Appendix

Drag and Drop API

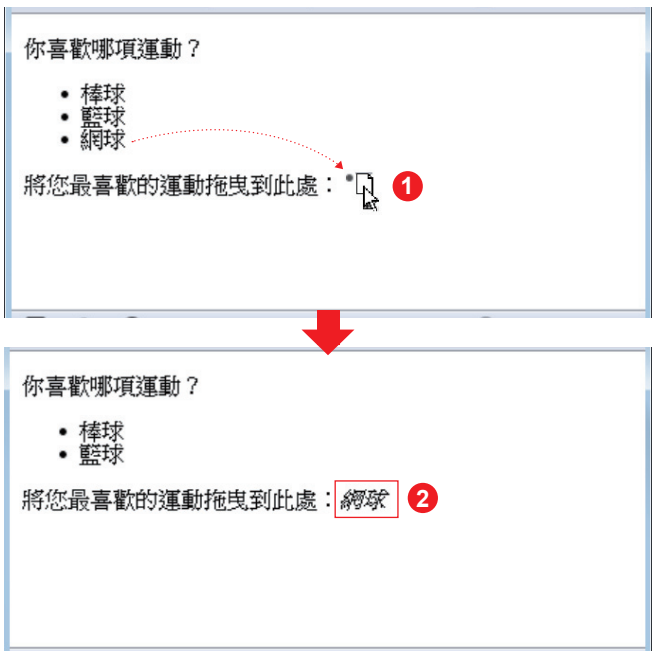
D

D-1 網頁元素的拖放操作

早在 HTML5 提供 Drag and Drop API 之前，Microsoft 公司就已經在 Internet Explorer 5 內建拖放功能的 API，而您或許也曾在一些網頁上體驗過拖放操作，例如將網頁上的某個區域拖放到另一個位置，只是這些拖放操作不一定是透過專用的 API，亦有可能是採用以事件為基礎的拖放機制 (event-based drag-and-drop mechanism)，也就是藉由諸如 mousedown、mousemove、mouseup 等一連串的滑鼠事件來達成的。

在 HTML5 將 Drag and Drop API 納入標準規格後，我們就可以將網頁上的元素拖放到其它位置，若再搭配其它 API，例如 File API，甚至可以讓使用者將檔案拖放到瀏覽器，類似的功能目前正在制定與測試中，未來會有更多瀏覽器支援。

現在，我們就以下面的例子 <\ 附錄 D\drag1.html > 示範如何在網頁上拖放元素。



1 將運動名稱從清單中拖放到 <p> 元素

2 運動名稱變成斜體並從清單中移除

這個例子主要包含下列三個步驟，以下各小節有詳細的說明：

1. 指定拖曳來源並處理拖曳開始事件 (設定來源元素的 draggable 屬性並撰寫 dragstart 事件處理程式)。
2. 指定放置目標並處理放置事件 (設定目標元素的 dropzone 屬性並撰寫 drop 事件處理程式)。
3. 處理拖曳結束事件 (撰寫 dragend 事件處理程式)。

D-1-1 指定拖曳來源並處理拖曳開始 (dragstart) 事件

首先，我們來說明拖放操作所涉及的兩個對象，第一個對象是拖曳來源，這有可能是網頁上的元素，也有可能是其它應用程式，比方說，這個例子的拖曳來源是位於網頁上方的項目，即 <li> 元素；而第二個對象是放置目標，這同樣有可能是網頁上的元素，也有可能是其它應用程式，比方說，這個例子的放置目標是位於網頁下方的段落，即 <p> 元素。

在預設的情況下，網頁上多數的元素是不接受拖曳的，若要將元素指定為拖曳來源，可以加上 draggable="true" 屬性，例如下面的敘述是令 <li> 元素接受拖曳：

```
<li draggable="true">棒球</li>
```

draggable 屬性的值有下列幾種：

- ❖ true：表示接受拖曳。
- ❖ false：表示不接受拖曳。
- ❖ 不指定：表示 auto，這取決於瀏覽器的實作，通常 <img> 元素或以 href 屬性指定的 <a> 元素預設接受拖曳，其它元素則預設不接受拖曳。

