**Fetch: Cross-Origin Requests**

Fetch: Cross-Origin is a topic that relates to how web browsers handle requests to resources from different origins, such as different domains, protocols, or ports.

Cross-Origin Requests in the context of the Fetch API refer to HTTP requests made by a web page or web application to a different origin or domain than the one from which the web page originated. This concept is closely related to Cross-Origin Resource Sharing (CORS), which is a security feature implemented by web browsers to control and secure such requests.

Here's how Cross-Origin Requests work with the Fetch API:

1. Same-Origin Policy: By default, web browsers adhere to the Same-Origin Policy (SOP), which restricts web pages from making requests to a different origin than the one that served the web page. An "origin" typically includes the same protocol (HTTP/HTTPS), domain, and port. Requests to a different origin are blocked by the browser to prevent potential security vulnerabilities.

2. Cross-Origin Requests with Fetch: When you use the Fetch API to make a network request from a web page, the browser checks if the request is a cross-origin request. If it is, the browser enforces security measures, and the request may be blocked or allowed based on CORS headers sent by the server hosting the resource.

3. CORS Headers: To enable cross-origin requests, the server that hosts the requested resource must include specific HTTP response headers known as CORS headers. These headers indicate which origins are permitted to access the resource. The most important CORS header is the `Access-Control-Allow-Origin` header, which specifies the origins that are allowed to access the resource.

For example, a server might include the following header in its response to allow access from any origin:

Access-Control-Allow-Origin:

Or it might specify a specific origin:

Access-Control-Allow-Origin:

4. Preflight Requests: Some cross-origin requests are considered "complex" and trigger a preflight request. This preflight request is an HTTP OPTIONS request that the browser sends to the server before making the actual request. The server responds to the preflight request with CORS headers to indicate whether the actual request is permitted. Only if the preflight request is successful will the browser proceed with the actual request.

5. Credentials: By default, CORS requests do not include credentials (e.g., cookies or HTTP authentication headers) to prevent unauthorized access. If a request needs to include credentials, both the client (JavaScript code) and the server must specify this using appropriate CORS headers and JavaScript options.

Here's an example of using the Fetch API to make a cross-origin request:

Javascript:

fetch('https://api.example.com/data', {

method: 'GET',

mode: 'cors', // Enables CORS handling

credentials: 'include', // Include credentials (e.g., cookies)

})

.then(response => {

if (!response.ok) {

throw new Error('Network response was not ok');

}

return response.json();

})

.then(data => {

// Process the response data

})

.catch(error => {

console.error('Fetch error:', error);

});

In summary, when making cross-origin requests using the Fetch API, it's important to be aware of CORS restrictions and ensure that the server hosting the requested resource is configured to allow access from the intended origins. This helps maintain security while enabling communication between web applications and different servers.