Chapter 10 Part 3: 事件模型(Event Model)

事件模型

事件是指在瀏覽器中發生的某些事情。

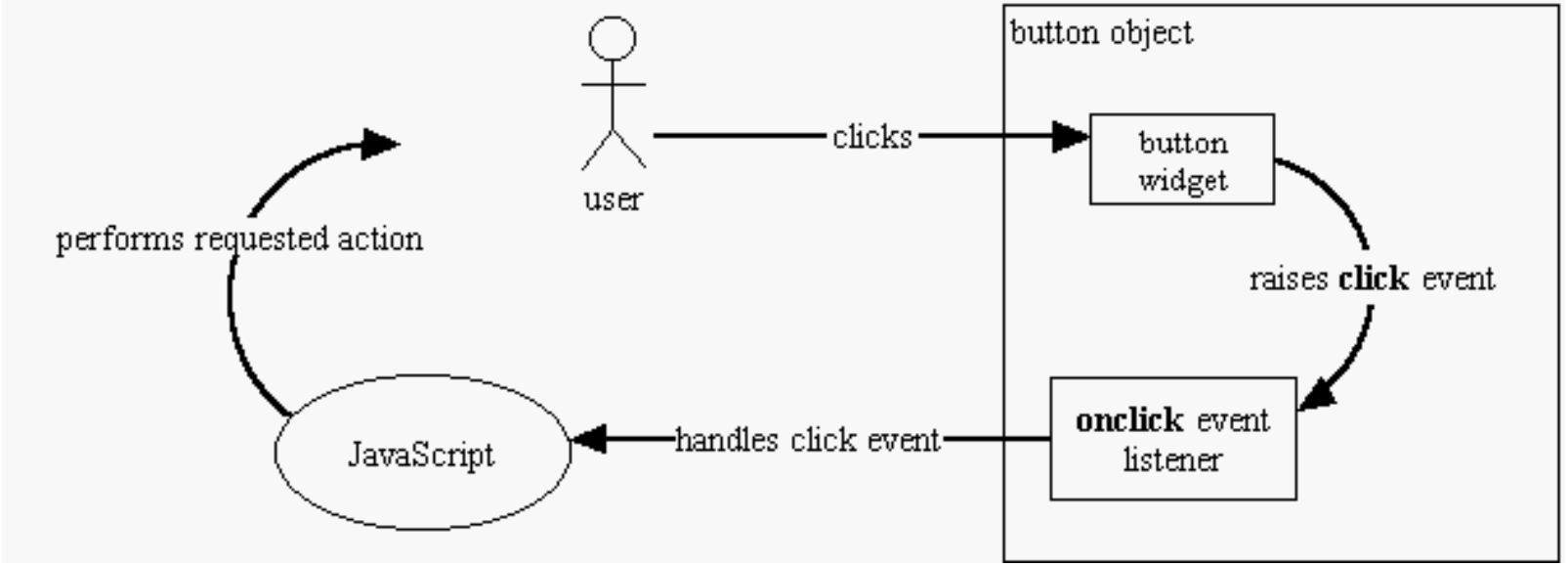
事件可以分為：

- 視窗或 DOM 事件（如 click、mouseover、keydown、drag-and-drop、scroll 等），表示使用者與視窗或 DOM 的互動。
- API 事件（如讀取用戶端檔案、從伺服器擷取資源、計時器等），用來通知開發者非同步操作已完成。

Ref: Event handling (overview) - Event reference | MDN

範例 6: 按鈕點擊事件

```
<button value="Click Me" onclick="alert('Thank you')" />
```



source: https://dotnettutorials.net/wp-content/uploads/2020/05/word-image-28.png?ezimgfmt=ngcb8/notWebP

事件鑑聽器 (Event Listener)

事件鑑聽器是用來監聽事件的函式，當事件發生時會執行相應的程式碼。

- 事件鑑聽器是回呼函式 (call-back function)。
- 當事件發生時，瀏覽器會呼叫事件鑑聽器函式，並傳遞一個 `Event` 物件作為參數。
- 事件鑑聽器函式的函數簽名如下：

```
functionName(event) {
    // code to execute when the event occurs
}
```

事件物件 (Event Object)

當事件發生時，瀏覽器會將一個 `Event` 物件傳遞給事件鑑聽器函式。

事件物件會提供兩個基本資訊：

- 事件的類型 (event type)
- 觸發事件的目標元素 (event target)

