Cours MERN - Semaine 8

Gestion d'État Globale en React

`useContext`, `Redux` et `Zustand`

Abdelweheb GUEDDES & Mohamed Ben Jazia / Ecole Polytechnique Sousse

8 novembre 2025

Table des matières

1 Objectifs Pédagogiques

À la fin de cette séance, vous serez capable de :

— **Comprendre** le problème du prop drilling
— **Créer** un Context avec useContext
— **Configurer** Redux Toolkit dans un projet
— **Utiliser** Zustand pour une gestion d'état simple
— **Gérer** des données d'API avec état global
— **Choisir** la solution adaptée selon votre projet

2 Partie 1 : Introduction et Concepts (40 min)

2.1 Le Problème : Prop Drilling

Problème : Passer des props à travers plusieurs niveaux de composants.

```
// App doit passer username à tous les enfants
function App() {
    const [username, setUsername] = useState("Alice");

    return <Header username={username}>
        <Navbar username={username}>
            <UserMenu username={username} />  {/* Utilise enfin ! */}
        </Navbar>
    </Header>;
}
```

Solution : État global accessible partout directement.

2.2 Solution 1 : useContext

2.2.1 Comment créer un Context ?

Étape 1 : Créer le Context

```
import { createContext, useContext, useState } from 'react';

const UserContext = createContext();

export const UserProvider = ({ children }) => {
    const [username, setUsername] = useState("Alice");

    return (
        <UserContext.Provider value={{ username, setUsername }}>
            {children}
        </UserContext.Provider>
    );
};

export const useUser = () => useContext(UserContext);
```

Étape 2 : Wrapper l'application

```
<UserProvider>
    <App />
</UserProvider>
```

Étape 3 : Utiliser

```
function UserMenu() {
    const { username } = useUser();
    return <div>Bonjour {username} !</div>;
}
```

2.3 Solution 2 : Redux Toolkit

Créer un Slice

```
import { createSlice } from '@reduxjs/toolkit';

const userSlice = createSlice({
    name: 'user',
    initialState: { username: 'Alice' },
    reducers: {
        setUsername: (state, action) => {
            state.username = action.payload;
        }
    }
});

```

```
export const { setUsername } = userSlice.actions;
export default userSlice.reducer;
```

Utiliser

```
const username = useSelector(state => state.user.username);
const dispatch = useDispatch();
dispatch(setUsername('Bob'));
```

2.4 Solution 3 : Zustand

Créer le Store

```
import { create } from 'zustand';

const useUserStore = create((set) => ({
    username: 'Alice',
    setUsername: (name) => set({ username: name })
}));

export default useUserStore;
```

Utiliser (pas de Provider !)

```
const username = useUserStore(state => state.username);
const setUsername = useUserStore(state => state.setUsername);
```

3 Partie 2 : Le Projet - Galerie de Personnages (2h20)

Projet : Créer une application qui charge des personnages depuis l'API Rick & Morty.

Fonctionnalités :

- Charger 20 personnages depuis l'API
- Afficher dans une grille avec image, nom, statut
- Système de likes (cœur) pour chaque personnage
- Header affichant le nombre total de likes
- Liste des personnages likés dans une barre latérale
- Filtrer par statut (alive/dead/unknown)

API : `https://rickandmortyapi.com/api/character`

3.1 Structure du Projet

```
src/
        styles/
             styles.css
        components/
             Header.jsx
             CharacterGrid.jsx
             CharacterCard.jsx
             FavoritesSidebar.jsx
             FilterBar.jsx
        App.jsx
```

3.2 Le Fichier CSS

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Arial', sans-serif;
    background-color: #1e1e1e;
    color: white;
}

.header {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    padding: 20px;
    text-align: center;
    box-shadow: 0 2px 10px rgba(0,0,0,0.3);
}

.header h1 {
    font-size: 32px;
    margin-bottom: 10px;
}

.likes-badge {
    display: inline-block;
    background-color: #e74c3c;
    padding: 8px 16px;
    border-radius: 20px;
    font-weight: bold;
}

.main-container {
    display: grid;
    grid-template-columns: 1fr 300px;
    gap: 20px;
    padding: 20px;
    max-width: 1400px;
    margin: 0 auto;
}

.filter-bar {
    display: flex;
    gap: 10px;
    justify-content: center;
    margin-bottom: 20px;
}

