JavaScript 事件機制的原理

Yu Hsiang 林郁翔

事件機制的JavaScript

JavaScript 是一個事件驅動 (Event-driven) 的程式語言當瀏覽器載入網頁開始讀取後，雖然馬上會讀取 JavaScript 事件相關的程式碼，但是必須等到「事件」被觸發(如使用者點擊、按下鍵盤等)後，才會再進行對應程式的執行。

事件制的行為像什麼？

辦公室擺了一台電話在桌上，但是電話要是沒響，我們不會主動去「接電話」(沒人打來當然也無法主動接)。

電話響了 (事件被觸發) -> 接電話 (去做對應的事)

什麼語言才可以寫事件驅動?

Event-driven programs can be written in any programming language, although the task is easier in languages that provide high-level abstractions, such as await and closures. Event-driven programming is the dominant paradigm used in graphical user interfaces.

引用自維基百科：https://en.wikipedia.org/wiki/Event-driven_programming

範例一

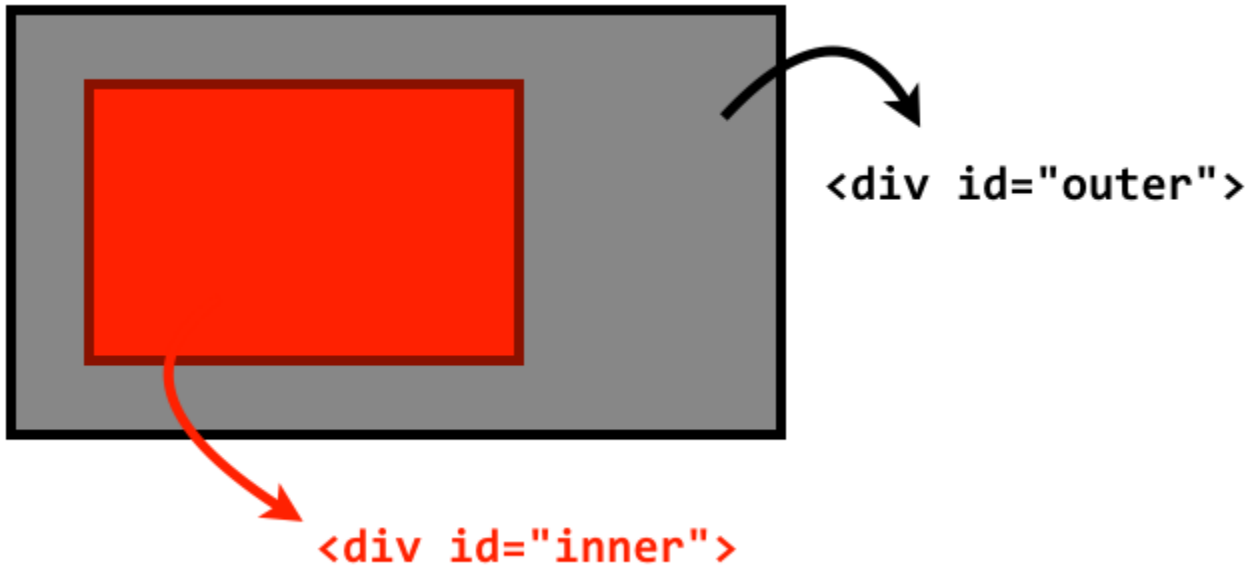
點擊按鈕->觸發事件，以Bootstrap測試：

https://getbootstrap.com/docs/4.0/components/modal/

檔案：/dist/demo01.txt

事件流程(Event Flow)

