TODO LIST EN REACT JS

Objectif :

Dans cet exercice vous devrez créer une application en REACT JS en utilisant les différents concepts abordés en classe.
Pour la réalisation de cette application, vous pouvez utiliser des classes ou exclusivement des fonctions avec les Hooks.

Le style de cette maquette est récupérable dans le dossier support fourni avec ces consignes.

TODO LIST EN REACT JS

Détails des différentes étapes à réaliser :

1 - Vous allez devoir décomposer cette application en différents composants réutilisables

Les différents composants à créer :

- App
- Header
- Button
- AddTodo
- Todos
- Todo
- Footer
- About

TODO LIST EN REACT JS

2 - Afficher une liste de Todos depuis le composant App

- Mettre en place un state : [todos, setTodos]
- Un todo est un objet avec les propriétés suivantes : id, text, day, reminder
- Pour cela vous allez importer le composants Todos
- Lui passer en props les différents todos de mon state todos
- Dans le composant Todos, vous allez afficher pour chaque todo un composant Todo en utilisant la méthode map()
- Si la liste est vide, afficher un message à l'écran indiquant 'Aucun élément dans la liste'. Pour cela vous devrez mettre en place une condition ternaire dans le return de votre fonction composant

3 - Ajouter un Todo à la liste de Todos

Il faudra ici permettre à l'utilisateur de rentrer un todo via le formulaire.

- Mettre en place le formulaire dans le composant AddTodo
- Capter l'événement submit
- Appeler une props définie depuis le composant App qui va réceptionner le nouveau todo que l'on va ajouter à notre state actuel
- Appeler la fonction dans App qui met à jour le state todos avec la fonction setTodos() en utilisant les spread operator (faire une copie du state actuel et y ajouter à l'intérieur mon nouveau todo

4 - Gérer la suppression d'un todo - onClick

Il faudra ici permettre à l'utilisateur de supprimer en cliquant sur un icon de suppression

- Installer npm i react-icons pour récupérer des icons
- ajouter un icon qui est la petite croix rouge (react-icons) dans le composant Todo (il faut pour ça installer un package (react-icons)
- capter l'événement click avec l'attribut JSX onClick sur chaque composant Todo
- l'événement click dans Todo va appeler un props depuis Todos et passer en paramètre l'id du Todo sur lequel j'ai cliqué
- la props appelée dans Todos va appeler une props dans App
- (nous voulons remonter l'événement click déclenché dans Todo jusqu'au composant App
- Todo -> Todos -> App
- la props dans App va appeler la fonction responsable de la suppression du Todo en mettant à jour le state todos en utilisant la méthode filter() et setTodos()

TODO LIST EN REACT JS

5 - Gérer le toggle sur la propriété reminder - onDoubleClick

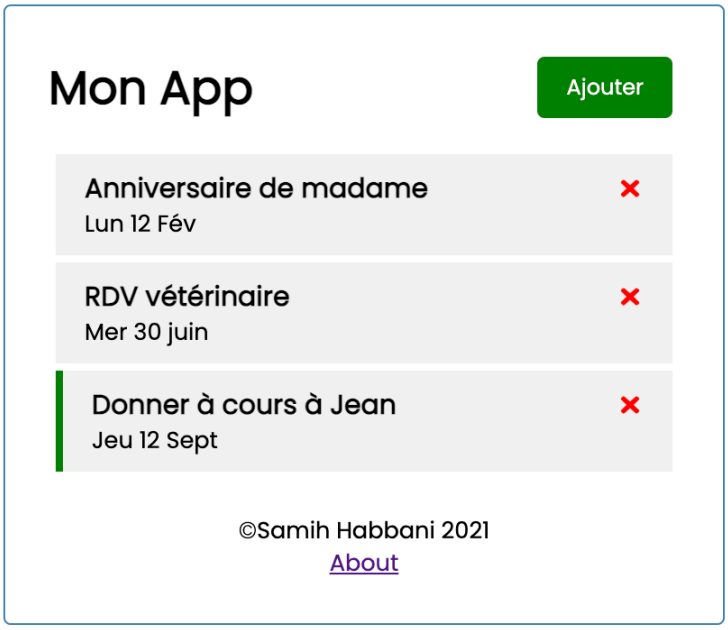
La propriétaire reminder permet de mettre un rappel sur un todo. Une fois activée, cela ajoute un bordure verte sur le todo. Il faut gérer le toggle sur cette propriété en double cliquant sur un todo.

