Atelier 15

Objectif d'apprentissage :

- Appel d'une API avec Redux-Toolkit

Les étapes à suivre

1- Créer un projet React avec la commande :

```
npx create-react-app example-call-api-redux
```

2- Installer la bibliothèque Axios :

```
npm i axios
```

3- Installer les bibliothèques de Redux

```
npm i react-redux @reduxjs/toolkit
```

4- Créer le Redux Store :

1. Créer un dossier redux dans le repertoire src.
2. Dans le répertoire redux créer le fichier store.jsx
3. Ecrire le code suivant dans ce fichier :

5- Ajouter dans le fichier index.js, Définir le Provider autour du composant App. Comme le montre le code suivant :

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { Provider } from 'react-redux';
import store from './redux/store'

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Provider store={store}>
    <App />
  </Provider>
);
```

6- Gérer les appels d'API asynchrones avec Redux-toolkit :

```
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit'
import axios from 'axios'
```

```
export const getProducts = createAsyncThunk('produits/getProduits', async ()
=> {
  const response = await axios.get('https://api.escuelajs.co/api/v1/products')
  return response.data
})


export const ProduitSlice = createSlice({
  name: 'produits',
  initialState: {
    data: [],
    loading: false,
    error: null,
  },
  reducers: {},
  extraReducers: (builder) => {
    builder.addCase(getProducts.pending, (state, action) => {
        state.loading = true
    })
    builder.addCase(getProducts.fulfilled, (state, action) => {
        state.data = action.payload
        state.loading = false
    })
    builder.addCase(getProducts.rejected, (state, action) => {
        state.loading = false
        state.error = 'Error occured'
      }
    })
  },
})
export default ProduitSlice.reducer
```

La fonction createAsyncThunk :

CreateAsyncThunk est l'endroit où nous effectuons des tâches asynchrones dans notre Slice. Il reçoit deux paramètres

- Le nom de l'action, la convention standard étant "[nom de la tranche]/[nom de l'action]", par exemple "posts/fetchPosts".
- La fonction de Callback qui effectue l'appel API et renvoie le résultat lorsqu'il est terminé. Notre appel API renvoie une promesse (qui est un objet représentant le statut d'une opération asynchrone, dans notre cas un appel API).

extraReducers :

Vous utilisez des extraReducers pour gérer les actions qui sont créées par createAsyncThunk. En fonction de l'état de la promesse, nous allons mettre à jour notre état.

7- Affichage des produits :

```jsx
import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { getProducts } from "../redux/ProduitSlice";

export default function ListProduits() {

  const dispatch = useDispatch();
  const { data, loading, error } = useSelector((state) => state.produits);

  useEffect(() => {
    dispatch(getProducts())
  }, [dispatch])


  let content


  if (loading) {
    content = (
      <div>
        <h1>Chargement en cours .....</h1>
      </div>
    )
  }
  if (!loading) {
    content = data.map((item) => {
      return <div>{item.id} - {item.title}</div>
    })
  }
  if (error !== null) {
    content = (
      <div>
       <h1>Erreur de chargement ... </h1>
      </div>
    )
  }
  return <div className="row">{content}</div>
}
```