

Chapter 13 互動式內容與事件監聽器 Part 4

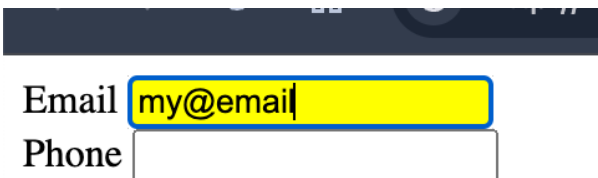
1 **input** 元素的 onblur/onfocus 與 onchange 事件

2 `onblur` 與 `onfocus` 事件

當 input 元素失去焦點時會觸發 `onblur` 事件。

相反地，當 input 元素獲得焦點時會觸發 `onfocus` 事件。

範例 6：當 input 元素獲得或失去焦點時改變背景顏色



Email

Phone

針對表單內所有 input 元素：

- 當 input 元素獲得焦點時，背景顏色變為黃色。
- 當 input 元素失去焦點時，背景顏色變回白色。

我們將 `focus` 與 `blur` 事件註冊到 form 元素(所有 `input` 的父元素)，以處理其所有子元素的事件。

在捕獲階段（capturing phase）呼叫事件處理函式，在事件到達目標元素之前改變背景顏色。

此範例的 JavaScript 程式碼如下：

```
const form = document.getElementById('myForm');
form.addEventListener('focus', function(event) {
    // 設定元素的 style 屬性
    event.target.style.backgroundColor = 'yellow';
}, true);
form.addEventListener('blur', function(event) {
    event.target.style.backgroundColor = '';
}, true);
```

- `addEventListener()` 的第三個參數設為 `true`，代表在捕獲階段註冊事件監聽器。

完整範例請見 [ex_11_6.html](#)

3 `input` 元素的 `onchange` 事件

當 `input` 元素的值改變且元素失去焦點時，會觸發 `onchange` 事件。

範例 10-7：當欄位變更時自動產生全名

- 每當名字或姓氏欄位的值改變時，會在輸出欄位顯示全名。

我們監聽 form 元素的 `change` 事件，就可以同時處理所有 input 元素的變更事件。

- 因為 input 元素的 `change` 事件會浮昇到 form 元素。

First Name	<input type="text" value="Michael"/>
Last Name	<input type="text" value="Wang"/>

Michael Wang

此範例的 JavaScript 程式碼如下：

```
const form = document.getElementById('myForm');
form.onchange = function (event) {
    const output = document.getElementById('output');
    const firstName = document.getElementById('firstName').value;
    const lastName = document.getElementById('lastName').value;
    output.innerHTML = `${firstName} ${lastName}`;
};
```

完整範例請見 [ex_11_7.html](#)