.filter-btn {
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-weight: bold;
    transition: all 0.3s;
}

.filter-btn.active {
    background-color: #667eea;
    color: white;
}

.filter-btn:not(.active) {
    background-color: #3a3a3a;
    color: #ccc;
}

.character-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
    gap: 20px;
```

```css
}

.character-card {
    background-color: #2a2a2a;
    border-radius: 10px;
    overflow: hidden;
    cursor: pointer;
    transition: transform 0.3s;
    position: relative;
}

.character-card:hover {
    transform: translateY(-5px);
}

.character-card img {
    width: 100%;
    height: 200px;
    object-fit: cover;
}

.character-info {
    padding: 15px;
}

.character-name {
    font-size: 18px;
    font-weight: bold;
    margin-bottom: 5px;
}

.character-status {
    font-size: 14px;
    color: #95a5a6;
}

.status-alive {
    color: #27ae60;
}

.status-dead {
    color: #e74c3c;
}

.like-btn {
    position: absolute;
    top: 10px;
    right: 10px;
    background-color: rgba(0,0,0,0.7);
    border: none;
    border-radius: 50%;
    width: 40px;
    height: 40px;
    font-size: 20px;
    cursor: pointer;
    transition: transform 0.2s;
}

.like-btn:hover {
    transform: scale(1.2);
}

.like-btn.liked {
    background-color: #e74c3c;
}

.favorites-sidebar {
    background-color: #2a2a2a;
    border-radius: 10px;
    padding: 20px;
    height: fit-content;
    position: sticky;
    top: 20px;
}

.favorites-sidebar h2 {
    margin-bottom: 15px;
    color: #667eea;
}

.favorite-item {
    display: flex;
    align-items: center;
    gap: 10px;
    padding: 10px;
    background-color: #3a3a3a;
    border-radius: 5px;
    margin-bottom: 10px;
}

.favorite-item img {
    width: 40px;
    height: 40px;
    border-radius: 50%;
    object-fit: cover;
}

.loading {
    text-align: center;
    padding: 50px;
    font-size: 24px;
```

```
}
```

Listing 1 – src/styles/styles.css

4 Version 1 : useContext (40 min)

4.1 Créer le Context

```
import { createContext, useContext, useState, useEffect } from 'react';

const CharactersContext = createContext();

export const useCharacters = () => {
    const context = useContext(CharactersContext);
    if (!context) {
        throw new Error('useCharacters must be used within CharactersProvider');
    }
    return context;
};

export const CharactersProvider = ({ children }) => {
    const [characters, setCharacters] = useState([]);
    const [likedIds, setLikedIds] = useState([]);
    const [filter, setFilter] = useState('all');
    const [loading, setLoading] = useState(true);

    // Charger les personnages
    useEffect(() => {
        fetch('https://rickandmortyapi.com/api/character')
            .then(res => res.json())
            .then(data => {
                setCharacters(data.results);
                setLoading(false);
            });
    }, []);

    const toggleLike = (id) => {
        setLikedIds(prev =>
            prev.includes(id)
                ? prev.filter(likedId => likedId !== id)
                : [...prev, id]
        );
    };

    const getLikedCharacters = () => {
        return characters.filter(char => likedIds.includes(char.id));
    };

    const getFilteredCharacters = () => {
        if (filter === 'all') return characters;
        return characters.filter(char => char.status.toLowerCase() === filter);
    };

    const value = {
        characters,
        likedIds,
        filter,
        loading,
        toggleLike,
        setFilter,
        getLikedCharacters,
        getFilteredCharacters
    };

    return (
        <CharactersContext.Provider value={value}>
            {children}
        </CharactersContext.Provider>
    );
};
```

Listing 2 – src/context/CharactersContext.jsx

4.2 Wrapper dans main.jsx

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { CharactersProvider } from './context/CharactersContext';
import './styles/styles.css';

ReactDOM.createRoot(document.getElementById('root')).render(
    <React.StrictMode>
        <CharactersProvider>
            <App />
        </CharactersProvider>
    </React.StrictMode>
);
```

