**🧩 Understanding useReducer: A Beginner's Guide**

**🤔 What is useReducer?**

useReducer is a React Hook that helps manage complex state logic in your components. Think of it like useState, but for more complicated state updates.

**📋 Basic Structure**

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

* **state**: Your current state (like in useState)
* **dispatch**: A function to update state (like setState)
* **reducer**: A function that specifies how the state gets updated
* **initialState**: Starting value of your state

**🔢 Simple Counter Example**

**1️⃣ Basic Counter**

function Counter() {

// Simple reducer function

function reducer(state, action) {

if (action === 'increment') {

return state + 1;

}

if (action === 'decrement') {

return state - 1;

}

return state;

}

// Initialize useReducer

const [count, dispatch] = useReducer(reducer, 0);

return (

<div>

<h1>Count: {count}</h1>

<button onClick={() => dispatch('increment')}>Add</button>

<button onClick={() => dispatch('decrement')}>Subtract</button>

</div>

);

}

**2️⃣ Counter with Action Objects**

function Counter() {

function reducer(state, action) {

switch (action.type) {

case 'INCREMENT':

return state + 1;

case 'DECREMENT':

return state - 1;

default:

return state;

}

}

const [count, dispatch] = useReducer(reducer, 0);

return (

<div>

<h1>Count: {count}</h1>

<button onClick={() => dispatch({ type: 'INCREMENT' })}>Add</button>

<button onClick={() => dispatch({ type: 'DECREMENT' })}>Subtract</button>

</div>

);

}

**📊 Managing Multiple Values**

**📝 Todo List Example**

function TodoList() {

// Initial state

const initialState = {

todos: [],

inputText: ''

};

// Reducer function

function reducer(state, action) {

switch (action.type) {

case 'ADD\_TODO':

return {

...state,

todos: [...state.todos, action.text],

inputText: '' // Clear input after adding

};

case 'UPDATE\_INPUT':

return {

...state,

inputText: action.text

};

default:

return state;

}

}

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

function handleSubmit(e) {

e.preventDefault();

if (state.inputText.trim()) {

dispatch({ type: 'ADD\_TODO', text: state.inputText });

}

}

return (

<div>

<form onSubmit={handleSubmit}>

<input

value={state.inputText}

onChange={(e) =>

dispatch({

type: 'UPDATE\_INPUT',

text: e.target.value

})

}

/>

<button type="submit">Add Todo</button>

</form>

<ul>

{state.todos.map((todo, index) => (

<li key={index}>{todo}</li>

))}

</ul>

</div>

);

}

**🤔 When to Use useReducer?**

Use useReducer when:

* You have complex state logic
* Different actions modify state in different ways
* State updates depend on multiple values
* You need to update multiple state values together

**🔄 Simple vs Complex Example**

**🔹 Simple (Use useState)**

function SimpleCounter() {

const [count, setCount] = useState(0);

return (

<button onClick={() => setCount(count + 1)}>

Count: {count}

</button>

);

}

**🔹 Complex (Use useReducer)**

function ShoppingCart() {

const initialState = {

items: [],

total: 0,

itemCount: 0

};

function reducer(state, action) {

switch (action.type) {

case 'ADD\_ITEM':

return {

items: [...state.items, action.item],

total: state.total + action.item.price,

itemCount: state.itemCount + 1

};

case 'REMOVE\_ITEM':

const item = state.items[action.index];

return {

items: state.items.filter((\_, i) => i !== action.index),

total: state.total - item.price,

itemCount: state.itemCount - 1

};

default:

return state;

}

}

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

return (

<div>

<h2>Shopping Cart ({state.itemCount} items)</h2>

<h3>Total: ${state.total}</h3>

<button onClick={() =>

dispatch({

type: 'ADD\_ITEM',

item: { name: 'Product', price: 10 }

})

}>

Add Item

</button>

{/\* Display items \*/}

</div>

);

}

**💡 Remember:**

* Reducers should be pure functions
* Always return a new state object
* Never modify the existing state
* Use descriptive action names
* Start with useState, move to useReducer when needed

🚀 Happy coding!