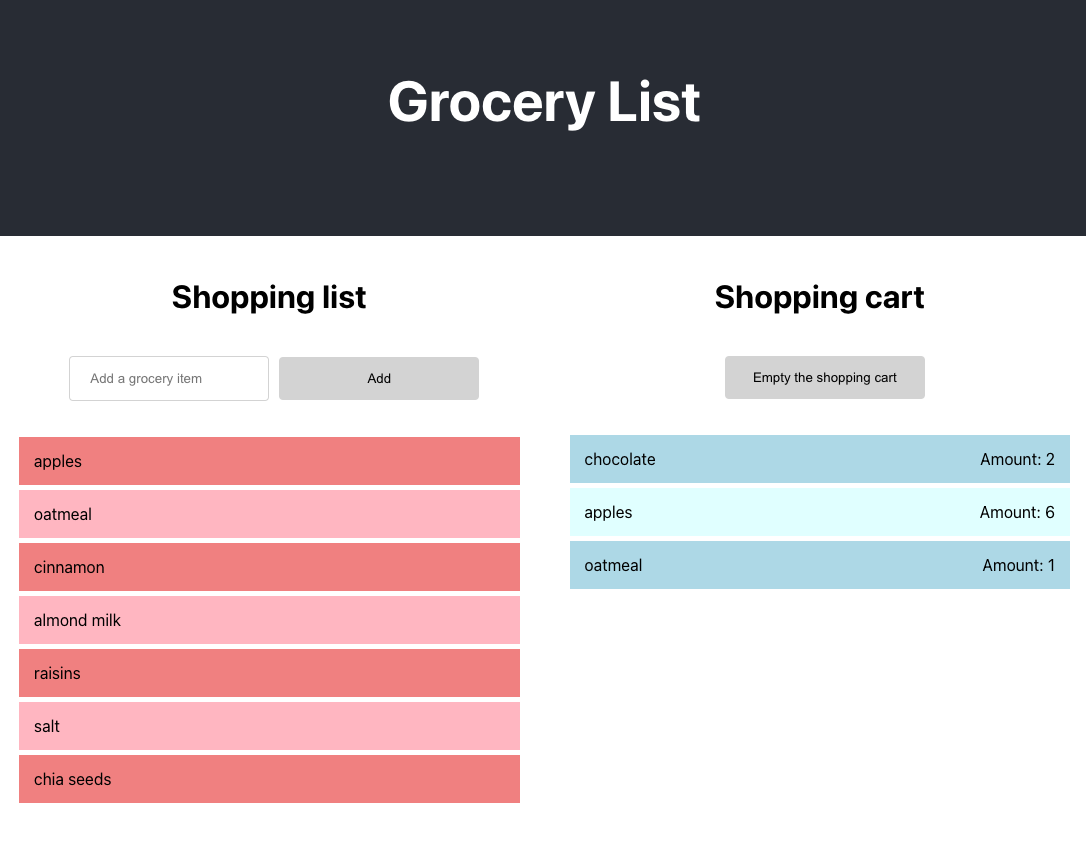
# REACT | REACT SHOPPING LIST AND SHOPPING CART

## #Exercise: React Shopping List and Shopping Cart

**Introduction**

In this exercise, you will create a shopping cart! You can add items from your shopping list. The exercise is divided into different parts, which become more and more difficult and less explicit.

**The end result:**



**Part 1: Make a shopping list**

1. Create a React App with $npx create-react-app [PROJECT NAME], give the app the project name, for example "my-groceries-app". When in doubt, follow the official documentation carefully: <https://reactjs.org/docs/create-a-new-react-app.html>
2. Make sure your app is always running on your [http://localhost:3000](http://localhost:3000/) (the default) as long as you are working with it. Your browser will refresh whenever you make a change in the project. Open a new terminal window for your git commands. – Several files have appeared. What was that again with node\_modules and package-lock.json? => Review the lessons on those subjects.
3. In the src folder, create a new folder called "components".
4. In the components folder, create a JavaScript file called List.js, it contains a class-based component List.
5. List renders a <ul> with <li>s (fill in some nice groceries you normally buy)
6. **Export** your component.
7. **Import** your new component into App.js.
8. Render your List in your App.js by means of an **import statement** and **JSX** =>\*\* \*\* <List/>

**Part 2: Make the items a separate component**

1. Create a new file ListItem.js with a **functional** component ListItem with a prop: **title.**
2. Import your ListItem component into your List.
3. Replace the <li> elements in List with your new imported component ListItem**.**
4. "Pass down props": give each <ListItem/> an appropriate prop for **title.**
5. For example: 'Apples', 'Carton of milk'.

