



CORS

Cross-Origin Resource Sharing

.NET CORE

Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to tell browsers to give access to a web application running at one origin to selected resources from a different origin.

[HTTPS://DEVELOPER.MOZILLA.ORG/EN-US/DOCS/WEB/HTTP/CORS](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS)

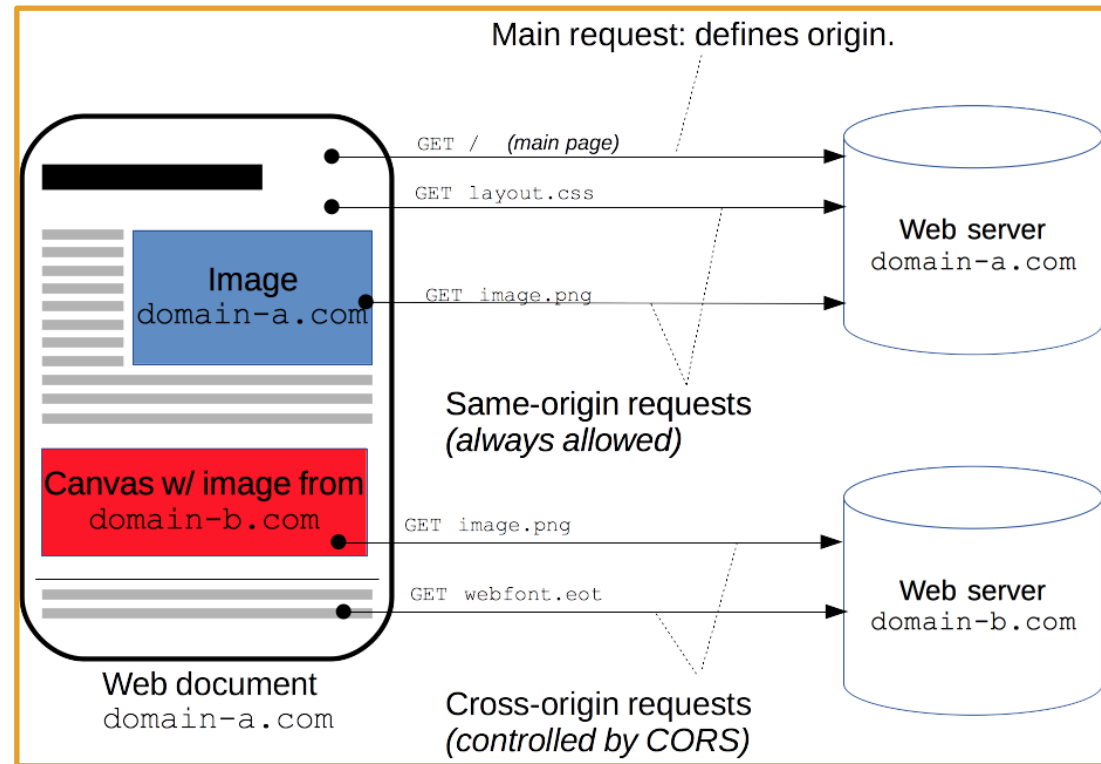
CORS - Overview

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

The specification for **CORS** is included as part of the WHATWG's **Fetch** Living Standard.

CORS is implemented if a document or web page needs resources from more than one source.

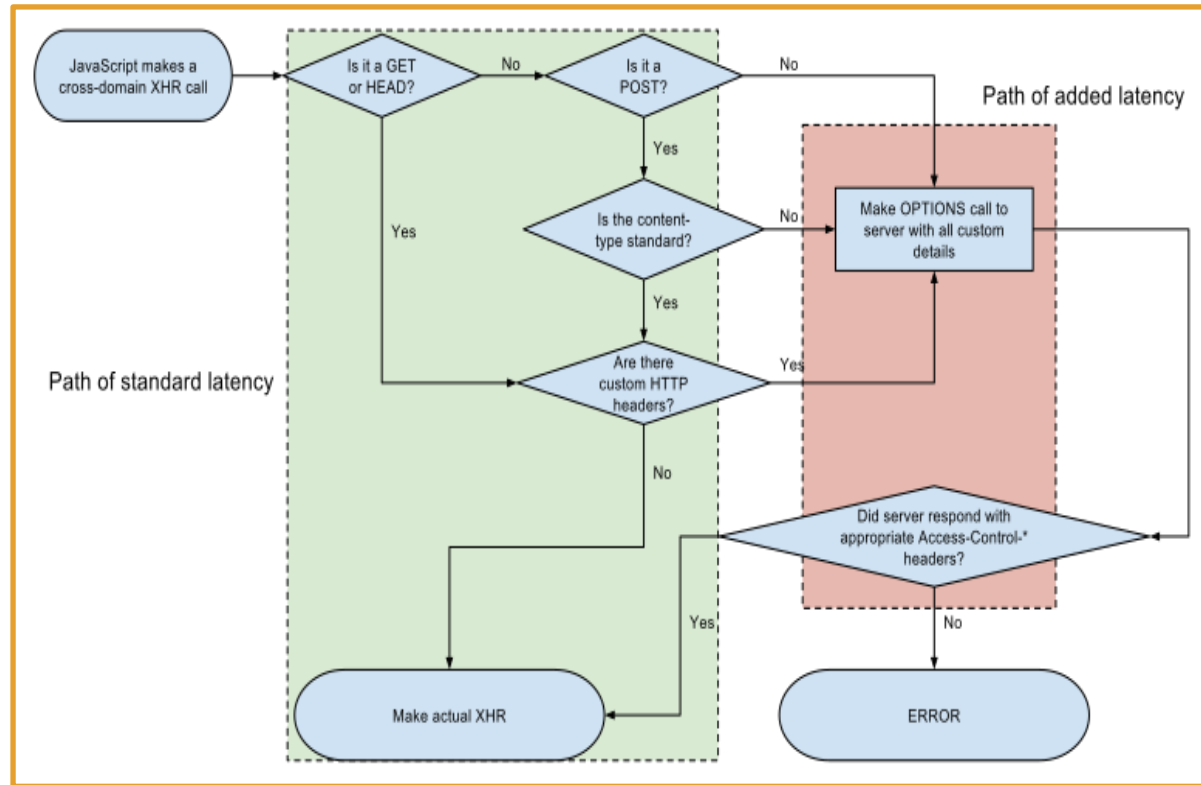
This was originally not allowed (for security reasons) but the **CORS Standard** has put protocols in place to make relax security and make **CORS** safe.



