跨來源資源共用

CORS（Cross-Origin Resource Sharing，跨來源資源共用），是一項在我們為了取得後端資料的規範。首先要了解同源政策（Same Origin Policy），簡單來說為了安全性的需求，舉個例子：正常情況下，駭客不能任意用一個惡意網站，取得和呼叫 IG 的網路服務，不然豈不是就可以隨意修改用戶的留言或新增貼文等動作了，另一個例子：如果今天恰好被駭客知道了公司內部的公開網站，那只有知道他的 Domain，就能經由 AJAX 去拿到資料，那豈不是許多機密文件都被駭客拿走了。

那重點就是甚麼是「同源」，這裡要先分成 DOM 和 Cookie，兩者在這件事情上定義不一樣，一個比較清晰的 DOM 同源定義是，當 Scheme、Domain 和 Port 是一致的資源，就被視為來自同個來源，第一個是協定，二是主機位置，三就是埠號。但不同來源也不是都會被禁止，像是跨來源的寫入通常是被允許的，具體的像是：重新導向（redirect）、連結（Link）是被允許的行為，還有跨來源的嵌入，像是放一段其他網站的影片和圖片，也是被允許的動作。

而 Cookie 的定義是允許子網域的程式碼，可以修改或者是覆蓋母網域的 Cookie。只要使用的 Cookie 的 Domain 跟 Path 一致，就被視為同源，但是如果有加入一些特別的設定時，就要判定 Scheme 是否一致，才能送出 Cookie。

在了解同源後，就會清楚 CORS 同樣是為了安全性的規範，而它的實際運作簡單來說是針對非同源的請求所設立之規範，前端工程師使用 Javascript 存取非同源資料時，伺服器主機必須明確告訴瀏覽器是何種請求，只有伺服器主機允許的請求才能夠被瀏覽器發送。大多數的時候，處理方法都是請後端加入 response header，抽象解釋這個行為，就像是如果小名直接問小美她的生日會被拒絕，但是請小美的朋友小黃去問他，就不會有問題，而小名也能得到小美的生日。

有一件很特別的事情是，單發生 CORS 問題時，在控制台會顯示”If an opaque response serves your needs, set the request’s mode to ‘no-cors’ to fetch the resource with CORS disabled.”，可是當你把 FETCH 加上 NO-cors 的 mode 時，會發現不會報錯，但是還是拿不到資料，甚至 status 是 0，原來是因為 no-cros 參數的意思並不是繞過 cors 來取得資料，而是瀏覽器知道沒辦法繞過 cors，但是也沒差，不要回傳錯誤給工程師和 response，這裡直接請後端加上”res.header('Access-Control-Allow-Origin', '*')”代表人何 origin 網站皆可使用該 AJAX 資源是比較好的處理方式。