

```
1 import { NextRequest, NextResponse } from "next/server";
2 import middlewareAuth from "./utils/middlewareAuth";
3
4 export async function middleware(req) {
5   const url = req.url;
6   const pathname = req.nextUrl.pathname;
7   // console.log({ url, pathname });
8
9   console.log(req.url, req.nextUrl.pathname);
10  if (pathname.startsWith("/profile")) {
11    console.log("profile request !!!");
12
13    const user = await middlewareAuth();
14    if (!user) return NextResponse.json({ message: "User not found" });
15
16    if (user && user.role === "ADMIN") {
17      return NextResponse.redirect("/");
18    }
19  }
20
21  if (pathname.startsWith("/admin")) {
22    const user = await middlewareAuth();
23    if (!user) return NextResponse.json({ message: "User not found" });
24    if (user && user.role !== "ADMIN") {
25      return NextResponse.redirect("/");
26    }
27  }
28
29  console.log("admin request !!!");
30}
```

Les fondamentaux du langage Javascript

Formation Développeur web et mobile

Boukari Dianka

Du 13 janvier 2025

SIELI : Formation professionnelle sur mesure



Objectif pédagogique

À l'issue du cours, l'apprenant sera capable de :

- Comprendre la différence entre `localStorage` et `sessionStorage`
- Apprendre à utiliser ces deux mécanismes pour stocker et récupérer des données
- Découvrir comment manipuler des objets JSON avec ces mécanismes

localStorage

localStorage

Qu'est-ce que `localStorage` ?

`localStorage` est un mécanisme de stockage qui permet de stocker des données persistantes dans le navigateur. Les données sont conservées même après la fermeture du navigateur ou du tab.

localStorage

Caractéristiques de **localStorage** :

- Capacité : environ 5 à 10 Mo.
- Les données sont disponibles tant que l'utilisateur n'efface pas les données du navigateur.
- Accessible par tous les onglets du même domaine.

localStorage

Méthodes de `localStorage`

1. `setItem(key, value)` : Ajoute un élément à `localStorage` ou met à jour sa valeur si elle existe déjà.
2. `getItem(key)` : Récupère la valeur associée à une clé.
3. `removeItem(key)` : Supprime l'élément spécifié.
4. `clear()` : Supprime tous les éléments stockés.
5. `key(index)` : Retourne la clé à l'index spécifié.
6. `length` : Retourne le nombre d'éléments dans `localStorage`.

localStorage

Exemple d'utilisation de `localStorage` avec des données simples

```
<h1>Stocker une donnée avec localStorage</h1>

<button id="setButton">Enregistrer dans localStorage</button>

<button id="getButton">Récupérer depuis localStorage</button>

<button id="removeButton">Supprimer depuis localStorage</button>

<button id="clearButton">Supprimer tout depuis localStorage</button>

<p id="output"></p>
```

localStorage

Enregistrer une donnée

```
document.getElementById('setButton').addEventListener('click', function() {  
    localStorage.setItem('username', 'JeanNaymard');  
    alert('Nom enregistré dans localStorage !');  
});
```

localStorage

Récupérer une donnée

```
document.getElementById('getButton').addEventListener('click', function() {
    const username = localStorage.getItem('username');
    document.getElementById('output').textContent = 'Nom utilisateur : ' + username;
});
```

localStorage

Supprimer une donnée

```
document.getElementById('removeButton').addEventListener('click', function() {  
    localStorage.removeItem('username');  
    alert('Nom supprimé de localStorage !');  
});
```

localStorage

Supprimer toutes les données

```
document.getElementById('clearButton').addEventListener('click', function() {
    localStorage.clear();
    alert('Toutes les données ont été supprimées de localStorage !');
});
```

sessionStorage

sessionStorage

Qu'est-ce que sessionStorage ?

sessionStorage est similaire à localStorage, mais avec une différence importante : les données sont supprimées lorsque l'onglet ou la fenêtre du navigateur est fermée.

