
# JavaScript 程式設計新手村

## 單元18 - 事件處理 Event Handle

@kdchang

# Outline

1. 事件處理基礎概念
2. 事件處理過程

# 事件處理基礎概念

- 事件處理（Event Handlers）是 JavaScript 非常重要的功能，事件是用來處理 JavaScript 與 HTML 之間的互動、建立動畫效果並和使用者互動
- 事件處理簡單說就是當一個事件發生時（網頁載入、按下右鍵等），程式會相對應做出怎樣的處理
- 例如：當使用者按下按鈕時會觸發 click 的事件（事件發生）並讓按鈕變成紅色（處理事件），這就是一種事件處理機制

# 建立事件處理

- 事件種類（Event Type）
  又稱事件名稱 (Event Name)，為一個字串，說明發生了什麼事件，例如：click (點擊)、mousemove (滑鼠滑過)
- 事件處理（Event Handlers）
  係指處理事件的函數名稱，當事件發生時要呼叫哪個函數進行處理

```
<button onclick="show()">Click Me!</button>
```

# 基本事件處理

```
<script>
 function changetext(id) { id.innerHTML="Ooops!"; }
</script> // 改變<h1>文字內容
<h1 onclick="changetext(this)">Click on this text!</h1>
//注意這邊的this指的是被點擊的元素
```

# 事件處理過程

當事件發生時，整個 DOM 架構中的節點都有機會來處理此事件，不同瀏覽器的事件處理模型上並不相同，因此有不同的處理過程，我們稱為『事件傳播(Event Propagation)』，它會影響HTML元素處理事件的順序

一般可分為：

1. 事件補抓模型 (Event Capturing)
2. 事件氣泡模型 (Event Bubbling)

# 建立事件處理

基本上JS建立事件處理有三種方式

1. HTML 屬性：在 HTML 屬性中指定 JavaScript 事件處理
2. JavaScript 屬性：在物件屬性中設定 JavaScript 事件處理
3. 呼叫DOM方法：使用addEventListener() 方法處理事件

# HTML 屬性指定事件

只要在屬性中指定事件處理即可以使用 (事件前面加上 on)，但當應用程式越來越複雜時不好管理，不建議使用

```
<h1 onclick="show();"> Hi JS </h1>
```

# JavaScript 屬性指定事件

透過在選取的 DOM 元素增加事件屬性，但不方便管理

```
var btn = document.getElementById('id');
btn.onclick = eventHandler;
```

# 呼叫 DOM 方法建立事件處理

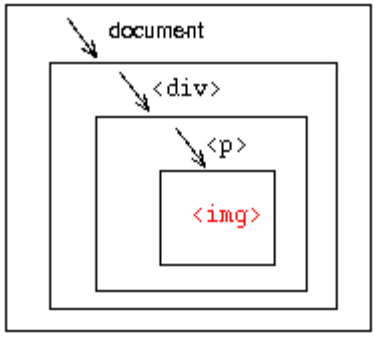
- 新增監聽函數(第三個參數false為使用氣泡事件)
  [選元素].addEventListener("event", Handler, 是否氣泡處理);
- 刪除監聽函數
  [選元素].removeEventListener("event", Handler, 是否氣泡);

( * 註冊監聽事件為建議寫法，同一事件可以註冊多個處理器，第三個參數 `flase` 為 `氣泡處理` )

# 練習一

使用三種監聽事件方式來監聽按鈕，當按鈕按兩下時，讓原來黑色 h1 文字變成紅色達到變換 CSS 效果

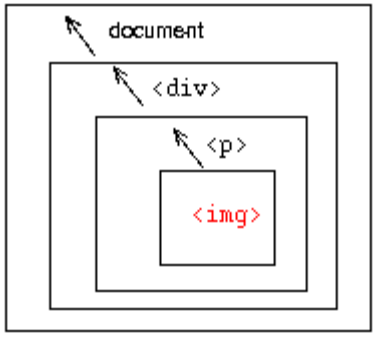
# 事件捕捉



# 事件捕捉

當事件發生時，首先是由Document物件開始(目前各瀏覽器實作上是由Window物件開始)，然後是最大範圍的元素有機會回應此事件，再來是下一層的，依序往下

# 氣泡事件




原本是 W3C 順序是先執行事件捕捉再執行氣泡處理，但就瀏覽器實作不多，目前實務上多用氣泡事件處理

範例程式

# 練習二

運用事件處理和新增刪除DOM的技巧，製作出一個可以新增刪除的代辦事項清單

Hint : click 事件、this、createElement() 和 appendChild()

