

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 | 3 |
| 2 Partie 1 : Introduction et Concepts (40 min) | 3 |
| 2.1 Le Problème : Prop Drilling | 3 |
| 2.2 Solution 1 : useContext | 4 |
| 2.2.1 Comment créer un Context ? | 4 |
| 2.3 Solution 2 : Redux Toolkit | 4 |
| 2.4 Solution 3 : Zustand | 5 |
| 3 Partie 2 : Le Projet - Galerie de Personnages (2h20) | 6 |
| 3.1 Structure du Projet | 6 |
| 3.2 Le Fichier CSS | 6 |
| 4 Version 1 : useContext (40 min) | 11 |
| 4.1 Créer le Context | 11 |
| 4.2 Wrapper dans main.jsx | 12 |
| 4.3 Les Composants | 12 |
| 5 Version 2 : Redux Toolkit (50 min) | 17 |
| 5.1 Installation | 17 |
| 5.2 Créer le Slice | 17 |
| 5.3 Configurer le Store | 18 |
| 5.4 Provider dans main.jsx | 18 |
| 5.5 Les Composants Redux | 19 |
| 6 Version 3 : Zustand (40 min) | 23 |
| 6.1 Installation | 23 |
| 6.2 Créer le Store | 23 |
| 6.3 main.jsx (Pas de Provider !) | 24 |
| 6.4 Les Composants Zustand | 24 |

| | |
|-----------------------------|-----------|
| 7 Comparaison Finale | 28 |
| 8 Travail à Rendre | 28 |
| 9 Ressources | 28 |

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.

```
1 // App doit passer username à tous les enfants
2 function App() {
3     const [username, setUsername] = useState("Alice");
4
5     return <Header username={username}>
6         <Navbar username={username}>
7             <UserMenu username={username} /> /* Utilise enfin !
8 */
9         </Navbar>
10    </Header>;
}
```

Solution : État global accessible partout directement.

2.2 Solution 1 : useContext

2.2.1 Comment créer un Context ?

Étape 1 : Créer le Context

```
1 import { createContext, useContext, useState } from 'react';
2
3 const UserContext = createContext();
4
5 export const UserProvider = ({ children }) => {
6     const [username, setUsername] = useState("Alice");
7
8     return (
9         <UserContext.Provider value={{ username, setUsername }}>
10            {children}
11        </UserContext.Provider>
12    );
13};
14
15 export const useUser = () => useContext(UserContext);
```

Étape 2 : Wrapper l'application

```
1 <UserProvider>
2   <App />
3 </UserProvider>
```

Étape 3 : Utiliser

```
1 function UserMenu() {
2     const { username } = useUser();
3     return <div>Bonjour {username} !</div>;
4 }
```

2.3 Solution 2 : Redux Toolkit

Créer un Slice

```
1 import { createSlice } from '@reduxjs/toolkit';
2
3 const userSlice = createSlice({
4     name: 'user',
5     initialState: { username: 'Alice' },
6     reducers: {
7         setUsername: (state, action) => {
8             state.username = action.payload;
9         }
10    }
11});
```

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

Utiliser

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

2.4 Solution 3 : Zustand

Créer le Store

```
1 import { create } from 'zustand';
2
3 const useUserStore = create((set) => ({
4     username: 'Alice',
5     setUsername: (name) => set({ username: name })
6 }));
7
8 export default useUserStore;
```

Utiliser (pas de Provider !)

```
1 const username = useUserStore(state => state.username);
2 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

```

1 src /
2   styles /
3     styles.css
4   components /
5     Header.jsx
6     CharacterGrid.jsx
7     CharacterCard.jsx
8     FavoritesSidebar.jsx
9     FilterBar.jsx
10    App.jsx

```

3.2 Le Fichier CSS

```

1 * {
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5 }
6
7 body {
8   font-family: 'Arial', sans-serif;
9   background-color: #1e1e1e;
10  color: white;
11 }
12
13 .header {
14   background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
15   padding: 20px;
16   text-align: center;
17   box-shadow: 0 2px 10px rgba(0,0,0,0.3);
18 }
19
20 .header h1 {

```

