3 minutes

XMLHttpRequest

L'API XMLHttpRequest

La Web API XMLHttpRequest permet également d'effectuer des requêtes AJAX comme fetch.

Il s'agit d'une API plus ancienne qui n'utilise pas les promesses.

Dans la majorité des cas nous vous conseillons d'utiliser l'API fetch.

Une exception est le besoin d'obtenir l'état d'avancement d'un upload qui n'est pas encore possible avec fetch.

Le nom XML est historique mais vous pouvez utiliser tous les formats de données supportés, par exemple JSON, FormData etc.

Étapes d'une requête XMLHttpRequest

Etape 1 - Pour commencer il faut créer un objet XMLHttpRequest :

```
const requete = new XMLHttpRequest();
```

Etape 2 - Il faut ensuite paramétrer la requête avec la méthode open() :

```
requete.open(méthode, URL, async, user, password)
```

Etape 3 - Vous ne devez préciser que la méthode et l'URL. Les autres options ne sont plus utilisées.

Il faut ensuite l'envoyer :

```
requete.send(body);
```

Pour rappel une requête GET ne prend pas de body.

Etape 4 - Il faut écouter les événements sur la requête pour les gérer. Les principaux sont load, error et progress.

Gérer les événements de la requête

Pour gérer les événements de la requête, avez d'abord à ajouter des écouteurs :

```
requete.addEventListener("progress", progress => console.log(progress));
requete.addEventListener("load", load => console.log("Terminé"));
requete.addEventListener("error", error => console.log("Erreur"));
requete.addEventListener("abort", abort => console.log("Annulée"));
```

L'événement `progress` permet de suivre la progression du transfert de données. Nous verrons un exemple dans la leçon suivante.

L'événement `load` permet de savoir lorsque la requête est terminée.

L'événement `error` permet de savoir si il y a eu une erreur lors de l'envoi.

L'événement `abort` permet d'être notifié si la requête est annulée.

Voici un exemple :
https://codesandbox.io/embed/js-c15-l9-1-p7q7l

Les types de réponse

Vous pouvez spécifier le type de la réponse attendue, elle sera alors automatiquement parsée au bon format :

```
requete.responseType = 'json';
requete.responseType = 'text';
requete.responseType = 'arraybuffer';
requete.responseType = 'blob';
```

Par défaut, si vous ne précisez pas, la réponse sera considérée au format `text` et vous aurez donc une chaîne de caractères.

Les états de la requête

Vous pouvez connaître où en est la requête avec la propriété `readyState` de l'objet.

Les états possibles sont :

```
UNSENT = 0;
OPENED = 1;
HEADERS_RECEIVED = 2;
```

```
LOADING = 3;
DONE = 4;
```

Vous pouvez suivre le changement entre les états en utilisant l'événement `readystatechange` :

```
requete.onreadystatechange = () => {
  if (requete.readyState === 1) {
    // Requête configurée
  }
  if (requete.readyState === 2) {
    // Entêtes de la réponse reçues
  }
  if (requete.readyState === 3) {
    // Chargement en cours
  }
  if (requete.readyState == 4) {
    // Terminée
  }
};
```

Définir les entêtes de la requête

Vous pouvez définir les entêtes de la requête en utilisant la méthode `setRequestHeader()` :

```
requete.setRequestHeader('Content-Type', 'application/json');
```

Lire les entêtes de la réponse

Vous pouvez également lire les entêtes de la réponse facilement avec les méthodes **`getResponseHeader()`** et **`getAllResponseHeaders()`** :

```
...
requete.onload = () => {
  if (requete.status !== 200) {
    console.error(`Erreur ${requete.status}: ${requete.statusText}`);
  } else {
    console.log(requete.getResponseHeader('Content-Type'));
    console.log(requete.getAllResponseHeaders());
```

```
  }
};
```