

Participant P23

00:02:14 **Researcher**
I'm going to start recording and count the time now, OK.

00:02:58 **Participant**
OK.

00:05:06 > **Viewing file** (webpack.common.js)

00:05:08 > **Viewing file** (webpack.dev.js)

00:05:20 **Participant**
How do I start the app

00:05:39 **Researcher**
You can ask the copilot

00:05:43 **Chat** (window: 0)
> **Agent**
github.copilot.default
> **Context**
viewport (webpack.common.js)
> **Message**
how do I run a js project
> **Response**
To run a JavaScript project like yours, which uses Webpack, follow the steps below:

1. **Install dependencies:** Make sure the project dependencies are installed. In the terminal, run:

npm install
2. **Compile the project:** Use Webpack to compile the project. Usually, the command is:

npm run build

Make sure the build script is configured in the [package.json](#) file. It is usually something like:


```
"scripts": {  
  "build": "webpack"  
}
```
3. **Run the development server (optional):** If you want to run the project on a local server for development, use:

npm start

Make sure the start script is configured in the [package.json](#), usually with something like:


```
"scripts": {  
  "start": "webpack serve"  
}
```
4. **Open in the browser:** After running the development server, access the address indicated in the terminal, usually `http://localhost:8080`.
If any of these steps do not work, check the [package.json](#) file to confirm the available scripts.

00:05:45 > **Response rendering**
- 00:05:53
00:05:59 **Participant**

00:06:01 Is the environment set up?
Researcher
Yes

00:06:06 **Participant**
OK, wait a second.

00:06:18 Alright, press me.

00:06:47 I think I understand now.

00:07:18 **Researcher**
The first task is for you to change the color of the initial box.
when you opened the application for the first time.

00:07:22 **Participant**
Is it this box here?

00:07:31 **Researcher**
That's right

00:07:37 **Participant**
Got it

00:07:50 Just that, then, right, change to blue.

00:07:51 **Researcher**
Yes

00:07:54 **Participant**
Ok

00:08:03 > **Viewing file** (webpack.common.js)

00:08:27 > **Viewing file** (webpack.dev.js)

00:08:32 > **Viewing file** (src/index.js)

00:08:38 > **Viewing file** (src/todo/app.jsx)

00:08:42 > **Viewing file** (src/todo/app.css)

00:08:47 > **Viewing file** (src/todo/components/input.jsx)

00:09:11 > **Viewing file** (src/todo/components/item.jsx)

00:09:36 > **Viewing file** (src/todo/components/paginationControls.jsx)

00:09:45 > **Viewing file** (src/todo/components/main.jsx)

00:10:04 > **Viewing file** (public/index.html)

00:10:11 > **Viewing file** (src/todo/constants.js)

00:10:15 > **Viewing file** (src/todo/reducer.js)

00:10:20 > **Viewing file** (src/todo/constants.js)

00:10:54 **Participant**
If I, for example, could I ask the copilot
to find a text in the project?

00:10:57 **Researcher**
Yes.

00:11:03 You can ask in the way you think will work.

00:11:15 **Chat** (window: 0)
> **Agent**
github.copilot.workspace
> **Context**
workspace, viewport (src/todo/constants.js)
> **Message**
Find for me where the phrase "Nenhum afazer aqui ainda" is in the scripts.
> **Response**
The phrase "Nenhum afazer aqui ainda" is located in the file
[src/todo/components/main.jsx](#), within the component [Main](#)
([src/todo/components/main.jsx](#)), in the following part of the code:

```
File: src/todo/components/main.jsx
<div className="todo-list-empty-container">
  <span className="todo-list-empty-message">Nenhum afazer aqui
```

```
ainda</span>
</div>
```

00:11:18

Researcher

As I said before, if after asking the questions, you think the copilot didn't help you, you can search the web

00:11:19
- 00:11:21

> Response rendering

00:11:22

Researcher

If you do that, let me know.

00:11:32

> **Viewing file** (src/todo/components/main.jsx) through Copilot link

00:12:27

Chat (window: 0)

> Agent

github.copilot.default

> Context

selection (src/todo/components/main.jsx)

> Message

I wanted to change the color of the button that has the text "Nenhum afazer aqui ainda"

> Response

In the provided code, "Nenhum afazer aqui ainda" is not on a button, but rather in a `<span>` element. If you want to change the color of this text, you can do so by adding or modifying the CSS style associated with the class [todo-list-empty-message \(src/todo/components/main.jsx\)](#).

