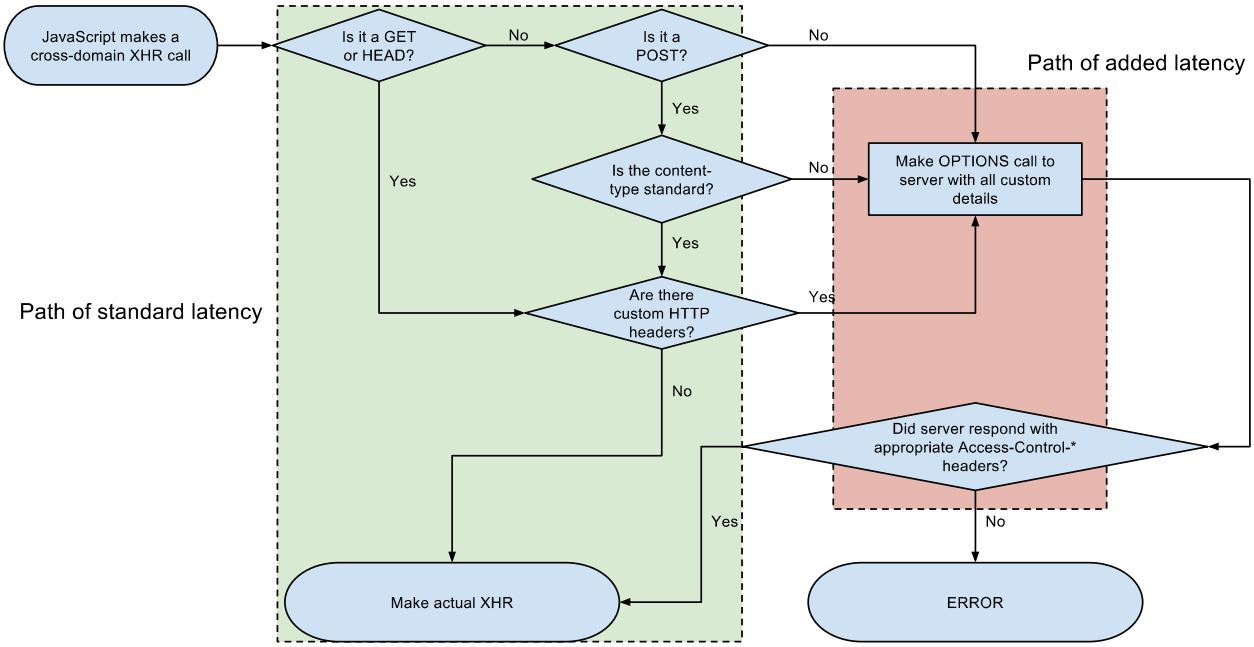
Cross-Domain requests met javascript.

Cross-Domain

Als je bijvoorbeeld een Ajax request naar een ander domein maakt is dat een cross-domain request. Dus als een script geladen in een browser op adres a.com een Ajax request doet naar een server "b.com" spreken we over een cross-domain request. **Enkel bepaalde cross-domain requests zijn toegelaten *in browsers*. Namelijk GET en HEAD cross-domain requests zonder custom(speciale) headers zijn toegelaten, alle andere niet**. Zie volgende figuur.



bron: https://upload.wikimedia.org/wikipedia/commons/c/ca/Flowchart_showing_Simple_and_Preflight_XHR.svg

(Merk op dat men buiten browsers elk domein kan nabootsen via custom http requests met één of andere programmeertaal te implementeren. Met andere woorden kan men buiten browsers deze requests toch maken, deze beperkingen zijn geimplementeerd in alle moderne browsers: Chrome, FireFox, IE,...)

Men kan het op twee manieren, die nu volgen, oplossen om toch cross-domain requests toe te laten.

