CHAPTER 03

流程控制

3-1 認識流程控制

3-2 if

3-3 switch

3-4 for

3-5 while

3-6 do…while

3-7 for…in

3-8 for…of

3-9 break、continue 與標記

3-1 認識流程控制

我們在前幾章所示範的例子都是很單純的程式，它們的執行方向都是從第一行敘述開始，由上往下依序執行，不會轉彎或跳行，但大部分的程式並不會這麼單純，它們可能需要針對不同的情況做不同的處理，以完成更多任務，於是就需要**流程控制** (flow control) 來協助控制程式的執行方向。

JavaScript 的流程控制分成下列兩種類型：

- **選擇結構** (decision structure)：用來檢查條件式，然後根據結果為 true 或 false 去執行不同的敘述。JavaScript 提供的選擇結構如下：
  - if
  - switch
  - try...catch...finally (例外處理，詳閱第 6 章)
- **迴圈結構** (loop structure)：用來重複執行指定的敘述。JavaScript 提供的迴圈結構如下：
  - for
  - while
  - do...while
  - for...in
  - for...of

流程控制經常需要檢查一些條件式或資料是 true 或 false，原則上，JavaScript 會將空字串 (" 或 "")、數值 0、NaN、undefined、null 等值視為 false，這些值以外的其它值則視為 true。

3-2 if

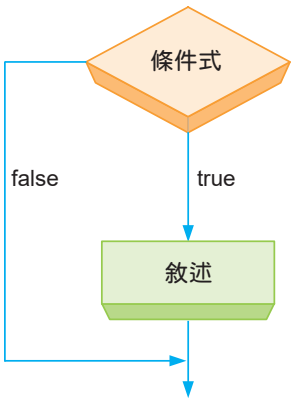
if 可以用來檢查條件式，然後根據結果為 true 或 false 去執行不同的敘述，又分成 if、if...else、if...else if 等類型。

3-2-1 if (若 ... 就 ...)

if 的語法如下，若條件式的結果為 true，就執行敘述，換句話說，若條件式的結果為 false，就不執行敘述。

```
if（條件式）{
    敘述;
}
```

關鍵字　結束大括號　條件式為 true 就執行敘述　開始大括號



NOTE

在條列 if 的語法時，為了提高可讀性，我們將大括號裡面的敘述以 if 關鍵字為基準向右縮排一個層次 (兩個空白)。此外，敘述可以是一個或多個，若只有一個，那麼大括號可以省略，不過，為了避免混淆，建議還是保留大括號。

下面是一個例子，它會要求使用者輸入數學分數 (0-100)，若該分數大於等於 60，就顯示「及格！」。

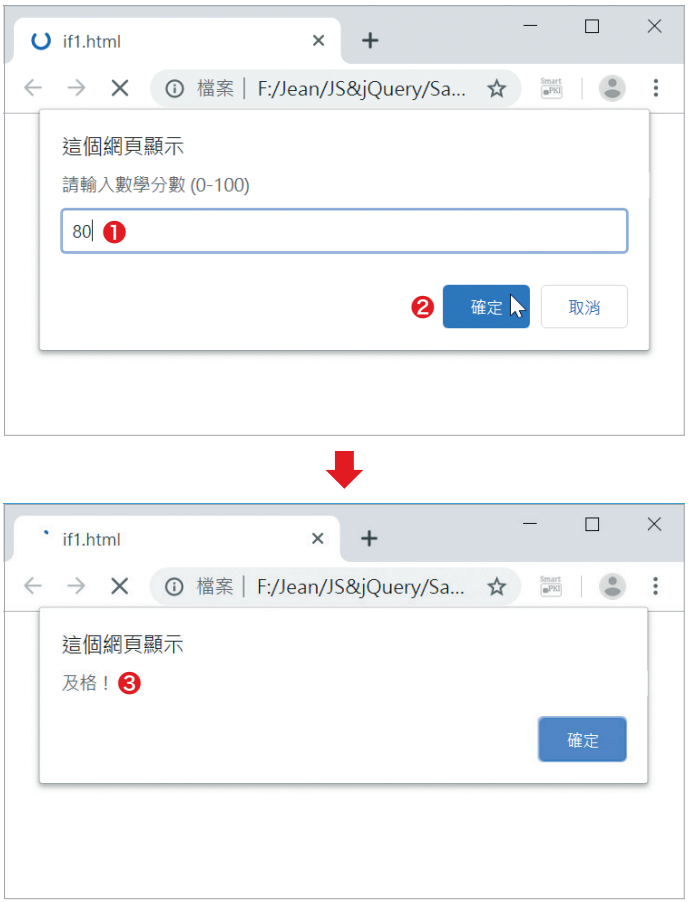
\Ch03\if1.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <script src="if1.js"></script>
  </body>
</html>
```

\Ch03\if1.js