Caractéristiques de sessionStorage :

- Capacité : environ 5 Mo.
- Les données ne sont disponibles que dans la même session (onglet ou fenêtre).
- Les données sont supprimées lorsque la session se termine (l'onglet est fermé).

sessionStorage

Méthodes de sessionStorage

Les méthodes de sessionStorage sont similaires à celles de localStorage :

1. **setItem(key, value)** : Ajoute un élément.
2. **getItem(key)** : Récupère la valeur associée à une clé.
3. **removeItem(key)** : Supprime l'élément spécifié.
4. **clear()** : Supprime tous les éléments.
5. **key(index)** : Retourne la clé à l'index spécifié.
6. **length** : Retourne le nombre d'éléments dans sessionStorage .

sessionStorage

Exemple d'utilisation de **sessionStorage** avec des données simples

```
<h1>Stocker une donnée avec sessionStorage</h1>

<button id="setButton">Enregistrer dans sessionStorage</button>

<button id="getButton">Récupérer depuis sessionStorage</button>

<button id="removeButton">Supprimer depuis sessionStorage</button>

<button id="clearButton">Supprimer tout depuis sessionStorage</button>

<p id="output"></p>
```

sessionStorage

Enregistrer une donnée

```
document.getElementById('setButton').addEventListener('click', function() {  
    sessionStorage.setItem('userEmail', 'jean.naymard@gmail.com');  
    alert('Nom enregistré dans sessionStorage !');  
});
```

sessionStorage

Récupérer une donnée

```
document.getElementById('getButton').addEventListener('click', function() {
  const username = sessionStorage.getItem('username');
  document.getElementById('output').textContent = 'Nom utilisateur : ' + username;
});
```

sessionStorage

Supprimer une donnée

```
document.getElementById('removeButton').addEventListener('click', function() {  
    sessionStorage.removeItem('username');  
    alert('Nom supprimé de sessionStorage !');  
});
```

sessionStorage

Supprimer toutes les données

```
document.getElementById('clearButton').addEventListener('click', function() {  
    sessionStorage.clear();  
    alert('Toutes les données ont été supprimées de sessionStorage !');  
});
```

localStorage, sessionStorage et JSON

`localStorage` / `sessionStorage` et `JSON`

`localStorage` et `sessionStorage` ne peuvent stocker que des chaînes de caractères. Cependant, nous pouvons convertir des objets JavaScript en chaînes JSON et inversement.

Sérialisation (Conversion d'objet en JSON)

Utilisez `JSON.stringify()` pour convertir un objet JavaScript en une chaîne JSON avant de le stocker.

Désérialisation (Conversion de JSON en objet)

Utilisez `JSON.parse()` pour reconvertis une chaîne JSON en objet JavaScript.

Exemple d'utilisation de `localStorage` avec un formulaire

```
<form id="userForm">
  <label for="name">Nom:</label>
  <input type="text" id="name" name="name" required><br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required><br>
  <button type="submit">Enregistrer</button>
</form>

<h2>Récupérer et afficher les informations:</h2>
<button id="getButton">Récupérer depuis localStorage</button>
<p id="output"></p>

<h2>Supprimer l'élément spécifique:</h2>
<button id="removeButton">Supprimer utilisateur de localStorage</button>

<h2>Supprimer tout:</h2>
<button id="clearButton">Supprimer toutes les données de localStorage</button>
```

localStorage / sessionStorage et JSON

Enregistrer les données dans localStorage

```
document.getElementById('userForm').addEventListener('submit', function(event) {  
    event.preventDefault();  
    const name = document.getElementById('name').value;  
    const email = document.getElementById('email').value;  
  
    const user = { name, email };  
  
    localStorage.setItem('user', JSON.stringify(user));  
    alert('Utilisateur enregistré dans localStorage !');  
});
```

localStorage / sessionStorage et JSON

Récupérer les données depuis localStorage

```
document.getElementById('getButton').addEventListener('click', function() {
    const storedUser = JSON.parse(localStorage.getItem('user'));
    if (storedUser) {
        document.getElementById('output').textContent =
            `Nom: ${storedUser.name},
             Email: ${storedUser.email}
            `;
    } else {
        document.getElementById('output').textContent = 'Aucune donnée trouvée dans localStorage.';
    }
});
```

localStorage / sessionStorage et JSON

Supprimer un élément spécifique

```
document.getElementById('removeButton').addEventListener('click', function() {
    localStorage.removeItem('user');
    alert('Utilisateur supprimé de localStorage !');
});
```

localStorage / sessionStorage et JSON

Supprimer toutes les données

```
document.getElementById('clearButton').addEventListener('click', function() {
    localStorage.clear();
    alert('Toutes les données ont été supprimées de localStorage !');
});
```



En conclusion

`localStorage` et `sessionStorage` sont des mécanismes puissants pour stocker des données côté client.

`localStorage` conserve les données de manière persistante, tandis que `sessionStorage` les conserve uniquement pendant la session du navigateur.

Vous pouvez stocker des objets JSON en sérialisant et désérialisant les données avec `JSON.stringify()` et `JSON.parse()`.