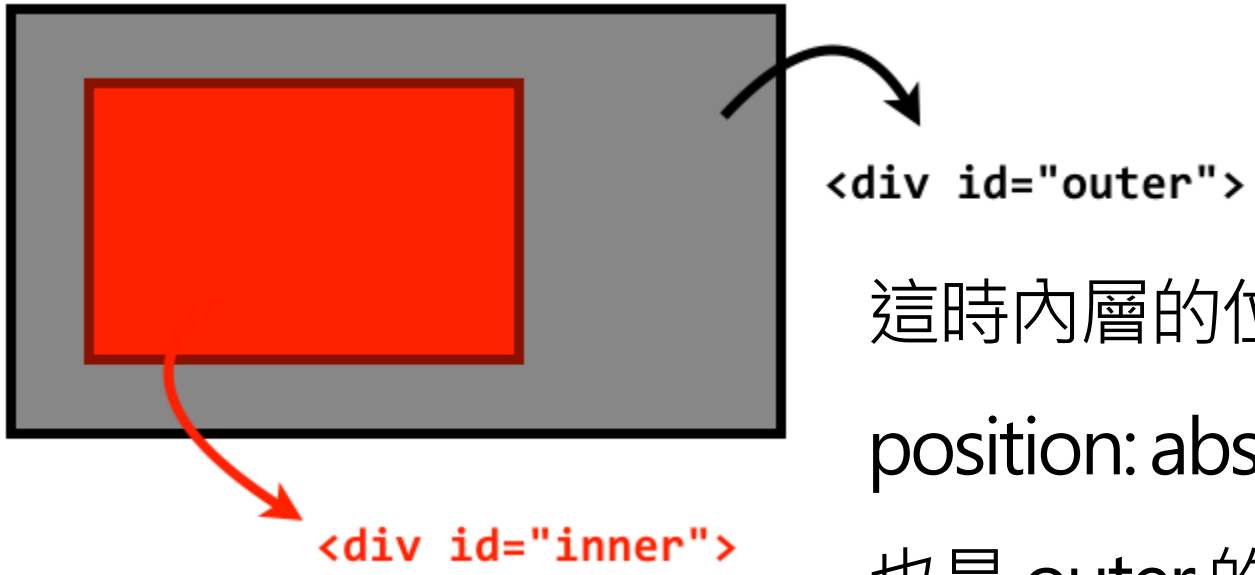
假設有兩個重疊的 div 元素，外層是 ，而內層是




```
<div id="outer">
   <div id="inner">

   </div>
</div>
```

事件流程(Event Flow)



這時內層的位置一定在外層裡面(不考慮css position: absolute; 的可能性)，這表示 inner 也是 outer 的一部分。 當我們點擊了 inner 的時候，代表我們也點擊到 outer，甚至再看遠一點，我們也點擊到整個網頁。

事件流程(Event Flow)

即是代表網頁元素接收事件的順序，分為

事件冒泡 (Event Bubbling)：從啟動事件的元素節點開始，逐層往上傳遞。

事件捕獲 (Event Capturing)：反之。

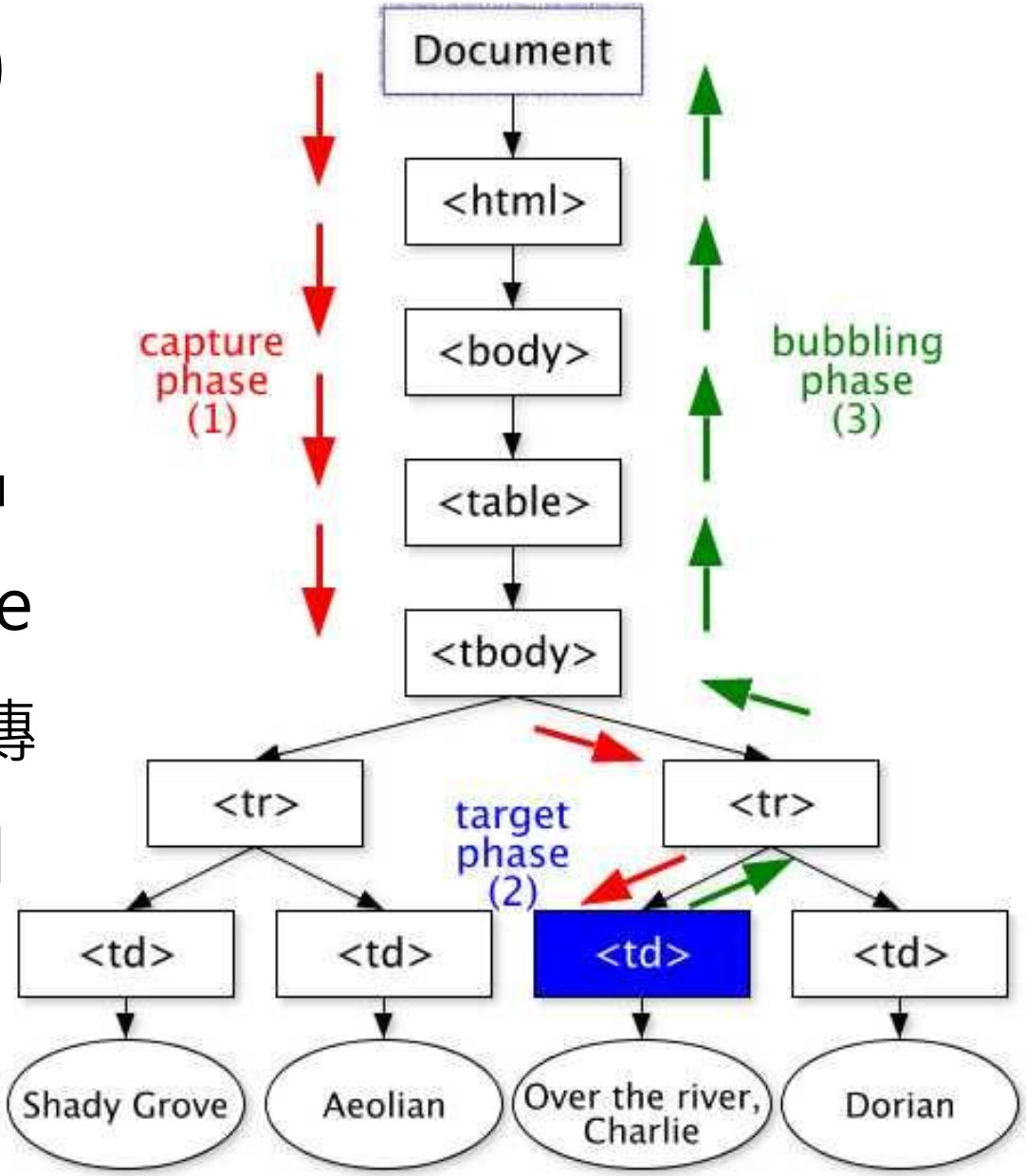
事件流程(Event Flow)