```
// 呼叫 Window 物件的 prompt() 方法要求輸入數學分數並指派給變數 score
var score = window.prompt('請輸入數學分數 (0-100)', '');
if (score >= 60) {
  window.alert('及格！');
}
```

執行結果如下圖，若所輸入的分數大於等於 60，條件式 (score >= 60) 的結果為 true，就會執行第 13 行，顯示「及格！」；相反的，若所輸入的分數小於 60，條件式 (score >= 60) 的結果為 false，就不會執行第 13 行。



❶ 輸入數學分數 (例如 80)　❷ 按 [確定]　❸ 顯示「及格！」

T I P

prompt() 是 Window 物件提供的方法，可以用來顯示對話方塊要求使用者輸入資料，然後傳回該資料。prompt() 方法的第一個參數是對話方塊中的提示文字，第二個參數是欄位預設的輸入值，此例的第二個參數為空字串，所以欄位一開始是空白的。

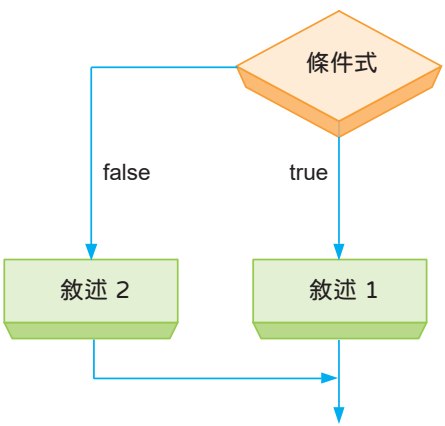
3-2-2 if...else (若 ... 就 ... 否則 ...)

if...else 的語法如下，若條件式的結果為 true，就執行敘述 1，否則執行敘述 2，所以敘述 1 和敘述 2 只有一組會被執行。

```
if (條件式) {
    敘述1;
}
else {
    敘述2;
}
```

❶ if 區塊 (條件式為 true 就執行敘述 1)

❷ else 區塊 (條件式為 false 就執行敘述 2)



下面是一個例子，它會要求使用者輸入數學分數 (0-100)，若該分數大於等於 60，就顯示「及格！」，否則顯示「不及格！」。

\Ch03\if2.js

```
01  var score = window.prompt('請輸入數學分數 (0-100)', '');
02  if (score >= 60) {
03    window.alert('及格！');
04  }
05  else {
06    window.alert('不及格！');
07  }
```

執行結果如下圖，若所輸入的分數大於等於 60，條件式 (score >= 60) 的結果為 true，就會執行 if 區塊的敘述，也就是第 03 行，顯示「及格！」，然後跳出 if...else 結構，不會再去執行第 06 行；相反的，若所輸入的分數小於 60，條件式 (score >= 60) 的結果為 false，就會執行 else 區塊的敘述，也就是跳過第 03 行，直接執行第 06 行，顯示「不及格！」。

❶ 輸入 80　❷ 按 [確定]　❸ 顯示「及格！」

❹ 輸入 50　❺ 按 [確定]　❻ 顯示「不及格！」

3-2-3 if...else if (若 ... 就 ... 否則 若 ... 就 ... 否則 ...)

if...else if 的語法如下，一開始先檢查條件式 1，若條件式 1 的結果為 true，就執行敘述 1，否則檢查條件式 2，若條件式 2 的結果為 true，就執行敘述 2，...，依此類推，若所有條件式的結果均為 false，就執行敘述 N+1，所以敘述 1 ~ 敘述 N+1 只有一組會被執行。

