Contexte

Imaginons que vous travaillez sur un site web de critique de films. Vous utiliserez une API publique pour récupérer une liste de films, puis vous afficherez ces films sur votre page web avec des détails tels que le titre, l'année, et un court résumé.

Exercice Pratique

Partie 1 : Récupération des données

1. **Faire une requête Fetch pour récupérer les données**

   Utilisez l'API Fetch pour faire une requête GET vers une API de films. Pour cet exercice, vous pouvez utiliser l'API gratuite The Movie Database (TMDb) ou toute autre API publique de films qui retourne des données au format JSON.

```
const url = "https://api.example.com/films"; // Remplacez par l'URL
réelle de l'API

fetch(url)
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((error) =>
    console.error("Erreur lors de la récupération des films :", error)
  );
```

Partie 2 : Traitement des données JSON

2. **Extraire et manipuler les données JSON**

   Traitez les données reçues pour extraire les informations pertinentes pour chaque film (par exemple, titre, année de sortie, résumé).

```
fetch(url)
  .then((response) => response.json())
  .then((data) => {
    const films = data.results.map((film) => {
      return {
        titre: film.title,
        annee: film.release_date.split("-")[0], // Supposant que la
date est au format YYYY-MM-DD
        resume: film.overview,
      };
    });
    console.log(films);
  })
  .catch((error) =>
    console.error("Erreur lors de la manipulation des données :",
```

```
error)
  );
```

Partie 3 : Affichage des résultats

3. **Afficher les données sur une page web**

   Modifiez le DOM pour afficher les films sur votre page. Vous pouvez créer des éléments HTML dynamiquement et les insérer dans votre page.

```
function afficherFilms(films) {
  const conteneur = document.getElementById("films-conteneur"); //
Assurez-vous d'avoir cet élément dans votre HTML
  films.forEach((film) => {
    const elementFilm = document.createElement("div");
    elementFilm.innerHTML = `
      <h3>${film.titre}</h3>
      <p>Année de sortie : ${film.annee}</p>
      <p>Résumé : ${film.resume}</p>
    `;
    conteneur.appendChild(elementFilm);
  });
}

// Après avoir récupéré et traité les films
fetch(url)
  .then((response) => response.json())
  .then((data) => {
    const films = data.results.map((film) => ({
      titre: film.title,
      annee: film.release_date.split("-")[0],
      resume: film.overview,
    }));
    afficherFilms(films);
  })
  .catch((error) =>
    console.error("Erreur lors de l'affichage des films :", error)
  );
```

Partie 4 : Gestion des erreurs

4. **Gérer les erreurs**

   Assurez-vous de gérer les erreurs à chaque étape du processus, notamment les erreurs réseau et les erreurs de traitement des données JSON.

   - Utilisez `.catch()` à la fin de vos chaînes de promesses pour attraper les erreurs.
   - Vérifiez le statut de la réponse HTTP avant de tenter de la lire comme JSON.