# Re-render

## Avoid re-render components belong to unrelated state changes

If you have a large list (like a million objects) in React state, and you want to prevent re-rendering that list when unrelated state (like count) changes, the best practices boil down to memoization, component isolation, and virtualization.

import React, { useState, useMemo } from 'react';

import { FixedSizeList as List } from 'react-window';

// Generate 1,000,000 items once

const generateItems = () => {

return Array.from({ length: 1\_000\_000 }, (\_, i) => ({

id: i,

name: `Item ${i}`,

active: i % 2 === 0 // only even-numbered items are "active"

}));

};

const VirtualizedList = React.memo(({ items }) => {

console.log('VirtualizedList rendered');

return (

<List

height={500}

itemCount={items.length}

itemSize={35}

width="100%"

>

{({ index, style }) => (

<div style={style}>{items[index].name}</div>

)}

</List>

);

});

function App() {

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

const [showActiveOnly, setShowActiveOnly] = useState(false);

const items = useMemo(() => generateItems(), []); // stable list

const filteredItems = useMemo(() => {

console.log('Filtering items...');

return showActiveOnly ? items.filter(item => item.active) : items;

}, [items, showActiveOnly]);

return (

<div>

<h1>Large List with useMemo + Virtualization</h1>

<button onClick={() => setCount(c => c + 1)}>Increment Count ({count})</button>

<button onClick={() => setShowActiveOnly(s => !s)}>

Toggle Active Only ({showActiveOnly ? 'On' : 'Off'})

</button>

<VirtualizedList items={filteredItems} />

</div>

);

}

export default App;

## If I use React.memo for child component, do I need to use useMemo in parent?

React.memo does a shallow prop comparison. So:

* Primitive props (string, number, boolean) are safe. No useMemo needed.
* Object/array/function props get a new reference on each render unless you memoize them.

→ That causes React.memo to fail its comparison, and the child will re-render unnecessarily.