```
21     font-size: 32px;  
22     margin-bottom: 10px;  
23 }  
24  
25 .likes-badge {  
26     display: inline-block;  
27     background-color: #e74c3c;  
28     padding: 8px 16px;  
29     border-radius: 20px;  
30     font-weight: bold;  
31 }  
32  
33 .main-container {  
34     display: grid;  
35     grid-template-columns: 1fr 300px;  
36     gap: 20px;  
37     padding: 20px;  
38     max-width: 1400px;  
39     margin: 0 auto;  
40 }  
41  
42 .filter-bar {  
43     display: flex;  
44     gap: 10px;  
45     justify-content: center;  
46     margin-bottom: 20px;  
47 }  
48  
49 .filter-btn {  
50     padding: 10px 20px;  
51     border: none;  
52     border-radius: 5px;  
53     cursor: pointer;  
54     font-weight: bold;  
55     transition: all 0.3s;  
56 }  
57  
58 .filter-btn.active {  
59     background-color: #667eea;  
60     color: white;  
61 }  
62  
63 .filter-btn:not(.active) {  
64     background-color: #3a3a3a;  
65     color: #ccc;  
66 }  
67  
68 .character-grid {  
69     display: grid;  
70     grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));  
71     gap: 20px;
```

```
72 }
73
74 .character-card {
75   background-color: #2a2a2a;
76   border-radius: 10px;
77   overflow: hidden;
78   cursor: pointer;
79   transition: transform 0.3s;
80   position: relative;
81 }
82
83 .character-card:hover {
84   transform: translateY(-5px);
85 }
86
87 .character-card img {
88   width: 100%;
89   height: 200px;
90   object-fit: cover;
91 }
92
93 .character-info {
94   padding: 15px;
95 }
96
97 .character-name {
98   font-size: 18px;
99   font-weight: bold;
100  margin-bottom: 5px;
101 }
102
103 .character-status {
104   font-size: 14px;
105   color: #95a5a6;
106 }
107
108 .status-alive {
109   color: #27ae60;
110 }
111
112 .status-dead {
113   color: #e74c3c;
114 }
115
116 .like-btn {
117   position: absolute;
118   top: 10px;
119   right: 10px;
120   background-color: rgba(0,0,0,0.7);
121   border: none;
122   border-radius: 50%;
```

```
123     width: 40px;
124     height: 40px;
125     font-size: 20px;
126     cursor: pointer;
127     transition: transform 0.2s;
128 }
129
130 .like-btn:hover {
131     transform: scale(1.2);
132 }
133
134 .like-btn.liked {
135     background-color: #e74c3c;
136 }
137
138 .favorites-sidebar {
139     background-color: #2a2a2a;
140     border-radius: 10px;
141     padding: 20px;
142     height: fit-content;
143     position: sticky;
144     top: 20px;
145 }
146
147 .favorites-sidebar h2 {
148     margin-bottom: 15px;
149     color: #667eea;
150 }
151
152 .favorite-item {
153     display: flex;
154     align-items: center;
155     gap: 10px;
156     padding: 10px;
157     background-color: #3a3a3a;
158     border-radius: 5px;
159     margin-bottom: 10px;
160 }
161
162 .favorite-item img {
163     width: 40px;
164     height: 40px;
165     border-radius: 50%;
166     object-fit: cover;
167 }
168
169 .loading {
170     text-align: center;
171     padding: 50px;
172     font-size: 24px;
```

173 }

Listing 1 – src/styles/styles.css

4 Version 1 : useContext (40 min)

4.1 Créer le Context

