openclassrooms.com

# Sauvegardez les données dans le localStorage

6–8 minutes

---

Votre page web est pleinement opérationnelle. Elle permet d'afficher les pièces automobiles, leurs avis, et de déposer de nouveaux avis.

Pour l'instant, tout cela fonctionne tant que votre navigateur est connecté à internet. Mais dans certaines situations, vous pouvez perdre cet accès à internet. Notamment, lorsque vous ne disposez pas d'une connexion stable.

Dans ce chapitre, nous allons découvrir comment garder notre page web interactive même lorsque vous n'êtes pas connecté à internet. Tout cela, grâce au **localStorage**. C'est parti ! 🚀

## Appréhendez la logique d'utilisation du localStorage

Le localStorage stocke uniquement des "valeurs" de type chaînes de caractères. Cela nous oblige à utiliser la sérialisation JSON pour stocker des données plus complexes. Chaque chaîne de caractères est identifiée par une "clé". On parle ainsi de **stockage clé-valeur**.

Les données du localStorage sont sauvegardées, y compris à la fermeture du navigateur. Ainsi, lorsque vous ouvrez à nouveau la page, les données du localStorage sont toujours présentes. Vous pouvez donc vous servir du localStorage pour sauvegarder des informations entre deux sessions d'un utilisateur. Cela est souvent utilisé pour stocker, par exemple, les préférences de langue de l'interface, ou de mode sombre ou de mode clair.

### Utilisez le localStorage

Il existe trois opérations principales pour interagir avec le localStorage : enregistrer, récupérer et supprimer. L'API propose une fonction pour chacune de ces opérations. On accède à ces fonctions grâce à l'objet window.localStorage.

### Enregistrez la réponse de l'API grâce à la fonction setItem

> Pour stocker la phrase "Les Bonnes Pièces !" avec la clé "nom", on écrira donc :

```
window.localStorage.setItem("nom", "Les Bonnes Pièces !");
```

Dans le cas de notre site web, nous devons rajouter un appel à cette fonction juste après la fonction fetch, qui récupère les pièces auprès de l'API HTTP :

```
// Récupération des pièces depuis le fichier JSON
const reponse = await fetch("http://localhost:8081/pieces");
const pieces = await reponse.json();
// Transformation des pièces en JSON
const valeurPieces = JSON.stringify(pieces);
// Stockage des informations dans le localStorage
window.localStorage.setItem("pieces", valeurPieces);
```

```
const pieces = await fetch("http://localhost:8081/pieces").then(pieces => pieces.jon());
```

### Récupérez les données des pièces auto grâce à la fonction getItem

> Ainsi, pour récupérer le nom de mon entreprise avec la clé "nom",

on écrira :

```
const nomEntreprise =
window.localStorage.getItem("nom");
```

Ajoutons la tentative de récupération des pièces dans le localStorage. L'extrait de code suivant se place juste avant l'appel à la fonction fetch, tout au début du fichier pieces.js :

```
// Récupération des pièces éventuellement stockées
dans le localStorage

const pieces = window.localStorage.getItem("pieces");
```

```
// Récupération des pièces éventuellement stockées
dans le localStorage

let pieces = window.localStorage.getItem("pieces");

if (pieces === null) {

  /* Code de récupération des pièces depuis l'API HTTP
*/

}
```

```
//Récupération des pièces eventuellement stockées dans
le localStorage

let pieces = window.localStorage.getItem('pieces');

if (pieces === null){

   // Récupération des pièces depuis l'API

   const reponse = await fetch('http://localhost:8081
/pieces/');

   pieces = await reponse.json();

   // Transformation des pièces en JSON

   const valeurPieces = JSON.stringify(pieces);

   // Stockage des informations dans le localStorage

   window.localStorage.setItem("pieces",
```

```
valeurPieces);
}else{
   pieces = JSON.parse(pieces);
}
```

Si je lance ma page une première fois et que je la rafraîchis (avec la touche F5, par exemple) plusieurs fois, l'API HTTP n'est appelée qu'une seule fois. Pourtant, ma page s'affiche correctement à chaque rafraîchissement.

**Supprimez les pièces auto du localStorage grâce à la fonction removeItem**

Maintenant que nous utilisons en priorité les données du localStorage lorsqu'elles sont présentes, il nous faut un moyen pour rafraîchir ces données. Pour y parvenir, nous devons supprimer les données déjà présentes dans le localStorage.

Nous allons donc ajouter un bouton sur la page web "Mettre à jour les pièces" dans le fichier HTML.

```
<h3>Ajouter un avis</h3>
<form class="formulaire-avis"><!-- …--></form>
<!-- code à ajouter-->
<h3>Données</h3>
<button class="btn-maj">Mettre à jour les
pièces</button>
```

Lors du clic sur ce bouton, nous supprimerons les données du localStorage. Lors du prochain chargement de la page, l'API HTTP sera à nouveau appelée pour obtenir la toute dernière version des pièces auto.

Pour supprimer le nom de l'entreprise Les Bonnes Pièces avec la clé "nom", on écrira donc :

```
window.localStorage.removeItem("nom");
```

Lors du prochain appel à getItem pour cette clé, nous obtiendrons la valeur null, indiquant qu'il n'y a pas de valeur stockée pour cette clé.

Dans le cas de notre site web, nous rajoutons un event listener à la fin du fichier pieces.js, sur le bouton "Mettre à jour les pièces", et nous appelons la fonction removeItem dans la fonction anonyme :

```
// Ajout du listener pour mettre à jour des données du localStorage

const boutonMettreAJour = document.querySelector(".btn-maj");

boutonMettreAJour.addEventListener("click", function () {

  window.localStorage.removeItem("pieces");

});
```

**Récapitulons en vidéo**

Vous pouvez revoir les différentes étapes de ce chapitre dans la vidéo ci-dessous :

**À vous de jouer !**

Dans le fichier avis.js de la branche P3C4-Exercice :

- sauvegardez les avis quand vous recevez la réponse de l'API HTTP à votre requête pour récupérer les avis d'une pièce auto ;
- affichez automatiquement les avis enregistrés lors du chargement de la page web.

**Corrigé**

Vous pouvez vérifier votre travail en consultant la branche P3C4-

Solution et en regardant la vidéo ci-dessous :

**En résumé**

- Utilisez le localStorage pour optimiser les performances de votre application :
- pour avoir toujours des données disponibles à afficher en cas de connexion intermittente ou instable ;
- pour sauvegarder les réponses des requêtes à l'API, et ainsi éviter de les demander en boucle.
- Vous pouvez réaliser différentes opérations dans le localStorage :
- enregistrer des données grâce à setItem ;
- récupérer des données enregistrées au préalable grâce à getItem ;
- supprimer des données grâce à removeItem.

*Vous savez désormais interagir avec un service web en utilisant une API HTTP. Vous pouvez être fier de vous ! 😃 Votre prochain défi dans la partie suivante ? Enrichir une page web en utilisant une librairie. Mais avant, testez vos connaissances en réalisant le quiz !*