Listing 3 – src/main.jsx

4.3 Les Composants

```
import { useCharacters } from '../context/CharactersContext';

function Header() {
    const { likedIds } = useCharacters();

    return (
        <header className="header">
            <h1>Rick & Morty Characters (Context)</h1>
            <div className="likes-badge">
                        {likedIds.length} personnages likés
```

```jsx
            </div>
        </header>
    );
}

export default Header;
```

Listing 4 – src/components/Header.jsx

```jsx
import { useCharacters } from '../context/CharactersContext';

function FilterBar() {
    const { filter, setFilter } = useCharacters();

    const filters = ['all', 'alive', 'dead', 'unknown'];

    return (
        <div className="filter-bar">
            {filters.map(f => (
                <button
                    key={f}
                    className={`filter-btn ${filter === f ? 'active' : ''}`}
                    onClick={() => setFilter(f)}
                >
                    {f.charAt(0).toUpperCase() + f.slice(1)}
                </button>
            ))}
        </div>
    );
}

export default FilterBar;
```

Listing 5 – src/components/FilterBar.jsx

```jsx
import { useCharacters } from '../context/CharactersContext';

function CharacterCard({ character }) {
    const { likedIds, toggleLike } = useCharacters();
    const isLiked = likedIds.includes(character.id);

    return (
        <div className="character-card">
            <button
                className={`like-btn ${isLiked ? 'liked' : ''}`}
                onClick={() => toggleLike(character.id)}
            >
                {isLiked ? '      ' : '      '}
            </button>

            <img src={character.image} alt={character.name} />

            <div className="character-info">
                <div className="character-name">{character.name}</div
>
                <div className={`character-status status-${character.
status.toLowerCase()}`}>
                    {character.status} - {character.species}
                </div>
            </div>
        </div>
    );
}

export default CharacterCard;
```

Listing 6 – src/components/CharacterCard.jsx

```
import { useCharacters } from '../context/CharactersContext';
import CharacterCard from './CharacterCard';

function CharacterGrid() {
    const { getFilteredCharacters, loading } = useCharacters();

    if (loading) {
        return <div className="loading">Chargement...</div>;
    }

    const filteredCharacters = getFilteredCharacters();

    return (
        <div className="character-grid">
            {filteredCharacters.map(character => (
                <CharacterCard key={character.id} character={
character} />
            ))}
        </div>
    );
}

export default CharacterGrid;
```

Listing 7 – src/components/CharacterGrid.jsx

```
import { useCharacters } from '../context/CharactersContext';

function FavoritesSidebar() {
    const { getLikedCharacters, toggleLike } = useCharacters();
    const likedCharacters = getLikedCharacters();

    return (
        <div className="favorites-sidebar">
            <h2>Mes Favoris ({likedCharacters.length})</h2>

            {likedCharacters.length === 0 ? (
```

```jsx
                <p style={{ color: '#95a5a6' }}>Aucun favori</p>
            ) : (
                likedCharacters.map(character => (
                    <div key={character.id} className="favorite-item"
>
                        <img src={character.image} alt={character.
name} />
                        <span>{character.name}</span>
                        <button
                            onClick={() => toggleLike(character.id)}
                            style={{
                                marginLeft: 'auto',
                                background: 'none',
                                border: 'none',
                                cursor: 'pointer',
                                fontSize: '20px'
                            }}
                        >

                        </button>
                    </div>
                ))
            )}
        </div>
    );
}

export default FavoritesSidebar;
```

Listing 8 – src/components/FavoritesSidebar.jsx

```jsx
import Header from './components/Header';
import FilterBar from './components/FilterBar';
import CharacterGrid from './components/CharacterGrid';
import FavoritesSidebar from './components/FavoritesSidebar';

function App() {
    return (
        <div>
            <Header />
            <div className="main-container">
                <div>
                    <FilterBar />
                    <CharacterGrid />
                </div>
                <FavoritesSidebar />
            </div>
        </div>
    );
}

