關於本書

JavaScript×ChatGPT 強強聯名，讓您在 AI 的神助攻下，華麗化身為 JavaScript 程式設計高手！如果您以為這種聯名方式只是搶搭生成式 AI 工具的熱潮，譁眾取寵，那麼我們要很慎重地告訴您，ChatGPT 真的會寫 JavaScript 程式，而且程式碼簡潔乾淨，不輸給程式設計高手。

在 ChatGPT 橫空出世後，有不少人驚覺「寫程式」即將由 AI 工具所取代，沒錯，AI 工具確實能夠寫程式，但這並不表示您就不用學程式設計，而是程式設計師必須要進化為 AI 工具的程式審查員或教 AI 學習的老師。

換句話說，您必須具備程式設計能力，才有辦法跟 AI 工具溝通，讓它寫出您需要的程式碼，也才有辦法閱讀或審查 AI 工具所生成的程式碼，確保程式碼是正確的、有效率的、經過完整測試的，而想要練就程式功力，您所需要的正是一本好書。

在本書中，我們除了告訴您如何使用 ChatGPT 撰寫程式、解讀程式、查詢語法、尋求技術支援、除錯、出題練習、在 JavaScript 與其它程式語言之間做轉換，更重要的是有計畫地帶您學習 JavaScript，無論您有無程式設計的經驗，只要約略具有 HTML 與 CSS 的基礎知識，都能看得懂、學得會，不會愈看愈挫折、半途而廢。

本書內容

- 第 1、2 章：介紹 JavaScript 的開發環境與編輯工具、程式碼撰寫慣例，以及使用 **ChatGPT** 撰寫程式、除錯、轉換程式語言等。
- 第 3 ~ 7 章：介紹 **JavaScript 的基本語法與內建物件**，例如變數、常數、型別、運算子、流程控制、函式、內建物件、錯誤處理等。
- 第 8 章：介紹 **DOM** (Document Object Model，文件物件模型)，這是一個與網頁相關的模型，當瀏覽器載入網頁時，會針對網頁和網頁的 HTML 元素建立對應的物件，JavaScript 可以透過 DOM 存取網頁的元素，例如段落、超連結、圖片、表格、表單等。

- 第 9 章：介紹**事件處理**，包括事件驅動模式、事件的類型、定義事件處理程式 / 事件監聽程式、事件流程 (事件氣泡 V.S. 事件捕捉)、Event 物件，以及一些事件處理範例。
- 第 10 章：介紹 **BOM** (Browser Object Model，瀏覽器物件模型)，這是一個與瀏覽器相關的模型，裡面有數個物件，JavaScript 可以透過 BOM 存取瀏覽器的資訊，例如瀏覽器類型、瀏覽歷程記錄、網址等。
- 第 11 章：介紹**網頁儲存**，這是一種在用戶端儲存資料的技術。
- 第 12 章：介紹 **Ajax 與 JSON**，這是一種讓瀏覽器與 Web 伺服器進行非同步溝通的技術。
- 第 13 章：使用 **jQuery** 所提供的 API 讓操作 HTML 文件、選擇 HTML 元素、處理事件、建立特效等動作變得更簡單。
- 第 14 章：使用 **Vue.js** 所提供的 API 進行資料繫結及操作網頁的元素，解決畫面顯示與資料狀態同步的問題。

排版慣例

本書在條列 JavaScript 語法時，遵循下列排版慣例：

- 斜體字表示自行輸入的參數、屬性值、敘述或名稱，例如 isNaN(*x*) 的 *x* 表示自行輸入的參數。
- 中括號 [] 表示可以省略不寫，例如 toString([*radix*]) 的 [*radix*] 表示參數可以有，也可以沒有。
- 垂直線 | 用來隔開替代選項。

聯絡方式

如果您有建議或授課老師需要 PowerPoint 教學投影片與學習評量，歡迎與我們洽詢：碁峰資訊網站 https://www.gotop.com.tw/；國內學校業務處電話－台北 (02)2788-2408、台中 (04)2452-7051、高雄 (07)384-7699。