Here's how you can do it:

1. **Locate the corresponding CSS file:** Look for the CSS file where the class [todo-list-empty-message \(src/todo/components/main.jsx\)](#) is defined. If it doesn't exist, you can create or edit the associated CSS file.
2. **Add or modify the class style:** For example, if you are using a CSS file, add something like:

```
/* filepath: /workspaces/copilot-study-todo/src/todo/styles.css */
.todo-list-empty-message {
  color: red; /* Change to the desired color */
}
```

3. **Ensure the CSS is being imported:** In the file [src/todo/components/main.jsx](#) or another relevant file, make sure the CSS is being imported:

```
import "../styles.css";
```

If you actually wanted to change the color of a button, please indicate where it is in the code.

00:12:29
- 00:12:32

> Response rendering

00:14:00

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

selection (src/todo/components/main.jsx)

> **Message**

In this project, where is the CSS file related to the button?

> **Response**

In the provided code, there is no button directly related to the phrase "No tasks here yet." This phrase is inside a `<span>` element with the class `todo-list-empty-message` ([src/todo/components/main.jsx](#)). To find the related CSS file, you can:

1. **Check the imported CSS files:** In the file [src/todo/components/main.jsx](#), there are no explicit imports of CSS files. This suggests that the style might be in a global CSS file or in a specific file that was not shown.

2. **Search the project:** Use the command below in the terminal to locate where the class `todo-list-empty-message` ([src/todo/components/main.jsx](#)) is defined:

```
grep -r "todo-list-empty-message" /workspaces/copilot-study-todo
```

3. **Check the global styles file:** If the project follows a common structure, the CSS file might be in a directory like `src/styles` or `src/todo/styles.css`.

If you do not find the file, you can create a new CSS file or add styles directly to the component using `style` or a library like `styled-components`.

> **Response rendering**

> **Viewing file** (src/todo/app.css)

Participant

Is that right?

> **Editing file** (src/todo/app.css)

Researcher

Ok, you can move on to task b

Participant

I'll copy placeholder

> **Viewing file** (src/todo/components/main.jsx) through Copilot link

> **Editing file** (src/todo/components/main.jsx)

Participant

Change to 'Type a to-do here'

ok.

This tip was practical.

ok, I'll do task c

Researcher

OK.

Participant

Currently, tasks are ordered in the order they were entered

Change to alphabetical order.

I'm thinking about how to ask the copilot in the best way

Researcher

ok.

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

viewport (src/todo/components/main.jsx)

> **Message**

What is the best way to sort the following tasks, in this order: "o", "rápido", "123", "marrom", "raposa", how do I do this?

00:14:02
- 00:14:08

00:15:01

00:15:42

00:15:46
- 00:15:47

00:15:58

00:16:06

00:16:35

00:16:42
- 00:16:42

00:16:42

00:17:05

00:17:08

00:17:12

00:17:14

00:17:15

00:17:26

00:18:40

00:18:44

00:19:22

> Response

To sort the tasks in the specific order you mentioned: "o", "rápido", "123", "marrom", "raposa", you can use an approach based on a **custom order**. This can be done by creating an array with the desired order and using the sort method with a comparison function.

Here's how you can implement this:

Step 1: Define the custom order

Create an array that defines the desired order.

Step 2: Sort the tasks

Use the sort method based on the position of the items in the custom order array.

Code example:

```
const todos = ["raposa", "rápido", "123", "marrom", "o"];
const customOrder = ["o", "rápido", "123", "marrom", "raposa"];

const sortedTodos = todos.sort((a, b) => {
  return customOrder.indexOf(a) - customOrder.indexOf(b);
});

console.log(sortedTodos);
// Output: ["o", "rápido", "123", "marrom", "raposa"]
```

How to apply in your project:

1. Adapt the code above to sort the [todos \(src/todo/components/main.jsx\)](#) in the [Main component \(src/todo/components/main.jsx\)](#).
2. Use [useMemo \(node_modules/@types/react/index.d.ts\)](#) to apply the custom sorting before rendering the visible items.

