2 minutes

Utilisations avancées de XMLHttpRequest

Définir un `timeout`

Vous pouvez définir un `timeout` en millisecondes pour la requête.

Si la période précisée est dépassée et qu'aucune réponse n'est reçue, la requête est annulée :

```
requete.timeout = 60000; // 1 minute
```

Annuler une requête en cours

Vous pouvez également annuler une requête en cours avec la méthode `abort()` :

```
requete.abort();
```

Vous pouvez savoir si une requête a été annulée avec l'événement `abord`, comms nous avons vu :

```
requete.addEventListener("abort", abort => console.log("Annulée"));
requete.abort();
```

Suivre la progression d'un téléchargement vers un serveur (`upload`)

Avec `XMLHttpRequest` vous pouvez suivre le téléchargement d'un fichier sur un chargeur ce qui peut être très pratique.

Pour ce faire, il faut écouter les événements émis sur la propriété `upload` :

```
requete.upload
```

Les événements possibles sont les suivants :

`loadstart` : émis lorsque le téléchargement est lancé.

`progress` : émis pendant le téléchargement pour suivre le progrès.

`abort` : émis si le téléchargement est annulé.

`error` : émis si il y a eu une erreur lors du téléchargement, avant la réponse du serveur.

`load` : émis si le téléchargement est réussi.

`loadend` : émis si le téléchargement est terminé, réussi ou échoué.

`timeout` : émis si le téléchargement est `timeout`.

Voici un exemple complet :
https://codesandbox.io/embed/js-c15-l10-rqzsw

Requête `CORS`

Il est possible de faire des requêtes `CORS` facilement avec `XMLHttpRequest`.

Vous n'avez rien de particulier à faire vu que le paramétrage est côté serveur.

Si vous voulez envoyer les `credentials` (`cookies` et entêtes d'authentification), il suffit d'ajouter :

```
requete.withCredentials = true;
```