Le projet - myMoviz
Day 4 - Communication entre composants

Objectif de la journée

L'application est devenue interactive: lorsque l'on clique sur les picto-coeurs ou que l'on sélectionne les items du menu, l'interface réagit et ses données se mettent à jour. Pour arriver à cette étape, il a fallu faire communiquer le composant App avec Movie, plus exactement faire redescendre les informations entre App et Movie au travers des propriétés.
L'objectif de cette étape est de synchroniser le nombre total de films sélectionnés avec le compteur de films, ainsi que la liste des 3 derniers titres visibles dans le header (composant pop over).
Pour réaliser cette étape, il sera nécessaire de mettre en oeuvre une communication inversée entre composants (inverse data flow). Le composant Movie devra envoyer une information pour que le composant App puisse mettre à jour son compteur.

1- Préparer l'environnement de travail

- Création du répertoire **part4** dans le répertoire **./lacapsule/project/mymoviz**
- Reprendre le livecode **part3** et le déposer dans le répertoire **part4**
- Se positionner sur le répertoire part4 via le terminal

```
cd ./lacapsule/project/mymoviz/part4
```

,

- Pour lancer la première fois l'application dans un navigateur, tapez dans le terminal la commande npm start

```
npm start
```

- Pour stopper le serveur, positionnez-vous dans le terminal et appuyez simultanément sur les touches suivantes:

```
ctrl + c
```

2- Mise à jour du compteur de films

L'objectif peut paraître simple: mettre à jour le compteur de films lorsqu'un film est liké ou disliké. Pour réaliser cela, il faut impérativement mettre en place la communication inversée entre **Movie** et **App**

Faire redescendre une partie de App vers Movie

- Dans le composant **App**, définissez une méthode `handleClick()`

- Dans cette méthode, affichez via un `console.log()` le message suivant "click detecté". Cette instruction est nécessaire pour valider cette étape

- Pour chaque composant Movie initialisé dans le composant **App**, créez une nouvelle propriété nommée **handleClickParent**.

- Initialisez la valeur de la propriété **handleClickParent** en y mettant le nom de la méthode `handleClick()`
  Note:
  C'est à cet instant précis que le code de la méthode **handleClickParent** de App va pouvoir être transmis aux composants Movies

Exploitez le code reçu

Dans le composant Movie, vous pouvez accéder au code reçu en utilisant l'instruction **this.props.handleClickParent**
La question que vous devez vous poser est la suivante: Quand doit-on exploiter le code contenu dans cette propriété ?
Pour avoir la réponse, il est nécessaire de garder en tête l'objectif: la mise à jour d'un compteur lorsque l'on clique sur le picto coeur.
Le code que l'on vient de passer va vous servir à terme à exécuter un code présent dans App depuis Movie dans l'objectif de mettre à jour le compteur.

- Localisez l'emplacement idéal pour y placer l'exécution de la méthode **handleClickParent()**
  Note:
  C'est à cet instant précis que le flux inversé prend tout son sens. Ce terme est bien entendu une image, mais il illustre cette mécanique qui permet d'exécuter le code d'un composant depuis un de ses composants enfants.
- Testez l'application en cliquant sur le coeur, le message "click detecté" devrait s'afficher dans la console.

Actualisez le compteur

A cette étape, il reste à mettre en oeuvre la partie qui va être responsable de la mise à jour du compteur. On souhaite stocker une information qui, dès qu'elle change, réactualise l'application. Il faudra donc utiliser la mécanique des états.

- Définissez dans le constructeur d'App un nouvel état **moviesCount** que vous initialiserez à 0
- Localisez l'emplacement idéal pour y placer la mise à jour l'état **moviesCount** afin d'y appliquer une incrémentation.
  Note:
  Attention lorsque vous faites une mise à jour d'un état, il ne faut pas assigner directement la propriété **this.state.moviesCount**
- Dans la partie `render()` de App, exploitez l'état **moviesCount** pour rendre dynamique le compteur

3- “disliker” un film

L'objectif est partiellement rempli, il y a un scénario qui n’est pas encore géré. Lorsque l’on clique de nouveau sur le même picto coeur celui-ci continu d’ajouter 1 au compteur alors que le résultat attendu serait plutôt d’enlever 1.