```
if（條件式1）{
   敘述1; ❶
}
else if（條件式2）{
   敘述2; ❷
}
...
else {
   敘述N+1; ❸
}
```

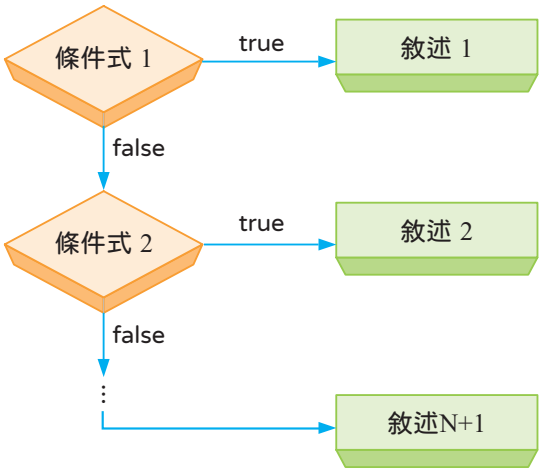
❶ 條件式 1 為 true 就執行敘述 1

❷ 條件式 2 為 true 就執行敘述 2

❸ 所有條件式均為 false 就執行敘述 N+1

if...else if 就是巢狀的 if...else，看似複雜但實用性也最高，因為 if...else if 可以處理多個條件式，而 if 和 if...else 只能處理一個條件式。

除了 if 之外，接下來要介紹的 switch、for、while、do...while 等也都能使用巢狀結構，只是層次盡量不要太多，而且要利用縮排來提高可讀性。



下面是一個例子，它會要求使用者輸入數學分數 (0-100)，然後根據級距判斷該分數的等第。

\Ch03\if3.js

```
01  var score = window.prompt('請輸入數學分數 (0-100)', '');
02  if (score >= 90) {
03    window.alert('優等！');
04  }
05  else if (score < 90 && score >= 80) {
06    window.alert('甲等！');
07  }
08  else if (score < 80 && score >= 70) {
09    window.alert('乙等！');
10  }
11  else if (score < 70 && score >= 60) {
12    window.alert('丙等！');
13  }
14  else {
15    window.alert('不及格！');
16  }
```

根據 90 以上 (含)、89 ~ 80、79 ~ 70、69 ~ 60、59 以下 (含) 的級距，判斷該分數的等第為「優等！」、「甲等！」、「乙等！」、「丙等！」或「不及格！」

執行結果如下圖，此例所輸入的分數為 85，表示在執行第 02 行時，條件式 (score >= 90) 的結果為 false，於是跳過第 03 行，直接執行第 05 行，此時條件式 (score < 90 && score >= 80) 的結果為 true，於是執行第 06 行，顯示「甲等！」，然後跳出 if...else if 結構，不會再去執行第 08 ~ 16 行。您不妨試著輸入其它數字，看看執行結果是否符合預期。

❶ 輸入 85　❷ 按 [確定]　❸ 顯示「甲等！」

3-3 switch

switch 結構可以根據運算式的值去執行不同的敘述，其語法如下，首先將運算式當作比較對象，接下來依序比較它有沒有等於哪個 case 後面的值，若有，就執行該 case 的敘述，然後執行 break 指令跳出 switch 結構，若沒有，就執行 default 的敘述，然後執行 break 指令跳出 switch 結構。

switch 結構的 case 區塊或 default 區塊的後面都要加上 break 指令，用來跳出 switch 結構。至於 if...else 結構則不需要加上 break 指令，因為在 if 區塊或 else 區塊執行完畢後，就會自動跳出 if...else 結構。

```
switch（運算式）{
  case 值1:
    敘述1;      ❶
    break;
  case 值2:
    敘述2;      ❷
    break;
  ...
  default:
    敘述N+1;    ❸
    break;
}
```

❶ case 區塊 (運算式等於值 1 就執行敘述 1)

❷ case 區塊 (運算式等於值 2 就執行敘述 2)

❸ default 區塊 (運算式不等於任何值就執行敘述 N+1)

下面是一個例子，它會要求使用者輸入 1-3 的數字，然後顯示對應的英文「ONE」、「TWO」、「THREE」，否則顯示「數字超過範圍」。

\Ch03\switch.js