```
1 import { createContext, useContext, useState, useEffect } from 'react';
2
3 const CharactersContext = createContext();
4
5 export const useCharacters = () => {
6   const context = useContext(CharactersContext);
7   if (!context) {
8     throw new Error('useCharacters must be used within
9   CharactersProvider');
10 }
11 return context;
12 };
13
14 export const CharactersProvider = ({ children }) => {
15   const [characters, setCharacters] = useState([]);
16   const [likedIds, setLikedIds] = useState([]);
17   const [filter, setFilter] = useState('all');
18   const [loading, setLoading] = useState(true);
19
20   // Charger les personnages
21   useEffect(() => {
22     fetch('https://rickandmortyapi.com/api/character')
23       .then(res => res.json())
24       .then(data => {
25         setCharacters(data.results);
26         setLoading(false);
27       });
28   }, []);
29
30   const toggleLike = (id) => {
31     setLikedIds(prev =>
32       prev.includes(id)
33         ? prev.filter(likedId => likedId !== id)
34         : [...prev, id]
35     );
36   };
37
38   const getLikedCharacters = () => {
39     return characters.filter(char => likedIds.includes(char.id));
40   };
41
42   const getFilteredCharacters = () => {
43     if (filter === 'all') return characters;
44     return characters.filter(char => char.status.toLowerCase()
45       === filter);
46   };
47 }
```

```

45 const value = {
46     characters,
47     likedIds,
48     filter,
49     loading,
50     toggleLike,
51     setFilter,
52     getLikedCharacters,
53     getFilteredCharacters
54 };
55
56 return (
57     <CharactersContext.Provider value={value}>
58         {children}
59     </CharactersContext.Provider>
60 );
61
62 };

```

Listing 2 – src/context/CharactersContext.jsx

4.2 Wrapper dans main.jsx

```

1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import App from './App';
4 import { CharactersProvider } from './context/CharactersContext';
5 import './styles/styles.css';
6
7 ReactDOM.createRoot(document.getElementById('root')).render(
8     <React.StrictMode>
9         <CharactersProvider>
10             <App />
11         </CharactersProvider>
12     </React.StrictMode>
13 );

```

Listing 3 – src/main.jsx

4.3 Les Composants

```

1 import { useCharacters } from './context/CharactersContext';
2
3 function Header() {
4     const { likedIds } = useCharacters();
5
6     return (
7         <header className="header">
8             <h1>Rick & Morty Characters (Context)</h1>
9             <div className="likes-badge">
10                 {likedIds.length} personnages likés

```

```

11         </div>
12     </header>
13   );
14 }
15
16 export default Header;

```

Listing 4 – src/components/Header.jsx

```

1 import { useCharacters } from './context/CharactersContext';
2
3 function FilterBar() {
4   const { filter, setFilter } = useCharacters();
5
6   const filters = [ 'all', 'alive', 'dead', 'unknown' ];
7
8   return (
9     <div className="filter-bar">
10       {filters.map(f => (
11         <button
12           key={f}
13           className={'filter-btn ${filter === f ? "active"' :
14             '}'}
15           onClick={() => setFilter(f)}
16         >
17           {f.charAt(0).toUpperCase() + f.slice(1)}
18         </button>
19       ))}
20     </div>
21   );
22 }
23
24 export default FilterBar;

```

Listing 5 – src/components/FilterBar.jsx

```

1 import { useCharacters } from './context/CharactersContext';
2
3 function CharacterCard({ character }) {
4   const { likedIds, toggleLike } = useCharacters();
5   const isLiked = likedIds.includes(character.id);
6
7   return (
8     <div className="character-card">
9       <button
10         className={'like-btn ${isLiked ? "liked" : "'}'}
11         onClick={() => toggleLike(character.id)}
12       >
13         {isLiked ? ' liked' : ' like'}
14       </button>
15
16       <img src={character.image} alt={character.name} />
17     </div>

```

```

18     <div className="character-info">
19         <div className="character-name">{character.name}</div>
20         <div className={'character-status status-' + character.status.toLowerCase()}>
21             {character.status} – {character.species}
22         </div>
23     </div>
24   </div>
25 );
26 }
27
28 export default CharacterCard;

```

Listing 6 – src/components/CharacterCard.jsx

