinMode(ledPin, OUTPUT);      // 把 ledPin 設置成 output pin

  TCCR1A = 0x00;                // Normal mode, just as a Timer

  TCCR1B |= _BV(CS12);          // prescaler = CPU clock/1024
  TCCR1B &= ~_BV(CS11);
  TCCR1B |= _BV(CS10);

  TCNT1 = 0;
}

void loop() {
  digitalWrite(ledPin, HIGH);   // 打開 LED 燈號
  delay1s();                    // 延遲一秒鐘
  digitalWrite(ledPin, LOW);    // 關閉 LED 燈號
```

```
  delay1s();                        // 延遲一秒鐘
}

void delay1s()
{
  while (TCNT1 <15625)          // Ticks for 1 second @16 MHz,prescale=10
24
    ;  // do nothing

  TCNT1 = 0;
}
```

以 Arduino UNO 或 Duemilanove 為例，它們的時脈頻率是 16 MHz，如果把 Timer1 的 prescaler 設成 CPU clock/1024，那麼 Timer1 的 clock 便是:

Timer1 的時脈 = 16 MHz/1024 = 15625Hz

所以 Timer1 從 0 數到 15625 就是一秒鐘的時間。Timer1 是 16-bit 的，最大值可以到 65535。

把 Prescaler 設成 CPU clock/1024 的方法是:

```
TCCR1B |= _BV(CS12);           // prescaler = CPU clock/1024
TCCR1B &= ~_BV(CS11);
TCCR1B |= _BV(CS10);
```

使用中斷

底下的程式是改用 Timer Overflow 中斷的版本:

```


/*
 * BlinkWithInterrupt.pde: 讓一顆燈號閃爍&#65292;每隔一秒切換一次燈號&#652
92;使用 Timer 算時間
 */

const int ledPin =  13;          // LED pin

void setup() {
  pinMode(ledPin, OUTPUT);       // 把 ledPin 設置成 output pin

  TCCR1A = 0x00;                 // Normal mode, just as a Timer
```

```
  TCCR1B |= _BV(CS12);          // prescaler = CPU clock/1024
  TCCR1B &= ~_BV(CS11);
  TCCR1B |= _BV(CS10);

  TIMSK1 |= _BV(TOIE1);         // enable timer overflow interrupt

  TCNT1 = -15625;               // Ticks for 1 second @16 MHz,prescale=10
24
}

void loop() {
  // do nothing
}

ISR (TIMER1_OVF_vect)
{
  PORTB ^= _BV(5);              // Toggle LED, PB5 = Arduino pin 13
  TCNT1 = -15625;               // Ticks for 1 second @16 MHz,prescale=10
24
}
```

這次 Prescaler 仍然使用 CPU clock/1024，所以，要算一秒鐘，一樣是讓 Timer1 從 0 數到 15625 就可算出。除了啟用 Timer overflow 中斷外，我們還得寫個 ISR:

```
ISR (TIMER1_OVF_vect)
{
  PORTB ^= _BV(5);              // Toggle LED, PB5 = Arduino pin 13
  TCNT1 = -15625;               // Ticks for 1 second @16 MHz,prescale=10
24
}
```

當發生中斷跑到 ISR 時，代表時間已經經過 1 秒鐘，所以接著就切換燈號，然後再把 -15625 載入到 TCNT1，讓 Timer1 在一秒鐘後再次觸發中斷。

我們也可以不用 Prescaler，像這樣:

```
/*
 * BlinkWithNoPrescaling.pde: 讓一顆燈號閃爍&#65292;每隔一秒切換一次燈號&#
65292;使用 Timer 算時間
```

```
 */

volatile unsigned int count = 0;
const int ledPin =  13;          // LED pin

void setup() {
  pinMode(ledPin, OUTPUT);       // 把 ledPin 設置成 output pin

  TCCR1A = 0x00;                 // Normal mode, just as a Timer

  TCCR1B &= ~_BV(CS12);          // no prescaling
  TCCR1B &= ~_BV(CS11);
  TCCR1B |= _BV(CS10);

