Event 事件

Event 事件

- 事件 (Events) 是指瀏覽器根據所偵測到使用者特定的動作而觸發特定的行為
- 常見的事件 e.g.
  - click, dblclick：點選某一個物件時
  - mousedown/mouseup：按下/鬆開 滑鼠按鈕時
  - keypress, keyup, keydown 敲鍵盤
  - change：改變物件選項或內容時
  - load/unload：瀏覽器 載入/離開 網頁時
  - …

Event Handler 事件處理程式

- 事件發生時，負責處理的相對程式碼稱為 Event Handlers (事件處理程式)，又稱為 Callback
- 定義 Event Handler 的方式
  - 定義在 HTML 標籤的屬性上
    - Event Handler 的函數名稱會以 "on" 開頭，e.g. click 事件的 Event Handler 就叫 onclick, load 事件的 Event Handler 就叫 onload，e.g. eventFirst.html
  - 使用 DOM Level Event Handler 的專用定義函數
    - e.g. 定義某個按鈕的 click Event Handler
      btn.addEventListener("click", sayHello);

DOM Level 2 Event Binding

- addEventListener()
  - e.g. btn.addEventListener('click', doHandler, false);
  - 同一 event 可登記多個 Event Handler
  - 第三個參數表示 event flow (稍後詳細解釋)
    - true: capture phase
    - false: bubbling phase (default)
- deleteEventListener()

- note: (IE5+) Event Binding
  - attachEvent()
  - detachEvent()

Event Binding - Crossing Browser

- e.g.

```
if (window.addEventListener)
    // DOM method for binding an event
    window.addEventListener("load", doHandler, false);
else if (window.attachEvent)
    // IE-exclusive method for binding an event
    window.attachEvent("onload", doHandler);
else
    // support older browsers
    window.onload=doHandler;
```

Event Object

- When an event is fired, all of the relevant information is gathered and stored on an object called event
- In IE, the event object is a global variable
  - e.g. ieEventObject.html
- In DOM, the event object is passed in as the sole argument to an event handler
  - In DOM, the event object is a local variable, so it must be passed in as a argument.
- For compatibility, most of modern browser implement both

Event Object - Crossing Browser

- crossing browser event handler pattern for event object handling, e.g.

```
function eventHandler(e) {
    e = e || window.event;
    …
}
```

Events

- Mouse Event
  - mousedown/mouseup：push/release mouse button
  - mousemove/mouseover/mouseout
    - mouseover：滑鼠移動經過某個物件時
  - click, dblclick
- Keyboard Event
  - keydown/keypress/keyup 敲鍵盤

Mouse Events

滑鼠事件	說明
mousedown	按下滑鼠按鍵
mouseup	釋放滑鼠按鍵
mouseover	移動滑鼠游標
mouseout	將滑鼠游標移出一個物件
click	單擊滑鼠按鍵
dblclick	雙擊滑鼠按鍵

Mouse Event Property

Property	說明
button	按下的滑鼠按鍵代碼
clientX	滑鼠游標相對於視窗的 X 座標
clientY	滑鼠游標相對於視窗的 Y 座標
offsetX	滑鼠游標相對於發送事件之物件的 X 座標
offsetY	滑鼠游標相對於發送事件之物件的 Y 座標

偵測滑鼠鍵

- e.g. mouseEvent.html

```
switch (e.button){
    case 0: alert("你用滑鼠左鍵！"); break;
    case 1: alert("你用滑鼠中鍵！"); break;
    case 2: alert("你用滑鼠右鍵！"); break;
    default: alert("未知的滑鼠鍵！");
}
```

  - 使用 mousedown 事件，當滑鼠點下按鈕，使用 e.button or event.button 偵測滑鼠按鍵

判斷不同滑鼠按鍵

DOM event.button 的值	說明
0	滑鼠左鍵被按下
1	滑鼠中鍵被按下
2	滑鼠右鍵被按下

IE event.button 的值	說明
1	滑鼠左鍵被按下
4	滑鼠中鍵被按下
2	滑鼠右鍵被按下

clientX, clientY

- event.clientX 及 event.clientY 分別代表滑鼠相對應於網頁視窗的 X 和 Y 座標
- e.g. 黏住游標的方塊 mouseCursor.html

```
<body onmousemove="newCursor()">
…
redSquare.style.posLeft=event.clientX-10;
redSquare.style.posTop=event.clientY-10;
```