在指定拖曳來源後，我們還要針對 dragstart (拖曳開始) 事件撰寫處理程式，主要的任務就是將拖曳來源的資料放進 DataTransfer 物件，同時設定所允許的動作，例如複製 (copy)、搬移 (move)、連結 (link) 或其它組合。

根據前面的討論，我們可以將程式碼撰寫成如下。

```
01:<!doctype html>
02:<html>
03:  <head>
04:    <meta charset="utf-8">
05:    <title> 拖放操作 </title>
06:  </head>
07:  <body>
08:    <p> 你喜歡哪項運動？ </p>
09:    <ul ondragstart="dragStartHandler(event)">
10:      <li draggable="true" data-sport="baseball"> 棒球 </li>
11:      <li draggable="true" data-sport="baseketball"> 籃球 </li>
12:      <li draggable="true" data-sport="tennis"> 網球 </li>
13:    </ul>
14:
15:    <script>
16:      function dragStartHandler(event) {
17:        // 檢查產生 dragstart 事件的元素是否為 <li> 元素
18:        if (event.target instanceof HTMLLIElement) {
19:          // 將拖曳來源的資料（此例為 data-sport 屬性的值）放進 DataTransfer 物件
20:          event.dataTransfer.setData('text/plain', event.target.dataset.sport);
21:          // 設定允許搬移的動作
22:          event.dataTransfer.effectAllowed = 'move';
23:        } else {
24:          // 取消不是 <li> 元素的拖放操作
25:          event.preventDefault();
26:        }
27:      }
28:    </script>
29:  </body>
30:</html>
```

<\ 附錄 D\drag1.html>

❖ 09：在 <ul> 元素加入 ondragstart="dragStartHandler(event)" 屬性，表示項目符號清單一產生 dragstart 事件，就執行 dragStartHandler(event) 處理程式。

- 10、11、12：在每個 <li> 元素加入 draggable="true" 屬性，令 <li> 元素接受拖曳。請注意，我們同時在每個 <li> 元素加入 HTML 5 新增的全域屬性 data-*，用來自訂屬性，以傳送資訊給 JavaScript 程式，比方說，第 10 行有加入一個自訂屬性 data-sport，其值為 "baseball"，當我們要在 JavaScript 程式碼中存取該自訂屬性的值時，可以透過 element.dataset.sport 的形式，例如第 20 行的 event.target.dataset.sport 指的就是自訂屬性 data-sport 的值。

- 16～27：定義 dragStartHandler(event) 處理程式，其中：

  - 18：檢查產生 dragstart 事件的元素是否為 <li> 元素。

  - 20、22：若是 <li> 元素，就執行第 20 行，呼叫 DataTransfer 物件的 setData() 方法將拖曳來源的資料 (此例為 data-sport 屬性的值) 放進 DataTransfer 物件，然後執行第 22 行，利用 DataTransfer 物件的 effectAllowed 屬性將所允許的動作設定為 move (搬移)。有關 DataTransfer 物件的方法與屬性，第 20-3 節有進一步的介紹。

  - 25：若不是 <li> 元素，就執行第 25 行，取消與事件關聯的預設動作，即取消處理拖放操作。

此時的瀏覽結果如下圖， 元素雖然接受拖曳，但由於尚未設定放置目標，因此，無論將項目拖曳到哪裡，指標都會呈現禁止的圖示。

D-1-2 指定放置目標並處理放置 (drop) 事件

接下來，我們要指定放置目標並處理放置 (drop) 事件。在預設的情況下，網頁上的元素是不接受放置的，若要將元素指定為放置目標，可以使用 dropzone 屬性集合，該屬性集合的值有下列幾種，預設為 copy：

- copy：表示放置的動作會把被拖曳的資料複製到新的位置。
- move：表示放置的動作會把被拖曳的資料搬移到新的位置。
- link：表示放置的動作會連結到被拖曳的資料。

