I see errors:

+ ToFixed(n) => missing input parameter value.

```
  }, [balances, prices]);

  const formattedBalances = sortedBalances.map((balance: WalletBalance) => {
    return {
      ...balance,
      formatted: balance.amount.toFixed()
    }
  })
```

+ lhsPriority => not declared

+ balancePriority => declared but not used

```
28              return 20
29            case 'Neo':
30              return 20
31            default:
32              return -99
33          }
34        }
35
36      const sortedBalances = useMemo(() => {
37        return balances.filter((balance: WalletBalance) => {
38            const balancePriority = getPriority(balance.blockchain);
39            if (lhsPriority > -99) {
40              if (balance.amount <= 0) {
41                return true;
42              }
43            }
44            return false
45          }).sort((lhs: WalletBalance, rhs: WalletBalance) => {
46              const leftPriority = getPriority(lhs.blockchain);
47            const rightPriority = getPriority(rhs.blockchain);
48            if (leftPriority > rightPriority) {
49              return -1;
```

+ formattedBalances => declared but not used

```
51              return 1;
52            }
53        });
54      }, [balances, prices]);
55
56      const formattedBalances = sortedBalances.map((balance: WalletBalance) => {
57        return {
58          ...balance,
59          formatted: balance.amount.toFixed()
60        }
61      })
62
63      const rows = sortedBalances.map((balance: FormattedWalletBalance, index: number) => {
64        const usdValue = prices[balance.currency] * balance.amount;
65        return (
66          <WalletRow
67            className={classes.row}
68            key={index}
```

+ For 2 nested if statements, we can use the && symbol.

```
30                return 20
31              default:
32                return -99
33            }
34          }
35
36        const sortedBalances = useMemo(() => {
37          return balances.filter((balance: WalletBalance) => {
38                const balancePriority = getPriority(balance.blockchain);
39                if (lhsPriority > -99) {
40                  if (balance.amount <= 0) {
41                    return true;
42                  }
43                }
44                return false
45              }).sort((lhs: WalletBalance, rhs: WalletBalance) => {
46                  const leftPriority = getPriority(lhs.blockchain);
47                const rightPriority = getPriority(rhs.blockchain);
48                if (leftPriority > rightPriority) {
```

+ sortedBalances => repeat through map 2 times

```
54      }, [balances, prices]);
55
9+ 56   const formattedBalances = sortedBalances.map((balance: WalletBalance) => {
57        return {
58          ...balance,
59          formatted: balance.amount.toFixed()
60        }
61      })
62
63      const rows = sortedBalances.map((balance: FormattedWalletBalance, index: number) => {
64        const usdValue = prices[balance.currency] * balance.amount;
65        return (
66          <WalletRow
67            className={classes.row}
68            key={index}
69            amount={balance.amount}
70            usdValue={usdValue}
71            formattedAmount={balance.formatted}
72          />
73        )
```