Qu'est-ce que CORS ?

CORS est un acronyme qui signifie Cross-Origin Resource Sharing.

Cela signifie que chaque fois que vous essayez de mettre sur votre site Web, avec une URL X, de demander diverses ressources à partir d'une URL Y. Un exemple classique est celui du site foo.com qui demande l'image souhaitée au site bar.com.

Il est basé sur le protocole HTTP, et son mécanisme est basé sur le HEADER fourni par le protocole. Il vérifie si l'entité requérante a le droit d'obtenir la ressource demandée.

Chaque fois que vous exécutez un fetch() , un XMLHTTPRequest ou ou axios en Javascript, vous utilisez CORS.

Comment cela fonctionne-t-il ?

Le mécanisme CORS, selon les règles principales et programmées, si l'entité appelante peut récupérer l'image qu'elle veut, à l'ensemble des données dont elle a besoin.

Cela peut être fait en manipulant l'origine (domaine, port, schéma) ou d'autres paramètres, y compris le type de données nécessaires, ou le type de demande.

Vous pouvez donc le considérer comme un filtre qui n'accepte que certains types de demandes, en fonction de ce que vous voulez.

Il y a quelques points principaux que vous devez connaître pour bien comprendre ce mécanisme. Il s'agit des points suivants :

Politique de même origine

Presque tous les navigateurs ont adopté la politique de confidentialité qu'apporte CORS. Le point principal de cette politique particulière est la politique de même origine.

Cela signifie que CORS autorise les demandes provenant de la même origine et partage librement les ressources avec elles. Elle bloque également tout ce qui provient d'une URL externe, sauf si certaines conditions sont remplies.

Access-Control-Allowed-Origin

Ce terme particulier est utilisé par le serveur pour vérifier si une requête entrante a été envoyée par un client qui a le droit de récupérer la ressource.

Il s'agit généralement d'une liste d'URL ou d'un caractère générique.

Lorsque le serveur reçoit une demande, il génère la réponse, avec cet attribut particulier défini dynamiquement. À ce stade, le serveur doit vérifier si l'origine de la demande est la même que celle de la réponse.

Si c'est le cas, il n'y a pas de problème et la réponse est envoyée. Dans le cas contraire, une erreur de console s'affiche dans le navigateur, mais avec seulement quelques détails pour une question de sécurité. Il ne s'agit donc guère de débogage, si vous ne connaissez pas CORS.

Si le paramètre est le joker, toute origine est acceptée.

Méthodes autorisées par le contrôle d'accès

Il s'agit du même argumentaire que le dernier dont nous avons discuté, mais il fait référence aux méthodes HTTP. Il fonctionne exactement de la même manière que le pitch discuté précédemment.

Il s'agit d'une liste de différentes méthodes possibles. Il peut s'agir d'un exemple de la ligne de la réponse dont nous parlons :

Access-Control-Allowed-Methods : GET, POST, PUT

Dans ce cas, si je fais une demande DELETE, le système la prévient et génère une erreur. Sinon, il passe ce contrôle et continue à créer la réponse.

Autres "filtres" CORS possibles

Il existe un grand nombre d'autres "filtres" CORS que vous pouvez définir, pour éviter de recevoir des demandes non appréciées que le serveur peut résoudre manuellement. Voici les plus connus :

Access-Control-Allowed-Headers

Access-Control-Max-Age

Accept-Language

Content-Language

Content-Type

Exemple pour les réglages dans node-express:

```
const express = require('express');

const cors = require('cors');

app = express(); app.use(cors());

// ici cors() sans option ne filtre pas
```

Pour gérer ces erreurs, il existe de nombreuses réponses de StackOverflow aux questions liées à CORS. La plupart du temps, elles consistent à rendre les règles moins restrictives, ce qui rend également le site web moins sûr.

Donc, si vous devez résoudre un bug CORS, faites également attention à la partie sécurité.