```
01  var number = window.prompt(' 請輸入 1-3 的數字 ', '');
02  switch (number) {
03    case '1':
04      window.alert('ONE');
05      break;
06    case '2':
07      window.alert('TWO');
08      break;
09    case '3':
10      window.alert('THREE');
11      break;
12    default:
13      window.alert(' 數字超過範圍 ');
14      break;
15  }
```

執行結果如下圖，此例所輸入的數字為 2，它會被當作 switch 結構的比較對象（第 02 行），接下來依序比較它有沒有等於哪個 case 後面的值，發現等於 case '2': （第 06 行），於是執行 case '2': 的敘述，顯示「TWO」（第 07 行），然後執行 break 指令跳出 switch 結構（第 08 行），不會再去執行第 09 ~ 15 行。

❶ 輸入 2　❷ 按 [確定]　❸ 顯示「TWO」

這個例子也可以使用 if...else if 來完成，如下。

\Ch03\if4.js

```
01  var number = window.prompt(' 請輸入 1-3 的數字 ', '');
02  if (number === '1') {
03    window.alert('ONE');
04  }
05  else if (number === '2') {
06    window.alert('TWO');
07  }
08  else if (number === '3') {
09    window.alert('THREE');
10  }
11  else {
12    window.alert(' 數字超過範圍 ');
13  }
```

❶ 輸入 2　❷ 按 [確定]　❸ 顯示「TWO」

NOTE

- switch 結構在比較有沒有等於哪個 case 後面的值時，所使用的是 === 運算子，而不是 == 運算子，值和型別都必須相同才會被判斷為相等。
- switch 結構的優點在於能夠清楚呈現出所要執行的效果，程式寫到愈大就愈能看到其優點，不過，它也有缺點，那就是只能執行一個條件式，而 if...else if 則無此限制。

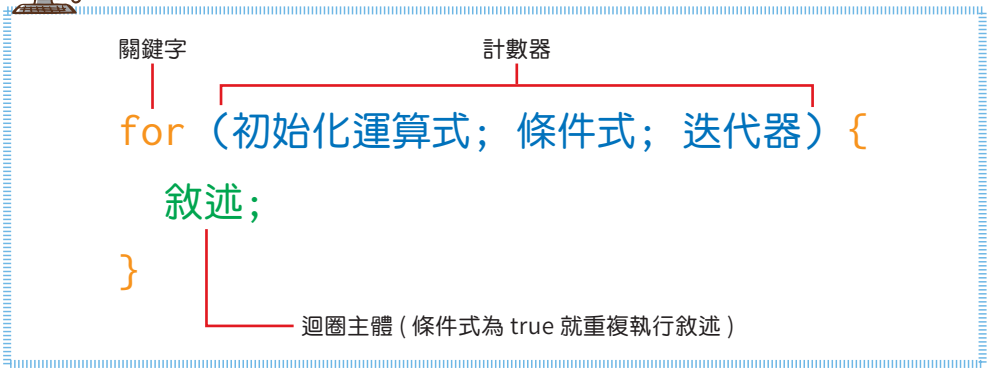
3-4 for

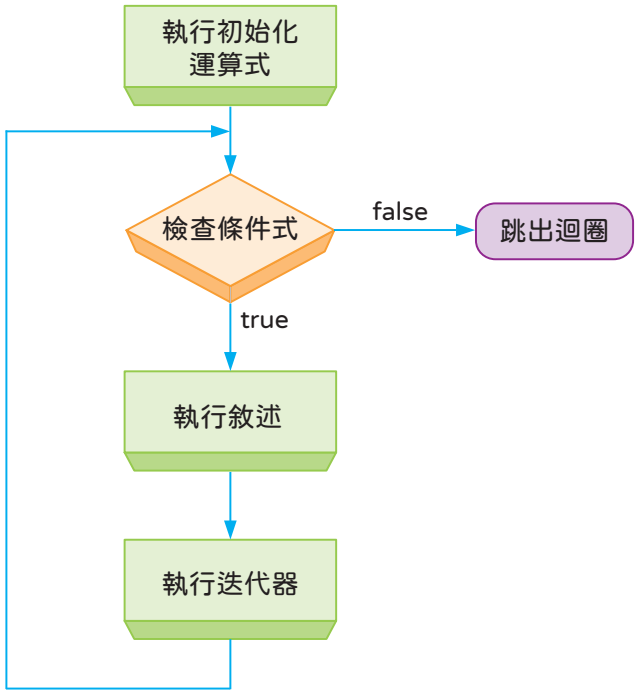
迴圈結構 (loop structure) 可以用來重複執行指定的敘述，它會檢查條件式，若結果為 true，就執行指定的敘述，然後再度檢查條件式，若結果仍為 true，就重複執行指定的敘述，然後再度檢查條件式，...，如此周而復始，直到條件式的結果為 false 才跳出迴圈。

JavaScript 提供的迴圈結構有 for、while、do...while、for...in 和 for...of，其中最常見的是 **for** 迴圈，適合應用在有指定重複次數的情況。舉例來說，假設要計算 1 加 2 加 3 一直加到 100 的總和，那麼可以使用 for 迴圈逐一將 1、2、3、...、100 累加在一起，就能求出總和，而且迴圈的執行次數就是 100 次。

由於我們通常會使用變數來控制 for 迴圈的執行次數，所以 for 迴圈又稱為**計數迴圈**，而此變數稱為**計數器**。

for 迴圈的語法如下，在進入 for 迴圈時，會先執行初始化運算式將計數器加以初始化，接著檢查條件式，若結果為 false，就跳出迴圈，若結果為 true，就執行迴圈內的敘述，完畢後執行迭代器將計數器加以更新，接著再度檢查條件式，若結果為 false，就跳出迴圈，若結果為 true，就重複執行迴圈內的敘述，完畢後執行迭代器將計數器加以更新，接著再度檢查條件式，...，如此周而復始，直到條件式的結果為 false 才跳出迴圈。





for 迴圈的計數器包含三個部分，例如：

