3 minutes

Options avancées des requêtes fetch

Toutes les options de `fetch`

Nous vous conseillons de lire la leçon brièvement et d'y revenir plus tard. Ce sont en effet des options très avancées que vous n'aurez pas l'occasion d'utiliser dans l'immédiat.

La liste complète des options qu'il est possible de passer à `fetch` est la suivante :

`method` : méthode de la requête (`GET/POST/HEAD/PATCH/PUT/DELETE`).

`headers` : Les entêtes à ajouter à la requête.

`body` : Le corps de la requête envoyé avec les méthodes autres que `GET/HEAD`.

`mode` : `cors`, `no-cors`, ou `same-origin`.

`credentials` : `omit`, `same-origin`, ou `include`.

`cache` : `default`, `no-store`, `reload`, `no-cache`, `force-cache` ou `only-if-cached`.

`redirect` : `follow`, `error`, ou `manual`.

`referrer` : `client`, `no-referrer`, ou une URL.

`referrerPolicy` : `no-referrer`, `no-referrer-when-downgrade`, `origin`, `origin-when-cross-origin` ou `unsafe-url`.

`integrity` : un hash.

`keepalive` : un booléen.

`signal` : un `AbortSignal` que nous verrons dans une leçon suivante.

Nous avons déjà vu les trois premières options et allons étudier les suivantes.

L'option `mode`

L'option **`mode`** permet d'empêcher les requêtes `CORS` non intentionnelles.

`cors` : défaut. Les requêtes `CORS` sont possibles.

`no-cors` : Seules les requêtes `CORS` simples sont possibles.

`same-origin` : Les requêtes `CORS` sont impossibles.

L'option `credentials`

L'option **`credentials`** permet de définir si les `cookies` et l'entête `HTTP-Authorization` seront envoyés.

`same-origin` : défaut. Ne pas les envoyer sur les requêtes `CORS`.

`include` : toujours envoyer. Nécessite que le serveur définisse l'entête `Accept-Control-Allow-Credentials` à `true`.

`omit` : jamais envoyés même pour les requêtes sur la même origine.

L'option `cache`

L'option **`cache`** permet de définir la mise en cache en ignorant les entêtes du mis par le serveur.

La mise en cache permet de conserver la réponse dans le navigateur pour certaines requêtes afin de ne pas devoir les refaire avant des temps déterminés. Par exemple pour les feuilles de styles etc.

`default` : `fetch` utilise les règles de cache standardes au protocole `HTTP` et respecte les entêtes de cache du serveur.

`no-store` : le navigateur n'utilise pas le cache pour la requête et ne met pas à jour le cache avec la réponse.

`reload` : le navigateur n'utilise pas le cache pour la requête mais il met à jour le cache avec la réponse.

`no-cache` : le navigateur regarde dans le cache, si il y a une réponse sauvegardée pour la requête il va effectuer une requête conditionnelle au serveur. Si le serveur indique que la ressource n'a pas changé alors la réponse sera chargée depuis le cache, sinon la requête sera poursuivie jusqu'à l'obtention d'une réponse et le cache sera mis à jour.

`force-cache` : le navigateur regarde dans le cache, si il y a une réponse sauvegardée pour la requête il va la récupérer en ignorant sa date de péremption et sans faire de requête conditionnelle au serveur pour demander si la ressource a changé. Si il n'y a pas de réponse en cache il fera la requête et mettra à jour le cache.

`only-if-cached` : le navigateur regarde dans le cache, si il y a une réponse sauvegardée pour la requête il va la récupérer sinon il renverra une erreur `504 Gateway timeout`.

L'option `redirect`

L'option **`redirect`** permet de définir comment les redirections sont gérées.

`follow` : défaut. Suit les redirections du serveur.

`error` : Erreur en cas de redirection.

`manual` : Ne pas suivre les redirections. Mais la propriété `url` de la réponse du serveur contiendra l'URL de redirection et `redirected` sera à `true`. Le `JavaScript` pourra ainsi manuellement faire la redirection si besoin.

L'option `referer`

L'option `referrer` permet de définir l'entête `Referer` qui est l'URL de la page qui a effectué la requête en `JavaScript`.

`no-referrer` : pas d'URL.

`client` : l'URL par défaut de la page qui a fait la requête.

une URL : vous pouvez changer l'URL mais il faut garder l'origine. C'est-à-dire que si l'origine est `https://fr.wikipedia.org` vous pouvez utiliser comme `referer` `https://fr.wikipedia.org/wiki/`.

L'option `referrerPolicy`

L'option `referrerPolicy` permet de définir comment le navigateur doit gérer l'entête `Referer`.

`no-referrer` : ne jamais envoyer d'entête `Referer`.

`no-referrer-when-downgrade` : défaut. Envoie l'entête `Referer` avec l'URL entière sauf si la requête est envoyée en utilisant un protocole moins sécurisé que l'origine. Par exemple si on passe de `HTTPS` à `HTTP`.

`origin` : défaut. Envoie l'entête `Referer` avec l'URL de l'origine sans le chemin (tout après le premier / est supprimé). Par exemple `https://fr.wikipedia.org/wiki/` devient `https://fr.wikipedia.org/`.

`origin-when-cross-origin` : Envoie l'URL entier si la requête est dirigée vers la même origine. Sinon envoie simplement l'URL de l'origine.

`same-origin` : Envoie l'URL entier si la requête est dirigée vers la même origine, sinon n'envoie aucun entête `Referer`.

`strict-origin` : Envoie uniquement l'URL de l'origine et uniquement lorsque le protocole n'est pas moins sécurisé. Sinon n'envoie aucun entête `Referer`.

`strict-origin-when-cross-origin` : Envoie l'URL entier si la requête est dirigée vers la même origine. Envoie uniquement l'URL de l'origine pour les requêtes `CORS`. N'envoie aucun entête `Referer` si le protocole est moins sécurisé.

`unsafe-url` : Envoie l'URL entier dans tous les cas.

L'option `integrity`

L'option `integrity` permet de passer un `hash` précédé par l'algorithme de hachage (`SHA-256` par exemple).

Exemple :

```
"sha256-BpfBw7ivV8q2jLiT13fxDYAe2tJllusRSZ273h2nFSE="
```

Après avoir téléchargé la ressource, le navigateur va calculer le `hash` de celle-ci en utilisant l'algorithme précisé (ici `SHA-256`). Il peut ainsi être certain que la ressource n'a pas été modifié par un attaquant.

L'option `keepalive`

L'option `keepalive` permet de préciser que la requête peut rester en vie même si la page qui l'a initiée est fermée.

Normalement, comme nous le verrons, toutes les requêtes `HTTP` en cours sont annulées lors de la fermeture d'une page. Cette option permet de maintenir les requêtes en cours.