當我們在 JavaScript 中透過 fetch 或 XMLHttpRequest 存取資源時，需要遵守 CORS (Cross-Origin Resource Sharing，跨來源資源共用)。瀏覽器在發送請求之前會先發送 preflight request (預檢請求)，確認伺服器端設定正確的 Access-Control-Allow-Methods、Access-Control-Allow-Headers 及 Access-Control-Allow-Origin 等 header，才會實際發送請求。使用 cookie 的情況下還需額外設定 Access-Control-Allow-Credentials header。需要注意的是，用 JavaScript 透過 fetch API 或 XMLHttpRequest 等方式發起 request，必須遵守同源政策。同源政策相同的通訊協定 、相同的網域、相同的通訊埠，當伺服器沒有正確設定時，請求就會因為違反 CORS 失敗，在 Chrome DevTool 就會看到以下的經典錯誤 CORS (Cross-Origin Resource Sharing) 是針對不同源的請求而定的規範，透過 JavaScript 存取非同源資源時，server 必須明確告知瀏覽器允許何種請求，只有 server 允許的請求能夠被瀏覽器實際發送，否則會失敗。在 CORS 的規範裡面，跨來源請求有分兩種：「簡單」的請求和非「簡單」的請求。所謂的「簡單」請求，必須符合下面兩個條件：第一只能是 HTTP GET, POST or HEAD 方法;第二自訂的 request header 只能是Accept、Accept-Language、Content-Language 或 Content-Type。違反簡單請求的地方有三個，分別是：第一是 http DELETE 方法；第二是 Content-Type 是 application/json；第三是帶了不合規範的 X-CUSTOM-HEADER。覽器發送跨來源請求時，會帶一個 Origin header，表示這個請求的來源。首先，瀏覽器發送跨來源請求時，會帶一個 Origin header，表示這個請求的來源。Origin 包含通訊協定、網域和通訊埠三個部分。當 server 端收到這個跨來源請求時，它可以依據「請求的來源」，亦即 Origin 的值，決定是否要允許這個跨來源請求。如果 server 允許這個跨來源請求，它可以「授權」給這個來源的 JavaScript 存取這個資源。授權的方法是在 response 裡加上 Access-Control-Allow-Origin header：如果 server 允許任何來源

的跨來源請求，那可以直接回 * ：當瀏覽器收到回應時，會檢查請求中的 Origin header 是否符合回應的 Access-Control-Allow-Origin header，相符的情況下瀏覽器就會讓這個請求成功，我們也可以順利地用 JavaScript 讀取到回應；反之，則瀏覽器會將這個 request 視為是不安全的而讓他失敗，即便 server 確實收到請求也成功地回應了，但基於安全性的理由 JavaScript 中沒有辦法讀到回應。非「簡單」的跨來源請求，例如：HTTP PUT/DELETE 方法，或是 Content-Type: application/json 等，瀏覽器在發送請求之前會先發送一個 「preflight request（預檢請求）」，其作用在於先問伺服器：你是否允許這樣的請求？真的允許的話，我才會把請求完整地送過去。Preflight request 是一個 http OPTIONS 方法，會帶有兩個 request header：Access-Control-Request-Method 和 Access-Control-Request-Headers。Server 必須告訴瀏覽器：我允許的方法和 header 有哪些。因此 Server 的回應必須帶有以下兩個 header。一般的 http request 會帶有該網域底下的 cookie；然而，跨來源請求預設是不能帶 cookie 的。因為帶有 cookie 的請求非常強大，如果請求攜帶的 cookie 是 session token，那這個請求可以以你的身份做很多機敏的事情，像是存取你的隱私資料、從你的銀行帳戶轉帳等。