# 事件物件

當事件觸發時會產生一個事件物件(Event Object)，這個物件記錄事件觸發的一些資料並傳入處理器方便做進一步處理，例如：位置、標籤、事件名稱

```
<button onclick="show();">Clik</button>
```

```
function show(event) {
  console.log(event.target);
}
```

# 動態事件註冊

除了網頁中現存的元素外，事件亦可以註冊在動態產生的元素上

```
<ul id="num"></ul>
```

```
const list = document.getElementById('num');

for(var i = 0; i < 5; i++){
  var option = document.createElement('LI');
  option.setAttribute('class', 'style');
  option.addEventListener('click', hand, false);

  option.innerHTML = i;
  list.appendChild(option);
}

function hand() {
  var dad = this.parentNode;
  dad.removeChild(this);
}
```

範例程式

# 事件處理的 this

事件處理 callback function 的 this 指向觸發事件的元素

```
<button id="btn">I love JS</button>
```

```
var head = document.getElementById('btn');

head.addEventListener('click', eventHandler, false);

function eventHandler(){
  console.log(this.innerHTML);
  this.style.color = 'red';
}
```

範例程式

# 事件物件屬性

- `currentTarget` 可以取得事件處理的元素物件
- `target` 觸發事件的元素
- `type` 觸發事件的名稱
- `clientX`, `clientY` 取得滑鼠點擊時游標位置

# 取消預設行為

有些 HTML 在觸發事件後會有預設行為，例如超連結會打開視窗，連到新網站

透過 `event.preventDefault()` 這個方法可以取消事件的預設行為

例如：按下超連結預設行為為連到新網頁

# 取消事件傳遞

`event.stopPropagation()` 可以取消事件捕捉或是氣泡事件傳遞

若使用 `return false;` ，則既取消事件傳遞也取消預設行為

# window 事件

window 物件支援數個事件，並於特定狀態下觸發

- `load` 網頁完全下載完成後觸發(含外部資源)
- `unload` 離開網頁觸發
- `DOMContentLoaded` 網頁文件(DOM)載入完成(不含其他外部資源)，注意和load的比較
- `resize` 調整視窗大小
- `scroll` 捲動視窗

# 滑鼠事件

- 滑鼠按一下：相關的「事件」有 `mousedown`，`mouseup`，`click`
- 滑鼠連按二下 ：相關的「事件」有 `mousedown`，`mouseup`，`click`，`dblclick`
- 滑鼠移動：相關的事件為 `mouseover`，`mousemove`，`mouseout`
- 滑鼠拖曳：相關的事件為 `mousedown` ，`dragstart`，`dragend`

範例程式

# 鍵盤事件

- `keydown` 按下鍵盤
- `keypress` 在按下與放開之間
- `keyup` 放開鍵盤按鍵

可以用 `event.keyCode` 取得鍵入的Unicode 對應值

範例程式

# HTML5：拖曳事件

- `dragstart` 開始拖曳操作時被觸發
- `dragover` 拖曳經過目標元素時被觸發
- `drop` 拖曳並且置放至目標元素時被觸發

# 練習三

使用事件處理的機制，設計一個有輸入框、送出鈕，可以計算還剩多少可以輸入字元的網頁 (類似 Twitter 只能輸入140字元)。當使用按送出後，輸入文字會出現在網頁上

Hint : 監聽鍵盤事件

# 總結

在這個章節中我們學會了：

1. 事件處理基礎概念
2. 事件處理過程