CHAPTER 02

使用 ChatGPT 撰寫 JavaScript 程式

2-3 撰寫 JavaScript 程式與除錯

我們可以請 ChatGPT 撰寫 JavaScript 程式、解讀 JavaScript 程式的意義、幫忙加上註解或進行除錯，以下有進一步的說明。

2-3-1 撰寫指定用途的 JavaScript 程式

當您要請 ChatGPT 撰寫 JavaScript 程式時，請詳細描述程式的用途，這樣它所撰寫出來的程式就會更符合要求，例如「請給我實現氣泡排序法的 JavaScript 程式」、「請給我使用 for 迴圈實現九九乘法表的 JavaScript 程式」、「請使用 JavaScript 撰寫一個計數器」、「請撰寫一個網頁，裡面有一個標題 1 格式的字串 "Hello!"，當滑鼠指標移到該字串時，字串會變成紅色，當滑鼠指標離開該字串時，字串會恢復成預設的顏色」等。

下面是我們在 ChatGPT 輸入「**請給我計算整數 1 加到 10 的總和的 JavaScript 程式**」所得到的回答，裡面除了有程式碼，還有相關的說明，若要複製程式碼，只要點取程式碼方塊右上角的 **[Copy code]** 即可。

2-3-2 解讀 JavaScript 程式的意義

除了撰寫程式之外，ChatGPT 也可以解讀 JavaScript 程式的意義。以下面的對話為例，我們問 ChatGPT 這段程式碼可以做什麼，然後附上一段程式碼，ChatGPT 馬上辨識出這段程式碼是以 JavaScript 撰寫而成，並逐行講解，說得非常清楚。

為了方便截圖做示範，所以我們附上的程式碼相當簡短，您也可以自己試著提問更複雜的程式碼，看看 ChatGPT 是否能夠正確解讀。

JE

這段程式碼可以做什麼?

```
let sum = 0;
for (let i = 1; i <= 10; i++) {
  sum += i;
}
console.log(sum);
```

這段程式碼是用JavaScript撰寫的。它的功能是計算從1到10（包括1和10）的所有整數的和，然後將結果打印到控制台。

程式碼解釋如下：

1. 定義一個變量 `sum` 並將其初始值設置為0。這個變量將用於存儲整數和的計算結果。
2. 使用 `for` 迴圈遍歷從1到10的整數範圍。迴圈變量 `i` 的初始值為1，迴圈條件為 `i <= 10`，每次迴圈迭代後，`i` 的值遞增1。
3. 在每次迴圈迭代中，將迴圈變量 `i` 的值累加到變量 `sum` 中。這樣可以在迴圈結束時得到1到10之間所有整數的和。
4. 迴圈結束後，使用 `console.log()` 函數將變量 `sum` 的值打印到控制台。

當這段程式碼執行時，它將計算出1到10之間所有整數的和，即1 + 2 + 3 + ... + 10 = 55。控制台將輸出結果 55。

3-3-8 陣列 (array)

陣列 (array) 可以用來儲存多個資料，這些資料叫做**元素** (element)，每個元素有各自的**索引** (index) 與**值** (value)。

索引可以用來識別元素，例如第 1 個元素的索引為 0，第 2 個元素的索引為 1，...，第 n 個元素的索引為 n - 1。當陣列最多儲存 n 個元素時，表示它的**長度** (length) 為 n。

例如下面的敘述是建立一個陣列並指派給變數 A：

```
var A = [10, 20, 30];
```

❶ ❷ ❸

❶ 陣列的名稱。

❷ 陣列的前後以中括號括起來。

❸ 包含 10、20、30 三個元素，中間以逗號隔開。我們可以透過陣列的名稱與索引來存取元素，例如 A[0]、A[1]、A[2] 分別代表 10、20、30。

元素	值
A[0]	10
A[1]	20
A[2]	30

陣列裡面也可以儲存其它陣列，形成**巢狀陣列** (nested array)，例如下面的敘述是建立一個巢狀陣列並指派給變數 B：

```
var B = [10, [21, 22], 30];
```