依據事件類型，不同的事件物件會有不同的屬性。例如：

- mouse event 包含滑鼠的座標、按鈕狀態、滾輪滾動的距離等資訊。
- keyboard event 包含被按下的鍵盤按鍵的代碼等。

要查閱完整的事件類型列表，請參考

- Event reference | MDN
- Event - Web APIs | MDN

註冊事件鑑聽器監聽某個元素的事件

有三種方法可以註冊事件鑑聽器來監聽某個元素的事件：

- 設定 HTML 標籤的事件處理屬性(event handler attribute)(行內事件處理器) 指向一個函數
- 設定 DOM 元素的事件處理特性(event handler property)指向一個函數
- 呼叫 DOM 元素的 `addEventListener()` 方法加入函數

接下來討論何時使用哪一種方法。

使用行內事件處理器：在 HTML 標籤的 `onXYZ` 屬性中撰寫 JavaScript 程式碼

當你只需要在事件發生時執行簡單的程式碼時，可以使用行內事件處理器。

你可以在 HTML 標籤的 `onXYZ` 屬性（如 `onclick`、`onmouseover`、`onkeydown` 等）中撰寫 JavaScript 程式碼。

範例 7: 行內事件處理器

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <a href="#" onclick="alert('Hello, world!');">Click me to say hello.</a> <br/>
        <a href="#" onclick="sayHello(this);"> Click me to invoke a function.</a> <br/>
    </body>

    <script>
        function sayHello(trigger) {
            // alert('Hello, world!');
            console.log('Hello, world!');
            console.log('this: ', trigger);
            console.log('event type: ', event.type);
            console.log('event target: ', event.target);
        }
    </script>
</html>
```

See file ex_12p2_07.html for the complete code.

第一個 `<a>` 標籤在使用者點擊連結時，會顯示提示訊息 "Hello, world!"。

第二個 `<a>` 標籤在使用者點擊連結時，會執行 `sayHello` 函式，並傳入兩個參數。

第一個參數是 `this` 關鍵字，代表執行行內事件處理器的元素本身。

- 在 `sayHello` 函式中，`this` 等同於 `event.target`。

自動轉換的行內事件處理器(Implicitly converted inline event handler)

瀏覽器會自動建立一個函數，將行內事件處理器中的程式碼加入該函數中，並將該函數註冊為元素的事件監聽器。

自動建立的函數有以下的模式：

```
function(event) {
  with(document) {
    with(this.form || {}) {
      with(this) {
        // code in the inline event handler
      }
    }
}
```

這表示，在行內處理器中，你可以直接存取觸發事件的元素 (`this`)、表單資料 (this.form)、 `document` 物件和 `event` 物件。

所以, 上述的行內事件處理器等同於以下的程式碼：

```
function sayHello(trigger) {
    ...
}
document.body.children[2].onclick = function(event){
  with(document) {
    with(this.form || {}) {
      with(this) {
        sayHello(this);
      }
    }
  }
}
```

缺點

大量使用行內事件處理器會讓你的程式碼難以維護與除錯，因為：

1. JavaScript 程式碼與 HTML 混雜在一起。
2. 無法集中管理事件監聽器的註冊。
3. 當多個標籤需要使用相同的事件處理器時，必須分別指定，非常不便。
4. 無法以程式方式動態新增或移除事件監聽器。

在 DOM 元素的事件處理特性(event property)上註冊事件監聽器是較佳的做法。

設定 DOM 元素的事件處理屬性來註冊事件監聽器

每個 HTML 標籤(tag)都有對應的 DOM 元素物件(DOM element object)。

HTML 標籤有一組事件處理屬性(event handler attributes)，而 DOM 元素物件也有一組事件處理特性(event handler properties)

- 如 `onclick`、`onmouseover`、`onkeydown` 等，全部為小寫。

你可以將一個函式指定給 DOM 元素的事件處理特性，來註冊事件監聽器。

範例 8: 設定 DOM 元素的事件處理特性

- 我們重寫前面的範例，並將 JavaScript 程式碼與 HTML 程式碼分開。
  - 這使得 HTML 及 JavaScript 程式碼更易於維護。

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <a href="#">Click me to say hello.</a> <br/>
        <a href="#">Click me to invoke a function.</a> <br/>

        <script src="ex_12p2_08.js"></script>
    </body>
</html>
```

```
const allAnchors = document.querySelectorAll('a');
let firstAnchor = document.querySelectorAll('a')[0]
let secondAnchor = document.querySelectorAll('a')[1]

// Assign the onclick event handler

firstAnchor.onclick = function(event) {
  console.log('Event type:', event.type);
  console.log('Event target:', event.target);
  window.alert('You clicked the first anchor element');
}

secondAnchor.onclick = function(event) {
    console.log('Hello, World!');
    console.log('Event type:', event.type);
    console.log('Event target:', event.target);
}
```