Change in the [Main component \(src/todo/components/main.jsx\)](#):

```
const customOrder = ["o", "rápido", "123", "marrom", "raposa"];

const visibleTodos = useMemo(
  () =>
    todos
      .filter((todo) => {
        if (route === "/active") return !todo.completed;
        if (route === "/completed") return todo.completed;
        return todo;
      })
      .sort((a, b) => customOrder.indexOf(a.text) -
customOrder.indexOf(b.text)), // Custom sorting
  [todos, route]
);
```

Note:

Ensure that each [todo \(src/todo/components/main.jsx\)](#) in the [todos array \(src/todo/components/main.jsx\)](#) has a text property (or equivalent) that contains the task text. Adjust the code according to the structure of your [todo object \(src/todo/components/main.jsx\)](#).

00:19:25
- 00:19:33

00:21:34

00:22:16

> Response rendering

Participant

First I'll check if it's sorting

Chat (window: 0)

> Agent

github.copilot.default

> Context

viewport (src/todo/components/main.jsx)

> Message

I want the tasks to be added in alphabetical order following the previous example.

> Response

To ensure that the tasks are added in alphabetical order, you can use the sort method on the array of [todos \(src/todo/components/main.jsx\)](#) before rendering them. Here's how you can implement this in the [Main component \(src/todo/components/main.jsx\)](#):

Change in [visibleTodos \(src/todo/components/main.jsx\)](#):

Add the alphabetical sorting logic in the [useMemo \(node_modules/@types/react/index.d.ts\)](#) that already filters the tasks. The sorting will be based on the text property (or equivalent) of each task.

Updated Code:

```
const visibleTodos = useMemo(
  () =>
    todos
      .filter((todo) => {
        if (route === "/active") return !todo.completed;
        if (route === "/completed") return todo.completed;
        return true;
      })
      .sort((a, b) => a.text.localeCompare(b.text)), //
  [todos, route]
);
```

Alphabetical sorting

Explanation:

1. [filter \(src/todo/components/main.jsx\)](#): Filters the tasks based on the route (/active, /completed, or all).
2. **sort**: Sorts the tasks alphabetically using `localeCompare`, which considers case and accent differences.

Note:

Ensure that each [todo object \(src/todo/components/main.jsx\)](#) in the [todos array \(src/todo/components/main.jsx\)](#) has a text property containing the task text. If the property has a different name, replace `a.text` and `b.text` with the correct name.

Result:

Now, the tasks will be displayed in alphabetical order automatically.

> Response rendering

> Implementing Copilot code using apply button (src/todo/components/main.jsx)

00:22:18
- 00:22:23

00:22:57
- 00:23:02

00:23:33 > **Viewing file** (src/todo/components/main.jsx)
00:23:34 **Researcher**
What are you thinking now?
00:25:10 **Participant**
The sorting isn't working, nothing changed
00:25:13 **Researcher**
ok.
00:25:21 **Participant**
It went back to normal.
00:25:57 The code should be compiling
00:26:02 **Researcher**
The code the copilot suggested didn't work
00:28:21 **Chat** (window: 0)
> **Agent**
github.copilot.default
> **Context**
viewport (src/todo/components/main.jsx)
> **Message**
I am having problems with this implementation when running the site it bugs.
> **Response**
The problem may be related to the handling of the array [todos](#) ([src/todo/components/main.jsx](#)) or the sorting/filtering logic in [useMemo](#) ([node_modules/@types/react/index.d.ts](#)). Let's review and correct the implementation to ensure it works correctly.

Here is a revised version of [useMemo](#) ([node_modules/@types/react/index.d.ts](#)) with alphabetical sorting logic:

Updated Code:

```
const visibleTodos = useMemo(() => {  
  // Filters the todos based on the route  
  const filteredTodos = todos.filter((todo) => {  
    if (route === "/active") return !todo.completed;  
    if (route === "/completed") return todo.completed;  
    return true;  
  });  
  
  // Sorts the todos alphabetically by the `text` field  
  return filteredTodos.sort((a, b) => a.text.localeCompare(b.text));  
}, [todos, route]);
```

Possible Problems and Solutions:

1. **Nonexistent text Property:** Ensure that each object in the array [todos](#) ([src/todo/components/main.jsx](#)) has the text property containing the task text. If the property name is different, replace `a.text` and `b.text` with the correct name.
2. **Mutation of the Original Array:** The sort method modifies the original array. To avoid this, create a copy of the array before sorting:

```
return [...filteredTodos].sort((a, b) =>  
  a.text.localeCompare(b.text));
```

3. **Dependencies of [useMemo](#) ([node_modules/@types/react/index.d.ts](#)):** Check if the dependencies of [useMemo](#) ([node_modules/@types/react/index.d.ts](#)) are correct. They should include

[todos \(src/todo/components/main.jsx\)](#) and [route \(src/todo/components/main.jsx\)](#), as shown in the code above.