```
for (let i = 0; i < 10; i++) {
}
```

❶ ❷ ❸

❶ **初始化運算式**：此例是宣告變數 i 做為計數器，初始值設定為 0。

❷ **條件式**：此例是將條件式設定為 i < 10;，只要變數 i 小於 10 就重複執行迴圈內的敘述，直到變數 i 大於等於 10 才跳出迴圈。

❸ **迭代器**：此例是將迭代器設定為 i++，表示迴圈每重複一次就將變數 i 的值遞增 1，所以變數 i 的值會從 0 遞增為 1、2、3、...、10。**迭代** (iteration) 一詞指的是要重複執行的一組敘述，亦可視為「重複」的同義字。

下面是一個例子，它會計算 1 ~ 10 的整數總和，然後顯示結果為 55。

\Ch03\for1.js

```
01  var total = 0;
02  for (let i = 1; i <= 10; i++){
03    total = total + i;
04  }
05  window.alert(total);
```

- 01：宣告變數 total 用來存放總和，初始值設定為 0。
- 02 ~ 04：let i = 1; 是宣告變數 i 做為計數器，初始值設定為 1，而 i <= 10; 是做為條件式，只要變數 i 小於等於 10 就重複執行迴圈內的敘述，至於 i++ 則是做為迭代器，迴圈每重複一次就將變數 i 的值遞增 1。

  for 迴圈的執行次數為 10 次，針對每一次的執行，第 03 行 total = total + i; 左右兩邊的 total 和 i 的值如下。

迴圈的執行次數	右邊的 total	i	左邊的 total	迴圈的執行次數	右邊的 total	i	左邊的 total
第一次	0	1	1	第六次	15	6	21
第二次	1	2	3	第七次	21	7	28
第三次	3	3	6	第八次	28	8	36
第四次	6	4	10	第九次	36	9	45
第五次	10	5	15	第十次	45	10	55

下面是另一個例子，它會顯示 1 ~ 10 之間所有奇數的總和。

\Ch03\for2.js