此外，我們還可以利用 dropzone 屬性集合指定資料的類型，例如：

- string:text/plain：表示純文字字串。
- file:image/png：表示 PNG 圖檔。
- file:image/gif：表示 GIF 圖檔。
- file:image/jpeg：表示 JPG 圖檔。

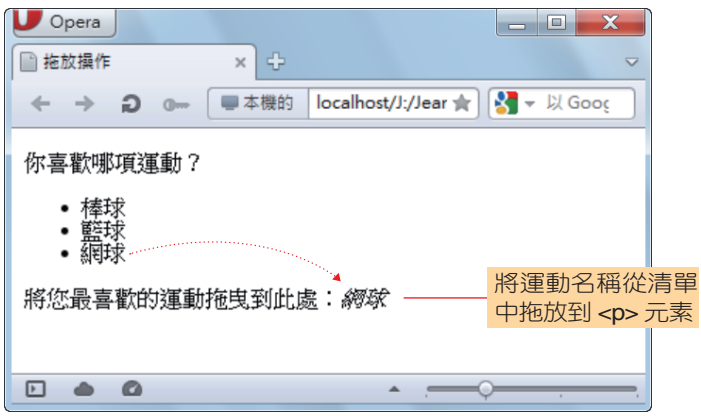
這麼一來，我們就可以透過類似 dropzone="move string:text/plain" 的寫法，來表示放置的動作會把被拖曳的純文字字串搬移到新的位置。

在指定放置目標後，我們還要針對 drop (放置) 事件撰寫處理程式，主要的任務就是從 DataTransfer 物件取出被拖曳的資料，然後做進一步的處理。

延續上一節的例子，我們可以在第 28 行的後面加入下列的程式碼，其中第一行敘述除了在 <p> 元素加上 dropzone="move string:text/plain" 屬性，將 <p> 元素指定為放置目標，還加上 ondrop="dropHandler(event)" 屬性，將 drop (放置) 事件的處理程式指定為 dropHandler(event)，然後在接下來的 JavaScript 程式碼中定義 dropHandler(event)，令它從 DataTransfer 物件取出被拖曳的資料，然後根據這些資料在放置目標顯示相對應的運動名稱。

```
<p dropzone="move string:text/plain" ondrop="dropHandler(event)">
  將您最喜歡的運動拖曳到此處：</p>
<script>
  function dropHandler(event) {
    var destination = document.createElement('i');
    // 從 DataTransfer 物件取出被拖曳的資料 （此例為 data-sport 屬性的值）
    var data = event.dataTransfer.getData('text/plain');
    // 根據從 DataTransfer 物件取出的資料顯示相對應的運動名稱
    if (data == 'baseball') {
      destination.innerHTML = '棒球';
    } else if (data == 'baseketball') {
      destination.innerHTML = '籃球';
    } else if (data == 'tennis') {
      destination.innerHTML = '網球';
    } else {
      destination.innerHTML = '其它運動';
    }
    event.target.appendChild(destination);
  }
</script>
```

此時的瀏覽結果如下圖，我們可以將最喜歡的運動名稱從清單中拖放到 <p> 元素，而且該運動名稱還會變成斜體。



D-1-3 處理拖曳結束 (dragend) 事件

在拖放操作的最後，我們要針對 dragend (拖曳結束) 事件撰寫處理程式，主要的任務就是處理一些拖曳結束的收尾工作，例如將被拖曳的元素從清單中移除。

延續上一節的例子，我們加入下列的程式碼 (以色字標示的部分)：

- 09：加入 ondragend="dragEndHandler(event)" 屬性，將 dragend (拖曳結束) 事件的處理程式指定為 dragEndHandler(event)。
- 29 ~ 32：定義 dragEndHandler(event) 處理程式，將被拖曳的元素從清單中移除。

