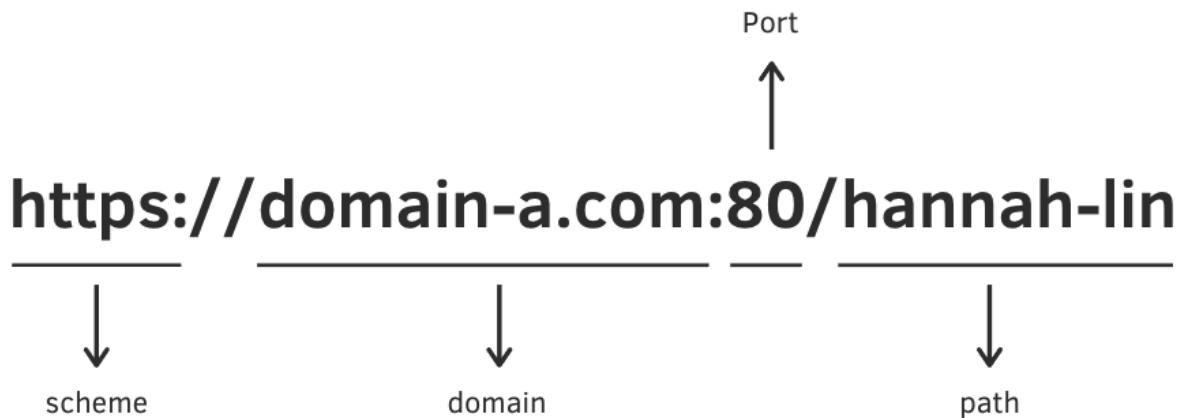


首先我們來認識瀏覽器的「同源政策」。當使用 JavaScript 透過 fetch API 或 XMLHttpRequest 等方式發起 request，必須遵守同源政策（same-origin policy）。同源政策是網站安全的基礎。例如 <https://domain-a.com> 只能存取自己網站裡的資源，而不允許網站 <https://domain-b.com> 來存取。只要通訊協定(protocol(scheme))、網域(domain)和通訊埠(port)一樣就會被視為同源 (same-origin)，其他則是不同源。



圖片來源: https://miro.medium.com/max/1896/1*2OD4i1N4B7aPDzZGRMEi8Q.png

而跨來源資源共用(Cross-Origin Resource Sharing (簡稱CORS))是一種使用額外 HTTP 標頭令目前瀏覽網站的使用者代理取得存取其他來源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，會建立一個跨來源 HTTP 請求cross-origin HTTP request。簡單地說，CORS 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。

瀏覽器將CORS請求分成兩類：簡單請求(simple request)和非簡單請求(not-so-simple request)。只要同時滿足以下兩大條件，就屬於簡單請求。

1. 請求方法是以下三種方法之一：HEAD、GET、POST。
2. HTTP的頭信息不超出以下幾種欄位：Accept、Accept-Language、Content-Language、Last-Event-ID、Content-Type(值只能是application/x-www-form-urlencoded、multipart/form-data、text/plain)

凡是不同時滿足上面兩個條件，就屬於非簡單請求。瀏覽器對這兩種請求的處理並不一樣。

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。

而非「簡單」的跨來源請求, 例如:HTTP PUT/DELETE 方法, 瀏覽器在發送請求之前會先發送一個「預檢請求(preflight request)」, 其作用在於先問伺服器是否允許這樣的請求, 允許的話, 才會把請求完整地送過去。預檢請求(preflight request)是一個 http OPTIONS 方法, 會帶有兩個 request header: Access-Control-Request-Method(非「簡單」跨來源請求的 HTTP 方法)和 Access-Control-Request-Headers(非「簡單」跨來源請求帶有的非「簡單」header)。

參考:

[跨來源資源共用\(CORS\) - HTTP | MDN \(mozilla.org\)](#)

[跨來源資源共享 - 維基百科, 自由的百科全書 \(wikipedia.org\)](#)

[簡單弄懂同源政策 \(Same Origin Policy\) 與跨網域 \(CORS\) | by Hannah Lin | Starbugs Weekly 星巴哥
技術專欄 | Medium](#)

[跨來源資源共用 \(CORS\) 是什麼? 如何設定 CORS? - Shubo 的程式開發筆記](#)

[跨域资源共享 CORS 详解 - 阮一峰的网络日志 \(ruanyifeng.com\)](#)