Stocker des données côté navigateur

Objectif : comprendre pourquoi on peut stocker des données dans le navigateur. Utiliser le localStorage et le sessionStorage. Stocker, lire, modifier et supprimer des données simples. Gérer des données structurées (ex. : objets ou tableaux).

Touchet Laurent / Afec 2025

Sommaire

Pourquoi stocker des données dans le navigateur ?

Le navigateur peut mémoriser certaines informations localement, sans avoir besoin d’un serveur distant. Cela permet par exemple :

- de garder un utilisateur connecté,
- de retenir des préférences (thème sombre, langue…),
- de sauvegarder un panier d’achat temporairement,
- de faciliter des tests ou démos locales.

Les deux principaux moyens de stockage

JavaScript propose deux mécanismes simples :

Méthode	Durée de vie
localStorage	Permanent (jusqu’à suppression)
sessionStorage	Temporaire (tant que l’onglet est ouvert)
CookieStore	un cookie store dans le navigateur.

Le localStorage

1/4) Stocker une donnée

```
localStorage.setItem("prenom", "nina");
```

2/4) Lire une donnée

```
const utilisateur = localStorage.getItem("prenom");
console.log(utilisateur); // "nina"
```

Le localStorage

3/4) Supprimer une donnée

```
localStorage.removeItem("prenom");
```

4/4) Vider tout le localStorage

```
localStorage.clear();
```

Gérer les objets ou tableaux

Le localStorage ne stocke que du texte. Pour enregistrer un tableau ou un objet, on doit le convertir en JSON.

1/2) Exemple avec un objet :

```
const utilisateur = {
 nom: "sébastien",
 age: 34,
 ville: "Lyon"
};

localStorage.setItem("profil", JSON.stringify(utilisateur));
```

Gérer les objets ou tableaux

2/2) Lire et reconstruire l’objet

```
const donneesBrutes = localStorage.getItem("utilisateur");

if (donneesBrutes !== null) {
 const utilisateurRecu = JSON.parse(donneesBrutes);
 console.log(utilisateurRecu.prenom); // "sébastien"
 console.log(utilisateurRecu.age);  // 34
}
```

Gérer les objets ou tableaux

1/2) Exemple pour stocker un tableau :

```
const eleves = ["guillaume", "angélique", "alvyn", "nina"];

localStorage.setItem("listeEleves", JSON.stringify(eleves));
```

2/2) Lire et reconstruire le tableau

```
const listeBrute = localStorage.getItem("listeEleves");

if (listeBrute !== null) {
 const liste = JSON.parse(listeBrute);
 console.log(liste); // ["guillaume", "angélique", "alvyn", "nina"]
 console.log(liste[0]); // "guillaume"
}
```

Le sessionStorage

Même fonctionnement que localStorage, mais la durée de vie est limitée à l’onglet courant.

```
sessionStorage.setItem("theme", "sombre");

const theme = sessionStorage.getItem("theme");
console.log(theme); // "sombre"
```

Bonnes pratiques

D’une manière générale et pas seulement pour le stockage de données côté navigateur, pensez à toujours vérifier si les données existent avant de les utiliser :

```
const profil = localStorage.getItem("profil");
if (profil !== null) {
 const utilisateur = JSON.parse(profil);
 console.log(utilisateur.nom);
}
```

- Ne jamais stocker de données sensibles (mot de passe, token) dans localStorage.
- Nettoyer les anciennes données si elles ne sont plus utilisées.

Ressources

MDN localStorage
https://developer.mozilla.org/fr/docs/Web/API/Window/localStorage

MDN sessionStorage
https://developer.mozilla.org/fr/docs/Web/API/Window/sessionStorage

CookieStore
https://developer.mozilla.org/en-US/docs/Web/API/CookieStore

Vidéo (fr, 15min) Que sont "localStorage" & "sessionStorage" ? WebStorage API
https://www.youtube.com/watch?v=ITmKqkmHlnY