4. **Structure of the Array [todos \(src/todo/components/main.jsx\)](#):** Ensure that the array [todos \(src/todo/components/main.jsx\)](#) is being passed correctly as a prop to the [Main \(src/todo/components/main.jsx\)](#) component. It should be an array of objects with the expected properties.

Test the Code:

After making the changes, test the site again. If the problem persists, check the browser console for error messages and share more details about the unexpected behavior.

00:28:23
- 00:28:28

> **Response rendering**

00:29:02
- 00:29:04

> **Implementing Copilot code** using apply button (src/todo/components/main.jsx)

00:29:36

Researcher

Apparently, the code it generated broke the application.

00:29:39

00:29:40

Participant

Yes, I'll go back to how it was before.

00:29:55
- 00:29:55

> **Implementing Copilot code** using apply button (src/todo/components/main.jsx)

00:30:37

Researcher

See, you can try other approaches. You can interact with the copilot as many times as necessary

00:30:54

Participant

ok.

00:31:14

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

viewport (src/todo/components/main.jsx)

> **Message**

Where does the addition of tasks occur in the code?

> **Response**

In the provided code, there is no explicit function that handles the **addition of tasks**.

The addition likely occurs in another component or function that manipulates the global state or the array [todos \(src/todo/components/main.jsx\)](#), which is passed as a prop to the [Main \(src/todo/components/main.jsx\)](#) component.

Possible locations where the addition occurs:

1. **In the parent component of [Main \(src/todo/components/main.jsx\)](#):** The array [todos \(src/todo/components/main.jsx\)](#) is passed as a prop to [Main \(src/todo/components/main.jsx\)](#). The addition logic might be in the parent component, where the state of [todos \(src/todo/components/main.jsx\)](#) is managed.
2. **In the [dispatch \(src/todo/components/main.jsx\)](#):** The [dispatch \(src/todo/components/main.jsx\)](#) is passed as a prop to [Main \(src/todo/components/main.jsx\)](#). It might be connected to a state manager like Redux or a reducer in React. The addition of tasks can be done through an action, such as:

```
dispatch({ type: "ADD_ITEM", payload: newTodo });
```

3. **In another component:** There might be a form or button in another component that calls a function to add a new task to the global or local state.

How to find where the addition occurs:

1. Look for [dispatch \(src/todo/components/main.jsx\)](#) with the type `ADD_ITEM`:
Use the command below in the terminal to locate where the addition action is triggered:

```
grep -r "ADD_ITEM" /workspaces/copilot-study-todo
```

2. Check the parent component: Open the component that renders [Main \(src/todo/components/main.jsx\)](#) and look for where the state [todos \(src/todo/components/main.jsx\)](#) is managed or updated.
3. Check the reducer (if applicable): If you are using Redux or a reducer, look for the logic associated with the action type `ADD_ITEM`.

If you share more details about how the state [todos \(src/todo/components/main.jsx\)](#) is managed, I can help identify exactly where the addition occurs.

00:31:16
- 00:31:20

00:32:04

> Response rendering

Chat (window: 0)

> Agent

github.copilot.default

> Context

codebase

> Message

#codebase where the addition of items occurs

> Response

The addition of items occurs in the Header component. Specifically, the [addItem \(src/todo/components/header.jsx\)](#) function is called when the user enters a title in the input field and presses "Enter". This function triggers the [ADD_ITEM \(src/todo/constants.js\)](#) action using the [dispatch \(src/todo/components/header.jsx\)](#).