  TIMSK1 |= _BV(TOIE1);          // enable timer overflow interrupt

  TCNT1 = 0;
}

void loop() {
  // do nothing
}

ISR (TIMER1_OVF_vect)
{
  count++;

  if (count == 244) {              // overflow frequency = 16 MHz/65536 =
244Hz
    PORTB ^= _BV(5);               // Toggle LED, PB5 = Arduino pin 13
    count = 0;
  }
}
```

不用 Prescaler 的時候，Timer 的 clock 就是 16 MHz:

Timer1 overflow frequency = 16 MHz/65536 = 244 Hz

所以只要發生 244 次 Timer overflow 中斷，就代表已經過了 1 秒鐘的時間。

補充說明

因為 Timer0 已經被 Arduino 拿去用了，所以這篇教學使用 Timer1 示範。

延伸閱讀

- Arduino 筆記 – Lab1 Blinking a LED

【本文作者為馬萬圳，原文網址為： http://coopermaa2nd.blogspot.tw/2011/07/41-blink-with-timer.html ，由陳鍾誠編輯後納入本雜誌】

Verilog 的電路合成研究 -- 以 MUX 多工器為例（作者：陳鍾誠）

雖然 Verilog 描述的是硬體，但對於沒有硬體背景的我而言，總是把它當成 C 語言來寫。但是、有時候就是會踢到鐵板，像是我設計的某幾個 CPU 在合成的時候跑超久的，放到 Altera Quartus II 裏合成，跑上半個小時都還出不來。而且有時候用 icarus 模擬都正常，但是放上 FPGA 之後執行結果就錯了。

對於這個現像，我總是百思不解，直到有人告訴我這可能和 if 有關，因為當 if 的條件沒有列完，或者沒有加上完整的 else 時，就會有 Latch 產生。

對、就是 Latch，這在「組合電路設計」上是一個很不好的情況。因為「組合電路」不應該會有內部狀態，但是當 if 或 case 語句沒有完整的「被封閉」時，就會留下某些「開放」的情況。這時候「電路合成工具」為了讓電路仍然可以運作，會加入 Latch 來處理這種「開放案例」，結果原本應該沒有狀態的組合電路就變成了「有狀態的循序電路」，這可能會造成很多奇怪的問題產生。

為了解釋何謂「未封閉的 if 會產生 Latch，我們用 Altera Quartus II 的『Tools/Netlist Viewers/RTL Viewer』合成電路檢視工具來觀察一個多工器 (MUX) 的不同寫法，並試著分析看看「未封閉的 if/case 語句會產生甚麼問題？」

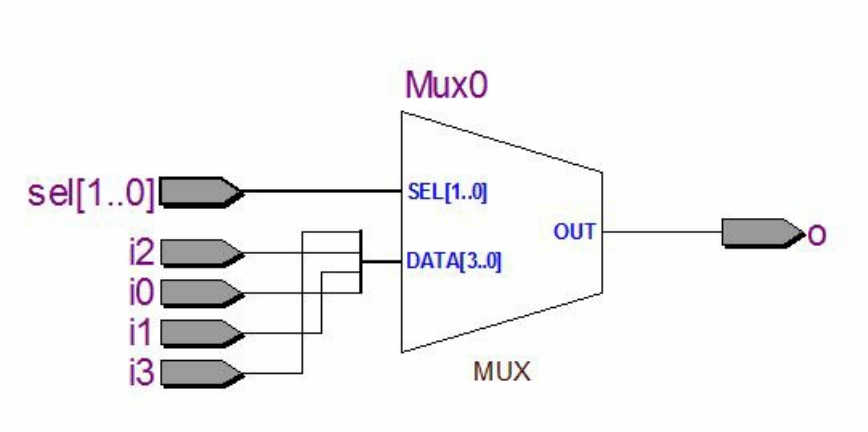
良好案例： mux_case.v

首先、讓我們看看用 case 語法設計的多工器，程式如下所示：

檔案： mux_case.v