```

1 import { useCharacters } from './context/CharactersContext';
2 import CharacterCard from './CharacterCard';
3
4 function CharacterGrid() {
5   const { getFilteredCharacters, loading } = useCharacters();
6
7   if (loading) {
8     return <div className="loading">Chargement...</div>;
9   }
10
11   const filteredCharacters = getFilteredCharacters();
12
13   return (
14     <div className="character-grid">
15       {filteredCharacters.map(character => (
16         <CharacterCard key={character.id} character={character} />
17       ))}
18     </div>
19   );
20 }
21
22 export default CharacterGrid;

```

Listing 7 – src/components/CharacterGrid.jsx

```

1 import { useCharacters } from './context/CharactersContext';
2
3 function FavoritesSidebar() {
4   const { getLikedCharacters, toggleLike } = useCharacters();
5   const likedCharacters = getLikedCharacters();
6
7   return (
8     <div className="favorites-sidebar">
9       <h2>Mes Favoris ({likedCharacters.length})</h2>
10      {likedCharacters.length === 0 ? (
11

```

```

12         <p style={{ color: '#95a5a6' }}>Aucun favori</p>
13     ) : (
14       likedCharacters.map(character => (
15         <div key={character.id} className="favorite-item">
16           <img src={character.image} alt={character.
17             name} />
18           <span>{character.name}</span>
19           <button
20             onClick={() => toggleLike(character.id)}
21             style={{
22               marginLeft: 'auto',
23               background: 'none',
24               border: 'none',
25               cursor: 'pointer',
26               fontSize: '20px'
27             }}
28           >
29           </button>
30         </div>
31       ))
32     )
33   </div>
34 );
35 }
36
37 export default FavoritesSidebar;

```

Listing 8 – src/components/FavoritesSidebar.jsx

```

1 import Header from './components/Header';
2 import FilterBar from './components/FilterBar';
3 import CharacterGrid from './components/CharacterGrid';
4 import FavoritesSidebar from './components/FavoritesSidebar';
5
6 function App() {
7   return (
8     <div>
9       <Header />
10      <div className="main-container">
11        <div>
12          <FilterBar />
13          <CharacterGrid />
14        </div>
15        <FavoritesSidebar />
16      </div>
17    </div>
18  );
19}

```

```
21 export default App;
```

Listing 9 – src/App.jsx

5 Version 2 : Redux Toolkit (50 min)

5.1 Installation

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

5.2 Créer le Slice

```
1 import { createSlice, createAsyncThunk } from '@reduxjs/toolkit';
2
3 // Thunk pour charger les personnages
4 export const fetchCharacters = createAsyncThunk(
5   'characters/fetchCharacters',
6   async () => {
7     const response = await fetch('https://rickandmortyapi.com/api/
8       /character');
9     const data = await response.json();
10    return data.results;
11  }
12);
13
14 const charactersSlice = createSlice({
15   name: 'characters',
16   initialState: {
17     characters: [],
18     likedIds: [],
19     filter: 'all',
20     loading: false
21   },
22   reducers: {
23     toggleLike: (state, action) => {
24       const id = action.payload;
25       if (state.likedIds.includes(id)) {
26         state.likedIds = state.likedIds.filter(likedId =>
27           likedId !== id);
28       } else {
29         state.likedIds.push(id);
30       }
31     },
32     setFilter: (state, action) => {
33       state.filter = action.payload;
34     }
35   },
36   extraReducers: (builder) => {
37     builder
38       .addCase(fetchCharacters.pending, (state) => {
39         state.loading = true;
40       })
41       .addCase(fetchCharacters.fulfilled, (state, action) => {
42         state.characters = action.payload;
43       })
44   }
45 );
```