❶ ❷

❶ 巢狀陣列的名稱。

❷ 第二個元素是另一個陣列，我們可以透過陣列的名稱與兩個索引來存取元素，例如 B[1][0]、B[1][1] 分別代表 21、22。

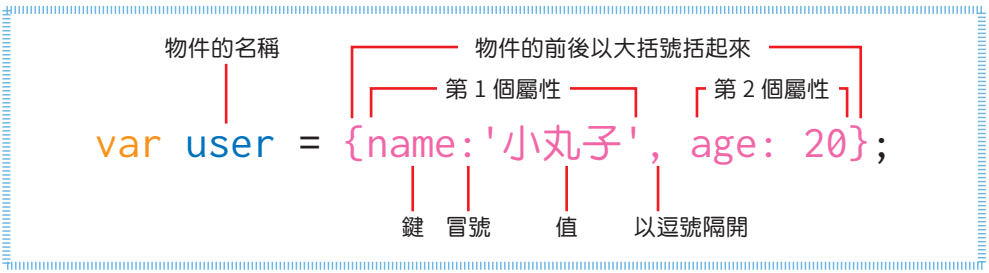
元素	值
B[0]	10
B[1]	[21, 22]
B[1][0]	21
B[1][1]	22
B[2]	30

3-3-9 物件 (object)

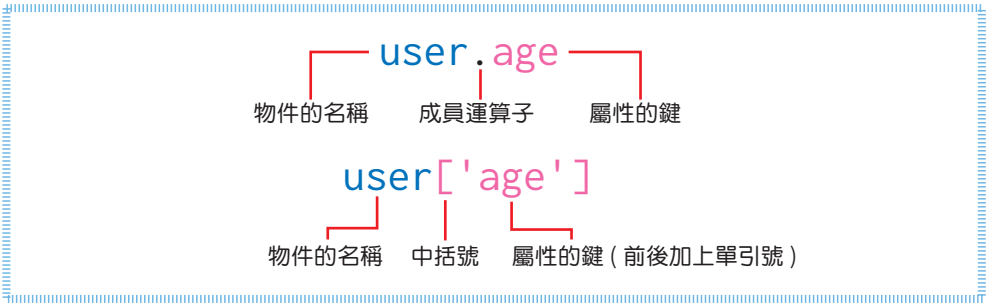
JavaScript 的**物件** (object) 是一種**關聯陣列** (associative array)，它和一般陣列的差別如下：

- 陣列所儲存的資料稱為**元素** (element)，而物件所儲存的資料稱為**屬性** (property)，屬性除了可以是數值、字串、布林等資料，也可以是函式，這種儲存了函式的屬性又稱為**方法** (method)。
- 陣列是使用**索引** (index) 來識別元素，而物件是使用**鍵** (key) 來識別屬性，索引是數字，而鍵是字串。事實上，物件的屬性就是一個**鍵 / 值對** (key/value pair)，分別代表屬性的名稱與值。

例如下面的敘述是建立一個物件並指派給變數 user：



我們可以使用**成員運算子 (.)** 或**中括號表示法**存取物件的屬性，例如下面兩個寫法均會傳回 age 屬性的值，也就是 20。有關如何操作物件的屬性與方法，第 6 章有進一步的說明。



