openclassrooms.com

Sauvegardez les données grâce à une API HTTP

7–8 minutes

L'API HTTP et la fonction fetch nous ont permis de récupérer dynamiquement les avis de nos pièces automobiles. Nous souhaitons maintenant permettre aux utilisateurs de déposer leurs avis sur le site. Pour y parvenir, nous allons devoir sauvegarder ces avis grâce à l'API HTTP et au verbe POST.

Personnalisez votre requête avec le deuxième argument de fetch

La sauvegarde de nouvelles données dans l'API HTTP nécessite de renseigner trois nouvelles informations en plus du chemin de la ressource : le verbe HTTP, la charge utile et les en-têtes.

Pour y parvenir, il faut utiliser le deuxième argument de la fonction fetch.

```
fetch("chemin de la ressource", { /* Objet de
configuration */ });
```

La première propriété concerne le protocole HTTP. Ce dernier utilise les **verbes** (GET, POST…) pour désigner l'opération demandée par la requête. Pour créer des avis, nous utiliserons le verbe POST, et nous le renseignerons grâce à la propriété **method** de l'objet de configuration. Sa valeur sera une chaîne de caractères contenant le verbe HTTP.

```
{
    /* Objet de configuration */
```

```
    method: "POST"
}
```

La deuxième propriété concerne la **charge utile** de la requête, c'est-à-dire l'ensemble de données que le serveur utilisera pour réaliser l'opération demandée par la requête. La charge utile est renseignée avec la propriété **body** de l'objet de configuration. Sa valeur sera une chaîne de caractères. Pour envoyer les informations d'un nouvel avis, nous utiliserons le format JSON.

```
{
    /* Objet de configuration */
    body: '{"commentaire":"Top produit !"}'
}
```

La troisième propriété concerne le format de la charge utile avec les en-têtes. Dans la majeure partie des cas, elle doit être spécifiée pour que le serveur l'interprète correctement. L'en-tête que nous utiliserons pour faire cela est "Content-Type". La propriété **headers** aura comme valeur un objet contenant lui-même une propriété "Content-Type" et une valeur indiquant le type MIME du format de la charge utile.

```
{
    /* Objet de configuration */
    headers: { "Content-Type": "application/json" }
}
```

Ainsi, l'appel à la fonction fetch ci-dessous permettra de configurer une requête pour qu'elle soit envoyée avec le verbe POST et un objet au format JSON comme charge utile :

```
fetch("/pieces/1/avis", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
```

```
    body: '{"commentaire": "Top produit !"}'
});
```

Envoyez vos avis à partir de la page web

Maintenant que vous en savez plus sur le deuxième argument de fetch, passez à présent sur l'étiquette P3C3-Debut, sur le dépôt GitHub. J'ai ajouté un formulaire dans le fichier index.html qui vous permettra d'écrire un nouvel avis. Pour l'ajouter vous-même, insérez le code ci-dessous dans la section filtres, après l'input de type range.

```
<input type="range" name="prix-max" id="prix-max"
min="0" max="60" step="5">
<!-- code a ajouter -->
<h3>Ajouter un avis</h3>
<form class="formulaire-avis">
    Identifiant de la pièce :<br>
    <input type="number" name="piece-id"><br>
    Nom d'utilisateur :<br>
    <input type="text" name="utilisateur"><br>
    Avis :<br>
    <input type="text" name="commentaire"><br>
    <button>Envoyer</button>
</form>
```

C'est parti ! Reprenons le code de notre application pour enregistrer de nouveaux avis. 😃

En premier lieu, dans le fichier avis.js, nous rajoutons un event listener sur l'événement "submit" de la balise form. Cet événement est déclenché lorsque le formulaire est validé.

```
export function ajoutListenerEnvoyerAvis() {
```

```
   const formulaireAvis =
document.querySelector(".formulaire-avis");
   formulaireAvis.addEventListener("submit", function
(event) {
   /* ... */
   });
}
```

N’oubliez pas d’importer la fonction dans le fichier pieces.js en ajoutant la fonction dans la ligne d’import. Sinon, le listener ne fonctionnera pas. Vous devez donc écrire :

```
import { ajoutListenersAvis, ajoutListenerEnvoyerAvis
} from "./avis.js";
// Récupération des pièces depuis le fichier JSON
const reponse = await fetch('http://localhost:8081
/pieces/');
const pieces = await reponse.json();
// on appelle la fonction pour ajouter le listener au
formulaire
ajoutListenerEnvoyerAvis()
```

Nous ajoutons donc un appel à la fonction preventDefault de l’objet event à l’intérieur de la fonction anonyme :

```
// Désactivation du comportement par défaut du
navigateur
event.preventDefault();
```

Nous allons maintenant construire la charge utile de la requête qui permettra d’ajouter l’avis dans l’API. La charge utile prend la forme d’une chaîne de caractères au format JSON, contenant un objet. Cet objet reprend les champs du formulaire, et crée une propriété pour chacun d’entre eux.

Nous construisons donc un objet en reprenant les valeurs des balises input du formulaire :

```
// Création de l’objet du nouvel avis.
const avis = {
   pieceId: parseInt(event.target.querySelector("
[name=piece-id]").value),
   utilisateur: event.target.querySelector("
[name=utilisateur").value,
   commentaire: event.target.querySelector("
[name=commentaire]").value,
};
```

Cet objet avis doit être converti en une chaîne de caractères au format JSON pour être transmis dans le body de la requête. Nous appelons donc la fonction JSON.stringify :

```
// Création de la charge utile au format JSON
const chargeUtile = JSON.stringify(avis);
```

Nous disposons maintenant de tous les éléments nécessaires pour créer et envoyer la requête avec la fonction fetch.

```
// Appel de la fonction fetch avec toutes les
informations nécessaires
fetch("http://localhost:8081/avis", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: chargeUtile
});
```

Si vous appuyez sur le bouton “Afficher les avis” de la pièce pour laquelle vous venez de déposer un nouvel avis, vous le verrez apparaître dans le texte sous le bouton.

Récapitulons en vidéo

Vous pouvez revoir les différentes étapes de cette démonstration dans la vidéo ci-dessous :

À vous de jouer

Dans le fichier index.html de la branche P3C3-Exercice :

- Ajoutez une balise input type=”number” qui servira à spécifier le nombre d’étoiles que l’utilisateur attribue à la pièce avec son commentaire.
- Modifiez en conséquence le fichier avis.js pour prendre en compte cette nouvelle information dans la charge utile. Vous utiliserez la propriété “nbEtoiles” dans l’objet de la charge utile.

Corrigé

Vous pouvez vérifier votre travail en consultant la branche P3C3-Solution et en regardant la vidéo ci-dessous :

En résumé

- Utilisez le deuxième argument de la fonction fetch pour personnaliser votre requête.
- Configurez l’objet du deuxième argument en renseignant trois propriétés :
- method pour le verbe HTTP ;
- body pour la charge utile ;
- headers pour les en-têtes.
- Pour créer une nouvelle ressource avec une API HTTP qui accepte le format JSON, vous devez utiliser :
- le verbe POST ;
- un body au format JSON ;

- un en-tête Content-Type application/json.

Et voilà, vous avez réussi à sauvegarder vos données en utilisant la fonction fetch. Découvrons dans le chapitre suivant comment garder notre page web active grâce au localStorage.