hw6-cors

跨來源資源共享（英語：Cross-origin resource sharing，縮寫：CORS），用於讓網頁的受限資源能夠被其他[域名](https://zh.wikipedia.org/wiki/%E5%9F%9F%E5%90%8D" \o "域名)的頁面存取的一種機制。

通過該機制，頁面能夠自由地使用不同源（英語：cross-origin）的圖片、樣式、指令碼、iframes以及影片。如跨域的請求（特別是Ajax）常常會被同源策略（英語：Same-origin policy）所禁止的。跨源資源共享定義了一種方式，為的是瀏覽器和伺服器之間能互相確認是否足夠安全以至於能使用跨源請求（英語：cross-origin requests）。比起純粹的同源請求，這將更為自由和功能性的，但比純粹的跨源請求更為安全。

跨來源資源共享是一份瀏覽器技術的規範，提供了 Web 服務從不同網域傳來[沙盒](https://zh.wikipedia.org/wiki/%E6%B2%99%E7%9B%92_(%E9%9B%BB%E8%85%A6%E5%AE%89%E5%85%A8)" \o "沙盒 (電腦安全))指令碼的方法，以避開瀏覽器的[同源策略](https://zh.wikipedia.org/wiki/%E5%90%8C%E6%BA%90%E7%AD%96%E7%95%A5)。

**狀況:**

1. 瀏覽器會擋你的跨來源請求，是因為安全性問題。因為 AJAX 你可以直接拿到整個 response，所以不擋的話會有問題，但像是 img 標籤你其實就拿不到 response，所以比較沒有問題
2. 今天會有 same-origin policy 跟 CORS，是因為我們「在瀏覽器上寫 JS」，所以受到執行環境的限制。如果我們今天寫的是 Node.js，就完全沒有這些問題，想拿什麼就拿什麼，不會有人擋我們
3. 在瀏覽器上面，CORS 限制的其實是「拿不到 response」，而不是「發不出 request」。所以 request 其實已經發出去了，瀏覽器也拿到 response 了，只是它因為安全性考量不給你（這講法也有一點不太精確，因為有分簡單請求跟非簡單請求，這個在第三篇會提到）。

**問題**:大部分情形下，CORS 都不是前端的問題，純前端是解決不了的。瀏覽器因為安全性的考量所以會把東西給擋住，因此，你必須要讓瀏覽器知道：「這其實是安全的」，它才會放行。

同源政策什麼的都只是「瀏覽器的限制」，一旦脫離了瀏覽器，就沒有任何限制了，proxy server 就是如此。

**解法**:Proxy server 的翻譯叫做代理伺服器，在不同的場合下用這個詞，代表的意思會有一點點不同，但是大方向都是一樣的。原本你是從 A 傳資料到 B，用代理以後變成你從 A 傳到 P（proxy server），P 再傳到 B，然後再回傳回來，中間那個 P 就擔任著「代理」的角色。

果你想拿 A 網站的資料，但是它沒有提供 Access-Control-Allow-Origin 這個 header，你就自己寫個 server，從後端去拿 A 網站的資料，再把資料丟回給自己的前端就行了。因為自己的後端可以自己控制，所以你想加什麼 header 就加什麼 header，想拿什麼資料就拿什麼。