**Part 3: Render the items dynamically**

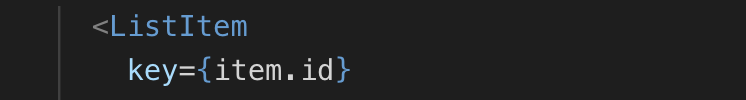
1. Make sure List component gets a state.
2. Initialize the state of List in the constructor => https://reactjs.org/docs/state-and-lifecycle.html.
   * example:
   * constructor(props) {
   * **super**(props);
   * **this**.state = {groceryItems: []};

}

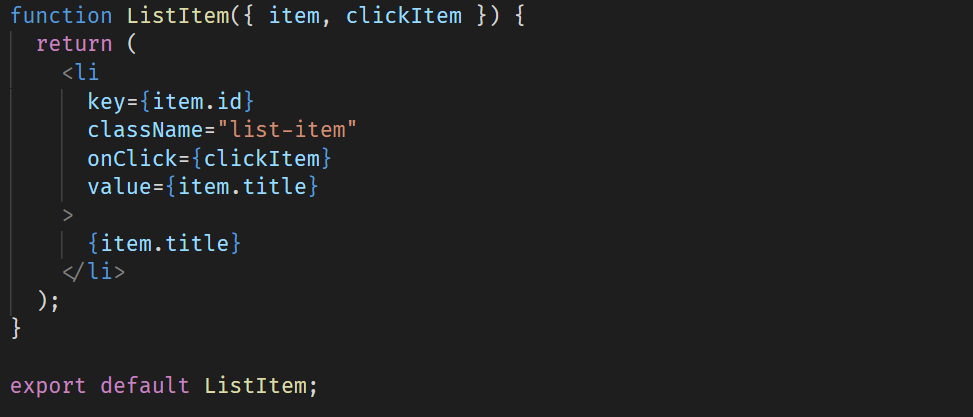
1. Initial state has an **array** called groceryItems **(note! Plural).**
2. Move the grocery titles as in 2.4 to your state and make it an array of objects, e.g:
3. groceryItems: [
4. { id: 1, title: "Appels" },

{ id: 2, title: "Carton of milk" }

1. Use an array method for each object in the array groceryItems to render a <ListItem/> component. Hint: array.map() is a good way to go. Each ListItem should again render the title.
2. **Give each component the correct id and title from the grocery as prop**.
3. You will notice that React now contains errors about unique "key" props. React does not know how to identify these children, so we have to help out. To fix the error, give each ListItem a key, specifically the id of the grocery item.

**

1. It is good practice to give a key and a value with a list item. This will be useful in the future. To do this, change the prop passed to the ListItem from just the title, to an entire grocery item, which you can name generic item.
2. Now, put the item.title in the list item, but in addition, add key=item.id, to the list item. While we are editing the ListItem, let's also add a classname and an onClick value. We can pass the onClick value via the props later, and we will call it clickItem. By now, you ListItem should looks something like this:



This functional component uses destructuring for its props, which allows you to specify the props directly. If you do not use destructuring, make sure to use use the props directly instead, e.g. props.item.title.

1. To make sure the onClick works, we can pass a function to the props of our ListItems. Open your List.js file and add a function onItemClick to your class component. This function accepts an item as its argument and logs the id and title to the console. Next, pass the function to the ListItem via the clickIem prop that you just created. After saving, open your browser devtools and make sure the onClick is triggered properly. The List component should now look like this:



For the following parts, you will have to use your React skills to implement the functionality of the app. There are different approaches you can take, and you have to make your own decisions. Try to get each part working before working on the next. Ideally, you make a commit for each part. That way, if something goes wrong, you can always revert to a working version!

**Part 4: Shopping cart**

**User Stories:**

* As a user, I want to be able to click on items in my shopping list click and add them to my shopping cart.
* As a user, I want to be able to empty my shopping cart with a click on a delete button

Try to figure out how to do this and what it takes. Try to answer the following questions:

1. Are there components you could reuse?
2. Which logic is needed in which components?
3. Where am I going to put state?
4. Which components will receive which props?

**Tip: draw the required components on an A4 sheet!** Compare your drawing and answer the above questions with the video below; how we use this shopping list and the shopping cart have thought out.

