Partie 7: finitions

Dans cette partie nous allons nous concentrer à finaliser l'application avant de pouvoir passer à la toute dernière partie: le offline.

Pour finaliser l'app il nous reste à:

- pouvoir naviguer dans les pages de notre app
- la rendre responsive à minima
- pouvoir supprimer un film

Les ancres

Pour naviguer dans notre application en utilisant l'historique de browser (les flèches donc ou alors sur les smartphones modernes le swipe) nous allons utiliser une méthode qui consiste à mettre des ancres dans le HTML et de manipuler ses ancres en JS.

1. il faut donc en premier lieu mettre des ancres dans le HTML. Une ancre est en fin de compte un lien donc une balise *<a href:"monancre">*. La seule différence est que l'on va ajouter un # avant et donner un nom choisi à l'ancre, ce qui donne:

```
//dans index.html
<a href="#add">
  <div id="sidenav">
    <span id="plus">&#43</span>
  </div>
</a>
```

Ce code va "simplememt" rajouter une ancre dans l'URL. Vous pouvez maintenant essayer de cliquer sur le bouton + et regarder le changement de l'URL.

2. Nous allons maintenant pouvoir manipuler et distinguer le changement de cette ancre grâce à *document.location.hash*

```
let anchor = document.location.hash;
```

Cette ligne va simplement prendre tout ce qui vient après le # et le mettre dans ma variable *anchor*, je vais donc pouvoir faire une fonction qui va cacher ou pas un bout du DOM quand l'ancre est sur mon URL ou pas.

```
const anchor = () => {
  let anchor = document.location.hash;
  if (!(anchor === "#add" || anchor === "#movies")) anchor = "#movies";
  if (anchor === "#movies") {
    document.querySelector("#add-movie").classList.add("hidden");
    document.querySelector("#global-page").classList.remove("hidden");
    document.querySelector("#sidenav").classList.remove("hidden");
    document.querySelector("#bytitle").focus();
  }
  if (anchor === "#add") {
    document.querySelector("#global-page").classList.add("hidden");
    document.querySelector("#sidenav").classList.add("hidden");
    document.querySelector("#add-movie").classList.remove("hidden");
  }
};

anchor();

window.addEventListener("popstate", () => {
  anchor();
});
```

QUESTION: Que fait cette fonction (anchor()) et les 3 lignes d'après ?

Exercice pratique n°6 A

DONNÉE: comme vous pouvez le voir, il y a un second *if* avec une seconde ancre. Modifier votre HTML pour qu'à l'ajout d'un film l'utilisateur soit redirigé automatiquement sur la liste de ses films.

ATENTION: il vous faudra supprimer

```
document.querySelector("#global-page").classList.remove("hidden");
document.querySelector("#add-movie").classList.add("hidden");
```

dans la fonction *domOn("#add-movie-button", "click", async () => {})*

mais aussi

```
domOn("#plus", "click", (evt) => {
  document.querySelector("#global-page").classList.add("hidden");
  document.querySelector("#add-movie").classList.remove("hidden");
});
```

et même en ayant supprimé tout cela, ça devrait marcher !

Responsive

Nous allons à présent parler un peu de responsive. Nous allons voir le responsive coté code et non coté design car cela prendrait largement un cours entier autrement.

Dans cette partie nous allons donc juste faire que les films se repositionnent comme il faut lorsqu'on resize la fenêtre du browser.

Nous allons utiliser comme base le CSS donné en exemple dans les corrections des exercices depuis le 3.

```css
/* dans stylesheet.css tout en bas du fichier après le footer */
@media screen and (max-width: 1500px), screen and (max-device-width: 1500px) {
  /* votre code CSS */
}
```

Voici la seule ligne à ajouter pour faire du responsive en CSS, enfin presque... 

1. *@media* est une règle CSS qui va définir et donc agir sur un type de média (écran (screen), impression (print), etc.).
2. *screen* ici en bleu va définir quel média veut-on manipuler
3. *max-width: 1500px* est notre règle en soit => cette règle va s'appliquer pour tous les écrans qui ont **une largeur en dessous de 1500px**
4. *screen and (max-device-width: 1500px)*, la suite est juste pour compléter la règle en disant que la taille du device doit aussi être prise en compte.
5. C'est donc entre les accolades de cette règle que vous allez pouvoir redéfinir des ordres CSS. Ces ordres ne seront pris en compte que si la règle vaut true.

```css
@media screen and (max-width: 1500px), screen and (max-device-width: 1500px) {
  /* je redéfinis les grille de l'id home pourqu'il n'y ait que 3 film
   par rangée quand la largeur de mon écran est inférieur à 1500   */
  #home {
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-areas: "cont cont cont";
  }
}
```