```

```
export default App;
```

Listing 9 – src/App.jsx

5 Version 2 : Redux Toolkit (50 min)

5.1 Installation

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

5.2 Créer le Slice

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

// Thunk pour charger les personnages
export const fetchCharacters = createAsyncThunk(
    'characters/fetchCharacters',
    async () => {
        const response = await fetch('https://rickandmortyapi.com/api/character');
        const data = await response.json();
        return data.results;
    }
);

const charactersSlice = createSlice({
    name: 'characters',
    initialState: {
        characters: [],
        likedIds: [],
        filter: 'all',
        loading: false
    },
    reducers: {
        toggleLike: (state, action) => {
            const id = action.payload;
            if (state.likedIds.includes(id)) {
                state.likedIds = state.likedIds.filter(likedId => likedId !== id);
            } else {
                state.likedIds.push(id);
            }
        },
        setFilter: (state, action) => {
            state.filter = action.payload;
        }
    },
    extraReducers: (builder) => {
        builder
            .addCase(fetchCharacters.pending, (state) => {
                state.loading = true;
            })
            .addCase(fetchCharacters.fulfilled, (state, action) => {
                state.characters = action.payload;
                state.loading = false;
            });
    }
});

export const { toggleLike, setFilter } = charactersSlice.actions;

// Selectors
export const selectCharacters = (state) => state.characters.
    characters;
export const selectLikedIds = (state) => state.characters.likedIds;
export const selectFilter = (state) => state.characters.filter;
export const selectLoading = (state) => state.characters.loading;

export const selectLikedCharacters = (state) => {
    return state.characters.characters.filter(char =>
        state.characters.likedIds.includes(char.id)
    );
};

export const selectFilteredCharacters = (state) => {
    const { characters, filter } = state.characters;
    if (filter === 'all') return characters;
    return characters.filter(char => char.status.toLowerCase() ===
    filter);
};

export default charactersSlice.reducer;
```

Listing 10 – src/store/charactersSlice.js

5.3 Configurer le Store

```
import { configureStore } from '@reduxjs/toolkit';
import charactersReducer from './charactersSlice';

export const store = configureStore({
    reducer: {
        characters: charactersReducer
    }
});
```

Listing 11 – src/store/store.js

5.4 Provider dans main.jsx

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

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

Listing 12 – src/main.jsx

5.5 Les Composants Redux

```
import { useSelector } from 'react-redux';
import { selectLikedIds } from '../store/charactersSlice';

function Header() {
    const likedIds = useSelector(selectLikedIds);

    return (
        <header className="header">
            <h1>Rick & Morty Characters (Redux)</h1>
            <div className="likes-badge">
                    {likedIds.length} personnages likés
            </div>
        </header>
    );
}

export default Header;
```

Listing 13 – src/components/Header.jsx

```
import { useSelector, useDispatch } from 'react-redux';
import { selectFilter, setFilter } from '../store/charactersSlice';

function FilterBar() {
    const filter = useSelector(selectFilter);
    const dispatch = useDispatch();

    const filters = ['all', 'alive', 'dead', 'unknown'];

    return (
        <div className="filter-bar">
            {filters.map(f => (
                <button
                    key={f}
                    className={`filter-btn ${filter === f ? 'active' : ''}`}
                    onClick={() => dispatch(setFilter(f))}
                >
```

```
                    {f.charAt(0).toUpperCase() + f.slice(1)}
                </button>
            ))}
        </div>
    );
}

export default FilterBar;
```

Listing 14 – src/components/FilterBar.jsx

```
import { useSelector, useDispatch } from 'react-redux';
import { selectLikedIds, toggleLike } from '../store/charactersSlice';

function CharacterCard({ character }) {
    const likedIds = useSelector(selectLikedIds);
    const dispatch = useDispatch();
    const isLiked = likedIds.includes(character.id);

    return (
        <div className="character-card">
            <button
                className={`like-btn ${isLiked ? 'liked' : ''}`}
                onClick={() => dispatch(toggleLike(character.id))}
            >
                {isLiked ? '      ' : '      '}
            </button>

            <img src={character.image} alt={character.name} />

            <div className="character-info">
                <div className="character-name">{character.name}</div>
                <div className={`character-status status-${character.status.toLowerCase()}`}>
                    {character.status} - {character.species}
                </div>
            </div>
        </div>
    );
}

export default CharacterCard;
```

Listing 15 – src/components/CharacterCard.jsx

```
import { useEffect } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import {
    selectFilteredCharacters,
    selectLoading,
    fetchCharacters
} from '../store/charactersSlice';
import CharacterCard from './CharacterCard';

function CharacterGrid() {
    const characters = useSelector(selectFilteredCharacters);
    const loading = useSelector(selectLoading);
    const dispatch = useDispatch();