- 說明
  - 使用 div 建立一紅色方塊(id=redSquare) ，使用 onmousemove 事件，當滑鼠移動同時移動方塊。

Keyboard Events 1/2

鍵盤事件	說明
keydown	按下鍵盤按鍵
keypress	輸入可見文字時
keyup	釋放鍵盤按鍵

Keyboard Events 2/2

- for character key
  - The **keydown** event occurs when the key is pressed, followed immediately by the **keypress** event.
    Then the **keyup** event is generated when the key is released.
  - if a key is held down, **keydown** & **keypress** are fired repeatedly
- for non-character key
  - **keydown** event followed by **keyup** event
  - if a key is held down, **keydown** is fired repeatedly
- e.g. keyboard.html

Keyboard Event Property

Property	說明
code	key name
key	key character
charCode ✗	keypress: character unicode
which/keyCode ✗	keypress: character unicode keydown/keyup: key code
shiftKey	SHIFT 鍵被按下時，值為 true
ctrlKey	CTRL 鍵被按下時，值為 true
altKey	ALT 鍵被按下時，值為 true

[註] Keyboard event properties are all READ-only.

https://www.w3.org/TR/uievents/

Example 偵測特定按鍵是否被按下

- 可利用 keyCode, which, charCode 得知按鍵代碼，並進行相關的處理。
- e.g. keyboardCode.html
  - keydown 事件執行程式 keyDownHandler()
  - keypress 事件執行程式 keyPressHandler()
- e.g. keyboardFunny.html
  - 利用 keypress 強制將輸入的小寫字母改為大寫

Modifier Keys 偵測複合鍵

- e.g. keyboardMeta.html

```
if ((event.shiftKey) && (event.keyCode!=16))
   alert("Shift + "+event.keyCode);
```

- 說明
  - 檢查 event.shiftKey 的值是否為 true，來確認 Shift 鍵是否被按下
  - 若不加第二個條件，會印出兩次警告視窗 ， 一次是按 Shift，第二次是所按字元鍵
  - CTRL 與 ALT 做法皆同於 SHIFT

Example -使用方向鍵來移動物件

- e.g. keyboardSquare.html 使用方向鍵來移動物件
- 程式碼重點

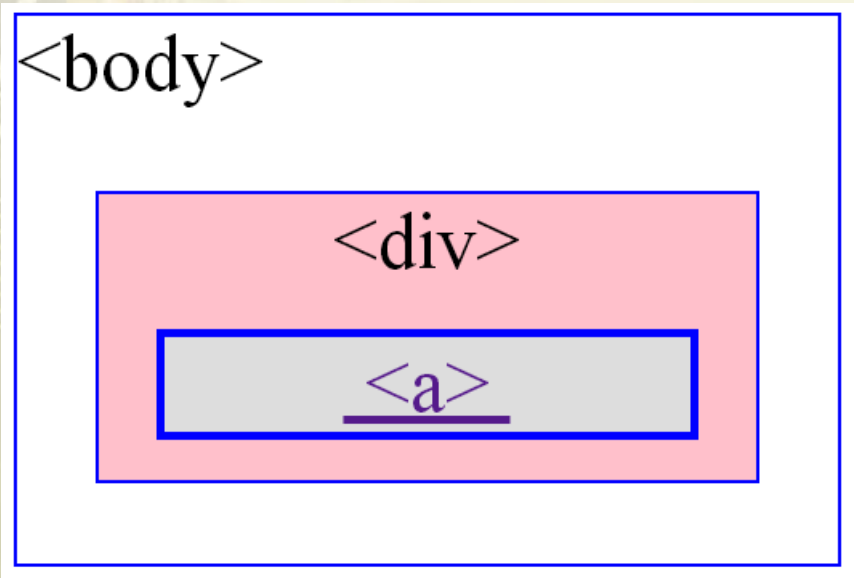
```
function followMe(e){
        e = e || window.event;
        ek = e.keyCode;
        if (ek==37) myArea.style.left = parseInt(myArea.style.left) - 5;
        if (ek==39) myArea.style.left = parseInt(myArea.style.left) + 5;
        if (ek==38) myArea.style.top  = parseInt(myArea.style.top) - 5;
        if (ek==40) myArea.style.top  = parseInt(myArea.style.top) + 5;
 }
```