Programming Pattern: 多個元素使用相同的事件處理器

- 當多個元素需要使用相同的事件處理器時，你應該將事件處理器(函數)獨立出來維護
- 然後取得所有相關的元素，並將事件處理器指定給這些元素的事件處理特性。

範例 9: 指定相同的事件處理器給多個元素

點擊每個 `<input type="checkbox">` 元素時，會在主控台中顯示該元素的 `value` 和 `data-flavor` 屬性。 file: ex_12p2_09.html

```
        <div>
            <input type="checkbox" id="cb1" name="flavor" value="10" data-flavor="vanilla">
            <label for="cb1">Vanilla</label>
            <input type="checkbox" id="cb2" name="flavor" value="20" data-flavor="strawberry">
            <label for="cb2">Strawberry</label>
        </div>
        <script>
            // define the listener function
            function logValueData(e){
                console.log(e.target.value);
                console.log(e.target.dataset.flavor);
            }
            // assign the listener function
            const checkboxes = document.getElementsByName('flavor');
            checkboxes.forEach(checkbox => {
                checkbox.onclick = logValueData;
            });
        </script>
```

注意事項：

- `document.getElementsByName('flavor')` 會回傳一個包含所有 name 為 `flavor` 的元素的 `NodeList` 物件。
- `NodeList` 物件提供 `forEach` 方法，可以用來遍歷清單中的每個元素。
  - 但 `HTMLCollection` 物件則沒有這個方法。

優點

- 程式碼與 HTML 分離，易於維護。
- 可以集中為元素註冊事件監聽器。
- 可以將相同的監聽函式指定給多個元素。

限制

- 你無法為同一個元素的同一事件類型指定多個事件監聽器。
  - 例如，無法為同一個元素的 `onclick` 事件指定兩個不同的函式。
- 你無法在指定事件監聽器時設定進階事件選項。
  - 例如 `capture`、`once`、`passive` 等選項。
  - 這些進階選項將在下一章介紹。

DOM 元素的 `addEventListener` 方法可以解決上述這些限制。

呼叫 DOM 元素的 `addEventListener()` 方法來註冊事件監聽器

第三種註冊事件監聽器的方法，是呼叫元素的 `addEventListener` 方法。

`addEventListener` 的語法如下：

```
addEventListener(type, listener)
addEventListener(type, listener, options)
addEventListener(type, listener, useCapture)
```

最後兩種語法(`options` 與 `useCapture` 參數)將在下一章說明。

`addEventListener` 方法的參數說明：

`type` 參數：

- 事件類型，例如 `click`、`mouseover`、`keydown` 等。
- 不需要加上 `on` 前綴。
- 事件類型的完整列表請參考 Event reference | MDN

`listener` 參數：

- **事件監聽器函式**，當事件發生時會被執行。
- 或是一個帶有 `handleEvent` 方法的**物件**。

Programming Pattern: 一個事件觸發多個事件監聽器

- 當一個事件需要觸發多個事件監聽器時，可以使用 `addEventListener` 方法來註冊多個事件監聽器到同一個元素的同一事件類型上。

範例 10: 使用 `addEventListener` 方法註冊多個事件監聽器

重寫前面的範例，並使用 `addEventListener` 方法為每個 `<input type="checkbox">` 元素註冊兩個事件監聽器。 ex_12p2_10.html

```
...
<script>
    // define the listener function
    function logValueData(e){...}
    function secondListener(e){...}
    // assign the listener function
    const checkboxes = document.getElementsByName('flavor');
    checkboxes.forEach(checkbox => {
        checkbox.addEventListener('click', logValueData);
        checkbox.addEventListener('click', secondListener);
    });
</script>
```

完整的程式碼可以參考 ex_12p2_10.html。

本章重點摘要

- 事件模型讓網頁能回應使用者互動與非同步操作。
- 事件鑑聽器是回呼函式，會收到事件物件參數。
- 註冊事件鑑聽器有三種方式：
    i. 行內事件處理器（HTML 標籤的 onXYZ 屬性）
    ii. DOM 元素的事件處理特性（如 element.onclick）
    iii. `addEventListener()` 方法（可註冊多個監聽器並支援進階選項）
- 建議將 JavaScript 與 HTML 分離，集中管理事件監聽器。
- `addEventListener` 方法最具彈性，支援多個監聽器與進階設定。
- 事件物件提供事件類型、目標元素及其他相關資訊。
- 查詢事件類型與屬性可參考 MDN 文件。