延伸閱讀

傳值指派 V.S. 傳址指派

基本型別和物件型別主要的差別在於變數是使用哪種方式來儲存值，基本型別的變數所儲存的是值本身，而物件型別的變數所儲存的是值在記憶體中的位址。

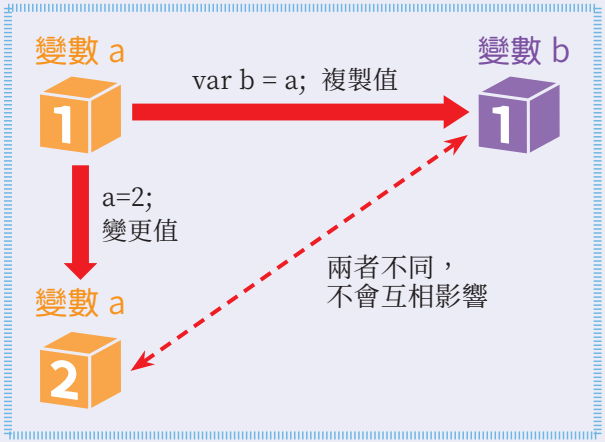
兩者在實際操作上會有些許不同，例如基本型別的指派運算是採取**傳值指派**，而物件型別的指派運算是採取**傳址指派**。

下面是一個例子，它示範了何謂傳值指派：

- 01：宣告變數 a，並將指定的值 (1) 儲存在變數 a。
- 02：宣告變數 b，並將變數 a 所儲存的值 (1) 複製一份給變數 b。
- 03：將變數 a 所儲存的值變更為 2。
- 04：由於變數 b 的值是複本，即使變數 a 的值改變了，也不會影響到變數 b，因而顯示變數 b 的值為 1。

\Ch03\assign.js

```
01  var a = 1;
02  var b = a;
03  a = 2;
04  window.alert(b);
```

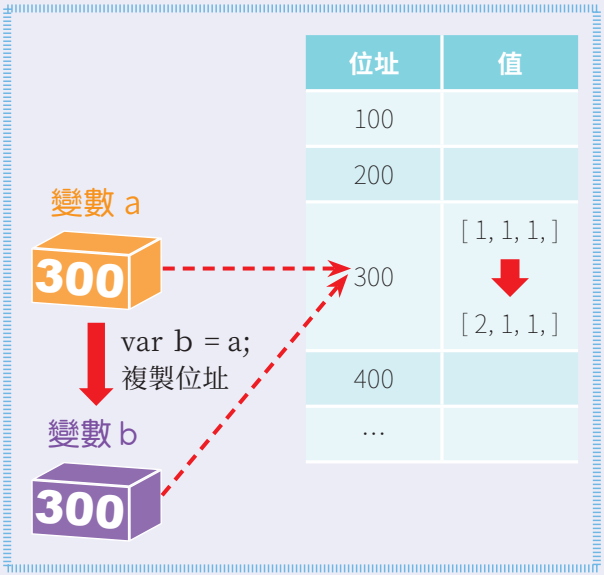


下面是另一個例子，它示範了何謂傳址指派：

- 01：宣告變數 a，並將陣列 [1, 1, 1] 的位址儲存在變數 a，表示變數 a 指向陣列 [1, 1, 1]，在下面的示意圖中，我們假設此陣列的位址為 300。
- 02：宣告變數 b，並將變數 a 所儲存的位址 (300) 複製給變數 b，表示變數 b 和變數 a 指向相同的陣列 [1, 1, 1]。
- 03：透過變數 a 將陣列的第一個元素變更為 2，此時，陣列的內容變成 [2, 1, 1]。
- 04：由於變數 b 和變數 a 指向相同的陣列，當陣列的內容改變了，也會連帶影響到變數 b，因而顯示變數 b 的值為 [2, 1, 1]。

\Ch03\assign2.js

```
01  var a = [1, 1, 1];
02  var b = a;
03  a[0] = 2;
04  window.alert(b);
```

