

```
import React, { useState } from "react"; //Komponente für die Anzeige der MyPantryList
const MyPantryList = ({ foodList, onDelete }) => { //der State für ausgewählte/ selected
  Elemente und Sortierung const [selectedItems, setSelectedItems] = useState([]); const
  [sortConfig, setSortConfig] = useState({ key: null, direction: 'asc' }); //hiermit wird die
  Checkbox-Änderung bearbeitet const handleCheckboxChange = (itemId) => { if
  (selectedItems.includes(itemId)) { setSelectedItems(selectedItems.filter((id) => id !==
  itemId)); } else { setSelectedItems([...selectedItems, itemId]); } }; //löscht ausgewählte
  Elemente const handleDelete = () => { onDelete(selectedItems); setSelectedItems([]); };
  //Funktion, die prüft, ob Lebensmittel über dem Verfallsdatum liegt const isExpired =
  (bestBeforeDate) => { const today = new Date(); const expirationDate = new
  Date(bestBeforeDate); return today > expirationDate; }; //Funktion, die die Sortierung
  anfordert const requestSort = (key) => { let direction = 'asc'; if (sortConfig.key === key &&
  sortConfig.direction === 'asc') { direction = 'desc'; } setSortConfig({ key, direction }); };
  //Funktion, die die Liste sortiert const sortedList = () => { let list = [...foodList]; if
  (sortConfig.key !== null) { list.sort((a, b) => { if (a[sortConfig.key] < b[sortConfig.key]) {
  return sortConfig.direction === 'asc' ? -1 : 1; } if (a[sortConfig.key] > b[sortConfig.key]) {
  return sortConfig.direction === 'asc' ? 1 : -1; } return 0; }); } return list; }; // Liste der
  Spaltenüberschriften der Tabelle/ Liste const columnHeaders = ["id", "name", "amount",
  "unit", "categorie", "bestBefore", "actions"]; //Rückgabewert der Komponente (gibt die Liste/
  Tabelle mit Sortierfunktion aus inklusive delete Button) return ( <div className="card-
  container"> <table> <thead> <tr> {/*Spaltenüberschriften mit Sortierfunktion*/}
  {columnHeaders.map((header, index) => ( <th key={index} onClick={() =>
  requestSort(header)}> {header} {sortConfig.key === header && (
  <span>{sortConfig.direction === 'asc' ? '▲' : '▼'}</span> )} </th> ))) </tr> </thead>
  <tbody> {/*Liste der eingegebenen Lebensmittel*/} {sortedList().map((foodItem) => ( <tr
  key={foodItem.id}> <td>{foodItem.id}</td> <td>{foodItem.name}</td>
  <td>{foodItem.amount}</td> <td>{foodItem.unit}</td> <td>{foodItem.categorie}</td> <td
  style={{ color: isExpired(foodItem.bestBefore) ? 'red' : 'black' }}> {foodItem.bestBefore}
  {isExpired(foodItem.bestBefore) && <span> - abgelaufen</span>} </td> <td> {/*
  Checkbox, zur Auswahl einer Zeile/ eines eingegebenen Elements*/} <input
  type="checkbox" onChange={() => handleCheckboxChange(foodItem.id)}
  checked={selectedItems.includes(foodItem.id)} /> </td> </tr> ))) </tbody> </table> <div
  className="input-container3"> <div className="buttons"> {/* Button zum löschen
  ausgewählter Elemente*/} <button className="deleteb" onClick={handleDelete}>Delete
  Selected</button> </div> </div> </div> ); }; export default MyPantryList; wie stelle ich ein,
  dass die Zahlen richtig sortiert werden? Die 10 sollte eine größere Zahl als 2 sein, wird aber
  davor gelistet beim Sortieren, wegen der führenden 1. Das Programm soll automatisch anhand
  der Zeichenanzahl erkennen, ob die Zahl im einer, zehner, hunderter etc. Bereich liegt und
  dementsprechend die Zahl einsortieren.
```

ChatGPT

Um sicherzustellen, dass die Zahlen in Ihrer Tabelle korrekt sortiert werden, müssen Sie die Sortierung unter Berücksichtigung der numerischen Werte vornehmen, anstatt sie als Zeichenketten zu vergleichen. Dafür können Sie die Funktion `localeCompare` verwenden und den numerischen Vergleich implementieren.