```
module mux_case(input i0, i1, i2, i3,  input [1:0] sel, output reg o);
 always@(i0 or i1 or i2 or i3 or sel) begin
  case (sel)
    2'b00   : o = i0;
    2'b01   : o = i1;
    2'b10   : o = i2;
    default : o = i3;
  endcase
 end
endmodule
```

此時若用 RTL Viewer 去檢視，會看到下列的電路圖。



不良案例： mux_case_no_default.v

接著、讓我們把 case 裏的 default 拿掉，形成一個有「空洞」的程式，如下所示：

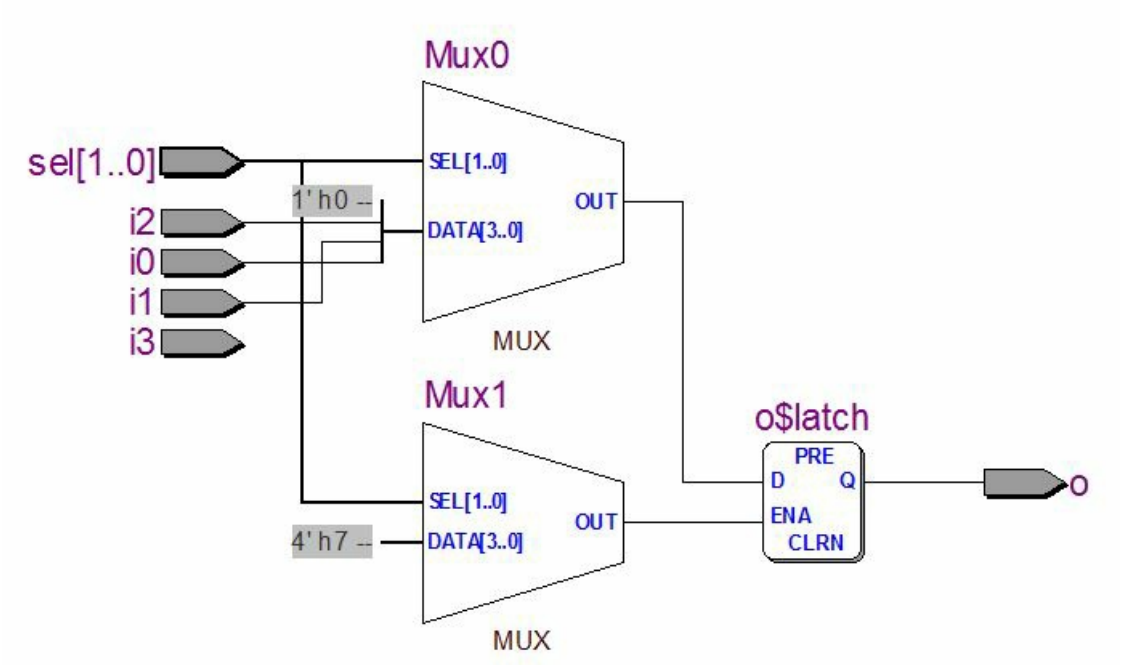
```
module mux_case_no_default(input i0, i1, i2, i3,  input [1:0] sel, outpu
t reg o);
 always@(i0 or i1 or i2 or i3 or sel) begin
  case (sel)
    2'b00   : o = i0;
    2'b01   : o = i1;
    2'b10   : o = i2;
//    default : o = i3;
  endcase
 end
endmodule
```

結果、按下 Start Compilation 的按鈕後，訊息框中就多出了下列的警告訊息。

```
...
Warning (10270): Verilog HDL Case Statement warning at mux_case_no_defaul
t.v(3): incomplete case statement has no default case item

Warning (10240): Verilog HDL Always Construct warning at mux_case_no_defa
ult.v(2): inferring latch(es) for variable "o", which holds its previous
value in one or more paths through the always construct
Info (10041): Inferred latch for "o" at mux_case_no_default.v(2)