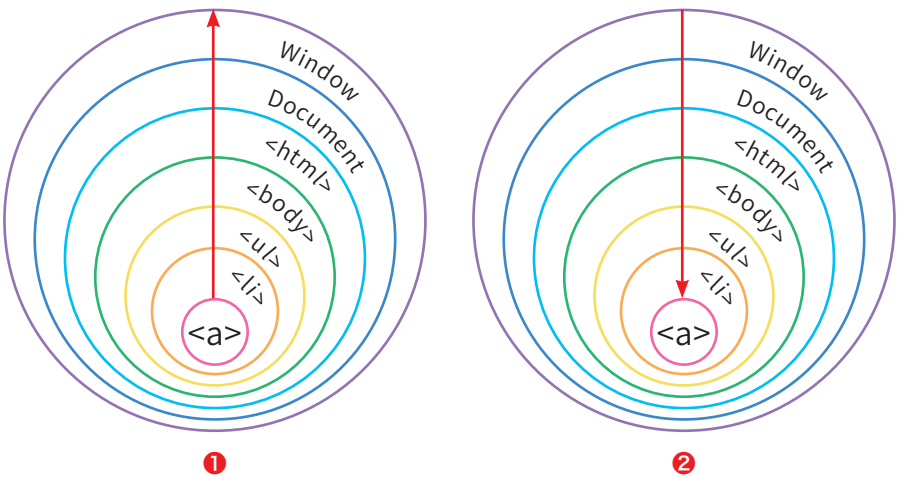


9-5 事件流程

HTML 文件屬於巢狀結構，當使用者以滑鼠移過或點按某個 HTML 元素時，也會連帶地移過或點按其外層的父元素。

舉例來說，假設網頁中有一個項目清單，裡面有幾個超連結項目，我們可以將事件處理程式或事件監聽程式繫結到 <a>、<li>、<ui>、<body>、<html> 等元素，以及 Document 和 Window 物件，當使用者點按項目清單中的一個超連結項目時，除了會觸發 <a> 元素的 click 事件，同時也會觸發外層元素的 click 事件，我們將這些元素之間的事件觸發順序稱為**事件流程**，而且事件流程有下列兩種方向：

- **事件氣泡** (event bubbling)：現代瀏覽器預設是採取事件氣泡流程，也就是事件會從目標元素開始往外循序傳遞，一直到最外層的 Window 物件為止，就像水面下的氣泡往上升一樣 (如圖❶)。
- **事件捕捉** (event capturing)：事件捕捉流程的事件會從最外層的元素開始往內循序傳遞，一直到最內層的目標元素為止 (如圖❷)。



下面是一個例子，我們分別針對 <a>、<li>、 等元素的 click 事件繫結事件監聽程式，而且 <a> 元素有兩個事件監聽程式。請注意，此處所有 addEventListener() 方法的第三個參數均為 false，表示採取預設的事件氣泡流程。

\Ch09\bubbling.html

```
<body>
  <ul id="foods">
    <li id="item1"><a id="a1" href="cake.html">蛋糕</a></li>
  </ul>
  <script src="bubbling.js"></script>
</body>
```

\Ch09\bubbling.js

```
var a1 = document.getElementById('a1');
var item1 = document.getElementById('item1');
var foods = document.getElementById('foods');

a1.addEventListener('click', function() {
  window.alert('<a> 元素的事件監聽程式 1');
}, false);

a1.addEventListener('click', function() {
  window.alert('<a> 元素的事件監聽程式 2');
}, false);

item1.addEventListener('click', function() {
  window.alert('<li> 元素的事件監聽程式 ');
}, false);

foods.addEventListener('click', function() {
  window.alert('<ul> 元素的事件監聽程式 ');
}, false);
```

❶ 繫結 <a> 元素的事件監聽程式 1

❷ 繫結 <a> 元素的事件監聽程式 2

❸ 繫結 <li> 元素的事件監聽程式

❹ 繫結 <ul> 元素的事件監聽程式

執行順序如下，click 事件會從目標元素 <a> 開始往外循序傳遞到最外層的元素：

❶ 顯示「<a> 元素的事件監聽程式 1」

❷ 顯示「<a> 元素的事件監聽程式 2」

❸ 顯示「 元素的事件監聽程式」

❹ 顯示「 元素的事件監聽程式」

❺ 離開 bubbling.html 網頁前往 cake.html 網頁

9-7-3 滑鼠事件

滑鼠事件是一些與使用者操作滑鼠相關的事件，例如 click、dblclick、mousedown、mouseup、mouseenter、mouseleave、mouseover、mouseout、mousemove、mousewheel 等，其中 **mouseover / mouseout** 會在使用者將滑鼠移入 / 移出元素時觸發。下面是一個例子，剛開始網頁上會顯示圖片 piece1.jpg，當指標移到圖片時會變成 piece2.jpg，而當指標離開圖片時又會變成原來的 piece1.jpg。

