useReducer

useReducer

Alternative à `useState` pour des états complexes avec plusieurs sous-valeurs ou des transitions d'état interdépendantes. Inspiré du pattern Redux.

Syntaxe

```
const [state, dispatch] = useReducer(reducer, initialState);
```

Exemple : gestion d'un formulaire

```
const initialState = { nom: "", email: "", loading: false, error: null };

function reducer(state, action) {
  switch (action.type) {
    case 'SET_FIELD':
      return { ...state, [action.field]: action.value };
    case 'SUBMIT_START':
      return { ...state, loading: true, error: null };
    case 'SUBMIT_SUCCESS':
      return { ...initialState };
    case 'SUBMIT_ERROR':
      return { ...state, loading: false, error: action.error };
    default:
      return state;
  }
}

function Formulaire() {
  const [state, dispatch] = useReducer(reducer, initialState);

  const handleChange = (e) => dispatch({
```

```
    type: 'SET_FIELD', field: e.target.name, value: e.target.value
  });

  const handleSubmit = async (e) => {
    e.preventDefault();
    dispatch({ type: 'SUBMIT_START' });
    try {
      await submitForm(state);
      dispatch({ type: 'SUBMIT_SUCCESS' });
    } catch (err) {
      dispatch({ type: 'SUBMIT_ERROR', error: err.message });
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input name="nom" value={state.nom} onChange={handleChange} />
      <input name="email" value={state.email} onChange={handleChange} />
      {state.error && <p>{state.error}</p>}
      <button disabled={state.loading}>Envoyer</button>
    </form>
  );
}
```

useReducer vs useState

useState	useReducer
État simple (string, number, bool)	État complexe (objet avec plusieurs champs)
Transitions simples	Transitions liées ou conditionnelles
Pas de logique dans le setter	Logique centralisée dans le reducer

Revision #1
Created 29 March 2026 12:57:36 by Hugo
Updated 29 March 2026 12:57:36 by Hugo