Relevant code in the header.jsx file:

```
export function Header({ dispatch }) {  
  const addItem = useCallback((title) => dispatch({ type: ADD_ITEM,  
    payload: { title } })), [dispatch]);  
  
  return (  
    <header className="header" data-testid="header">  
      <h1>Afazeres</h1>  
      <Input onSubmit={addItem} label="Novo afazer">
```

```
placeholder="O que precisa ser feito?" />
    </header>
  );
}
```

Addition flow:

1. The [Input \(src/todo/components/input.jsx\)](#) component (imported from [input.jsx](#)) calls the [onSubmit \(src/todo/components/header.jsx\)](#) function (which is [addItem \(src/todo/components/header.jsx\)](#)) when the user presses "Enter".
2. The [addItem \(src/todo/components/header.jsx\)](#) function triggers the [ADD_ITEM \(src/todo/constants.js\)](#) action with the new task's title as [payload \(src/todo/components/header.jsx\)](#).
3. The [dispatch \(src/todo/components/header.jsx\)](#) is connected to the [todoReducer \(src/todo/reducer.js\)](#) in the [reducer.js](#) file, which adds the new item to the state.

Logic in the [todoReducer \(src/todo/reducer.js\)](#):

In the [reducer.js](#) file, the [ADD_ITEM \(src/todo/constants.js\)](#) action is handled like this:

```
case ADD_ITEM:
  newState = state.concat({ id: generateUniqueID(), title:
    action.payload.title, completed: false });
  break;
```

Summary:

- **User input:** Input field in the [Header \(src/todo/components/header.jsx\)](#) component.
- **Action triggered:** [ADD_ITEM \(src/todo/constants.js\)](#).
- **State update:** Done by the [todoReducer \(src/todo/reducer.js\)](#), which adds the new item to the [todos \(src/todo/app.jsx\)](#) array.

00:32:10
- 00:32:18
00:32:43

> Response rendering

Chat (window: 0)

> Agent

github.copilot.default

> Context

codebase

> Message

#codebase and how do I sort the added items

> Response

In your project, the items are added to the global state through the action [ADD_ITEM \(src/todo/constants.js\)](#), which is handled in the [todoReducer \(src/todo/reducer.js\)](#) in the [reducer.js](#) file. To sort the added items, you can

modify the [todoReducer \(src/todo/reducer.js\)](#) to apply sorting whenever a new item is added.

Step 1: Modify the [todoReducer \(src/todo/reducer.js\)](#) to sort the items

In the `reducer.js` file, update the case [ADD_ITEM \(src/todo/constants.js\)](#) to sort the items after adding the new item.

Updated Code:

```
case ADD_ITEM:
  newState = state.concat({ id: generateUniqueID(), title:
action.payload.title, completed: false });
  // Sort the items by title in alphabetical order
  newState.sort((a, b) => a.title.localeCompare(b.title));
  break;
```

Step 2: Test the functionality

1. Add new items in the application.
2. Check if the items are displayed in alphabetical order automatically.

Explanation:

- [concat \(src/todo/reducer.js\)](#): Adds the new item to the state without mutating the original array.
- `sort`: Sorts the items in the array [newState \(src/todo/reducer.js\)](#) based on the property [title \(src/todo/components/header.jsx\)](#) in alphabetical order.

Alternative: Sort in the component [Main \(src/todo/components/main.jsx\)](#)

If you prefer, you can sort the items in the component [Main \(src/todo/components/main.jsx\)](#) before rendering them, as previously discussed. This can be done in the [useMemo \(node_modules/@types/react/index.d.ts\)](#) that calculates [visibleTodos \(src/todo/components/main.jsx\)](#):

```
const visibleTodos = useMemo(
  () =>
    todos
      .filter((todo) => {
        if (route === "/active") return !todo.completed;
        if (route === "/completed") return todo.completed;
        return true;
      })
      .sort((a, b) => a.title.localeCompare(b.title)), // Sort by
  title
  [todos, route]
);
```

Which approach to choose?

- Sort in the [todoReducer \(src/todo/reducer.js\)](#): Ensures that the global state is always sorted.
- Sort in the component [Main \(src/todo/components/main.jsx\)](#): Useful if sorting is only for display and does not need to be reflected in the global state.

Choose the approach that best suits your needs!

00:32:50
- 00:32:56

> **Response rendering**

00:34:07
- 00:34:08

> **Implementing Copilot code** using apply button (src/todo/components/main.jsx)

00:34:21

Researcher

What are you thinking now?

00:34:24

Participant

So the copilot gave me 2 options here to do the sorting

00:34:30

He said there's a global array

00:34:40

It's like in the display, it didn't work, I thought about sorting in the global array.

00:34:45

Researcher

ok.

00:34:45

Participant

I know it apparently broke everything.

00:35:13

I tried to use this here.

00:35:14

it's the same implementation. This one and that other one there.

00:35:17

Researcher

ok.

