**Authorization & CORS**

**Cross-Origin Resource Sharing (CORS):**

Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to tell browsers to give a web application running at one origin, access to selected resources from a different origin. A web application executes a cross-origin HTTP request when it requests a resource that has a different origin (domain, protocol, or port) from its own.

An example of a cross-origin request: the front-end JavaScript code served from https://domain-a.com uses XMLHttpRequest to make a request for https://domain-b.com/data.json.

For security reasons, browsers restrict cross-origin HTTP requests initiated from scripts. For example, XMLHttpRequest and the Fetch API follow the same-origin policy. This means that a web application using those APIs can only request resources from the same origin the application was loaded from unless the response from other origins includes the right CORS headers.

**Authorization:**

The Authorization HTTP header provides authentication information on a request. There are several types of authentication that use this header, and some are supported by browsers, such as basic authentication. When an unauthenticated request is received by the server, it will respond with an HTTP 401 Unauthorized response with a WWW-Authenticate header. This will trigger the browser to ask the user for credentials. The browser will then perform the same request but include an Authorization header with the entered credentials.

In contrast, some applications use the Authorization header without any intervening from the browser. A JavaScript app may obtain a token from the server and send that with each request to authenticate the request. This if called bearer authentication and the Authorization header is often used to send the token.