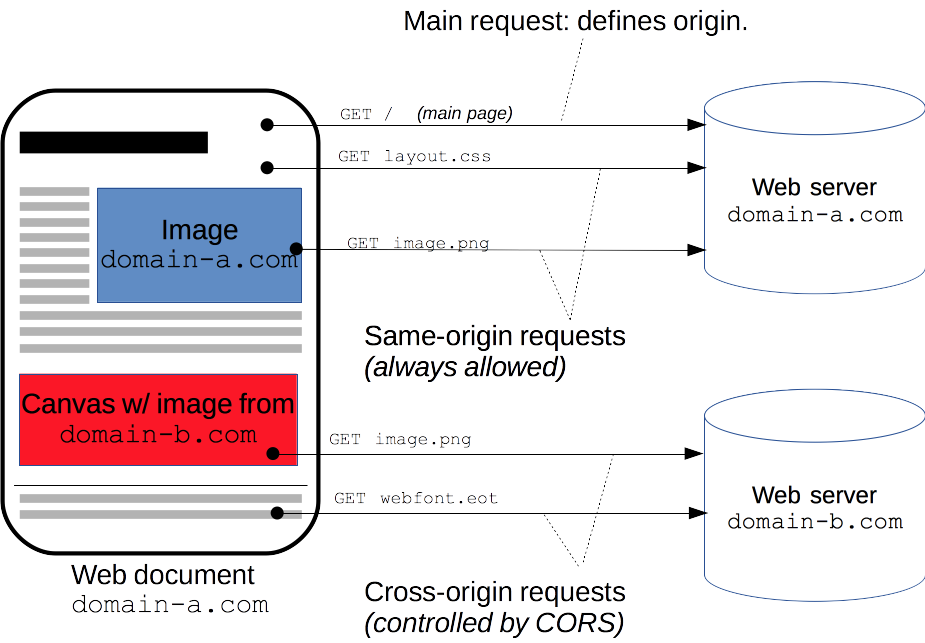
**AUTH and CORS in javascript**

**Cross-Origin Resource Sharing** ([CORS](https://developer.mozilla.org/en-US/docs/Glossary/CORS)) is a mechanism that uses additional [HTTP](https://developer.mozilla.org/en-US/docs/Glossary/HTTP) headers to tell browsers to give a web application running at one [origin](https://developer.mozilla.org/en-US/docs/Glossary/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](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest) to make a request for <https://domain-b.com/data.json>.



The CORS mechanism supports secure cross-origin requests and data transfers between browsers and servers. Modern browsers use CORS in APIs such as XMLHttpRequest or [Fetch](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API) to mitigate the risks of cross-origin HTTP requests.

The Cross-Origin Resource Sharing standard works by adding new [HTTP headers](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers) that let servers describe which origins are permitted to read that information from a web browser. Additionally, for HTTP request methods that can cause side-effects on server data (in particular, HTTP methods other than [GET](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET), or [POST](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST) with certain [MIME types](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types)).

**HTTP AUTHENTICATION**

HTTP Authentication provides mechanism to protect web pages and resources.

* **Basic**

The browser sends the username and password as Base64-encoded text, without any encryption. **Basic authentication** should only be used with HTTPS, otherwise the password can be exposed to everyone.

If the client request protected resource without providing credentials, the server will reject the request and send back **401** HTTP status and **WWW-Authenticate** header.

HTTP/1.1 401 Unauthorized

WWW-Authenticate: Basic realm="Restricted area"

Then the browser will display popup asking for user credentials used to retry the request with **Authorization** header.

Authorization: Basic bXl1c2VyOm15cHN3ZA==

* **Digest**

The client sends the hashed variant of the username and password. Encryption instead of encoding makes the **digest authentication** safer than basic auth.

jQuery API provides username and password parameters as part of the ajax settings object, which intends to do the job instead of sending a header by yourself, unfortunatelly they doesn't works.

### RESPONSE

To trigger the basic authentication use your prefered method.

### CROSS-DOMAIN REQUESTS WITH CORS

In case, the protected resource or page is accessible through a domain that differs from the origin, a restriction from same origin policy is applied. To circumvent the same-origin policy, use the [Cross-origin resource sharing](https://zinoui.com/blog/cross-origin-resource-sharing).