00:35:21

Participant

Practically the same implementation.

00:35:29

Researcher

ok

00:35:31

Participant

I'm trying to use it here.

00:35:32

But I don't know.

00:35:33

I think it broke everything.

00:35:36

I can't say very well.

00:35:37

Let me see if it goes, it really doesn't.

00:36:05

I'll try this again.

00:36:08
- 00:36:12

> **Implementing Copilot code** using apply button (src/todo/components/main.jsx)

00:36:12

Researcher

ok

00:36:18

Participant

Let me see.

00:36:28

Researcher

You can try checking the browser log

00:37:04

Participant

Let me see if it.

00:37:36

And can I take and send the error to the copilot?

00:37:40

Researcher

yes

00:37:45

Participant

Try to do this again, then.

00:37:47
- 00:37:49

> **Implementing Copilot code** using apply button (src/todo/components/main.jsx)

00:38:16

Participant

Now it worked.

00:38:50

Researcher

ok, I'll send you the other task list now

00:38:52

You can do it in the order you think is best. Just move on to the next task after doing all the checks ok.

00:39:22

Ask below, before moving on to the next.

00:39:28

But you don't need to do it in order, no.

00:39:49

Participant

ok

00:39:54

I'll do the persistence task

Chat (window: 0)

> Agent

github.copilot.default

> Context

codebase

> Message

#codebase make the created items persist

> Response

To persist the created items, you can use [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) to save and load the items. The project already has a [loadState](#) ([src/todo/reducer.js](#)) and [saveState](#) ([src/todo/reducer.js](#)) function in the `reducer.js` file, but they are not being used correctly. Let's fix this.

Step 1: Load the initial state from [localStorage](#) ([node_modules/@types/node/globals.d.ts](#))

In the `app.jsx` file, pass the initial state loaded from [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) to [useReducer](#) ([node_modules/@types/react/index.d.ts](#)).

Update the code to:

```
File: src/todo/app.jsx
import { useReducer } from "react";
import { Header } from "../components/header";
import { Main } from "../components/main";
import { Footer } from "../components/footer";

import { todoReducer, loadState } from "../reducer";

import "../app.css";

export function App() {
  const [todos, dispatch] = useReducer(todoReducer, loadState()); //
  // Load initial state from localStorage

  return (
    <>
      <Header dispatch={dispatch} />
      <Main todos={todos} dispatch={dispatch} />
      <Footer todos={todos} dispatch={dispatch} />
    </>
  );
}
```

Step 2: Save the state in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) after each change

In the `reducer.js` file, the [saveState](#) ([src/todo/reducer.js](#)) function is already being called after each action in the [todoReducer](#) ([src/todo/reducer.js](#)). Make sure it is correctly saving the state in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)).

The relevant snippet is already like this:

```
File: src/todo/reducer.js
const saveState = (state) => {
  localStorage.setItem("todos", JSON.stringify(state));
};
```

```

export const todoReducer = (state, action) => {
  let newState;
  switch (action.type) {
    case ADD_ITEM:
      newState = state.concat({ id: generateUniqueID(), title:
action.payload.title, completed: false });
      break;
    // ... other actions
  }

  saveState(newState); // Save the updated state in localStorage
  return newState;
};

```

Step 3: Test persistence

1. Add new items in the app.
2. Reload the page.
3. Check if the added items remain in the list.

Explanation

- **loadState (src/todo/reducer.js)**: Loads the saved state from [localStorage \(node_modules/@types/node/globals.d.ts\)](#) when starting the app.
- **saveState (src/todo/reducer.js)**: Saves the updated state in [localStorage \(node_modules/@types/node/globals.d.ts\)](#) after each action in the [todoReducer \(src/todo/reducer.js\)](#).

With these changes, the created items will be automatically persisted in [localStorage \(node_modules/@types/node/globals.d.ts\)](#).

00:40:51
- 00:40:58

> **Response rendering**

00:41:16

> **Viewing file** (src/todo/app.jsx) through Copilot link

00:41:19
- 00:41:21

> **Implementing Copilot code** using apply button (src/todo/app.jsx)

00:41:35
- 00:41:44

> **Implementing Copilot code** using apply button (src/todo/reducer.js)

00:41:35

> **Viewing file** (src/todo/reducer.js) through Copilot link

00:42:15

Participant

I'll see the single one, add our favorite text.