```

41         state.loading = false;
42     });
43 }
44 });
45
46 export const { toggleLike, setFilter } = charactersSlice.actions;
47
48 // Selectors
49 export const selectCharacters = (state) => state.characters.
    characters;
50 export const selectLikedIds = (state) => state.characters.likedIds;
51 export const selectFilter = (state) => state.characters.filter;
52 export const selectLoading = (state) => state.characters.loading;
53
54 export const selectLikedCharacters = (state) => {
55     return state.characters.characters.filter(char =>
56         state.characters.likedIds.includes(char.id)
57     );
58 };
59
60 export const selectFilteredCharacters = (state) => {
61     const { characters, filter } = state.characters;
62     if (filter === 'all') return characters;
63     return characters.filter(char => char.status.toLowerCase() ===
64         filter);
65 };
66 export default charactersSlice.reducer;

```

Listing 10 – src/store/charactersSlice.js

5.3 Configurer le Store

```

1 import { configureStore } from '@reduxjs/toolkit';
2 import charactersReducer from './charactersSlice';
3
4 export const store = configureStore({
5     reducer: {
6         characters: charactersReducer
7     }
8 });

```

Listing 11 – src/store/store.js

5.4 Provider dans main.jsx

```

1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import { Provider } from 'react-redux';
4 import { store } from './store/store';
5 import App from './App';

```

```

6 import './styles/styles.css';
7
8 ReactDOM.createRoot(document.getElementById('root')).render(
9     <React.StrictMode>
10        <Provider store={store}>
11            <App />
12        </Provider>
13    </React.StrictMode>
14 );

```

Listing 12 – src/main.jsx

5.5 Les Composants Redux

```

1 import { useSelector } from 'react-redux';
2 import { selectLikedIds } from '../store/charactersSlice';
3
4 function Header() {
5     const likedIds = useSelector(selectLikedIds);
6
7     return (
8         <header className="header">
9             <h1>Rick & Morty Characters (Redux)</h1>
10            <div className="likes-badge">
11                {likedIds.length} personnages likés
12            </div>
13        </header>
14    );
15 }
16
17 export default Header;

```

Listing 13 – src/components/Header.jsx

```

1 import { useSelector, useDispatch } from 'react-redux';
2 import { selectFilter, setFilter } from '../store/charactersSlice';
3
4 function FilterBar() {
5     const filter = useSelector(selectFilter);
6     const dispatch = useDispatch();
7
8     const filters = ['all', 'alive', 'dead', 'unknown'];
9
10    return (
11        <div className="filter-bar">
12            {filters.map(f => (
13                <button
14                    key={f}
15                    className={'filter-btn ${filter === f ? 'active'
16 : ''}'}
17                    onClick={() => dispatch(setFilter(f))}
18                >

```

```

18             { f.charAt(0).toUpperCase() + f.slice(1) }
19         </button>
20     ))
21 </div>
22 );
23 }
24
25 export default FilterBar;

```

Listing 14 – src/components/FilterBar.jsx

```

1 import { useSelector, useDispatch } from 'react-redux';
2 import { selectLikedIds, toggleLike } from '../store/charactersSlice'
3 ;
4
5 function CharacterCard({ character }) {
6     const likedIds = useSelector(selectLikedIds);
7     const dispatch = useDispatch();
8     const isLiked = likedIds.includes(character.id);
9
10    return (
11        <div className="character-card">
12            <button
13                className={'like-btn ${isLiked ? 'liked' : ''}'}
14                onClick={() => dispatch(toggleLike(character.id))}>
15                {isLiked ? 'Unlike' : 'Like'}
16            </button>
17
18            <img src={character.image} alt={character.name} />
19
20            <div className="character-info">
21                <div className="character-name">{character.name}</div>
22            >
23                <div className={'character-status status-${
24                    character.status.toLowerCase()
25                }'}>
26                    {character.status} – {character.species}
27                </div>
28            </div>
29        );
30    }
31
32    export default CharacterCard;

```

Listing 15 – src/components/CharacterCard.jsx

```

1 import { useEffect } from 'react';
2 import { useSelector, useDispatch } from 'react-redux';
3 import {
4     selectFilteredCharacters,
5     selectLoading,
6     fetchCharacters

```