- capter l'événement double click sur un composant Todo
- renvoyer l'événement vers App comme pour la suppression en passant par les props du composant Todos et App en passant en paramètre l'id du Todo qui a été double cliqué
- la props dans le composant App va appeler la fonction responsable de mettre à jour la propriété reminder pour le todo concerné
- il va falloir faire un toggle sur la valeur de cette propriété
- et mettre à jour le state (pour cela vous allez utiliser la méthode map() et setTodos()
- il faudra également mettre la classe .reminder sous conditions sur le composant Todo
- si la propriété reminder du todo est true alors ajouter cette classe (cela ajoutera la bordure verte sur le côté, sinon ne pas l'ajouter)

6 - Gérer le toggle de l'affichage du formulaire - onClick

Objectif : faire apparaitre disparaître le formulaire en faisant évoluer le style du bouton.

- mettre en place un state : [showAddTodo, setShowAddTodo] et initialiser sa valeur à true (par défaut j'afficherai le formulaire au chargement du composant)
- passer en props la valeur de ce state showAddTodo au composant Header qui contient notre Button
- mettre une condition sur le composant Header : si showAddTodo est true alors le texte à afficher dans le bouton est 'Fermer' et la couleur sera 'rouge' sinon le texte du bouton sera 'Ajouter' et le bouton sera vert. La couleur du background du bouton et le texte devront être passés en props dans le composant Button.
- capter l'événement click sur le composant Buton et faire un toggle du state showAddTodo dans App en passant par les props de Header et App.
- la props définie dans App appellera la fonction responsable de faire un toggle sur le state showAddTodo en utilisant la fonction setShowAddTodo
- si le state showAddTodo est true alors j'afficherai le composant addTodo sinon je ne l'afficherai pas à l'écran

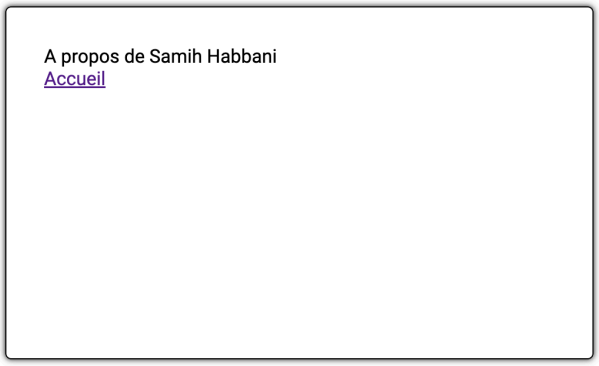


TODO LIST EN REACT JS

7 - Gérer la navigation dans le footer

L'objectif est de créer une navigation qui permette de naviguer entre la page d'accueil et la page Home.

- Installer le package npm install react-router-dom@6
- Importer le module BrowserRouter, et Routes, Route
- Imbriquer la div parent du composant App avec <BrowserRouter>
- Créer les différentes <Route> de l'application dans le composant <Routes>
- Ajouter un lien de navigation dans le Footer avec un composant <Link>
- Créer la page About
- Créer un lien de navigation dans la page About avec un composant <Link>



8 - Mettre en place faux server JSON avec une API Mock

L'objectif est de mettre en place un faux Backend avec JSON-SERVER

- Vous aller devoir installer le package npm install -g json-server
- Vous aller devoir ajouter un raccourcis de commande dans votre package.json au niveau des sripts : "server": "json-server --watch db.json --port 8000"
- Pour lancer le serveur : npm run server
- Cela va vous générer un fichier db.json
- A l'intérieur de ce fichier il faudra déplacer notre liste de todos sous forme d'objet json
- Ce fichier db.json simulera votre BDD
- Une fois mis en place, vous pourrez requêter de manière asynchrone via l'API FETCH en utilisant les différents méthodes POST - GET - PUT - DELETE…

TODO LIST EN REACT JS

9 - Connecter l'application à ce faux serveur pour avoir des données dynamiques venant d'un serveur

L'objectif est de connecter toutes nos fonctionnalités (ajout, suppression, modification, récupération (CRUD) à une BDD via l'api FETCH de manière asynchrone.

- Charger de manière asynchrone les données de notre liste via une requête GET
- Supprimer chaque todo de manière asynchrone via une requête DELET
- Modifier chaque todo de manière asynchrone via une requête PUT
- Ajouter un todo de manière asynchrone via une requête POST

BONUS :

- Mettre en place un button ON/OFF qui permet d'ajouter/enlever le rappel sur tous les todos.
- Mettre en place un button supprimer qui permette de supprimer tous les todos
- Agrandir le todo (mettre en relief) si sa date correspond à la date du jour