Mini-projet: Notes (Classes & objets)

Dans ce mini-projet qui mettra en application vos connaissances des classes et des objets vous créerez le front-end d'une application de prise de notes et de checklist.

Il vous sera fourni un fichier Javascript contenant sous format JSON un array mimiquant des notes/checklists déjà créées et récupérées depuis un serveur dans ordre croissant (les plus anciennes d'abord). Votre application devra également être capable d'en créer de nouvelles et de les stocker dans cet array également sous format JSON.

Une note se caractérise par un titre, un texte, une date de création, une couleur indiquant son importance et une date de rappel (par défaut undefined).
Une checklist se caractérise par un titre, un array contenant chacune des choses à accomplir sous forme d'objet (inspirez vous des notes existantes), une date de création, une couleur indiquant son importance et une date de rappel (par défaut undefined).
Utilisez des propriétés "privées", des getters et des setters et vérifiez que les conditions soient bonnes: types, couleurs correctes, etc.

Les codes couleurs sont les suivants:

- Rouge = urgent et important
- Orange = non-urgent et important
- Jaune = urgent et non-important
- Vert = non-urgent et non-important

Il vous est demandé de créer une classe ancêtre (classe **Note**) regroupant les caractéristiques communes entre ces deux types avant de définir des classes filles pour les notes (classe **TextNote**) et les checklist (classe **ChecklistNote**).

Commencez par instancier à partir des classes filles les notes composant l'array JSON et stockez les instances dans un array portant le nom **notesArray**. L'ordre de création décroissant (les plus récentes d'abord) devra êtres respecté.

La classe ancêtre devra possèder une méthode (fonction **displayNotes**) prenant deux arguments:

- Un array contenant des instances **TextNote** & **ChecklistNote**
- Un élément HTML qui servira de container dans lequel les afficher

Les notes devront avoir une méthode (fonction **render**) qui renverra un élément HTML crée à partir de l'objet. Cet élément HTML devra posséder une référence à l'objet sur lequel il est basé nommée "**_noteReference_**". La couleur de la note doit être visible.

Les checklists devront avoir une méthode (fonction **render**) qui renverra un élément HTML crée à partir de l'objet. Les éléments de la checklist devront être considérés comme étant soient accomplis soit non accomplis (par défaut). On pourra cliquer sur ces éléments pour changer leur état. Cet élément HTML devra posséder une référence à l'objet sur lequel il est basé nommée "**_noteReference_**". La couleur de la checklist doit être visible.

Permettez finalement à l'utilisateur de modifier les objets à travers l'affichage avec l'attribut **contenteditable**. L'utilisateur devrait être capable de modifier chacun des champs depuis l'HTML.

Bonus

Faites utilisation de la propriété de date de rappel pour notifier l'utilisateur quand le temps est venu.

Information utiles

N'oubliez pas que les copies d'objets sont en fait des copies de références (un pointeur vers un emplacement mémoire spécifique) et non une duplication de l'objet. L'objet n'est présent qu'à un seul endroit dans la mémoire donc les modifications à partir d'une référence impactent à toutes les références de cet objet.

document.createElement: Est une fonction permettant de créer un élément HTML sans le placer directement. Vous pouvez modifiez ses caractéristiques avec outerHTML, innerHTML ou directement à travers ses propriétés (avec le point/crochets ou un **setAttribute** quand approprié)

element.insertAdjacentElement: Est une fonction pour insérez un élément HTML positionné par rapport à un autre.

childNode.replaceWith: Permet de remplacer un élément HTML par un ou plusieurs autres éléments HTML. Elle pourrait être utile si vous désirez pouvoir également modifier le HTML à partir de l'objet et pas uniquement de l'HTML vers l'objet. (En utilisant la méthode **render** des notes)