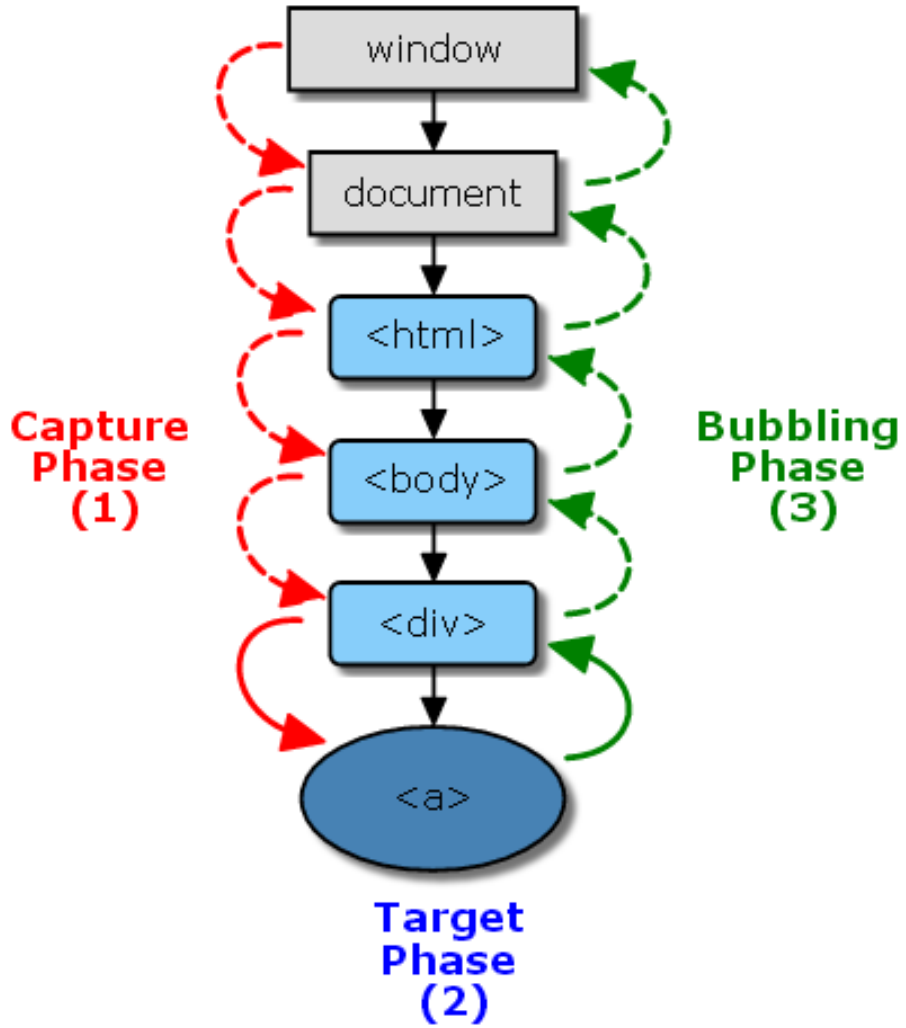
- 說明
  - 使用 event.keyCode 判定使用者所按的方向鍵，並移動綠色的 div 方塊 (id=myArea)。

Event Flow

Event Flow 1/3

- 接收到 Event 後的程式執行順序，可分三個 phase
  - capture phase (NN)
  - target phase
  - bubbling phase (IE)





Event Flow 2/3

- DOM method for binding an event

  addEventListener(*event*, *Handler*, *true/false*);

- The 3rd argument indicate phase. The phase says in what order the event should be detected.
  - **False** means the bubbling phase, and is usually the one you want to use.
  - **True** means the capture phase.
- Any event handlers added using the traditional techniques are treated as a bubbling event listener.

Event Flow 3/3

- You can add as many listeners for the same event on the same element as you want
  - e.g. eventFlow.html
  - but note that there is no way to guarantee in what order they will be run.
- it is not possible to obtain a list of the event handler functions

中斷 Event Flow

- 有二種中斷 Event Flow 的方法
- 設定 event.cancelBubble 屬性為 true 中斷上溯
  - event.cancelBubble = true;
  - 只能中斷 bubbling phase
- 使用 event.stopPropagation() 中斷 flow
  - capture phase, bubbling phase 皆可

Prevent Default

- use preventDefault() to prevent the default action of a particular event
- any event that can be canceled using preventDefault() will have its cancelable property set to true
- e.g. cancel the link's default action (eventFlow.html)

```
function eventHandler(e) {
    e = e || window.event;
    if (e.cancelable) e.preventDefault();
}
```

取消滑鼠右鍵功能

- 在 oncontextmenu event handler 中執行 preventDefault()
- e.g. disableRightButton.html

```
if (e.button==2) {
        e.preventDefault();
}
```