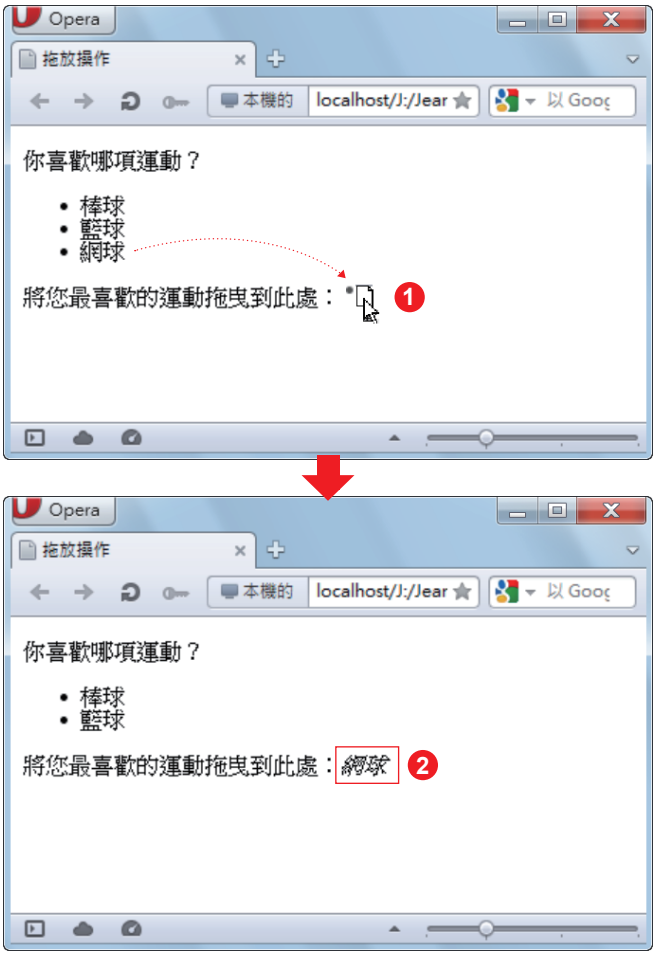
```
01:<!doctype html>
02:<html>
03:  <head>
04:    <meta charset="utf-8">
05:    <title>拖放操作</title>
06:  </head>
07:  <body>
08:    <p>你喜歡哪項運動？</p>
09:    <ul ondragstart="dragStartHandler(event)" ondragend="dragEndHandler(event)">
10:      <li draggable="true" data-sport="baseball">棒球</li>
11:      <li draggable="true" data-sport="baseketball">籃球</li>
12:      <li draggable="true" data-sport="tennis">網球</li>
13:    </ul>
14:
15:    <script>
16:      function dragStartHandler(event) {
17:        // 檢查產生 dragstart 事件的元素是否為 <li> 元素
18:        if (event.target instanceof HTMLLIElement) {
19:          // 將拖曳來源的資料 ( 此例為 data-sport 屬性的值 ) 放進 DataTransfer 物件
20:          event.dataTransfer.setData('text/plain', event.target.dataset.sport);
21:          // 設定允許搬移的動作
22:          event.dataTransfer.effectAllowed = 'move';
```

<\ 附錄 D\drag1.html>(下頁續 1/2)

```
23:         } else {
24:           // 取消不是 <li> 元素的拖放操作
25:           event.preventDefault();
26:         }
27:       }
28:
29:       function dragEndHandler(event) {
30:         // 移除被拖曳的元素
31:         event.target.parentNode.removeChild(event.target);
32:       }
33:     </script>
34:
35:     <p dropzone="move string:text/plain" ondrop="dropHandler(event)">
36:       將您最喜歡的運動拖曳到此處：</p>
37:
38:     <script>
39:       function dropHandler(event) {
40:         var destination = document.createElement('i');
41:         // 從 DataTransfer 物件取出被拖曳的資料 （此例為 data-sport 屬性的值）
42:         var data = event.dataTransfer.getData('text/plain');
43:         // 根據從 DataTransfer 物件取出的資料顯示相對應的運動名稱
44:         if (data == 'baseball') {
45:           destination.innerHTML = ' 棒球 ';
46:         } else if (data == 'baseketball') {
47:           destination.innerHTML = ' 籃球 ';
48:         } else if (data == 'tennis') {
49:           destination.innerHTML = ' 網球 ';
50:         } else {
51:           destination.innerHTML = ' 其它運動 ';
52:         }
53:         event.target.appendChild(destination);
54:       }
55:     </script>
56:   </body>
57:</html>
```

