跨來源資源共用（Cross-Origin Resource Sharing (CORS)）是一種使用額外 HTTP 標頭令目前瀏覽網站的使用者代理 (en-US)取得存取其他來源（網域）伺服器特定資源權限的機制。當使用者代理請求一個不是目前文件來源—例如來自於不同網域（domain）、通訊協定（protocol）或通訊埠（port）的資源時，會建立一個跨來源 HTTP 請求（cross-origin HTTP request）。

基於使用者的資訊安全，程式碼所發出的跨來源 HTTP 請求會受到限制。例如，XMLHttpRequest 及 Fetch 都遵守同源政策（same-origin policy）。這代表網路應用程式所使用的 API 除非使用 CORS 標頭，否則只能請求與應用程式相同網域的 HTTP 資源，這就是很常聽到的跨域請求問題，當開發者透過 JavaScript 針對不同於當前位置的來源發送請求，這個請求的回應就會被瀏覽器攔截掉，不交給 JavaScript 處理。這邊的「不同來源」，指的是目標資源與當前網頁的網域（domain）、通訊協定（protocol）或通訊埠（port）只要有任一項不同，就算是不同來源。瀏覽器要這麼雞婆把跨域請求資源擋掉，是為了防止在惡意開發者所撰寫的網站在運作的過程中，嘗試透過 XHR 打向 FaceBook、Instagram 等目標網站；如果使用者原先就有目標網站的登入狀態，這樣便能窺探該使用者的隱私，取得不該取得的資訊，若沒有瀏覽器限制跨域請求的保護，惡意開發者便能為所欲為。

關於跨域請求，解決方案有不少種，例如鼎鼎大名的 JSONP，就是透過 HTML 中沒有跨域限制的標籤如 img、script 等，再藉由指定 callback 函式，將回應內容介接回 JavaScript 中；或是透過 iframe，繞過跨域保護取得目標資源等；但最標準、正確的解決方法是透過跨來源資源共用，其運作方式是藉由加入新的 HTTP 標頭讓伺服器能夠描述來源資訊以提供予瀏覽器讀取，此機制提供了網頁伺服器跨網域的存取控制，增加跨網域資料傳輸的安全性。

跨域是實務上很常遇到的需求，畢竟網頁字體、WebGL 紋理、CSS 樣式表和指令碼等，都可以使用 CORS 開啟跨站 HTTP 請求，對於網站管理員、伺服器端開發者和前端網頁開發者，處理 CORS 也會有不同的做法，例如:前端開發者處理 CORS 有兩種作法：請後端工程師在 API 加入 CORS 標頭或使用 CORS Anywhere，如果能夠搞清楚 CORS 規範中的 Headers，並在伺服器做相對應的調整，就可以順利的完成跨域請求。

參考資料:
https://developer.mozilla.org/zh-TW/docs/Web/HTTP/CORS#%E8%AA%B0%E6%87%89%E8%A9%B2%E9%96%B1%E8%AE%80%E9%80%99%E7%AF%87%E6%96%87%E7%AB%A0%EF%BC%9F
https://ithelp.ithome.com.tw/articles/10226262