

Partie 6: Tri de l'information

Dans cette partie, nous allons nous concentrer sur comment trier les données de notre app. Nous allons donc trier les films par étoiles, par titre, par date de visionnage et par date de sauvegarde dans la DB.

Champ de recherche

Les champs de recherche, que vous avez déjà sûrement dû utiliser sur des sites de e-commerce par exemple, sont des types de tri de l'information.

Nous avons déjà le squelette de notre champ de recherche puisque nous l'avons codé au début de ce cours, manque plus qu'à le faire marcher.

Exercice pratique n°5 A

DONNÉE: le but est de coder une fonction qui sera appelée chaque fois que l'utilisateur rentrera de l'information dans le champ de recherche. Cette fonction devra cacher les films qui ne correspondent pas à la chaîne de caractère entrée par l'utilisateur est au contraire laissé visible le/les films qui matche l'entrée de l'utilisateur.

Règles: cet exercice est un exercice de groupe (2 pers minimum, 4 pers maximum). Le but est que vous vous entraïdiez pour réussir à coder la fonction et que vous construisiez qqch ensemble !

Ressources: Pour vous aider, voici un petit texte de la fonction:

*" Il faut déjà me donner la **valeur** que je devrais analyser. Puis faire une transformation dessus pour que je puisse me dépatouiller entre les grandes et les petites. À la suite de cela il restera à me donner les **éléments** à faire disparaître. Pour finir un petit tour de magie et hop hidden seront les films. "*

Solution

```
// à chaque "keyup" de l'utilisateur sur le champ de recherche la fonction est lancée
domOn("#bytitle", "keyup", () => {
  //je récupère la valeur
  const letters = document.querySelector("#bytitle").value;
  // je la tranfome en majuscule
  const filter = letters.toUpperCase();
  //je chope tout les films
  const movies = document.querySelectorAll(".content-image");
  // je fais une boucle sur chaque film
  for (const movie of movies) {
    if (
      // si le titre du film en question que je mets aussi en majuscule matche l'entrée de l'utilisateur...
      movie.querySelector(".title").innerHTML.toUpperCase().indexOf(filter) > -1
    ) {
      //je laisse le film visible
      movie.classList.remove("hidden");
    } else {
      //je cache le film en question
      movie.classList.add("hidden");
    }
  }
});
```

Tri dans la base de données

Dans la partie d'avant, nous avons créé des index dans notre DB. Allons jeter un coup d'œil à ceux-ci.

Nous pouvons voir que Indexed DB a automatiquement indexé / trié les éléments de la base par rapport au nombre d'étoiles qui leur avaient été attribués.

ATTENTION: il faut bien que le nom de l'index (ici rate => Key path) ait le même nom que la clé dans l'objet JS (ici rate => dans value)

Le tri est donc fait dans la base manque plus qu'à appeler le bon index au bon moment pour trier les films !

Pour commencer, il faut ajouter un paramètre à notre fonction *callDOM* et rajouter un tri par index dans cette même fonction

```
//AVANT
const callDOM = async () => {
  //pour supprimer le DOM puis le mettre à jour
  if (document.querySelectorAll(".content-image") != null) {
    document.querySelectorAll(".content-image").forEach((el) => el.remove());
  }

  db.transaction("movies").objectStore("movies").getAll().onsuccess = (
    event
  ) => {
    //code...
  };
};
```



```

//APRES
//nouveau paramètre = choosenindex
const callDOM = async (choosenindex) => {
  //pour supprimer le DOM puis le mettre à jour
  if (document.querySelectorAll(".content-image") != null) {
    document.querySelectorAll(".content-image").forEach((el) => el.remove());
  }

  db
    .transaction("movies")
    .objectStore("movies")
    .index(`${choosenindex}`) //nouvelle ligne
    .getAll().onsuccess = (event) => {
      //code
    };
};

```

Avec un nouveau paramètre à la fonction `callDOM` cela veut dire qu'il faut envoyer un élément à chaque fois qu'on l'appelle. Nous l'appelons en effet pendant le *onsuccess* de la DB.

```
//AVANT
request.onsuccess = (event) => {
  db = event.target.result;
  callDOM();
  db.onerror = (event) => {
    console.error("Database error: " + event.target.errorCode);
  };
};
```

```
//APRES
request.onsuccess = (event) => {
  db = event.target.result;
  callDOM(chosenindex); //changement
  db.onerror = (event) => {
    console.error("Database error: " + event.target.errorCode);
  };
};
```

chosenindex ici étant une variable il faut la définir (tout au début du script après les imports). ON peut donc rajouter cette ligne.

```
let chosenindex = "storagedate"; //ici on mettra dans la variable le nom d'un des indexs de la DB
```

Comme codé dans cet exemple, le DOM appelé par défaut sera la liste des films triée par leur date d'ajout dans la DB.

MAIS nous voulons pouvoir grâce au bouton de tri, trier nos films !!

Il y a juste à faire 3 fonctions qui changent le *chosenindex* et qui appellent le DOM avec l'index voulu quand l'utilisateur clic sur le bouton.

```
// A la fin du code global
//appel du DOM avec le bon tri dessus
domOn("#bystars", "click", () => {
  chosenindex = "rate";
  callDOM(chosenindex);
});

domOn("#bydate", "click", () => {
  chosenindex = "watchingdate";
  callDOM(chosenindex);
});

domOn("#reset", "click", () => {
  chosenindex = "storagedate";
  callDOM(chosenindex);
});
```

Exercice pratique n°5 B

DONNÉE: mettez en place le tri par étoile, par date de visionnage et le reset du tri dans votre application.