Warning (13012): Latch mux_case_no_default:mux1|o has unsafe behavior
```

```
Warning (13013): Ports D and ENA on the latch are fed by the same signal
sel[0]
...
```

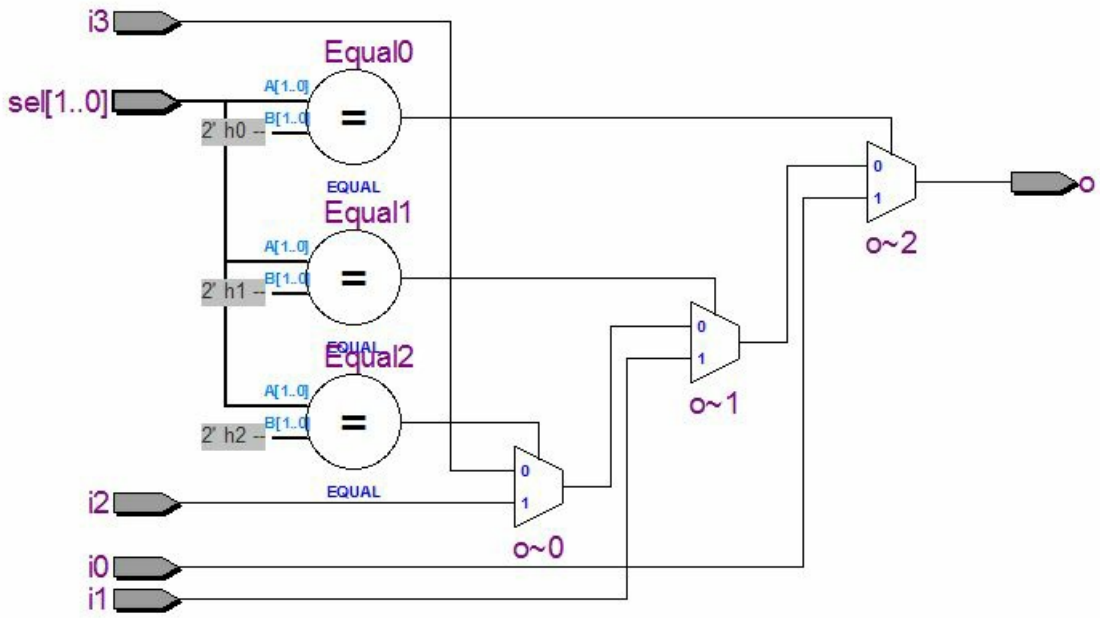
此時若用 RTL Viewer 去檢視，會看到下列的電路圖。



良好案例： mux_if1.v

檔案：mux_if1.v

```
module mux_if1(input i0, i1, i2, i3,  input [1:0] sel, output reg o);
 always@(i0 or i1 or i2 or i3 or sel) begin
  if (sel == 2'b00)
     o = i0;
  else if (sel == 2'b01)
     o = i1;
  else if (sel == 2'b10)
     o = i2;
  else
     o = i3;
 end
endmodule
```

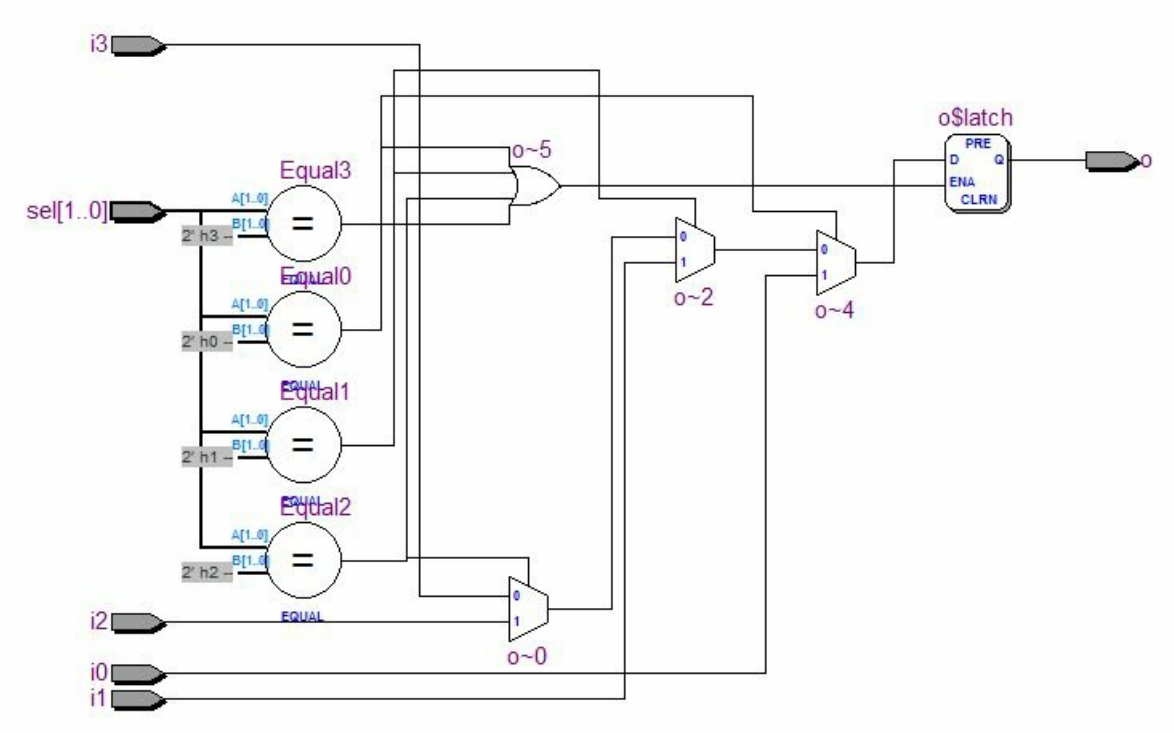


不良案例： mux_if1_no_default.v

檔案： mux_if1_no_default.v

