openclassrooms.com

Envoyez une requête depuis le navigateur

8–10 minutes

Votre site commence à prendre forme ! Vous avez réussi à générer automatiquement les fiches produits sur votre page web. Vous l'avez également rendue interactive en utilisant des filtres pour réordonner les fiches.

Nous souhaitons à présent afficher les avis déposés sur les produits par les utilisateurs du site. Dans la première partie du cours, nous avions récupéré les produits sous la forme d'un fichier JSON. À présent, nous allons récupérer les avis à l'aide d'une API en ligne. Grâce à ce service web, nous serons en mesure d'afficher en continu des données actualisées.

Dans cette partie du cours, votre page web aura donc besoin de communiquer avec une API en ligne disponible sur l'adresse http://localhost:8081. Vous devrez héberger cette API en clonant ce dépôt GitHub. Avec un terminal, placez-vous dans le dossier parent du projet de votre page web et lancez les commandes :

```
git clone https://github.com/OpenClassrooms-Student-Center/7697016-Back-End.git api-http

cd api-http

npm install

npm start
```

Et voilà, nous sommes prêts à récupérer les données ! Mais avant de rentrer dans le vif du sujet, découvrons ensemble en quoi consiste une **API HTTP**, et comment elle s'utilise.

Utilisez un service web pour persister vos données

La persistance de la donnée

Dans les deux premières parties de ce cours, nous récupérions les données depuis le fichier JSON pieces-autos.json :

```
const reponse = await fetch("pieces-autos.json");

const pieces = await reponse.json();
```

Cependant, cette méthode a ses limites : le fichier JSON est présent sur votre disque dur, il ne peut pas recevoir de mise à jour des fiches produits. Heureusement, en utilisant un service web, grâce à son API en ligne, nous allons changer la donne. 😉

Comme je vous l'expliquais dans le 2ème chapitre de la première partie de ce cours, un service web permet de manipuler les données qu'il expose à travers son API en ligne. Il peut les créer, les récupérer, les mettre à jour et les supprimer. Ces quatre opérations permettent de **persister vos données**, c'est-à-dire de les **stocker durablement**. Ainsi, lorsque mon application web communique avec le service web, ces manipulations sont sauvegardées jusqu'à la prochaine utilisation.

L'API HTTP

Une API est, pour faire simple, un moyen de communication entre deux logiciels, que ce soit entre différents composants d'une application ou entre deux applications différentes. C'est ce qu'on appelle une **abstraction** : une façade qui simplifie des mécanismes complexes permettant de fournir un service demandé.

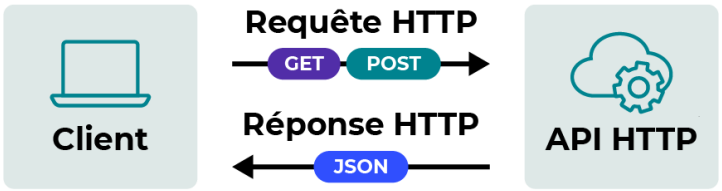
Si je reprends mon exemple de la météo pour illustrer cela : je n'ai pas besoin de savoir comment calculer la vitesse du vent pour avoir cette information, l'API me la fournit directement.

Le fonctionnement d'un serveur web

Pour récupérer les avis des clients, nous utiliserons un service web qui exposera ses fonctionnalités de stockage de l'information via une API HTTP. Jusqu'à présent, nous n'utilisions qu'un seul programme pour exécuter notre application : le navigateur. En utilisant un service web, un deuxième programme intervient : le **serveur web**.

Ouh là là… Mais ça part dans tous les sens, cette histoire de serveur web.

Pas d'inquiétude ! 😉 Le schéma ci-dessous vous aidera à mieux comprendre comment cela fonctionne :



Dans ce cours, nous n'utiliserons que les verbes GET et POST pour récupérer les avis et en créer de nouveaux.

Une requête HTTP concerne toujours une **ressource**. Il s'agit d'un type de donnée que le service web gère, par exemple :

- la ressource "pièces automobiles" ;
- la ressource "avis utilisateur".

Les ressources sont accessibles à travers des **chemins**. Il s'agit d'une partie de l'URL qui identifie la ressource concernée par la

requête. Voici les chemins correspondant aux ressources précédentes :

- `/pieces`
- `/avis`

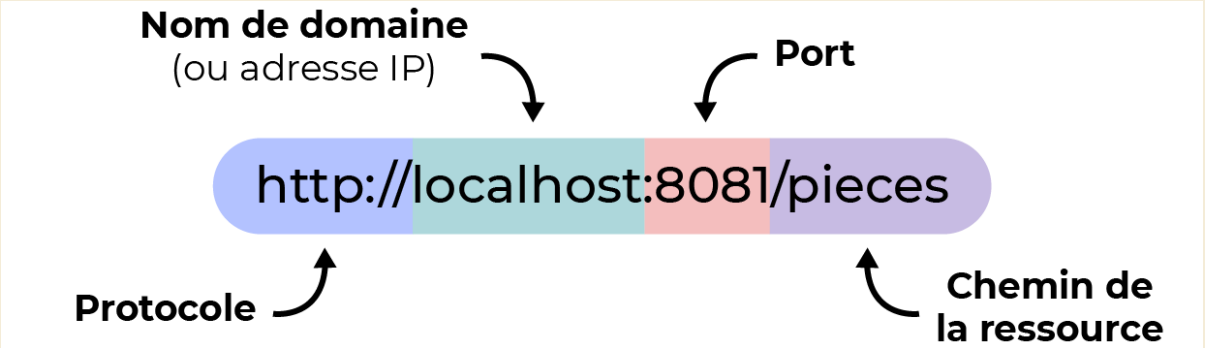
L'association de la ressource et du verbe permet de décrire l'opération demandée par le navigateur et qui sera traitée par le service web. Par exemple :

- `GET /pieces` permet de récupérer toutes les pièces automobiles ;
- `POST /avis` permet de créer un nouvel avis ;
- `DELETE /avis/42` permet de supprimer l'avis numéro 42.