4 Key Events 鍵盤事件

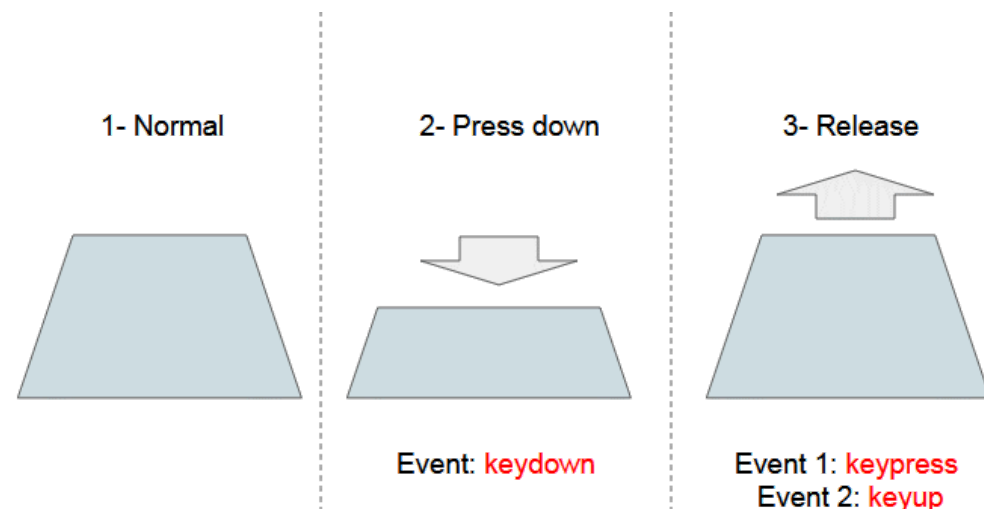
捕獲使用者按下的按鍵，並執行必要的操作。

5 `keydown`, `keyup`, and `keypress` 事件

- `keydown` 事件在按下鍵時觸發。
 - `keyup` 事件在釋放鍵時觸發。
 - 這兩個事件都會產生 `KeyboardEvent` 物件。

`keypress` 事件則是在按下並釋放鍵時觸發 (2014 年起已棄用)。

<https://o7planning.org/12319/javascript-keyboardevent>



警告：

- `keypress` 事件 已在最新版 JavaScript 中被棄用。

取得按下的按鍵字元(key)或實體鍵碼(code)

- `KeyboardEvent` 事件提供特性，讓開發者取得按下的按鍵字元或按鍵代碼。
- `keyboardEvent.key` 屬性：返回按下的鍵的字元值。
 - 例如：按下 b 鍵，返回 "b"；按下 shift+b 鍵，返回 "B"
 - 按下左控制鍵，返回 "Control"
- `keyboardEvent.code` 屬性：返回按下的實體鍵碼。
 - 例如：按下 b 鍵，返回 "KeyB"；按下 shift+b 鍵，返回 "KeyB"
 - 按下左控制鍵，返回 "ControlLeft"

`keyboardEvent.key` 特性的按鍵值規則

如果按下的按鍵有可列印的字元表示，則回傳非空的 Unicode 字元字串，包含該按鍵的可列印表示。

- 按下 b 鍵，回傳 "b"
- 按下 shift+b 鍵，回傳 "B"

如果按下的是**控制鍵或特殊字元鍵**，則回傳預先定義的鍵值：

- Enter 鍵 -> "Enter"
- Backspace 鍵 -> "Backspace"
- 完整的預設鍵值請參見 [Key values for keyboard events - Web APIs | MDN](#)

如果無法辨識該按鍵，則回傳 `Unidentified`。

keyboardEvent.code 特性

`KeyboardEvent.code` 屬性回傳的是實體鍵碼(code).

例如：

- 按下 b 鍵，回傳 "KeyB"
- 同時按下左側 Shift 和 b 鍵，會分別產生 "ShiftLeft" 和 "KeyB"（會產生兩個事件）

6 範例 10-8：顯示按下鍵時的 key 與 code 值

在輸入欄位中輸入任意鍵，會在頁面上顯示所按下鍵的 key 與 code 值。



A screenshot of a web form. It features a text input field with a blue border containing the text "234 a". A yellow cursor is positioned at the end of the text. Below the input field is a button labeled "Clear Logs".

Code value:

Digit2 Digit3 Digit4 Space KeyA ShiftLeft ControlLeft

key value:

2 3 4 a Shift Control

此範例的 JavaScript 程式碼如下：

```
const input = document.querySelector("input");
const keyLog = document.getElementById("keyCodeLog");
const charLog = document.getElementById("charCodeLog");

input.addEventListener("keydown", logKey);

function logKey(e) {
  keyLog.textContent += ` ${e.code}`;
  charLog.textContent += ` ${e.key}`;
}

function clearLogs() {
  keyLog.textContent = "";
  charLog.textContent = "";
}
```

完整範例請見 [ex_11_8.html](#)

7 範例 10-9：僅允許在輸入欄位中輸入數字，不允許空格、字母或特殊字元

- 此範例僅允許在輸入欄位中輸入數字。
- 例外按鍵包括 Backspace、Delete、ArrowLeft 和 ArrowRight，可用於編輯。

Code value:

KeyA KeyB KeyC Space Digit1 Digit2 Digit3 ControlLeft ShiftLeft

key value:

a b c 1 2 3 Control Shift

驗證輸入欄位的 JavaScript 程式碼如下：

```
// 為輸入欄位加入 keydown 事件監聽器
input.addEventListener("keydown", isNumberKey);

// 檢查按下的按鍵是否為數字的函式
function isNumberKey(event) {
  const exceptionKeys = ['Backspace', 'Delete', 'ArrowLeft', 'ArrowRight'];

  if ((isNaN(event.key) && !exceptionKeys.includes(event.code)) || event.code == 'Space') {
    // 阻止 keydown 事件的預設行為，並停止事件傳遞
    event.preventDefault();
    event.stopPropagation();
  }
}
```