```
module mux_if1_no_default(input i0, i1, i2, i3,
                          input [1:0] sel, output reg o);
 always@(i0 or i1 or i2 or i3 or sel) begin
  if (sel == 2'b00)
     o = i0;
  else if (sel == 2'b01)
     o = i1;
  else if (sel == 2'b10)
     o = i2;
  else if (sel == 2'b11)
     o = i3;
 end
endmodule
```

此時若用 RTL Viewer 去檢視，會看到下列的電路圖。



示範影片

筆者用 Altera Quartus II 建立的一個專案，並將上述四個範例放入，一個一個用 Quartus II 分析後並用 RTL Viewer 檢視電路圖，以下是完整的教學錄影，請讀者參考。

- YouTube : Verilog 的電路合成研究-以 MUX 多工器為例 (使用 Altera Quartus II / RTL Viewer 檢視)

結語

所以、當您用 Verilog 「寫程式」的時候，請務必對 case 語法加上 default，然後對每個 if 語法都要加上 else，這樣才能形成「封閉性良好」的程式。

當 Verilog 程式的「封閉性良好」時，合成工具才不會產生 Latch 去試圖彌補開放性的情況，也才不會讓你的組合電路變成「狀態電路」，造成電路的浪費。

更重要的是，這樣的 Verilog 程式才比較不容易出現「執行時期錯誤」或者「合成電路非常緩慢」的情況。

這是我個人慘痛經驗所換取的教訓，希望讀者不會重蹈覆轍啊！

參考文獻

- 真 OO无双 - 博客园: (原創) 多工器MUX coding style整理 (SOC) (Verilog) (Quartus II)
- Stack Exchange : Why are inferred latches bad?

Random variable of normal distribution（作者：Bridan）

發表 亂數產生器 一文後，吸引不少讀者參考，近日發現網友 PeiLingLiu 的部落格 ，分析她的筆記， Box-Muller transform 這方法也不錯。簡單的說， 將兩獨立均勻分佈變數，經變換計算與極座標轉換可產出常態

分佈變數 。於 嵌入式系統 ，除非 MCU (Micro Control Unit) 的速度夠快，不然 中央極限定理 的方法， 計算時間較短 。亂數分佈情形，請參考 EXCEL 檔 內容。

$$x \sim N(0,1)$$

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}, -\infty < x < \infty$$

$$F(x) = \int_{-\infty}^{x} f(x)dx$$

$$\int_{-\infty}^{\infty} f(x)dx = 1$$

$$(\int_{-\infty}^{\infty} f(x)dx)^2 = 1$$

$$let \quad x \perp y$$

$$\Rightarrow \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x)f(y)dxdy = 1$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (\frac{1}{\sqrt{2\pi}})^2 e^{\frac{-x^2-y^2}{2}} dxdy = 1$$

$$let \quad x = r\cos\theta, y = r\sin\theta$$