```
7 } from './store/charactersSlice';
8 import CharacterCard from './CharacterCard';
9
10 function CharacterGrid() {
11     const characters = useSelector(selectFilteredCharacters);
12     const loading = useSelector(selectLoading);
13     const dispatch = useDispatch();
14
15     useEffect(() => {
16         dispatch(fetchCharacters());
17     }, [dispatch]);
18
19     if (loading) {
20         return <div className="loading">Chargement...</div>;
21     }
22
23     return (
24         <div className="character-grid">
25             {characters.map(character => (
26                 <CharacterCard key={character.id} character={character}>
27                     ))}
28             </div>
29     );
30 }
31
32 export default CharacterGrid;
```

Listing 16 – src/components/CharacterGrid.jsx

```
1 import { useSelector, useDispatch } from 'react-redux';
2 import { selectLikedCharacters, toggleLike } from '../store/
    charactersSlice';
3
4 function FavoritesSidebar() {
5     const likedCharacters = useSelector(selectLikedCharacters);
6     const dispatch = useDispatch();
7
8     return (
9         <div className="favorites-sidebar">
10            <h2>Mes Favoris ({likedCharacters.length})</h2>
11
12            {likedCharacters.length === 0 ? (
13                <p style={{ color: '#95a5a6' }}>Aucun favori</p>
14            ) : (
15                likedCharacters.map(character => (
16                    <div key={character.id} className="favorite-item">
17
18                        <img src={character.image} alt={character.
    name}>
19
20                        <span>{character.name}</span>
21                        <button>
```

```
20         onClick={() => dispatch(toggleLike(
21             character.id))} style={{
22                 marginLeft: 'auto',
23                 background: 'none',
24                 border: 'none',
25                 cursor: 'pointer',
26                 fontSize: '20px'
27             }}>
28
29         </button>
30     </div>
31   )) )
32 )
33 </div>
34 );
35
36 }
37
38 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

```
1 npm install zustand
```

6.2 Créer le Store

```
1 import { create } from 'zustand';
2
3 const useCharactersStore = create((set, get) => ({
4     characters: [] ,
5     likedIds: [] ,
6     filter: 'all' ,
7     loading: true ,
8
9     // Charger les personnages
10    fetchCharacters: async () => {
11        set({ loading: true });
12        const response = await fetch('https://rickandmortyapi.com/api
13        /character');
14        const data = await response.json();
15        set({ characters: data.results , loading: false });
16    },
17
18    // Toggle like
19    toggleLike: (id) => set((state) => {
20        likedIds: state.likedIds.includes(id)
21            ? state.likedIds.filter(likedId => likedId !== id)
22            : [...state.likedIds , id]
23    )),
24
25    // Changer le filtre
26    setFilter: (filter) => set({ filter }) ,
27
28    // Selectors
29    getLikedCharacters: () => {
30        const { characters , likedIds } = get();
31        return characters.filter(char => likedIds.includes(char.id));
32    },
33
34    getFilteredCharacters: () => {
35        const { characters , filter } = get();
36        if (filter === 'all') return characters;
37        return characters.filter(char => char.status.toLowerCase()
38        === filter);
39    }
}));
```

```
40 export default useCharactersStore;
```

Listing 18 – src/store/useCharactersStore.js

6.3 main.jsx (Pas de Provider !)

```
1 import React from 'react';
2 import ReactDOM from 'react-dom/client';
3 import App from './App';
4 import './styles/styles.css';
5
6 ReactDOM.createRoot(document.getElementById('root')).render(
7   <React.StrictMode>
8     <App />
9   </React.StrictMode>
10 );
```

Listing 19 – src/main.jsx

6.4 Les Composants Zustand

```
1 import useCharactersStore from '../store/useCharactersStore';
2
3 function Header() {
4   const likedIds = useCharactersStore(state => state.likedIds);
5
6   return (
7     <header className="header">
8       <h1>Rick & Morty Characters (Zustand)</h1>
9       <div className="likes-badge">
10         {likedIds.length} personnages likés
11       </div>
12     </header>
13   );
14 }
15
16 export default Header;
```