```
01  var total = 0;
02  for (let i = 1; i <= 10; i+=2){
03    total = total + i;
04  }
05  window.alert(total);
```

這個例子的差別在於將第 02 行的 i++ 改為 i+=2，令計數器每次遞增 2，因此，for 迴圈的執行次數為 5 次，針對每一次的執行，第 03 行 total = total + i; 左右兩邊的 total 和 i 的值如下。

迴圈的執行次數	右邊的 total	i	左邊的 total	迴圈的執行次數	右邊的 total	i	左邊的 total
第一次	0	1	1	第四次	9	7	16
第二次	1	3	4	第五次	16	9	25
第三次	4	5	9				

TIP

原則上，在我們撰寫迴圈後，程式就會依照設定將迴圈執行完畢，不會中途跳出迴圈。不過，有時我們可能需要在迴圈內檢查其它條件式，一旦成立就強制跳出迴圈，此時可以使用 break 指令，第 3-9 節有進一步的說明。

3-9 break、continue 與標記

break

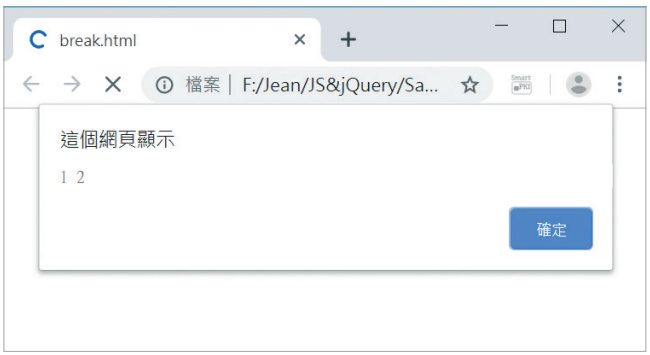
原則上，在我們撰寫迴圈後，程式就會依照設定將迴圈執行完畢，不會中途跳出迴圈。不過，有時我們可能需要在迴圈內檢查其它條件式，一旦成立就強制跳出迴圈，此時可以使用 **break** 指令。

下面是一個例子，其中第 03 ~ 08 行的 for 迴圈並沒有執行到 10 次，因為在第 04 行檢查到變數 i 等於 3 時，就會執行第 05 行的 break 指令，強制跳出迴圈，然後執行第 09 行顯示結果為 1、2。

\Ch03\break.js

```
01  // 宣告用來存放結果的變數 str，初始值為空字串
02  var str = '';
03  for (var i = 1; i <= 10; i++) {
04    if (i === 3) {
05      break;
06    }
07    str = str + i + '\t';
08  }
09  window.alert(str);
```

若變數 i 等於 3，就強制跳出迴圈



continue

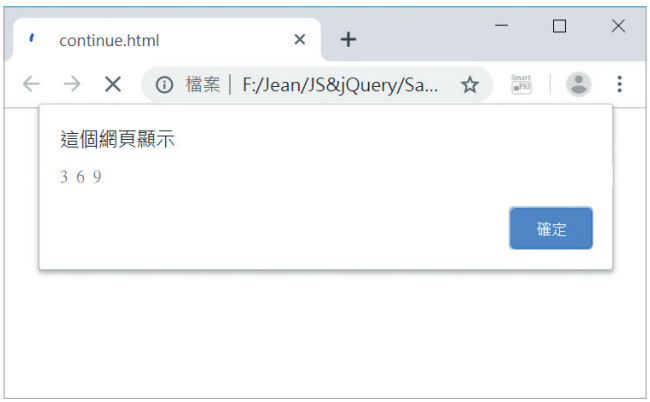
除了 break 指令，JavaScript 還提供了另一個經常使用於迴圈的 **continue** 指令，用來在迴圈內跳過後面的敘述，直接返回迴圈的開頭。

下面是一個例子，它會顯示 1~ 10 之間有哪些整數是 3 的倍數，因為在第 04 行檢查到變數 i 除以 3 的餘數不等於 0 時 (表示不是 3 的倍數)，就會執行第 05 行的 continue 指令，跳過第 07 行，直接返回迴圈的開頭，繼續檢查下一個變數 i，直到變數 i 大於 10 才會跳出迴圈，然後執行第 09 行顯示結果為 3、6、9。

\Ch03\continue.js

