**Issues and Anti-Patterns**

1. Incorrect Filtering Logic:

The filtering logic within the useMemo hook is incorrect. It filters balances based on the condition balance.amount <= 0, which seems to be intended to exclude those balances, but the return value of the filtering function should be the opposite (i.e., balance.amount > 0).

1. Inefficient Use of useMemo:

The useMemo hook is used to memoize the sorting of balances, but it also relies on the prices dependency, which may not be necessary for sorting. This can cause unnecessary recalculations.

1. Inefficient Sorting Logic:

The sorting function uses a switch statement to determine the priority, but this could be refactored into a mapping object to improve readability and potentially performance.

1. Unused Dependencies:

The prices dependency in useMemo is unused within the function, which could cause unnecessary recalculations of the sorted balances when prices change.

1. Incorrect Type for rows Mapping:

The rows mapping uses WalletBalance instead of FormattedWalletBalance, causing potential type issues.

1. Redundant Mapping of formattedBalances:

The formattedBalances mapping is redundant if the formatted string is only used within the rendering. This can be integrated into the rendering logic directly.

**Refactored Code**

import React, { useMemo } from 'react';

interface WalletBalance {

currency: string;

amount: number;

blockchain: string;

}

interface FormattedWalletBalance extends WalletBalance {

formatted: string;

}

interface Props extends BoxProps {}

const WalletPage: React.FC<Props> = (props: Props) => {

const { children, ...rest } = props;

const balances = useWalletBalances();

const prices = usePrices();

const blockchainPriority: { [key: string]: number } = {

Osmosis: 100,

Ethereum: 50,

Arbitrum: 30,

Zilliqa: 20,

Neo: 20,

};

const getPriority = (blockchain: string): number => {

return blockchainPriority[blockchain] ?? -99;

};

const sortedBalances = useMemo(() => {

return balances

.filter((balance: WalletBalance) => balance.amount > 0)

.sort((lhs: WalletBalance, rhs: WalletBalance) => {

const leftPriority = getPriority(lhs.blockchain);

const rightPriority = getPriority(rhs.blockchain);

return rightPriority - leftPriority;

});

}, [balances]);

const rows = sortedBalances.map((balance: WalletBalance, index: number) => {

const usdValue = prices[balance.currency] \* balance.amount;

const formattedAmount = balance.amount.toFixed();

return (

<WalletRow

className={classes.row}

key={index}

amount={balance.amount}

usdValue={usdValue}

formattedAmount={formattedAmount}

/>

);

});

return <div {...rest}>{rows}</div>;

};

export default WalletPage;

**Explanation of Improvements**

1. Fixed Filtering Logic:

Changed the filtering logic to correctly filter out balances with amounts less than or equal to zero.

1. Optimized useMemo Dependencies:

Removed prices from the dependency array of useMemo since it was not used in the sorting logic.

1. Simplified Priority Mapping:

Replaced the switch statement with a blockchainPriority object for more efficient priority lookup.

1. Combined Formatting in Mapping:

Integrated the formatting of the amount directly into the rows mapping, removing the need for a separate formattedBalances mapping.

1. Corrected Typing for rows Mapping:

Ensured the correct type is used for the rows mapping, avoiding potential type issues.