    useEffect(() => {
        dispatch(fetchCharacters());
    }, [dispatch]);

    if (loading) {
        return <div className="loading">Chargement...</div>;
    }

    return (
        <div className="character-grid">
            {characters.map(character => (
                <CharacterCard key={character.id} character={character} />
            ))}
        </div>
    );
}

export default CharacterGrid;
```

Listing 16 – src/components/CharacterGrid.jsx

```
import { useSelector, useDispatch } from 'react-redux';
import { selectLikedCharacters, toggleLike } from '../store/charactersSlice';

function FavoritesSidebar() {
    const likedCharacters = useSelector(selectLikedCharacters);
    const dispatch = useDispatch();

    return (
        <div className="favorites-sidebar">
            <h2>Mes Favoris ({likedCharacters.length})</h2>

            {likedCharacters.length === 0 ? (
                <p style={{ color: '#95a5a6' }}>Aucun favori</p>
            ) : (
                likedCharacters.map(character => (
                    <div key={character.id} className="favorite-item">
                        <img src={character.image} alt={character.name} />
                        <span>{character.name}</span>
                        <button
```

```
                              onClick={() => dispatch(toggleLike(
    character.id))}
                              style={{
                                  marginLeft: 'auto',
                                  background: 'none',
                                  border: 'none',
                                  cursor: 'pointer',
                                  fontSize: '20px'
                              }}
                          >
                          
                          </button>
                      </div>
                  ))
              )}
          </div>
      );
}

export default FavoritesSidebar;
```
Listing 17 – src/components/FavoritesSidebar.jsx

Points clés Redux :
- `createAsyncThunk` pour les appels API
- `extraReducers` pour gérer les états async (pending/fulfilled)
- `Selectors` pour calculer les données dérivées
- Performance optimale avec sélection précise

6 Version 3 : Zustand (40 min)

6.1 Installation

```
npm install zustand
```

6.2 Créer le Store

```
import { create } from 'zustand';

const useCharactersStore = create((set, get) => ({
    characters: [],
    likedIds: [],
    filter: 'all',
    loading: true,

    // Charger les personnages
    fetchCharacters: async () => {
        set({ loading: true });
        const response = await fetch('https://rickandmortyapi.com/api/character');
        const data = await response.json();
        set({ characters: data.results, loading: false });
    },

    // Toggle like
    toggleLike: (id) => set((state) => ({
        likedIds: state.likedIds.includes(id)
            ? state.likedIds.filter(likedId => likedId !== id)
            : [...state.likedIds, id]
    })),

    // Changer le filtre
    setFilter: (filter) => set({ filter }),

    // Selectors
    getLikedCharacters: () => {
        const { characters, likedIds } = get();
        return characters.filter(char => likedIds.includes(char.id));
    },

    getFilteredCharacters: () => {
        const { characters, filter } = get();
        if (filter === 'all') return characters;
        return characters.filter(char => char.status.toLowerCase() === filter);
    }
}));

```

```
export default useCharactersStore;
```

Listing 18 – src/store/useCharactersStore.js

6.3 main.jsx (Pas de Provider !)

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import './styles/styles.css';

ReactDOM.createRoot(document.getElementById('root')).render(
    <React.StrictMode>
        <App />
    </React.StrictMode>
);
```

Listing 19 – src/main.jsx

6.4 Les Composants Zustand

```
import useCharactersStore from '../store/useCharactersStore';

function Header() {
    const likedIds = useCharactersStore(state => state.likedIds);

    return (
        <header className="header">
            <h1>Rick & Morty Characters (Zustand)</h1>
            <div className="likes-badge">
                        {likedIds.length} personnages likés
            </div>
        </header>
    );
}

export default Header;
```

Listing 20 – src/components/Header.jsx

```
import useCharactersStore from '../store/useCharactersStore';

function FilterBar() {
    const filter = useCharactersStore(state => state.filter);
    const setFilter = useCharactersStore(state => state.setFilter);

    const filters = ['all', 'alive', 'dead', 'unknown'];

    return (
        <div className="filter-bar">
            {filters.map(f => (
```

```jsx
                <button
                    key={f}
                    className={`filter-btn ${filter === f ? 'active' : ''}`}
                    onClick={() => setFilter(f)}
                >
                    {f.charAt(0).toUpperCase() + f.slice(1)}
                </button>
            ))}
        </div>
    );
}

export default FilterBar;
```