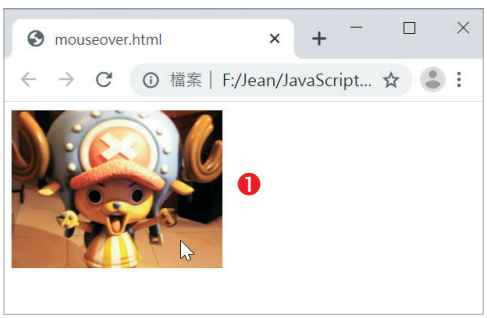
\Ch09\mouseover.html

```
<body>
  <img id="fig" src="piece1.jpg" width="200">
  <script src="mouseover.js"></script>
</body>
```

\Ch09\mouseover.js

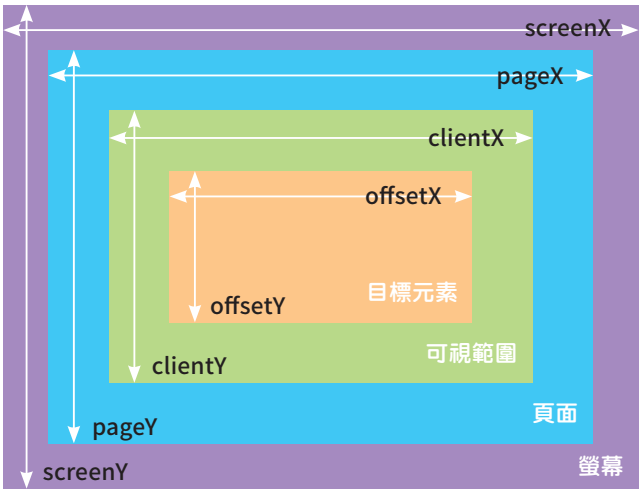
```
var fig = document.getElementById('fig');

fig.addEventListener('mouseover', function() {
  fig.src = 'piece2.jpg';
}, false);

fig.addEventListener('mouseout', function() {
  fig.src = 'piece1.jpg';
}, false);
```

❶ 當指標移到圖片時會顯示 piece2.jpg
❷ 當指標離開圖片時會顯示 piece1.jpg



此外，當滑鼠事件被觸發時，我們可以透過 Event 物件提供的屬性取得指標的座標：

- **screenX、screenY**：指標在螢幕中的 X、Y 座標，以螢幕左上角為原點，而非瀏覽器中的位置。
- **pageX、pageY**：指標在頁面中的 X、Y 座標，頁面可能會超出瀏覽器可視範圍，所以 pageX、pageY 和下面的 clientX、clientY 可能會不同。
- **clientX、clientY**：指標在瀏覽器可視範圍中的 X、Y 座標。
- **offsetX、offsetY**：指標在目標元素中的 X、Y 座標。



下面是一個例子，當使用者在 區塊中移動滑鼠時，就會即時顯示指標的各項座標。

\Ch09\mousemove.html

```
<body>
  <div id="region" style="position: absolute; top: 100px; left: 100px;
    width: 300px; height: 200px; border: 1px solid black;"></div>
  <script src="mousemove.js"></script>
</body>
```

\Ch09\mousemove.js

```
var region = document.getElementById('region');

region.addEventListener('mousemove', function(e) {
  region.innerHTML = 'screenX/screenY:' + e.screenX + '/' + e.screenY + '<br>'
                   + 'pageX/pageY:' + e.pageX + '/' + e.pageY + '<br>'
                   + 'clientX/clientY:' + e.clientX + '/' + e.clientY + '<br>'
                   + 'offsetX/offsetY:' + e.offsetX + '/' + e.offsetY;
}, false);
```

瀏覽結果如下圖，只要在 區塊中移動滑鼠，就會不斷地顯示最新的指標座標，您可以仔細比較各項座標的差異。