$$J = \begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{vmatrix} = \begin{vmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{vmatrix} = r$$

$$\Rightarrow \int_0^{2\pi}\int_0^{\infty} \frac{1}{2\pi} e^{\frac{-r^2}{2}} \|J\| dr d\theta$$

$$= \int_0^{2\pi}\int_0^{\infty} \frac{1}{2\pi} e^{\frac{-r^2}{2}} r dr d\theta = \frac{2\pi}{2\pi}\int_0^{\infty} e^{\frac{-r^2}{2}} r dr$$

$$= \int_0^{\infty} e^{\frac{-r^2}{2}} r dr = -e^{\frac{-r^2}{2}}\Big|_0^{\infty} = -(0-1) = 1$$

$$Box-Muller \quad Transformation$$

$$\int_0^{2\pi}\int_0^{\infty}\frac{1}{2\pi}e^{\frac{-r^2}{2}}rdrd\theta=\int_0^{2\pi}\frac{1}{2\pi}d\theta\times\int_0^{\infty}e^{\frac{-r^2}{2}}rdr$$

$$\begin{cases}\theta\sim U(0,2\pi) & ,r\perp\theta\\ r\sim g(r)=re^{\frac{-r^2}{2}} & ,r>0\end{cases}$$

$$G(r)=\int_0^r e^{\frac{-u^2}{2}}udu=-e^{\frac{-u^2}{2}}\Big|_0^r=-(e^{\frac{-r^2}{2}}-1)$$

$$u=G(r)=1-e^{\frac{-r^2}{2}}\sim U(0,1)$$

$$\Rightarrow e^{\frac{-r^2}{2}}=1-u \quad \Rightarrow\frac{-r^2}{2}=\ln(1-u)$$

$$\Rightarrow r^2=-2\ln(1-u)$$

$$\Rightarrow r=\sqrt{-2\ln(1-u)}$$

$$\Rightarrow r=\sqrt{-2\ln u}$$

$$\therefore x=r\cos\theta=\sqrt{-2\ln u}\times\cos\theta$$

$$or \quad y=r\sin\theta=\sqrt{-2\ln u}\times\sin\theta$$

(本文來自「研發養成所」 Bridan 的網誌，原文網址為 http://4rdp.blogspot.tw/2008/06/random-variable-of-normal-distribution.html ，由陳鍾誠編輯後刊登於程式人雜誌)

Visual Basic 6.0: 奇數魔術方塊(Odd Magic Square) 詳細解法 (作者：廖憲得 0xde)

什麼是奇數魔術方陣 (?)

魔術方塊是許多人想要解決的一個古老的數學問題，您可能在一些雜誌上看過，也可能您的老師有介紹過。一個魔術方塊是在於安排數字在一矩陣 [n*n] 中，從 1 到 n^2, 每一數字僅出現一次,而且，任一列、任一行或任一對角線的總和都相同。求總和的公式要證明為 n*[(n^2 + 1) / 2] ，並不是很困難，若我們利用這個公式，對 [5*5] 矩陣而言，其總和為 5*[(5^2 + 1) / 2] = 65 ，其對應的魔術方塊輸出如下：

17	24	1	8	15	65
23	5	7	14	16	65
4	6	13	20	22	65
10	12	19	21	3	65
11	18	25	2	9	65
65	65	65	65	65	65