<\ 附錄 D\drag1.html>(接上頁 2/2)

此時的瀏覽結果如下圖，在我們將最喜歡的運動名稱從清單中拖放到 <p> 元素後，該運動名稱不僅會變成斜體，而且會被從清單中移除。



1 將運動名稱從清單中拖放到 <p> 元素

2 運動名稱變成斜體並從清單中移除

D-2 拖放操作相關的事件

拖放操作相關的事件有下列幾種：

- dragstart：這個事件的通知對象為拖曳來源元素，表示拖曳開始。
- drag：這個事件的通知對象為拖曳來源元素，表示拖曳中。
- dragenter：這個事件的通知對象為拖曳中指標行經的元素，表示拖曳進入元素的範圍。
- dragleave：這個事件的通知對象為拖曳中指標行經的元素，表示拖曳離開元素的範圍。
- dragover：這個事件的通知對象為拖曳中指標行經的元素，表示拖曳經過元素的範圍。
- drop：這個事件的通知對象為放置目標元素，表示放置時。
- dragend：這個事件的通知對象為拖曳來源元素，表示拖曳結束。

這些事件均定義在 DragEvent 介面，而 DragEvent 介面有一個重要的屬性－dataTransfer，該屬性可以用來存取與事件關聯的 DataTranfer 物件。DataTranfer 物件對於拖放操作扮演著極為重要的角色，拖曳來源元素的資料會被放進 DataTranfer 物件，而放置目標元素的資料則是從 DataTranfer 物件取出，下一節就為您介紹 DataTranfer 物件的屬性與方法。