Listing 20 – src/components/Header.jsx

```
1 import useCharactersStore from '../store/useCharactersStore';
2
3 function FilterBar() {
4   const filter = useCharactersStore(state => state.filter);
5   const setFilter = useCharactersStore(state => state.setFilter);
6
7   const filters = [ 'all', 'alive', 'dead', 'unknown' ];
8
9   return (
10     <div className="filter-bar">
11       {filters.map(f => (
```

```

12         <button
13             key={f}
14             className={'filter-btn ${filter === f ? 'active'
15 : ''}`}
16             onClick={() => setFilter(f)}
17             >
18                 {f.charAt(0).toUpperCase() + f.slice(1)}
19             </button>
20         ))}
21     </div>
22 );
23
24 export default FilterBar;

```

Listing 21 – src/components/FilterBar.jsx

```

1 import useCharactersStore from '../store/useCharactersStore';
2
3 function CharacterCard({ character }) {
4     const likedIds = useCharactersStore(state => state.likedIds);
5     const toggleLike = useCharactersStore(state => state.toggleLike);
6     const isLiked = likedIds.includes(character.id);
7
8     return (
9         <div className="character-card">
10            <button
11                className={'like-btn ${isLiked ? 'liked' : ''}'}
12                onClick={() => toggleLike(character.id)}
13                >
14                  {isLiked ? ' liked' : ' like'}
15            </button>
16
17            <img src={character.image} alt={character.name} />
18
19            <div className="character-info">
20                <div className="character-name">{character.name}</div>
21            >
22                <div className={'character-status status-' + character.status.toLowerCase()}>
23                    {character.status} - {character.species}
24                </div>
25            </div>
26        </div>
27    );
28
29 export default CharacterCard;

```

Listing 22 – src/components/CharacterCard.jsx

```

1 import { useEffect } from 'react';
2 import useCharactersStore from '../store/useCharactersStore';

```

```
3 import CharacterCard from './CharacterCard';
4
5 function CharacterGrid() {
6     const getFilteredCharacters = useCharactersStore(state => state.
7         getFilteredCharacters);
8     const loading = useCharactersStore(state => state.loading);
9     const fetchCharacters = useCharactersStore(state => state.
10        fetchCharacters);
11
12     useEffect(() => {
13         fetchCharacters();
14     }, [fetchCharacters]);
15
16     if (loading) {
17         return <div className="loading">Chargement...</div>;
18     }
19
20     const characters = getFilteredCharacters();
21
22     return (
23         <div className="character-grid">
24             {characters.map(character => (
25                 <CharacterCard key={character.id} character={
26                     character} />
27             )));
28         </div>
29     );
30 }
31
32 export default CharacterGrid;
```

Listing 23 – src/components/CharacterGrid.jsx

```
1 import useCharactersStore from './store/useCharactersStore';
2
3 function FavoritesSidebar() {
4     const getLikedCharacters = useCharactersStore(state => state.
5         getLikedCharacters);
6     const toggleLike = useCharactersStore(state => state.toggleLike);
7     const likedCharacters = getLikedCharacters();
8
9     return (
10         <div className="favorites-sidebar">
11             <h2>Mes Favoris ({ likedCharacters.length })</h2>
12
13             { likedCharacters.length === 0 ? (
14                 <p style={{ color: '#95a5a6' }}>Aucun favori</p>
15             ) : (
16                 likedCharacters.map( character => (
17                     <div key={character.id} className="favorite-item">
18                         <img src={character.image} alt={character .
```

```
18     name} />
19         <span>{character.name}</span>
20         <button
21             onClick={() => toggleLike(character.id)}
22             style={{
23                 marginLeft: 'auto',
24                 background: 'none',
25                 border: 'none',
26                 cursor: 'pointer',
27                 fontSize: '20px'
28             }}>
29         </button>
30     </div>
31   )) )
32 )
33 </div>
34 );
35 );
36 }
37
38 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/>