Listing 21 – src/components/FilterBar.jsx

```jsx
import useCharactersStore from '../store/useCharactersStore';

function CharacterCard({ character }) {
    const likedIds = useCharactersStore(state => state.likedIds);
    const toggleLike = useCharactersStore(state => state.toggleLike);
    const isLiked = likedIds.includes(character.id);

    return (
        <div className="character-card">
            <button
                className={`like-btn ${isLiked ? 'liked' : ''}`}
                onClick={() => toggleLike(character.id)}
            >
                {isLiked ? '      ' : '      '}
            </button>

            <img src={character.image} alt={character.name} />

            <div className="character-info">
                <div className="character-name">{character.name}</div>
                <div className={`character-status status-${character.status.toLowerCase()}`}>
                    {character.status} - {character.species}
                </div>
            </div>
        </div>
    );
}

export default CharacterCard;
```

Listing 22 – src/components/CharacterCard.jsx

```jsx
import { useEffect } from 'react';
import useCharactersStore from '../store/useCharactersStore';
import CharacterCard from './CharacterCard';

function CharacterGrid() {
    const getFilteredCharacters = useCharactersStore(state => state.getFilteredCharacters);
    const loading = useCharactersStore(state => state.loading);
    const fetchCharacters = useCharactersStore(state => state.fetchCharacters);

    useEffect(() => {
        fetchCharacters();
    }, [fetchCharacters]);

    if (loading) {
        return <div className="loading">Chargement...</div>;
    }

    const characters = getFilteredCharacters();

    return (
        <div className="character-grid">
            {characters.map(character => (
                <CharacterCard key={character.id} character={character} />
            ))}
        </div>
    );
}

export default CharacterGrid;
```

Listing 23 – src/components/CharacterGrid.jsx

```
import useCharactersStore from '../store/useCharactersStore';

function FavoritesSidebar() {
    const getLikedCharacters = useCharactersStore(state => state.getLikedCharacters);
    const toggleLike = useCharactersStore(state => state.toggleLike);
    const likedCharacters = getLikedCharacters();

    return (
        <div className="favorites-sidebar">
            <h2>Mes Favoris ({likedCharacters.length})</h2>

            {likedCharacters.length === 0 ? (
                <p style={{ color: '#95a5a6' }}>Aucun favori</p>
            ) : (
                likedCharacters.map(character => (
                    <div key={character.id} className="favorite-item">
                        <img src={character.image} alt={character.
    name} />
                                <span>{character.name}</span>
                                <button
                                      onClick={() => toggleLike(character.id)}
                                      style={{
                                           marginLeft: 'auto',
                                           background: 'none',
                                           border: 'none',
                                           cursor: 'pointer',
                                           fontSize: '20px'
                                      }}
                                >

                                </button>
                           </div>
                      ))
                 )}
            </div>
      );
}

export default FavoritesSidebar;
```

Listing 24 – src/components/FavoritesSidebar.jsx

7 Comparaison Finale

Critère	**Context**	**Redux**	**Zustand**
Lignes de code	80	100	60
Gestion API	useEffect	createAsyncThunk	async function
Provider requis			
Selectors	Fonctions	Intégrés	get()
Complexité	Moyenne	Haute	Faible

Observations :
- **Context :** Simple mais tous les consommateurs re-rendent
- **Redux :** Gestion API élégante avec createAsyncThunk
- **Zustand :** Le plus concis, async functions intégrées naturellement

8 Travail à Rendre

Projet Comparatif

Créer une application de films (API TMDb ou OMDb) avec les 3 solutions.

Fonctionnalités :
- **Charger des films populaires depuis l'API**
- **Système de favoris (étoile)**
- **Filtrer par genre**
- **Recherche par titre**
- **Afficher les favoris dans une section séparée**

Livrables :
- **3 dossiers séparés**
- **CSS commun**
- **README comparatif détaillé**
- **Captures d'écran**

Échéance : La veille de la prochaine séance à 23h59

9 Ressources

- Rick & Morty API : `https://rickandmortyapi.com/`
- React Context : `https://react.dev/reference/react/useContext`
- Redux Toolkit : `https://redux-toolkit.js.org/`
- Zustand : `https://zustand-demo.pmnd.rs/`