Envoyez une information complémentaire à App

En plus d'exécuter un code de son composant parent, le composant Movie va devoir envoyer à `handleClick()` l’état **like** dans lequel il se trouve. En effet cette information va permettre d’en conclure qu’il faudra réaliser une incrémentation ou décrémentation de l’état **moviesCount**

- Depuis Movie, envoyez en argument à `handleClick` la valeur de l’état **like**
  Note:
  Rappelez-vous dans Movie, handleClick est stocké dans la propriété **this.props.handleClickParent**

- Dans le composant App, modifiez la méthode `handleClick()` pour récupérer la valeur de l’état **like** envoyé précédemment. Stockez cette valeur dans un argument que vous nommerez **isLike**

- Mettez en place un traitement permettant de vérifier la valeur de **isLike** afin de déclencher une incrémentation ou une décrémentation de **moviesCount**

4- Mettre à jour la liste des 3 derniers films

Le composant pop-over chargé de lister les 3 dernier films sélectionnés n’est pas encore totalement dynamique. Cette liste de film est gérée intégralement par le composant App. Il faudrait que le composant Movie puisse lui envoyer à chaque like le nom du film sélectionné pour que le traitement localisé dans la méthode `handleClick()` puisse mettre à jour cette liste.

Envoyez une deuxième information à App

Le composant Movie va devoir en plus envoyer le nom du film à `handleClick()`
Il faut absolument que le composant App puisse le récupérer afin de mettre à jour la structure de données permettant de stocker l’information reçue.

- Depuis Movie, envoyez un deuxième argument à `handleClick` Cet argument fera référence au nom du film qui est une information déjà présente dans le composant **Movie**
  Note:
  Rappelez-vous dans Movie, handleClick est stocké dans la propriété **this.props.handleClickParent**
- Dans le composant App, modifiez la méthode `handleClick()` pour récupérer le nom du film envoyé par Movie. Stockez cette valeur dans un argument que vous nommerez **name**.

Gérez une liste de nom de films

A cette étape, App renvoie bien le nom du film lorsque l'on clique sur picto coeur.
Il reste à mettre en oeuvre les instructions qui vont permettre de stocker ces informations et également mettre à jour l'application pour afficher les 3 derniers films
De nouveau, lorsque l'on souhaite mettre en oeuvre une actualisation de l'application il faut automatiquement penser à la mécanique des états.

- Définissez dans le constructeur d'App un nouvel état **moviesNameList.**
  Note:
  A vous de choisir la bonne structure de données. Souvenez-vous, on souhaite stocker la liste des noms des films qui ont été sélectionnés.
- Localisez l'emplacement idéal pour y placer la mise à jour l'état **moviesNameList** afin d'y ajouter le nom du film.
  Note:
  Attention la structure de **moviesNameList** va vous obliger à réaliser une copie de cet état avant de pouvoir le modifier.

Exploitez la liste des films

Dernière étape pour rendre l'application totalement dynamique: modifier le traitement réalisé dans le render qui permet d'afficher la liste des 3 derniers films ainsi que le compteur total.

- Dans la partie `render()` de App, supprimez la ligne d'initialisation de la variable **moviesCount**
  Note:
  Attention à ne pas confondre la variable **moviesCount** avec l'état **this.state.moviesCount** que vous venez de créer. Même si leurs noms se ressemblent, les informations qu'elles possèdent ne sont pas les mêmes.

- A l'endroit où la variable **moviesCount** est utilisée dans le JSX, remplacez la par l'état **moviesCount**

- Toujours dans la partie `render()` de App, supprimez la ligne d'initialisation de la variable **moviesNameList**
  Note:
  Attention à ne pas confondre la variable **moviesNameList** avec l'état **this.state.moviesNameList** que vous venez de créer. Même si leurs noms se ressemblent, les informations qu'elles possèdent ne sont pas les mêmes.

- A l'endroit où la variable **moviesNameList** est utilisée dans le JSX, remplacez la par l'état **moviesNameList**