Hier ist eine mögliche Überarbeitung Ihrer `sortedList` Funktion:

```
javascript
const sortedList = () => {
```

```

let list = [...foodList];
if (sortConfig.key !== null) {
  list.sort((a, b) => {
    if (sortConfig.key !== 'amount' && sortConfig.key !== 'id') {
      // Wenn es nicht um 'amount' oder 'id' geht, führe den normalen
Vergleich durch
      if (a[sortConfig.key] < b[sortConfig.key]) {
        return sortConfig.direction === 'asc' ? -1 : 1;
      }
      if (a[sortConfig.key] > b[sortConfig.key]) {
        return sortConfig.direction === 'asc' ? 1 : -1;
      }
    } else {
      // Für 'amount' und 'id', führe den numerischen Vergleich durch
      const numA = parseFloat(a[sortConfig.key]);
      const numB = parseFloat(b[sortConfig.key]);

      if (numA < numB) {
        return sortConfig.direction === 'asc' ? -1 : 1;
      }
      if (numA > numB) {
        return sortConfig.direction === 'asc' ? 1 : -1;
      }
    }
    return 0;
  });
}
return list;
};

```

Diese Überarbeitung berücksichtigt die spezielle Behandlung der Spalten 'amount' und 'id' und führt einen numerischen Vergleich durch, indem `parseFloat` verwendet wird. Dadurch sollten Zahlen korrekt sortiert werden, unabhängig von ihrer Länge oder führenden Nullen.

JA

geht das auch noch für das Datum? zuerst muss anhand des Jahres sortiert werden, zweitrangig ist danach Monat und abschließend drittrangig ist der Tag.

ChatGPT

Ja, Sie können eine ähnliche Logik auch für die Sortierung von Datumsangaben implementieren. Hier ist eine mögliche Überarbeitung Ihrer `sortedList` Funktion:

```

javascript
const sortedList = () => {
  let list = [...foodList];
  if (sortConfig.key !== null) {
    list.sort((a, b) => {
      if (sortConfig.key === 'bestBefore') {
        // Wenn es sich um die Spalte 'bestBefore' handelt, führe einen
Datumssortierung durch
        const dateA = new Date(a.bestBefore);
        const dateB = new Date(b.bestBefore);

        if (dateA < dateB) {
          return sortConfig.direction === 'asc' ? -1 : 1;
        }
        if (dateA > dateB) {
          return sortConfig.direction === 'asc' ? 1 : -1;
        }
      }
    });
  }
  return list;
};

```

```

    } else if (sortConfig.key !== 'amount' && sortConfig.key !== 'id') {
      // Wenn es nicht um 'amount', 'id' oder 'bestBefore' geht, führe
den normalen Vergleich durch
      if (a[sortConfig.key] < b[sortConfig.key]) {
        return sortConfig.direction === 'asc' ? -1 : 1;
      }
      if (a[sortConfig.key] > b[sortConfig.key]) {
        return sortConfig.direction === 'asc' ? 1 : -1;
      }
    } else {
      // Für 'amount' und 'id', führe den numerischen Vergleich durch
      const numA = parseFloat(a[sortConfig.key]);
      const numB = parseFloat(b[sortConfig.key]);

      if (numA < numB) {
        return sortConfig.direction === 'asc' ? -1 : 1;
      }
      if (numA > numB) {
        return sortConfig.direction === 'asc' ? 1 : -1;
      }
    }
  }
  return 0;
});
}
return list;
};

```

Hier wird der Vergleich für die 'bestBefore'-Spalte durchgeführt, indem `new Date()` verwendet wird, um die Daten in JavaScript-Datumobjekte zu konvertieren. Dadurch erfolgt die Sortierung zunächst nach dem Jahr, dann nach dem Monat und schließlich nach dem Tag.