Pour finaliser, voici les 3 règles pour que nos films soient bien réarrangés sur tout type d'écran (surtout mobile)

```css
@media screen and (max-width: 1500px), screen and (max-device-width: 1500px) {
  #home {
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-areas: "cont cont cont";
  }
}

@media screen and (max-width: 1000px), screen and (max-device-width: 1000px) {
  #home {
    grid-template-columns: 1fr 1fr;
    grid-template-areas: "cont cont";
  }
}

@media screen and (max-width: 700px), screen and (max-device-width: 700px) {
  #home {
    grid-template-columns: 1fr;
    grid-template-areas: "cont";
  }

  #sidenav {
    transform: translate(-50%, 630%);
    top: 40vh;
  }
}
```

Suppression d'un film

Pour terminer notre finalisation, il nous reste à pouvoir supprimer un film.

Exercice pratique n°6 B

DONNÉE: mettez-vous en groupe (2 pers min, 4 pers max) et essayez de trouver une solution pour supprimer les films. Après un moment de réflexion, veuillez présenter votre solution au reste de la classe en expliquant pourquoi votre solution marche et quels en sont les avantages et inconvénients.

RESSOURCES:

- Pour aider 1
- Pour aider 2

Solution

1. Il faut pouvoir récupérer l'id (la clef) du film en question pour pouvoir par la suite l'identifier.
2. Pour faire cela, il faut avoir enregistré l'id au préalable.
   Pour enregistrer l'id nous pouvons lors de l'appel de la fonction *callDOM* sauvegarder l'id du film sur le bouton de suppression du film par exemple. Comme ça quand nous cliquerons sur le bouton supprimer nous pourrons récupérer l'id du film et le donné à la DB pour suppression.
3. L'enregistrement de l'id sur un élément HTML se fait via les *data-attributs*, a contrario des attributs "normaux" qui sont compris d'une certaine façon par le HTML (les classes, les id, etc.) les data-attributs eux sont là pour stocker de la data très très peu volumineuse (un id exemple)

```
<!-- dans mon template de film html -->
<!-- je peux trouver cette ligne -->
  <button id="delete">&#10005</button>
```

Modifions-la de telle sorte qu'un data-attribut vienne s'y loger

```
  <button id="delete" data-action="delete">&#10005</button>
  <!-- je peux en mettre tant que je veux, j'ai juste à mettre le mot clef data- avant -->
```

4. Maintenant je veux pouvoir le rajouter dynamiquement à tous mes films quand ils sont appelés vis ma fonction callDOM. Je rajoute donc cette ligne dans ma fonction *callDOM* juste après la création de ma variable *templateDone*

```
templateDone
  .querySelector("#delete") //je chope l'id de mon bouton delete
  .setAttribute("data-moviekey", element.numKey); // et j'y ajouter un data-attributs
//qui s'appelle data-moviekey et qui prends comme valeur le numéro de la clef (l'id) de mon film
//ATTENTION la donnée sera tranformée en chaine de caractère!
```

5. J'ai donc mon id sur mon bouton (je peux voir que cela a bien marché en inspectant mon code). Manque plus qu'à le récupérer lors du clic de l'utilisateur sur la croix de suppression et de le passer à la DB.

```
domOn("#home", "click", async (evt) => {
  // au clic sur mon élément global présent dans le DOM dèsdébut
  const target = evt.target; // je souvegarde l'élément exact qui a été cliqué
  if (target.dataset.action != "delete") return; // je vérife cet élément, si c'est un bouton de supression
  if (!confirm("Are you sure you want to delete me ?")) return; // je demande confirmation de supression
  const moviekeyString = target.dataset.moviekey;
  const movieKeyInt = parseInt(moviekeyString); //je tranforme ma chaine de caractère en nombre
  const request = db // je supprime le film de ma database
    .transaction("movies", "readwrite")
    .objectStore("movies")
    .delete(movieKeyInt);
  request.onsuccess = () => {
    //si cela à marché comme il faut je rappelle le DOM pour qu'il s'actualise.
    callDOM(choosenindex);
  };
});
```

❗ ATTENTION IMPORTANT ❗

Au moment où je charge le DOM la première fois il se peut qu’aucun bouton supprimé ne soit présent (c'est normal puisqu’aucun film n'a encore été ajouté). Je dois donc (très important!) agir sur un élément qui est présent dès le début sur le DOM ici: *#home*. Puis à la suite faire un tri en ne prenant le clic que s’il est fait un bouton de suppression; que je reconnais grâce à l'ajout de mon *data-attributs* (data-action="delete").

6. Je peux enfin supprimer les films que je ne veux plus !