  - 檢查 e.button 是否為滑鼠右鍵，是就執行 e.preventDefault() 取消預設的功能。

Form with JavaScript

大綱

- 本部分將介紹表單及其相關控制項的性質、方法，並說明如何建立動態選單、如何進行表單資料驗證。
- 主題
  - 表單及其屬性與性質
  - 表單元素及其相關物件
  - 動態下拉式選單
  - 簡易表單資料驗證

表單標籤與物件

- 使用 JavaScript 時，表單被視為一個物件
  - 此物件有各種性質 (property) 和方法 (method)
- 表單標籤的屬性 (attribute) 和表單物件的性質，幾乎有著一對一的關係。
  - 屬性用於 HTML，性質用在 JavaScript 之中。
- 表單物件也有一些性質無法和表單標籤對應。
  - 例如：elements (由表單元素所形成的陣列) 和 length (表單元素的個數)

Form 屬性與性質對應列表 1/3

表單標籤屬性	說明	對應的表單物件性質及說明
name	表單的名稱，可被 JavaScript 或 VBScript 用以存取表單及其相關物件。	我們可用 document.formName 或 document.forms["formName"] 取得此表單物件。
id	表單的名稱，可被 JavaScript 或 VBScript 用以存取表單及其相關物件。	使用 id 時，可以不必經 DOM 的階層式架構取得表單物件，可直接使用 formId 或 document.all["formId"] 取得此表單物件。id 的值在整個網頁必須是唯一。
target	伺服器回傳資訊必須出現的位置，可以是一個視窗 、框架 或是 _top、_parent、_self、_blank。若無此屬性，則回傳結果將出現於原視窗。	對應的性質是 target，例如： formId.target = _blank document.formName.target = _blank

Form 屬性與性質對應列表 2/3

表單標籤屬性	說明	對應的表單物件性質及說明
action	指定伺服器端的處理程式。此處理程式可位於網路上的任一台伺服器，也可使用 mailto:xxx@xxx.xxx 將表單經電子郵件寄出。	對應的性質是 action，例如： formId.action = example/formact.asp document.formName.action = example/formact.asp
method	•指定資料傳送的方式，可有兩種方式： get: 表單資料經由 QUERY_STRING 的環境變數送至伺服器，這是預設的方式，但傳送資料量有限，通常只限於 1 KB 左右。 •post: 表單資料經由 standard input 傳送，資料長度儲存於 CONTENT_LENGTH 的環境變數，傳送資料可大於 1 MB。	對應的性質是 method，例如： formId.method = post document.formName.method = post

Form 屬性與性質對應列表 3/3

表單標籤屬性	說明	對應的表單物件性質及說明
enctype	指定 MIME 的編碼方式來傳送資料，可以有兩種值："application/x-www-form-urlencoded" (預設值) 或 "multipart/form-data"。	對應的性質是 encoding，例如：formId.encoding = application/x-www-form-urlencoded document.formName.encoding = application/x-www-form-urlencoded
onSubmit	JavaScript 或 VBScript 的事件處理器，若回傳值為 false，則表單將不會被送出。此屬性通常用來檢查使用者所填入的表單資料是否正確，若不正確，就不將表單資料送到伺服器。	對應的性質是 onSubmit。

表單的命名

- HTML 中包含數個表單，最好取用不同的名稱使 JavaScript 根據名稱存取不同表單
- e.g. 列出表單所有性質 formProperty.html
  - 程式碼重點

```
for (var prop in document.formName)
    document.writeln('document.formName. '+prop+' = '+document.formName[prop]+'<br>');
```

- 預設(當不設定表單的 name 或 id 屬性)
  - 可用 document.forms[0], document.forms[1], document.forms[2]取得第０、１、２個表單，以此類推

Example - Text 1/2

- 主題：使用 this.form 傳送兩個文字欄位的訊息
  - 檔名： formControlText.html
  - 程式碼重點

```
<input type="button" value="======>"
onClick="this.form.text2.value=this.form.text1.value">
```

- 說明
  - this.form 就是代表按鈕所在的表單
  - 一般而言，以「a.b」的方式來指到一個物件，例如 form1.input1 等，是由大（表單）到小（控制項）的方式，但唯一的例外，就是 this.form，這是由小（控制項）到大（表單）的方式。

Example - Text 2/2

- 主題：使用 onChange/onKeyPress/onKeyUp 事件將文字欄位送到狀態列
- 說明
  - 設定 window.status 文字的動作，可定義於文字欄位的 onChange 屬性，可以省掉按鈕的使用(此事件在文字欄位失去滑鼠焦點時才起作用）
  - 如果希望在文字欄位填入文字時，狀態列能夠立即改變，可將 onChange 改成 onKeyPress 即可。

Note

- Note: The window.status property does not work in the default configuration of IE, Firefox, Chrome, Safari or Opera 15 and newer.
- To allow scripts to change the text of the status, the user must set the dom.disable_window_status_change preference to false in the about:config screen. (or in Firefox: "Tools - Options - Content -Enable JavaScript / Advanced - Allow scripts to change status bar text").
- There's no status bar in Chrome.