Utilisez la fonction fetch pour envoyer une requête

La fonction fetch

Le navigateur vous offre un moyen simple d'envoyer des requêtes HTTP, avec la fonction fetch. Cette dernière prend en argument d'appel une chaîne de caractères comprenant l'adresse du serveur web et le chemin qui décrit la ressource que nous souhaitons manipuler. Le verbe utilisé par défaut par la fonction fetch est GET.



> Par exemple, l'appel à la fonction fetch ci-dessous permettra d'envoyer une requête GET sur le serveur localhost:8081 pour demander la ressource /pieces.

```
fetch("http://localhost:8081/pieces");
```

Récupérez les avis des autres utilisateurs sur les pièces automobiles

Revenons à notre site web pour ajouter la récupération des avis avec la fonction fetch. Mais avant, passez sur la branche P3C1-Debut du dépôt GitHub.

J'y ai ajouté un bouton sur chaque fiche produit, intitulé "Afficher les avis". Chaque bouton est créé avec un attribut "data-id" qui contient l'identifiant de la pièce automobile. Cet attribut permettra de faire le lien entre un bouton sur la page web et une pièce automobile lorsque l'on réagira au clic de ce bouton.

Pour ajouter le code vous-même, insérez le code ci-dessous dans avis.js :

```
export function ajoutListenersAvis() {
    const piecesElements = document.querySelectorAll(".fiches article button");
    for (let i = 0; i < piecesElements.length; i++) {
      piecesElements[i].addEventListener("click", async function (event) {
          /* ... */
      });
    }
}
```

Dans un deuxième temps, nous appelons cette fonction ajoutListenersAvis dans le fichier pieces.js après la boucle for de génération des éléments du DOM qui se trouve dans la fonction *genererPieces*. Comme ceci :

```
function genererPieces(pieces){
   for (let i = 0; i < pieces.length; i++) {
   //...
    }
    // Ajout de la fonction ajoutListenersAvis
```

```
    ajoutListenersAvis();

}
```

Pour que l'ajout de cette fonction ne génère pas d'erreur dans notre code, nous devons l'importer avant de l'utiliser. Cet import doit être réalisé à la première ligne du fichier pieces.js :

```
import { ajoutListenersAvis } from "./avis.js";
```

Cette syntaxe moderne de JavaScript permet d'importer des variables et des fonctions dans nos fichiers sans rajouter d'autres balises script dans notre HTML. Pour cela, il nous suffit d'ajouter la propriété *type* à notre balise script avec la valeur *module*, comme ceci :

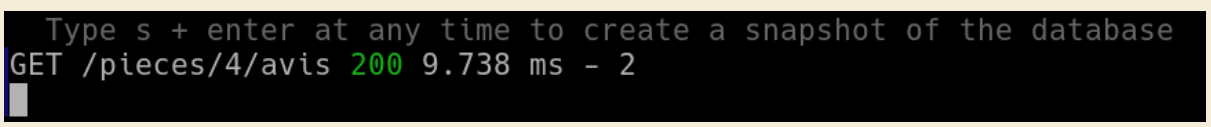
```
<script type="module" src="pieces.js"></script>
```

Enfin, dans l'event listener du fichier avis.js, nous récupérons l'identifiant de la pièce automobile pour laquelle l'utilisateur a cliqué sur le bouton "Afficher les avis". Nous récupérons la valeur de l'attribut data-id grâce à la propriété "dataset.id". Nous utilisons ensuite cet identifiant pour construire le chemin de la ressource sur laquelle créer la requête HTTP avec la fonction fetch.

```
const id = event.target.dataset.id;

fetch(`http://localhost:8081/pieces/${id}/avis`);
```

Si je clique sur le bouton "Afficher les avis" de la pièce Liquide de frein, je vois dans la console de l'API HTTP, qui se trouve dans le terminal avec lequel vous avez lancé votre serveur, qu'une requête est arrivée avec l'identifiant 4.



Félicitations, vous avez réussi à envoyer une requête depuis le navigateur ! 🥳 Vous pouvez retrouver le code développé dans ce chapitre sur la branche P3C1-Fin.

Récapitulons en vidéo

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

En résumé

- Utilisez les API en ligne pour persister vos données :
- Les services web permettent de stocker durablement les données grâce à leur API en ligne.
- Un navigateur web utilise le protocole HTTP pour envoyer des **requêtes** à destination du serveur.
- Pour que le serveur identifie le type de requête envoyé, vous devez utiliser des verbes HTTP (POST, GET, PUT, DELETE).
- Utilisez la fonction fetch pour envoyer des requêtes au service web :
- Les requêtes permettent de manipuler les données.
- Elle utilise le verbe HTTP GET par défaut.

Vous avez envoyé votre première requête depuis votre navigateur. Il vous faut maintenant traiter la réponse du serveur. Suivez-moi dans le chapitre suivant pour découvrir comment faire.