***Note: this is the final result after part 7 of this******exercise. Your app doesn't have to look like the end result yet, but you can use it as inspiration.***

***Graphical user interface, application, website

Description automatically generated***

Component structure:

> App

> Container

> ShoppingCart

> List

> ListItem

> ListItem

> ListItem

> ListItem

> GroceryList

> List

> ListItem

> ListItem

> ListItem

**What steps are needed to make this?**

1. Create 2 new components ShoppingCart and GroceryList . They will both use the List component in their render methods. What props do they need?
2. You want to render different items in both lists. To do this, move the items in the state of List to a higher level. We will do that in the next step, by creating a new component to maintain the state.
3. Create a Container component at the level of the App component, where you put the GroceryList and ShoppingCart.
4. Render your Container component inside the App component, and remove the top level List component. Your structure is now as follows:
5. > Container
6. > ShoppingCart
7. > List
8. > ListItem
9. > GroceryList
10. > List

> ListItem

1. In the Container make a state with groceryItems (make sure to remove the state from List and level it up in the Container).
2. Add a new array to the state of your Container component: shoppingListItems. This array will be updated when we add and remove items from the cart.
3. The logic to move items from the GroceryList to the ShoppingCart should be created in the Container component (the place where both lists are saved). Make sure that, when a ListItem is clicked, the click event is passed from ListItem > List > GroceryList > Container. Check if your click works with console.log and make sure you have the title of the grocery what has been clicked passes into a newly created handleClickGroceryItem function in the Container.
4. When the GroceryListItem is clicked, add the grocery to the shopping cart (shoppingListItems). Pay attention! Modifying State automatically re-renders your component. You cant simply modify state (as you know): use the React [setState](https://reactjs.org/docs/react-component.html#setstate) function. You will now see your Shopping List update when you click on an item in your shopping list.
5. You only want to move items from your shopping list to your shopping cart and not the other way around. Because we dont pass the response to clicking in the ShoppingCart component on a groceryItem to the Container, you will see that adding only works when you click on an item from your shopping list and it doesn't work when you click on an item in your shopping cart. Even though it both use the same component, namely List
6. Finally, you would like to empty the shoppingListItems when a button is clicked that is called "empty cart". In ShoppingCart add a button. When the button is clicked, a function should go off in Container that says emptyCart.
7. In this emptyCart function, you want the state of shoppingListItems change to an empty array.
8. Make sure that the shopping list and the shopping cart are both styled differently. Use the nth-child selector to make sure that they are colored alternately.

**Part 5: Create an InputField to add items to your shopping list.**

**User Stories:**

* As a user, I want to be able to add items to my shopping list.

What steps are needed to make this so?

1. Create an InputField component in a new file InputField.js. (Function component or a class-based component? The choice is yours).
2. Render the InputField in the GroceryList component.
3. When the user adds a new grocery item with this, it must appear in the List of GroceryList .
4. **Pass a function** from the parent component GroceryList to the InputField component. Make sure you can work with what the user has entered in the relevant component. In this case: Container here namely the grocerylist stored in the state.
5. Make sure that this function can not only be called, but also can take an argument with it, namely what the user has filled in. Use console.log to check if you have input from the user received in the Container.
6. Update the state with this.setState of the GroceryList with the new grocery.
7. Knowledge check: what happens to a React component when the state is being updated? The new item will automatically be rendered after the state change.

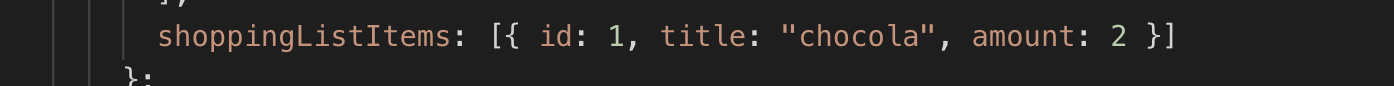
**Part 6: Show duplicate shopping cart items with the text "quantity: 3" in the shopping cart.**

**User Stories:**

* As a user, I want, when I add an item to my shopping cart that already is in it, to see the quantity of the item increased by 1.

**What steps are needed to make this?**

1. Information about the quantity of a certain item is not yet included in the current state shoppingListItems. Add this as property per item.



1. Add the text "quantity: X" to the shopping cart. The shopping list and shopping cart both use List. We must therefore pass a prop to List that tells it whether it is a shopping cart or not. And based on this whether or not the text to render. This is called conditional rendering. To this end, we can add a boolean prop showQuantities to the List component.
2. In the current function where an item is added to the shopping cart, check whether the current shopping cart already contains the clicked item.
3. If not, continue as you already did. With 1 rule there: also add an amount: 1 when you add a new item to the shoppingList adds.
4. If yes, then call a new function that we are going to create now.
5. In this new function addAmountToItem you will do the following:

* find the relevant item in the current state
* increase the amount of this item by 1
* update the new state with the new amount.

**Part 7: Styling**

* Finally, you can go wild with the styling!
* Use your creativity and what you have learned from CSS.
* Creativity Block? Then try to copy our app as closely as possible as shown in the screenshot above

[**Completed. Let's continue.**](javascript:void(0))