jQuery 事件處理 (Events)

jQuery 的事件處理函式大都提供兩種用途，一種是呼叫**帶有參數**的函式 – **綁定**事件處理函式；另一種則是呼叫**不帶有參數**的函式 – **觸發**該事件。

帶有參數：(例. 綁定所有段落觸發 click 事件時，將背景顏色改為藍色。)

```
$("p").click(function() {
  $(this).css("background-color","blue");
});
```

不帶有參數：(例. 觸發所有段落的 click 事件)

```
$("p").click();
```

事件處理函數中的「this」為被觸發的「DOM元素」。(非jQuery物件)

上述的程式碼，我們用到 jQuery 定義好的 click 函式來處理 click event，然而 jQuery 也同樣對 DOM 其它的 event 都有相關的函式，如以下的 jQuery 事件函式也是同樣的使用方法：

事件	觸發條件
(on) blur	物件失去焦點時
(on) change	物件內容改變時
(on) click	滑鼠點擊物件時
(on) dblclick	滑鼠連點二下物件時
(on) error	當圖片或文件下載產生錯誤時
(on) focus	當物件被點擊或取得焦點時
(on) keydown	按下鍵盤按鍵時
(on) keypress	按下並放開鍵盤按鍵後
(on) onkeyup	按下並放開鍵盤按鍵時
(on) onload	網頁或圖片完成下載時
(on) mousedown	按下滑鼠按鍵時
(on) mousemove	介於over跟out間的滑鼠移動行為
(on) mouseout	滑鼠離開某物件四周時
(on) mouseover	滑鼠離開某物件四周時
(on) mouseup	放開滑鼠按鍵時
(on) resize	當視窗或框架大小被改變時
(on) scroll	當捲軸被拉動時
(on) select	當文字被選取時
(on) submit	當按下送出按紐時
(on) unload	當使用者關閉網頁時

jQuery 的 event object

對於所有的 jQuery event handler，你都能傳入一個參數作為 event 物件，如下例：

```
$(document).click(function(event) {
  alert(event.pageX);
});
```

上面這個例子是用來取得滑鼠游標相對於頁面的位置，**重點是！這段程式碼在IE上也能執行** (註：原本的IE瀏覽器事件無pageX屬性)。WHY？因為 jQuery 又事先幫你解決掉事件處理中跨瀏覽器的問題了 – 「**jQuery 修改了 event object 以使其符合 W3C 的標準**」，所以從此你不用再重複 if(window.event)，你只需了解並使用標準的事件屬性！

jQuery 其它幾個一般性的事件處理函式

hover(fn1, fn2)

hover = mouseover + mouseout。當滑鼠移動到一個匹配的元素上面時，會觸發第一個函數 (fn1)；當滑鼠移出該元素時，會觸發第二個函數 (fn2)。

toggle(fn1, fn2, [fn3,fn4,...])

綁定元素每當觸發「click 事件」時輪流切換執行函數。如點第 1 次執行fn1；點第 2 次執行 fn2.....。

例：

```
$("something").toggle(
  function () {
    $(this).css(......);
  },
  function () {
    $(this).css(......);
  },
  function () {
    $(this).css(......);
  }
);
```

bind(eventType, fn)、unbind(eventType)

事件的綁定與移除。bind 及 unbind 還可以讓我們達到**自訂事件**處理的功能：

```
$('#sth').bind('myEvent', doSomething);
```

如上，我們自訂一個叫作 myEvent 的事件，但這不是 DOM 標準事件啊，怎麼觸發它？答案是用接著會談到的「trigger」函式來觸發 myEvent。

trigger(eventType, [data])

觸發事件，其中 data 為要傳給事件處理函式的參數 (一個陣列)。

one(eventType, [data], fn)

如果只是觸發「一次」事件，就使用 one 函式來作 bind 的動作，當該事件被觸發一次之後就會自己自動 unbind。

```
Namespacing events - 替事件命名
我們雖然可以用unbind來移除事件，但是它會一次移除全部綁定的事件處理耶，何解?
答案是利用jQuery提供的Namespacing events！看個例子就明白它是什麼：
$('.class').bind('click.namespace', function(){}); //綁定
$('.class').trigger('click.namespace'); //觸發
$('.class').unbind('click.namespace'); //移除
Namespacing events 簡單的說就是幫事件取個名字(namespace)，了解吧 :)
```

最後，但是很重要也很常用的

$(document).ready(function(){})

jQuery 中，大部分的操作都基於 HTML DOM，所以我們必須確定頁面文件已經完全下載好才開始執行你的程式，jQuery 提供下面這個函式來處理 DOM ready 事件 (DOMContentLoaded)：

```
$(document).ready(function() {
  // 這裡放你要執行的程式碼
});
```

你也可以這樣寫：

```
$(function() {

  // 這裡放你要執行的程式碼
});
```

> jQuery的DOM ready event是等HTML DOM準備好就可以開始執行程式，不像一般常用的window.onload要連圖片、外部檔案等全部都下載完畢才會觸發onload事件。

DOMContentLoaded & 外部樣式表的問題？

如上述，DOMContentLoaded 事件的觸發不必等到圖片檔案等下載完畢，但這裡有個問題就是，那需要等到「外部樣式表 (External CSS Styles)」下載完畢嗎？關於這一點，各家瀏覽器對於 DOMContentLoaded 事件的觸發也的確不一致，在 Opera 中瀏覽器不會等外部樣式表下載完即觸發 DOM ready；而 Firefox 則會等到外部 CSS 都加載完畢。因此若你有在 jQuery ready 事件中改變了元素的 CSS 屬性，對於 Opera，你修改的屬性很可能會被接著加載的外部樣式表覆蓋過去，這也就造成跨瀏覽器處理的問題。但如果你無法避免在頁面初始階段改變 CSS 屬性，這問題我們可以利用 load 事件來避開，以確定事件的觸發會在外部樣式表下載完畢之後：

```
$(element).load(fn);
```