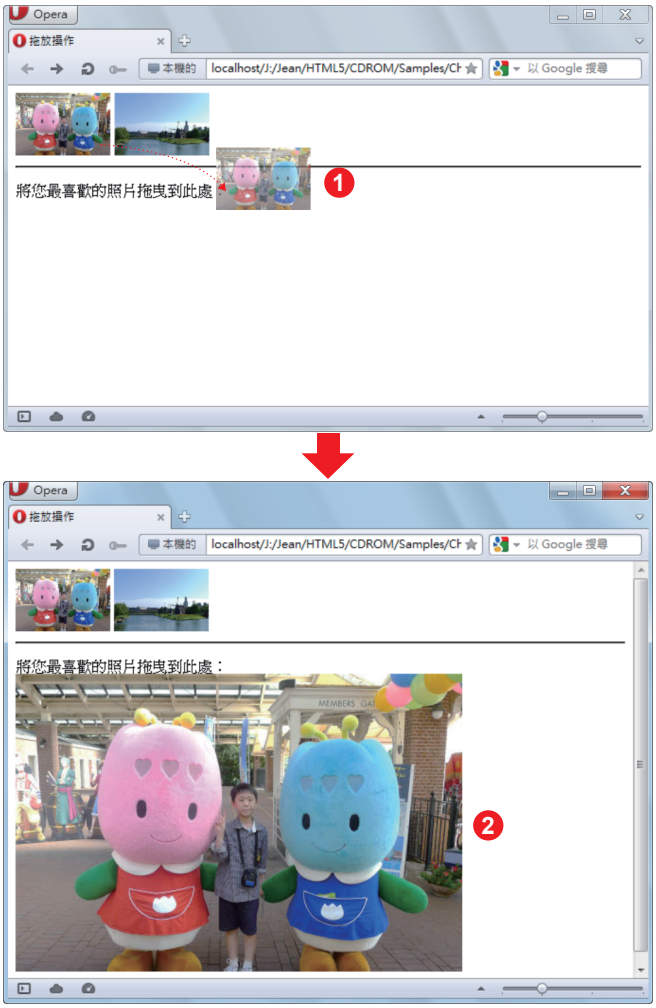
D-3 DataTransfer 物件的屬性與方法

為了增進您對拖放操作的瞭解，我們來介紹 DataTransfer 物件的屬性與方法：

- dropEffect [= *value*]：這個屬性用來設定或傳回拖放操作的類型，可能的值如下，其它類型則會操作失敗：
  - "none"
  - "copy"
  - "link"
  - "move"
- effectAllowed [= *value*]：這個屬性是在拖放操作的 dragenter 和 dragover 事件期間，用來初始化 dropEffect 屬性，在建立 DataTransfer 物件時，effectAllowed 屬性會被設定為 *value* 所指定的值，可能的值如下，其它值則會被忽略：
  - "none"
  - "copy"
  - "copyLink"
  - "copyMove"
  - "link"
  - "linkMove"
  - "move"
  - "all"
  - "uninitialized"

- items：這個屬性會傳回一個與 DataTransfer 物件關聯的 DataTransferItemList 物件。
- setDragImage(*element*, *x*, *y*)：這個方法會將參數 *element* 指定的圖像設定為拖曳中的回饋圖示，參數 *element* 通常是 <img> 元素，至於參數 *x*、*y* 則是用來調整顯示位置的 x、y 座標。
- addElement(*element*)：這個方法會將參數 *element* 指定的元素加入拖曳中的回饋圖示，它和 setDragImage() 方法的差別在於會自動根據目前加入的元素產生圖像，並在拖曳過程中更新圖像，舉例來說，假設參數 *element* 為 <video> 元素，那麼在拖曳過程中就會隨著影片的播放自動更新圖像，而 setDragImage() 方法是在被呼叫的當下就載入指定的圖像，不會在拖曳過程中更新圖像。
- setData(*format*, *data*)：這個方法會將參數 *data* 指定的資料放進 DataTransfer 物件，若參數 *format* 為 "text"，就變更為 "text/plain" 格式，若參數 *format* 為 "url"，就變更為 "text/uri-list" 格式。
- types：這個屬性會傳回在 dragstart 事件中所設定的格式，若有檔案被拖曳，那麼其中一個格式將是 "Files"。
- getData(*format*)：這個方法會從 DataTransfer 物件取出資料，若沒有資料，就傳回空字串。
- clearData([*format*])：這個方法會從 DataTransfer 物件移除參數 *format* 指定之格式的資料，若省略參數 *format*，就移除所有資料。
- files：這個屬性會傳回一個 FileList 物件，該物件為被拖曳的檔案集合。

下面是一個例子，用來示範除了文字之外，我們也可以拖曳網頁上的圖像，其瀏覽結果如下圖。



1 按住滑鼠左鍵將圖片拖曳到此處

2 放開滑鼠左鍵將圖片放置在此處

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>拖放操作</title>
  </head>
  <body>
    <img src="jp1.jpg" width="100">
    <img src="jp2.jpg" width="100">
    <hr>
    <p dropzone="move string:text/plain" ondrop="dropHandler(event)"
       ondragover="dragoverHandler(event)">
      將您最喜歡的照片拖曳到此處：</p>

    <script>
      //drop 事件處理程式 （用來從 DataTransfer 物件取出被拖曳的資料並顯示出來）
      function dropHandler(event) {
        // 建立一個 <img> 元素做為放置目標
        var destination = document.createElement('img');
        // 取出被拖曳的資料 （此例為圖像的路徑） 並指派給 <img> 元素的 src 屬性
        destination.src = event.dataTransfer.getData('text/plain');
        event.target.appendChild(destination);
      }

     //dragover 事件處理程式 （用來取消瀏覽器預設的動作）
      function dragoverHandler(event) {
        event.preventDefault();
      }
    </script>
  </body>
</html>
```

<\ 附錄 D\drag2.html>