Example - checkbox, radio

- 主題：
  - 檢查 checkbox,radio 的 checked 性質，判斷是否勾選
  - 使用 this.form 擷取表單
- 檔名：formControlCheckbox.html
- 說明
  - checked 的值，true 代表勾選，false 代表不勾選
  - 如果不是用 this.form ，程式碼比較繁雜
  - radio 與 checkbox 在操作上大同小異

Example - Select - Single

- 主題：單選的下拉式選單
  - 檔名： formControlSelectSingle.html
  - 程式碼重點

```
<select name=courseList size=4 onChange="alert('你選的課程是
「'+this.options[this.selectedIndex].text+'」')">
<option> 1. Linear Algebra
  …
</select>
```

- 說明
  - 下拉式選單不是使用input標籤，而是用select和option
  - options[n]代表select中第幾個option標籤
  - selectedIndex代表所選取option標籤的index值

Example - Select - Multiple

- 主題：多選的下拉式選單
  - 檔名: SingleformSelectMultiple.html
  - 程式碼重點

    <select name=courseList size=4 multiple onChange="listSummary(this.form)">
- 說明
  - 從單選變多選只要加入multiple 屬性。
  - listSummary()函式中，我們在 result 變數之前加上了 var，這代表 result 是一個區域變數(Local Variable)，如果沒有，這個變數就預設成全域變數(Global Variable)，可以在函式以外的任何地方存取此變數。
  - 一般我們建議在函式內的變數都盡量設定成局部變數，以減少函式呼叫後可能產生的非預期副作用。

Example - formControlTextarea.html

- 主題：多列文字欄位(textarea)
  - 程式碼重點

    <textarea name=courseList cols=80 rows=10
    onChange="alert('更改後的文字：\n'+this.value)">
    This is the text within the textarea tag. 這是位於
    textarea 標籤內的文字。 </textarea>
- 說明

控制元件一覽表

- 檔名： formControls.html
- 說明
  - 例中用不同方式存取控制元件的 Property，請多參考
  - 點選表中連結可觀看各元件的 Property

動態下拉式選單

以 JavaScript 動態改變 select 標籤的選單內容，以產生動態的下拉式選單

動態下拉式選單

- 主題：動態地增加或刪除選項
  - 檔名: dynamicListBox01.html
  - 程式碼重點

    list.options[index]=null;

    list.options[index]=new Option(text, value);
- 說明
  - 用 list.options[i]=null 直接刪除第 i 個選項。
  - 用 list.options[index]=new Option(text, value)增加選項。
  - 範例中使用 id 而不用 name 代表表單控制項，好處是可直接用 id 來使用表單元素，不必經由文件物件模型的階層結構由上到下、一層一層指定。

選單基本性質

- list.options：此選單的選項，是一個陣列，其中每個元素是代表選項的物件。
- list.options.length：選項的個數。
- list.options[i].text：第 i 個選項的文字。
- list.options[i].value：第 i 個選項的值。
- list.options.selectedIndex：反白選項的索引值。若無反白選項，則此變數值為 -1。（當反白選項存在時，list.options[list.options.selectedIndex].text 就是反白選項的文字。）

刪除與增加選單的選項

- 刪除做法：
  - 使用 list.options[i]=null 直接刪除第 i 個選項
  - 使用 list.options.length=n 可將選項個數設定為 n，其餘多的選項將會被刪除。
- 增加做法
  - list.options[i]=new Option(text, value)
    - 其中 text 和 value 分別代表新選項的文字和值。
    - 如0<=i<list.options.length，原先第 i 個選項將被新選項取代
    - 如i = list.options.length，將會產生一個新的選項

動態下拉式選單-兩框連動

- 主題：動態下拉式選單-兩框連動（兩層樹狀選項）
  - 連結：dynamicListBox02.html
  - 程式碼重點

```
function renew(index){
    for(var i=0;i<department[index].length;i++)
    document.myForm.member.options[i]=new Option(
                                department[index][i],
department[index][i]);
    document.myForm.member.length=department[index].length;
}
onChange="renew(this.selectedIndex);"
```

- 說明
  - 利用onChange及時改變第二個選單的內容。
  - 範例中用二維陣列，方便存取相關資料。