



TP N°3

Gestion d'un Panier E-commerce avec IndexedDB

Objectif : Cette activité vise à :

- Approfondir les connaissances en JavaScript
- Comprendre et implémenter un système de stockage local persistant
- Réaliser des opérations CRUD (Create, Read, Update, Delete) avec IndexedDB
- Gérer un panier e-commerce avec IndexedDB
- Manipuler des données en mode hors ligne

Contexte

Ce TP est une suite logique du TP2 (Il est important de compléter le TP2 avec succès), où vous avez créé l'interface de la boutique de café en ligne et affiché les produits depuis une API, vous allez maintenant ajouter une fonctionnalité essentielle pour tout site e-commerce : la gestion du panier d'achat. Les données du panier seront stockées localement en utilisant IndexedDB, permettant ainsi aux utilisateurs de conserver leur panier même après avoir fermé leur navigateur.

Ressources et outils

Le code du TP2 (fichiers HTML, CSS et JavaScript)

Navigateur chrome : Pour ce TP, veuillez utiliser le navigateur Chrome pour éviter tout problème de compatibilité.

Extension Chrome : IndexedDB viewer (optionnel mais recommandé)

Type de l'activité : Le travail doit être réalisé en binôme.

Résultat attendu : Les étudiants doivent soumettre avant le **30 novembre 2024**

- Un fichier compressé qui contient le répertoire du projet.
- Un fichier PDF contenant des captures écran, démontrant la réalisation de l'activité.
- Soumission via formulaire Google Forms : <https://forms.gle/xHN85g4JNp6VH2ySA>

Tâches à réaliser :

Dans le TP2, vous avez créé un fichier `script.js` qui gère l'affichage des produits depuis une API. vous allez modifier ce fichier pour ajouter la persistance des données avec IndexedDB.

Tâche 1 : Initialisation de la Base de Données

La première étape consiste à créer une fonction `openDB()` qui initialisera la connexion à la base de données et créera les object stores :

```
function openDB() {
  const dbRequest = indexedDB.open("CoffeeShopDB", 1);

  dbRequest.onupgradeneeded = (event) => {
    const db = event.target.result;
    if (!db.objectStoreNames.contains("products")) {
      db.createObjectStore("products", { keyPath: "id" });
    }
    if (!db.objectStoreNames.contains("cart")) {
      db.createObjectStore("cart", { keyPath: "id" });
    }
  };

  return dbRequest;
}
```

Tâche 2 : Modification de la Fonction `getProducts()`

Modifiez la fonction `getProducts()` existante pour ajouter le stockage des produits dans IndexedDB. Pour cela, vous devez appeler la fonction `addProductsToDB(products)` après avoir reçu les données de l'API.

Tâche 3 : Création de la Fonction `addProductsToDB()`

Créer une fonction `addProductsToDB(products)` qui permet de :

- Ouvrir une connexion à la base de données `const dbRequest = openDB();`
- Stocker les produits reçus en paramètre dans cet object store
- Afficher un message de succès dans la console

Tâche 4 : Validation de l'Implémentation

Pour tester votre implémentation :

- Ouvrez les DevTools (F12)
- Allez dans l'onglet "Application"
- Vérifiez sous "IndexedDB" que :

- a. La base "CoffeeShopDB" existe
- b. L'object store "products" contient les données des produits

Tâche 5 : Chargement des Produits en Mode Hors Ligne

- Modifiez la fonction `getProducts()` existante pour appeler la fonction `loadProductsFromDB()` notamment en cas d'échec de la connexion à l'API (mode hors ligne).

```
.catch(error => {
  console.error("Erreur lors de la récupération des produits:",
    error);
  loadProductsFromDB();
});
```

- Créer une fonction `loadProductsFromDB()` qui permet de récupérer les produits stockés dans IndexedDB et les afficher, L'affichage utilise la fonction `displayProducts()` existante.

Tâche 6 : Ajout au Panier

Implémentez la fonction `addToCart(productId)` pour gérer l'ajout des produits dans l'object store "cart" dans IndexedDB.

- Créez un objet avec la structure suivante :

```
// Créer l'objet à stocker dans le panier
let product_detail= products.find(p => p.id == productId);
const cartItem = {
  id: productId,
  image_url: product_detail.product_detail,
  name:product_detail.name,
  price: product_detail.price,
  quantity: 1
};
```

- Ajouter le produit dans l'object store "cart"
 - Cette fonction est appelée par le bouton "+" déjà implémenté dans `createProductCard()`
- `<button onclick="addToCart('${product.id}')" class="add-to-cart">+</button>`

Tâche 7 : Test de Validation

1. Cliquez sur le bouton "+" d'un produit
2. Inspectez dans DevTools > Application > IndexedDB > cart
3. Vérifiez que :
 - o L'item est ajouté au panier
 - o La quantité = 1

Tâche 8 : Implémentation de l'Affichage du Panier

- La page panier.html permet d'afficher les produits stockés dans l'object store "cart" de IndexedDB.

Créer le fichier panier.js

- Implémentez les fonctions suivantes dans panier.js :

Function loadProductsFromCart () :




- Charger les produits depuis l'object store "cart"
- Afficher chaque produit avec displayCartItem()

- Ajouter la fonction displayCartItem()

```
function displayCartItem(pr) {  
  const cartContainer = document.getElementById('cart-items');  
  cartContainer.innerHTML = ''; // Vider le contenu actuel  
  pr.forEach(product => {  
    cartContainer.appendChild(createCartItemRow(product));  
  });  
}
```

- Implementer createCartItemRow() qui permet de créer et retourner une ligne du tableau pour un produit du panier

Après l'exécution de votre code, le résultat affiché dans le navigateur doit être visuellement identique à l'image suivante :

Terres de Café				Retour aux produits
PRODUIT	PRIX	QUANTITÉ		TOTAL
 Golden Sunrise	10.99 dh	-	1 +	10.99 dh ×
 Harvest Moon	9.99 dh	-	1 +	9.99 dh ×
 Breezy Beans	11.99 dh	-	1 +	11.99 dh ×
				Total: 0 dh

Tâche 9 : Implémentation des fonctions de mise à jour et de suppression des articles du panier

Ajoutez les fonctionnalités de gestion du panier :

Function updateQuantity(itemId, newQuantity) :

- Met à jour la quantité d'un article dans IndexedDB
- Met à jour l'affichage

Function removeFromCart(itemId) :

- Supprime un article de l'object store "cart"
- Met à jour l'affichage