```
'# [Visual Basic 6.0] 奇數魔術方塊(Odd Magic Square)
'# 0xDe
Dim InputN
Dim Squate()
Private Sub Form_Activate()
'------------------------------
InputN = 3 ' 輸入 (必須為奇數)
'------------------------------



'------------------------------
If InputN Mod 2 = 0 Then Exit Sub ' 判斷是否為奇數
'------------------------------
ReDim Square(InputN - 1, InputN - 1)
'------------------------------
Print "N= " & InputN & "的奇數魔術方陣" & vbCrLf
Randomize Timer ' 亂數產生
```

```
TempX = Int(Rnd * InputN) ' 隨機起始 X
TempY = Int(Rnd * InputN) ' 隨機起始 Y
'-----------------------------
Do Until N = (InputN ^ 2) ' 直到放滿
    If Square(TempX, TempY) = "" Then
        N = N + 1
        Square(TempX, TempY) = N

        TempX = TempX - 1 ' 向上移
        If TempX < 0 Then TempX = InputN - 1
        TempY = TempY + 1 ' 向右移
        If TempY > InputN - 1 Then TempY = 0
    Else
        ' 恢復原本的狀態往下
        TempX = TempX + 1
        If TempX > InputN - 1 Then TempX = 0
        TempY = TempY - 1
        If TempY < 0 Then TempY = InputN - 1
        ' 往下
        TempX = TempX + 1
        If TempX > InputN - 1 Then TempX = 0
    End If
Loop
'-----------------------------
    For I = 0 To InputN - 1 ' 將結果輸出
        For J = 0 To InputN - 1
            Print Square(I, J);
        Next J
            Print
    Next I
'-----------------------------
End Sub
```

- 檔案下載：Visual Basic 6.0:奇數魔術方塊(Odd Magic Square).rar

【本文作者為「廖憲得」，原文網址為： http://www.dotblogs.com.tw/0xde/archive/2013/11/13/129187.aspx ，由陳鍾誠編輯後納入本雜誌】

雜誌訊息

讀者訂閱

程式人雜誌是一個結合「開放原始碼與公益捐款活動」的雜誌，簡稱「開放公益雜誌」。開放公益雜誌本著「讀書做善事、寫書做公益」的精神，我們非常歡迎程式人認養專欄、或者捐出您的網誌，如果您願意成為本雜誌的專欄作家，請加入 程式人雜誌社團 一同共襄盛舉。

我們透過發行這本雜誌，希望讓大家可以讀到想讀的書，學到想學的技術，同時也讓寫作的朋友的作品能產生良好價值 － 那就是讓讀者根據雜誌的價值捐款給慈善團體。 讀雜誌做公益也不需要有壓力，您不需要每讀一本就急著去捐款，您可以讀了十本再捐，或者使用固定的月捐款方式，當成是雜誌訂閱費，或者是季捐款、一年捐一次等都 OK！甚至是單純當個讀者我們也都很歡迎！

本雜誌每期參考價：NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體。例如可捐贈給「羅慧夫顱顏基金會 彰化銀行(009) 帳號：5234-01-41778-800」。(若匯款要加註可用「程式人雜誌」五個字)

投稿須知

給專欄寫作者： 做公益不需要有壓力。如果您願意撰寫專欄，您可以輕鬆的寫，如果當月的稿件出不來，我們會安排其他稿件上場。

給網誌捐贈者： 如果您沒時間寫專欄或投稿，沒關係，只要將您的網誌以 [創作共用的「姓名標示、非商業性、相同方式分享」授權] 並通知我們，我們會自動從中選取需要的文章進行編輯，放入適當的雜誌當中出刊。

給文章投稿者： 程式人雜誌非常歡迎您加入作者的行列，如果您想撰寫任何文章或投稿，請用 markdown 或 LibreOffice 編輯好您的稿件，並於每個月 25 日前投稿到程式人雜誌社團 的檔案區，我們會盡可能將稿件編入隔月1號出版程式人雜誌當中，也歡迎您到社團中與我們一同討論。

如果您要投稿給程式人雜誌，我們最希望的格式是採用 markdown 的格式撰寫，然後將所有檔按壓縮為 zip 上傳到社團檔案區給我們， 如您想學習 markdown 的撰寫出版方式，可以參考 看影片學 markdown 編輯出版流程 一文。

如果您無法採用 markdown 的方式撰寫，也可以直接給我們您的稿件，像是 MS. Word 的 doc 檔或 LibreOffice 的 odt 檔都可以，我們 會將這些稿件改寫為 markdown 之後編入雜誌當中。

參與編輯

您也可以擔任程式人雜誌的編輯，甚至創造一個全新的公益雜誌，我們誠摯的邀請您加入「開放公益出版」的行列，如果您想擔任編輯或創造新雜誌，也歡迎到 程式人雜誌社團 來與我們討論相關事宜。

公益資訊

公益團體	聯絡資訊	服務對象	捐款帳號

財團法人羅慧夫顱顏基金會	http://www.nncf.org/ lynn@nncf.org 02-27190408分機232	顱顏患者 (如唇顎裂、小耳症或其他罕見顱顏缺陷）	銀行：009彰化銀行民生分行 帳號：5234-01-41778-800
社團法人台灣省兒童少年成長協會	http://www.cyga.org/ cyga99@gmail.com 04-23058005	單親、隔代教養.弱勢及一般家庭之兒童青少年	銀行：新光銀行 戶名：台灣省兒童少年成長協會 帳號：103-0912-10-000212-0