```
01  // 宣告用來存放結果的變數 str，初始值為空字串
02  var str = '';
03  for (var i = 1; i <= 10; i++) {
04    if ((i % 3) !== 0){
05      continue;
06    }
07    str = str + i + '\t';
08  }
09  window.alert(str);
```

若變數 i 除以 3 的餘數不等於 0，就返回迴圈的開頭，繼續檢查下一個變數 i



標記 (label)

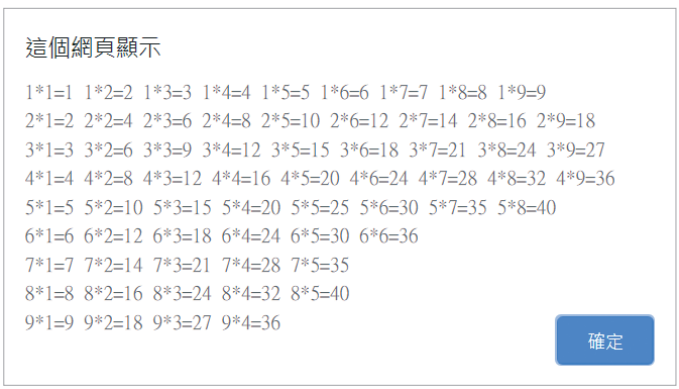
當我們在巢狀迴圈中使用 break 或 continue 指令時，預設會跳出或返回最內層的迴圈，若要設定跳出或返回整個巢狀迴圈，可以搭配**標記** (label)。

下面是一個例子，它改寫自第 3-4 節的九九乘法表，差別在於加入第 05 ~ 07 行，若第 05 行檢查到乘積大於 40，就執行第 06 行的 break 指令跳出內層迴圈，所以執行結果會顯示乘積小於等於 40 的九九乘法表。

\Ch03\nestedfor2.js

```
01  var str1 = '', str2 = '';
02  for (var i = 1; i <= 9; i++) {                  // 外層迴圈的開始
03    str1 = '';
04    for (var j = 1; j <= 9; j++) {                 // 內層迴圈的開始
05      if ((i * j) > 40){
06        break;
07      }
08      str1 = str1 + i + '*' + j + '=' + (i * j) + '\t';// '\t' 表示 [Tab] 鍵
09    }                                              // 內層迴圈的結尾
10    str2 = str2 + str1 + '\n';                     // '\n' 表示 [Enter] 鍵
11  }                                                // 外層迴圈的結尾
12  window.alert(str2);
```

若乘積大於 40，就跳出內層迴圈



若要改成一旦乘積大於 40，就停止顯示九九乘法表，該怎麼辦呢？此時，break 指令就不能只是跳出內層迴圈，而是要跳出整個巢狀迴圈，我們可以使用標記將程式改寫成如下，關鍵在於加入第 02 行，表示將外層迴圈標記為 outerloop，如此一來，若第 06 行檢查到乘積大於 40，就執行第 07 行的 break outerloop; 跳出標記的整個區塊 (即外層迴圈)，得到執行結果如下圖。

標記的命名規則和變數相同，只是後面要加上冒號。此外，標記不能單獨使用，必須搭配 break 或 continue 指令，而且要寫在同一行不能分行，不然會被當作兩個獨立的敘述，導致無法預期的結果。

\Ch03\nestedfor3.js

```
01  var str1 = '', str2 = '';
02  outerloop:
03  for (var i = 1; i <= 9; i++) {                    // 外層迴圈的開始
04    str1 = '';
05    for (var j = 1; j <= 9; j++) {                   // 內層迴圈的開始
06      if ((i * j) > 40){
07        break outerloop;
08      }
09      str1 = str1 + i + '*' + j + '=' + (i * j) + '\t';  // '\t' 表示 [Tab] 鍵
10    }                                                // 內層迴圈的結尾
11    str2 = str2 + str1 + '\n';                       // '\n' 表示 [Enter] 鍵
12  }                                                  // 外層迴圈的結尾
13  window.alert(str2);
```

若乘積大於 40，就跳出外層迴圈