CORS Standard Protocols

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

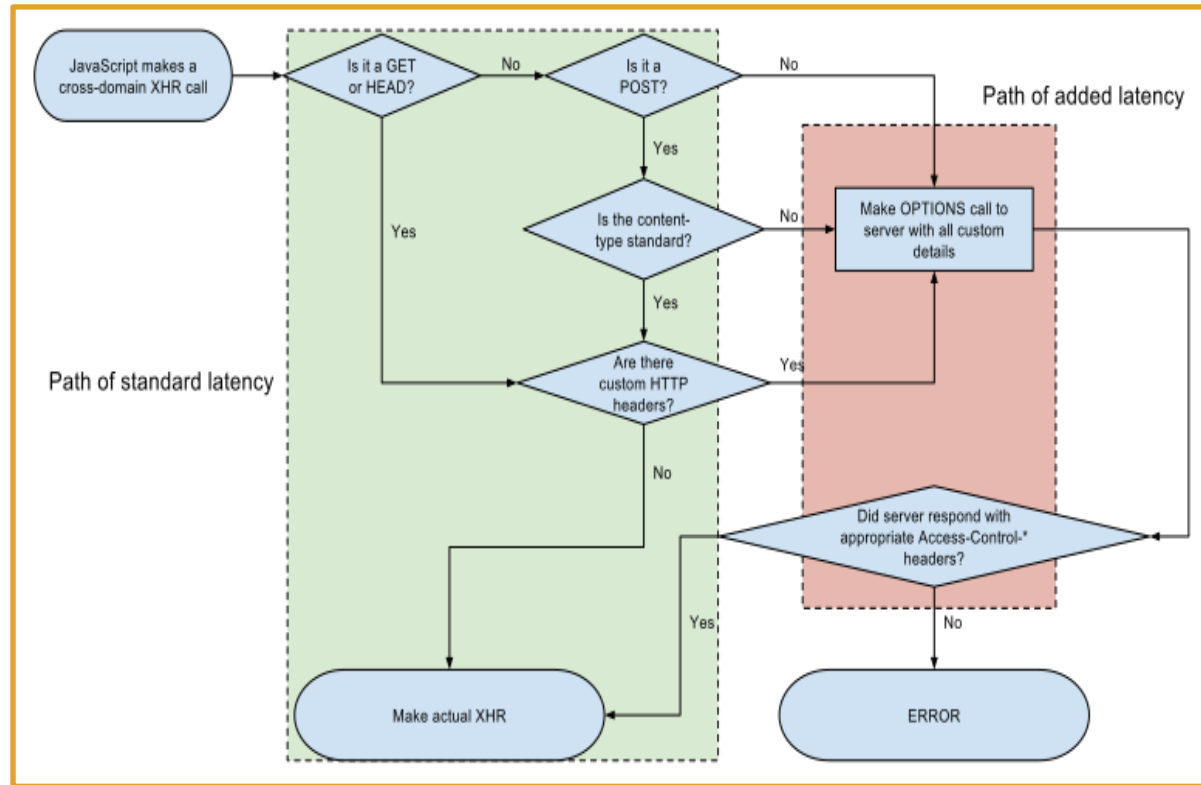
- The **CORS** Standard adds new HTTP headers
 - servers describe which origins are permitted to read information from web browsers.
- Browsers must "preflight" any requests that can alter data.
 - Browsers solicit supported methods from the server. Upon "approval" from the server, the browser sends the actual request.



CORS Standard Protocols

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

- Servers can inform clients whether "credentials" (like Cookies or HTTP Authentication) should be sent with requests.
- CORS failures result in errors (rejected HTTP Requests) but the details are not available on JavaScript for security reasons.
- You can see what went wrong in the Browsers Console display.



TODO - CORS in .NET

<https://docs.microsoft.com/en-us/aspnet/core/security/cors?view=aspnetcore-3.1>

CORS Simple Example

https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS> (more examples)

1. A user visits <http://www.revature.com>. The Revature.com needs to validate the information of the user so it sends a request to another site, http://www.verify_users.com, to verify the data. This is a cross-origin request.
2. Browsers are the enforcers of CORS. A CORS-compatible browser will attempt to make a cross-origin request to www.verify_users.com.
3. The browser sends the GET request with an extra **Origin:** HTTP header to www.verify_users.com. This extra header looks like this - **Origin: http://www.revature.com**.
4. Depending on www.verify_users.com's policies, it can respond in three ways.
 1. With the requested data and another header, **Access-Control-Allow-Origin: http://www.revature.com**. This response says that only <http://www.revature.com> is allowed to use the data. [Www.verify_users.com](http://www.verify_users.com) uses **CORS** to permit the browser to authorize www.revature.com to make requests to www.verify_users.com.
 2. With the requested data and another header, **Access-Control-Allow-Origin: ***. The * is a 'wildcard' that allows any other site to access the data.
 3. An error.