首先，將 `keydown` 事件註冊到輸入欄位。當按下按鍵時，會呼叫 `isNumberKey` 函式。

`isNumberKey` 函式會檢查按下的按鍵是否為數字。

如果按鍵不是數字，則會呼叫 `preventDefault()` 方法來阻止該按鍵的預設行為。

- 也就是說，該按鍵不會被輸入到欄位中。

完整範例請見 [ex_11_9.html](#)

8 關於已棄用的 `keyCode` 與 `charcode` 特性

`keyCode` 和 `charcode` 屬性在新版 JavaScript 中已被棄用。

- 請勿再使用這些屬性。
- 請改用 `key` 屬性來取得按鍵資訊。



複習問題

- 如果你想在使用者於輸入欄位輸入時顯示建議清單，應該監聽哪些事件？

- A. `keydown` 、 `keyup` 或 `keypress`
- B. `blur` 和 `focus`
- C. `change`

► Ans

- 你想知道使用者是否按下左側的 Control 鍵。你應該使用 `keyboardEvent` 物件的哪個屬性？

- A. `keyboardEvent.key`
- B. `keyboardEvent.code`

► Ans

9 表單提交

表單提交是網頁開發中的基本功能，允許使用者將資料傳送到伺服器進行處理。

10 建立表單

要建立表單，請使用 `<form>` 元素，並在其中加入各種輸入元素，例如文字欄位、單選按鈕、核取方塊和提交按鈕。

設定：

- `action` 屬性：指定伺服器端處理表單資料的 URL，或表單提交後的導向網址。
- `method` 屬性：指定用來傳送表單資料的 HTTP 方法，例如 `GET` 或 `POST`。

範例: 建立一個表單

```
<body>
  <form id="exampleForm"
    action="https://formtester.goodbytes.be/post.php"
    method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>
    <br>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <br>
    <input type="submit" value="Submit">
  </form>
</body>
```


監聽表單的提交事件

- 當使用者點擊提交按鈕（`type="submit"` 的 `input` 元素）時，會觸發表單的提交事件 `submit`。
- 監聽器應該註冊在 `<form>` 元素上，而不是在 `<input>` 元素上。

11 監聽事件可以做什麼事

在表單提交事件處理函式中，你可以執行以下操作：

- 驗證表單資料(validation)
 - 檢查使用者輸入的資料是否符合預期格式。
- 處理表單資料(data processing)
 - 計算或轉換使用者輸入的資料。
- 其他操作(other operations)
 - 例如：顯示提示訊息、清除表單欄位等。

12 如何在 `submit` 事件中取得表單資料

方法 1: 使用 `FormData` 物件自動將表單資料轉換為鍵值對

步驟 1. 使用 `new FormData(formElementObject)` 建構子，從現有的表單元素建立一個新的 `FormData` 物件。

步驟 2. 使用 `FormData` 物件所提供的方法來存取表單資料。

- `formData` 物件是 JavaScript 內建的物件，用來以鍵值對的方式表示表單資料。

FormData 物件的方法

- `get(name)`：根據欄位名稱取得特定表單欄位的值。
- `getAll(name)`：取得所有同名表單欄位的值（例如多選框）。
- `entries()`：回傳一個可用於遍歷所有鍵值對的迭代器。

更多細節請參考 [FormData - Web APIs | MDN](#)