假設點擊圖中藍色的 <td>。

那麼當 td 的 click 事件發生時，

會先走紅色的「capture phase」

然後再繼續執行綠色的「bubble phase」，反方向由 <td> 往上傳至 Document，整個事件流程到此結束。



以上說的，似乎有點道理

但沒有經過**實際驗證**，我能相信嗎？

範例二

驗證Event Flow的事件冒泡 (Event Bubbling)及事件捕獲 (Event Capturing)。

檔案：/dist/demo02.html

從範例二可知:

點擊子元素的時候，父層的 Capturing 會先被觸發，然後再到子層內部的 Capturing 或 Bubbling 事件。 最後才又回到父層的 Bubbling 結束。

但，有個問題

子層(點擊目標)的 Capturing 或 Bubbling 是誰先誰後呢?

範例三

驗證Event Flow的事件冒泡 (Event Bubbling)及事件捕獲 (Event Capturing)，將子層(點擊目標) addEventListener順序顛倒。

檔案：/dist/demo03.html

從範例三可知:

子層(點擊目標)的事件冒泡 (Event Bubbling)及事件捕獲 (Event Capturing)要看你程式碼addEventListener的順序而定。

事件註冊的方式：

除了addEventListener可以註冊事件外

還可以透過HTML on-event註冊事件

```
<button id="btn"
onclick="console.log('HI');">Click</button>
```

兩種方式抉擇的考量：

基於程式碼的使用性與維護性考量，現在已經不建議用此方式來綁定事件，詳情可參考「維基百科: 非侵入式 JavaScript」條文，或自行 Google 相關資訊。

非侵入式JavaScript：非侵入式JavaScript是一種將Javascript從HTML結構抽離的設計概念，避免在HTML標籤中夾雜一堆onchange、onclick等屬性去掛載Javascript事件，讓HTML與Javascript分離，依模型-视图-控制器的原則將功能權責清楚區分，使HTML也變得結構化容易閱讀

添加事件addEventListener的陷阱

範例四

當addEventListener遇到removeEventListener

檔案：/dist/demo04.html

從範例四發現:

addEventListener後removeEventListener，事件沒有被移除掉。

因為removeEventListener() 解除事件的時候，第二個參數的 handler 必須要與先前在 addEventListener() 綁定的 handler 是同一個「實體」。

正確版：

範例5：

檔案：/dist/demo05.html

從範例五發現:

當addEventListener遇到removeEventListener，事件被移除掉，了！因綁定到同一個實體。

延伸閱讀：

「重新認識 JavaScript: Day 05 JavaScript 是「傳值」或「傳址」？」與「重新認識 JavaScript: Day 10 函式 Functions 的基本概念」

參考資料：

重新認識 JavaScript: Day 14 事件機制的原理
https://ithelp.ithome.com.tw/articles/10191970

E N D