CORS

Een manier om toch een cross-domain toe te laten is CORS. (Het rode gedeelte in vorige figuur.) Hierbij stuurt de client eerst een OPTIONS request met daarin een "Origin" header. De server kan hierop controleren welk domein deze header bevat en al dan niet een antwoord sturen om deze aanvraag toe te laten. Deze voorafgaande request heet men de preflight. Hier ziet u een mooi voorbeeld van de request en resonses:

```
OPTIONS /resources/post-here/ HTTP/1.1
Host: bar.other
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US;
rv:1.9.1b3pre) Gecko/20081130 Minefield/3.1b3pre
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
```

```
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
Origin: http://foo.example
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER


HTTP/1.1 200 OK
Date: Mon, 01 Dec 2008 01:15:39 GMT
Server: Apache/2.0.61 (Unix)
Access-Control-Allow-Origin: http://foo.example
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGOTHER
Access-Control-Max-Age: 86400
Vary: Accept-Encoding, Origin
Content-Encoding: gzip
Content-Length: 0
Keep-Alive: timeout=2, max=100
Connection: Keep-Alive
Content-Type: text/plain

POST /resources/post-here/ HTTP/1.1
Host: bar.other
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US;
rv:1.9.1b3pre) Gecko/20081130 Minefield/3.1b3pre
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: keep-alive
X-PINGOTHER: pingpong
Content-Type: text/xml; charset=UTF-8
Referer: http://foo.example/examples/preflightInvocation.html
Content-Length: 55
Origin: http://foo.example
Pragma: no-cache
Cache-Control: no-cache

<?xml version="1.0"?><person><name>Arun</name></person>


HTTP/1.1 200 OK
Date: Mon, 01 Dec 2008 01:15:40 GMT
Server: Apache/2.0.61 (Unix)
Access-Control-Allow-Origin: http://foo.example
Vary: Accept-Encoding, Origin
Content-Encoding: gzip
Content-Length: 235
Keep-Alive: timeout=2, max=99
Connection: Keep-Alive
Content-Type: text/plain

[Some GZIP'd payload]
```

bron: https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS

Het implementeren van CORS hangt af van de serveromgeving en eventueel bijhorende omgeving/platform. Dus het is soms mogelijk dit te configureren op de server al dan niet gebruik makende van modules. Maar het is ook mogelijk dit software matig te implementeren op de server. Dit beide afhankelijk van omgeving.

JSONP

Een andere manier, volledig softwarematig bij mijn weten, is JSONP. Hierbij gaat de client geen Ajax request doen maar een javascript toevoegen door middel van een *script* tag aan de webpagina. Dit laat toe een script te laden vanop de server met een gewone GET request welke toegelaten is cross-domain. In het src attribuut kan men in de URL parameters mee geven aan de server. Naast parameters in verband met de request geeft men meestal een functie naam mee. De client moet ook deze functie voorzien die zal aangeroepen worden bij een antwoord van de server. (Voor de functie naam zie callback in het voorbeeld. Indien men deze niet mee geeft is deze naam vast bepaald op de server.) Zie hier een voorbeeld van een JSONP request op de client.

```
function makeRequest() {
    var script = document.createElement('script');
    script.src = '//b.com/jsonscript.php?callback=foo';
    document.getElementsByTagName('head')[0].appendChild(script);
}

function foo(data) {
    alert(data.message);
}
```

De server antwoord dan met een script die deze functie aanroept. Zie dit php script voor een voorbeeld van een JSONP response:

```
header('Content-Type: application/javascript');
echo $_GET['callback'] . '({\"message\":\"jsonp says hello\"});';
```

References

- https://en.wikipedia.org/wiki/Cross-origin_resource_sharing
- https://upload.wikimedia.org/wikipedia/commons/c/ca/Flowchart_showing_Simple_and_Preflight_XHR.svg
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS
- http://stackoverflow.com/questions/6132796/how-to-make-a-jsonp-request-from-javascript-without-jquery
- http://stackoverflow.com/questions/6809053/simple-jquery-php-and-jsonp-example
- http://stackoverflow.com/questions/111302/best-content-type-to-serve-jsonp