範例 11-11：從表單資料建立 `FormData` 物件

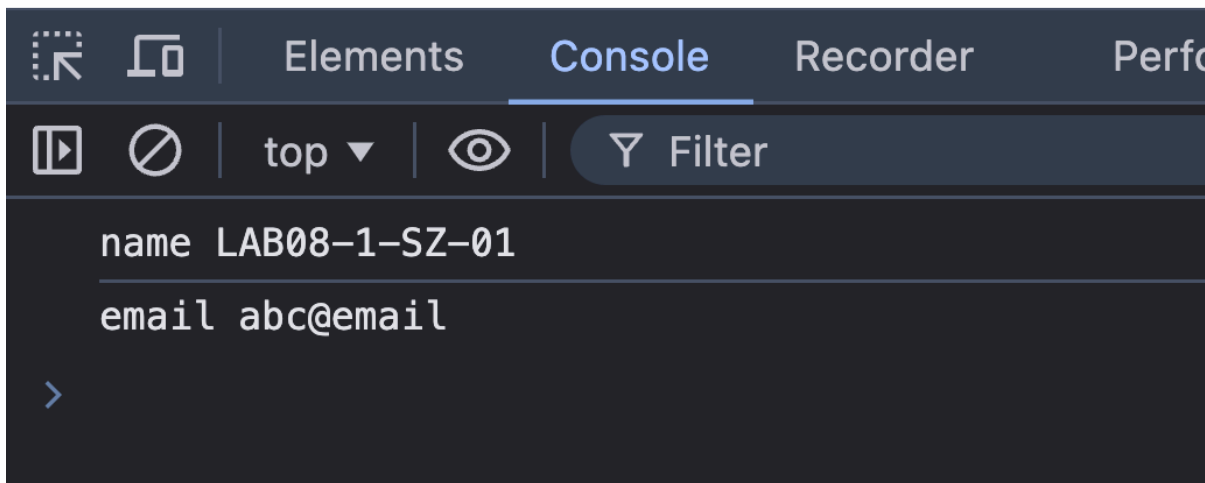
當使用者提交表單時，表單資料會顯示在主控台（console）中。

```
const form = document.getElementById("exampleForm");
form.addEventListener("submit", function (event) {
  event.preventDefault();
  const formData = new FormData(form);
  for (const [key, value] of formData.entries()) {
    console.log(key + ': ' + value);
  }
});
```

Name:

Email:

Go to <https://formtester.goodbytes.be/post.php> to see the form submission.



方法 2: 使用 `HTMLFormElement` 物件的 `elements` 屬性

- `HTMLFormElement` 的 `elements` 屬性會回傳一個 `HTMLFormControlsCollection`，列出該 `<form>` 元素中所有的表單控制項。
- 使用 HTML tag 的 `name` 屬性值為鍵值，向 `HTMLFormControlsCollection` 物件索引，取得對應的表單控制項的 `value` 屬性值。

Example:

```
const form = document.getElementById("exampleForm");
form.addEventListener("submit", function (event) {
  event.preventDefault();
  const name = form.elements["name"].value;
  const email = form.elements["email"].value;
  console.log("Name: " + name);
  console.log("Email: " + email);
});
```


範例 11-12：在提交前驗證表單資料

當使用者提交表單時，會先驗證表單資料。如果資料無效，則不會提交表單。

S1. 建立驗證函式，檢查 name 和 email 欄位是否為空。

```
function validate(formData) {  
  const name = formData.get('name');  
  const email = formData.get('email');  
  if (name === "" || email === "") {  
    alert("姓名與電子郵件為必填欄位。");  
    return false; // 回傳 false 以阻止表單提交  
  } else  
    return true; // 回傳 true 以允許表單提交  
}
```

S2. 在表單提交事件處理函式中驗證表單資料。

```
form.addEventListener('submit', function(event) {  
    const form = document.getElementById('exampleForm');  
    const formData = new FormData(form);  
    if (!validate(formData)) {  
        event.preventDefault(); // 阻止表單提交  
    } else {  
        form.submit();           // 允許表單提交  
        alert("表單已成功提交。");  
    }  
});
```

完整範例請見 [ex_11_11.html](#)

複習問題

Q1 `FormData` 物件在 JavaScript 中的責任是什麼？（可複選）

- A. 以鍵值對的方式儲存表單資料
- B. 建立新的表單元素
- C. 將表單資料提交到伺服器

► Ans

Q2 在表單提交事件處理函式(submit handler)中可以做哪些事？（可複選）

- A. 驗證表單資料
- B. 處理表單資料
- C. 阻止表單提交
- D. 呼叫 `fetch()` 將表單資料送到伺服器

► Ans

13 本章重點摘錄

- 了解 `input` 元素的 `onblur` 、 `onfocus` 、 `onchange` 事件及其應用。
- 掌握事件捕獲階段 (capturing phase) 與事件浮昇 (bubbling) 的差異。
- 熟悉鍵盤事件 (`keydown` 、 `keyup`) 及 `KeyboardEvent` 物件的 `key` 和 `code` 屬性。
- 能夠限制輸入欄位僅允許特定按鍵 (如數字)。
- 理解 `FormData` 物件的用途與常用方法，能在表單提交時取得與驗證資料。
- 熟悉表單提交事件的攔截、驗證與資料處理流程。
- 知道已棄用的 `keypress` 、 `keyCode` 、 `charCode` 屬性，應改用現代屬性。
- 能根據需求選擇合適的事件監聽器與資料處理方式。