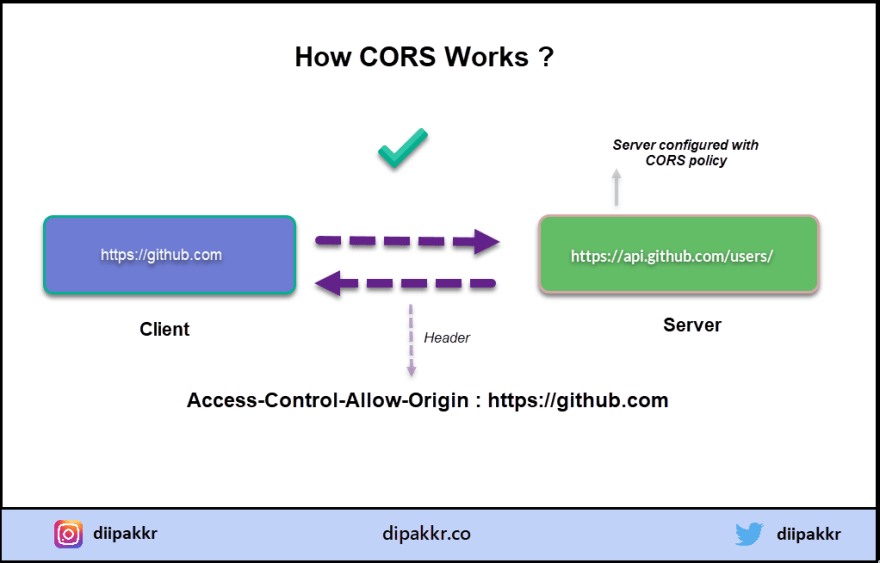
# CORS

Cross origin resource sharing

Cross-Origin Resource Sharing (CORS) is a protocol that enables scripts running on a browser client to interact with resources from a different origin. This is useful because, thanks to the [same-origin policy](https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy) followed by XML Http Request and fetch, JavaScript can only make calls to URLs that live on the same origin as the location where the script is running. For example, if a JavaScript app wishes to make an AJAX call to an API running on a different domain, it would be blocked from doing so thanks to the same-origin policy.



## CORS - Why Is It Needed?

Most of the time, a script running in the user's browser would only ever need to access resources on the same origin (think about API calls to the same backend that served the JavaScript code in the first place). So the fact that JavaScript can't normally access resources on other origins is a good thing for security.

In this context, "other origins" means the URL being accessed differs from the location that the JavaScript is running from, by having:

* **a different scheme (HTTP or HTTPS)**
* **a different domain**
* **a different port**

## Identifying a CORS Response

When a server has been configured correctly to allow cross-origin resource sharing, some special headers will be included. Their presence can be used to determine that a request supports CORS. Web browsers can use these headers to determine whether or not an XML Http Request call should continue or fail.

There are [a few headers that can be set](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS#The_HTTP_response_headers), but the primary one that determines who can access a resource is [Access-Control-Allow-Origin](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS#Access-Control-Allow-Origin). This header specifies which origins can access the resource. For example, to allow access from any origin, you can set this header as follows:

Ac[Access-Control-Allow-Origin](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS#Access-Control-Allow-Origin): \*

Access-Control-Allow-Origin: <https://example.com>

## CORS Request Types:

There are two types of CORS request: "simple" requests, and "preflight" requests, and it's the browser that determines which is used. As the developer, you don't normally need to care about this when you are constructing requests to be sent to a server. However, you may see the different types of requests appear in your network log and, since it may have a performance impact on your application, it may benefit you to know why and when these requests are sent.

### **Simple requests (GET, POST, and HEAD)**

The browser deems the request to be a ["simple" request](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS#Simple_requests) when the request itself meets a certain set of requirements:

* One of these methods is used: GET, POST, or HEAD
* A [CORS safe-listed header](https://fetch.spec.whatwg.org/#cors-safelisted-request-header) is used
* When using the Content-Type header, only the following values are allowed: application/x-www-form-urlencoded, multipart/form-data, or text/plain
* No event listeners are registered on any XMLHttpRequestUpload object
* No [ReadableStream](https://developer.mozilla.org/en-US/docs/Web/API/ReadableStream" \t "_blank) object is used in the request

The request is allowed to continue as normal if it meets these criteria, and the Access-Control-Allow-Origin header is checked when the response is returned.

### **Preflight requests (OPTIONS)**

If a request does not meet the criteria for a simple request, the browser will instead make an automatic [preflight request](https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS" \l "Preflighted_requests" \t "_blank) using the OPTIONS method. This call is used to determine the exact CORS capabilities of the server, which is in turn used to determine whether or not the intended CORS protocol is understood. If the result of the OPTIONS call dictates that the request cannot be made, the actual request to the server will not be executed.

The preflight request sets the mode as OPTIONS and sets a couple of headers to describe the actual request that is to follow:

* Access-Control-Request-Method: The intended method of the request (e.g., GET or POST)
* Access-Control-Request-Headers: An indication of the custom headers that will be sent with the request
* Origin: The usual origin header that contains the script's current origin

An example of such a request might look like this:

# Request

curl -i -X OPTIONS localhost:3001/api/ping \

-H 'Access-Control-Request-Method: GET' \

-H 'Access-Control-Request-Headers: Content-Type, Accept' \

-H 'Origin: http://localhost:3000'

This request basically says "I would like to make a GET request with the Content-Type and Accept headers from http://localhost:3000 - is that possible?".

The server will include some Access-Control-\* headers within the response to indicate whether the request that follows will be allowed or not. These include:

* Access-Control-Allow-Origin: The origin that is allowed to make the request, or \* if a request can be made from any origin
* Access-Control-Allow-Methods: A comma-separated list of HTTP methods that are allowed
* Access-Control-Allow-Headers: A comma-separated list of the custom headers that are allowed to be sent
* Access-Control-Max-Age: The maximum duration that the response to the preflight request can be cached before another call is made

The response would then be examined by the browser to decide whether to continue with the request or to abandon it.

So a response to the earlier example might look like this:

HTTP/1.1 204 No Content

Access-Control-Allow-Origin: \*

Access-Control-Allow-Methods: GET,HEAD,PUT,PATCH,POST,DELETE

Vary: Access-Control-Request-Headers

Access-Control-Allow-Headers: Content-Type, Accept

Content-Length: 0

Date: Fri, 05 Apr 2019 11:41:08 GMT

Connection: keep-alive

The Access-Control-Allow-Origin header, in this case, allows the request to be made from any origin, while the Access-Control-Allow-Methods header describes only the accepted HTTP methods. If a given HTTP method is not accepted, it will not appear in this list.

In this example, Access-Control-Allow-Headers echos back the headers that were asked for in the OPTIONS request. This indicates that all the requested headers are allowed to be sent. If for example, the server doesn't allow the Accept header, then that header would be omitted from the response and the browser would reject the call.

# AUTH

Authentication is common way to handle security for all applications. This is only way to answer the question “who you are?” to the application

Allow-Origin