00:42:25
- 00:42:27

> **Implementing Copilot code** using apply button (src/todo/reducer.js)

00:42:43
- 00:42:44

> **Implementing Copilot code** using apply button (src/todo/app.jsx)

00:43:00

Participant

It's not, it's going, I don't know what it is.

00:43:06

Researcher

Try to run it again. Maybe there's an error in the interface.

00:43:10

Participant

I think that was it really

00:43:27

Researcher

Our time is up, I'll ask you to stop programming now ok.

00:43:30

Participant

OK.

00:45:01

Researcher

You can close the application now. And please answer the post-study questionnaire.

00:45:11

Feel free to make comments

00:45:17

Participant

Alright, how would you rate the copilot's responses?

00:45:20

in relation to length length. What do you mean by length?

00:45:25

Researcher

Length if the responses are too long.

00:45:29

Participant

ok

00:45:31

I think they are good responses.

00:45:39

Details?

00:45:46

I think it was good also the use of technical terms.

00:45:52

There wasn't much use. I think it's neutral.

00:45:55

Researcher

ok.

00:45:59

Participant

I don't think he is he was.

00:46:11

But he was neutral there because he

00:46:16 because at the beginning he had a problem
00:46:25 Following your instructions.
00:47:12 I think neutral is good.
00:47:15 I think it was good and bad.
00:47:23 copilot in my work.
00:47:26 It would allow me to perform tasks more quickly.
00:47:30 Look, I think so. Because in the tasks I didn't know anything about what I
00:47:35 was doing. I don't deal much with this language,
00:47:39 so it helped me with what I had to do. did the right part.
00:47:50 in studies.
00:47:51 I think so. I think I really loved it.
00:47:54 **Researcher**
Do you use it in your work during study?
00:48:00 **Participant**
sometimes to find different ways to solve a problem.
00:48:04 Seeing that a problem is being solved one way,
00:48:07 I'll try to find other ways to manage to solve
00:48:14 **Researcher**
did you use copilot before this research?
00:48:19 **Participant**
No.
00:48:20 Never used copilot.
00:48:22 **Researcher**
other LLM, Gpts?
00:48:25 **Participant**
yes, I use ChatGPT a lot.
00:48:47 Yes, learning to operate would be
00:48:49 easy for me. I didn't understand
00:49:09 **Researcher**
Do you think you could master
00:49:12 copilot
00:49:15 **Participant**
I think it would be easy.
00:51:26 How much did you have to strive to start your performance level?
00:51:32 I didn't strive a lot, just a little.
00:51:37 I felt a little stressed and bothered and bothered.
00:51:39 **Researcher**
Is the environment to develop new complicate?
00:51:43 **Participant**
Staying is a little difficult. The app's language I don't master
00:51:48 I think someone with more experience would have a better performance
00:52:18 This one I found easy
00:52:20 **Researcher**
Ah, these. You managed to do them all.
00:52:23 **Participant**
Yes.
00:52:26 This one I found difficult.
00:53:51 **Researcher**
Ok, thank you for answering. I'll ask you a few more questions now.
00:54:02 How would you have responded to the tasks without the help of the copilot? And how did the
copilot influence your strategy to solve the tasks?
00:54:22 **Participant**
I think starting from the point that I don't know the app's language
00:54:31 Then I would have to research on the internet.
00:54:35 How to solve this problem
00:54:39 And it would be more or less that. I would spend a good amount of time.

00:54:43 Researching how to solve it then I would have to,
00:54:45 try to understand how the code works there
00:54:49 I tried a little of that at the beginning
00:54:52 but I realized that no, I wouldn't be able to
00:54:57 So I had to use the.
00:54:58 The copilot.
00:55:17 **Researcher**
What aspects of the copilot did you like and which did you not like?
00:55:24 **Participant**
I really liked that you can copy the
00:55:28 code directly, exactly as
00:55:32 it suggested.
00:55:38 I also liked that it references your code
00:55:43 Because in ChatGPT I find it very bad.
00:55:46 you sometimes have to keep copying a
00:55:49 part and pasting
00:55:50 The copilot does everything automatically directly through the IDE.
00:55:52 This is much simpler.
00:56:00 **Researcher**
It's the last question. Did you change the way you interact with the copilot during the
experiment?
00:56:20 **Participant**
I tried at the beginning.
00:56:23 to ask a little about what was happening in the code
00:56:27 tried to understand a little
00:56:31 For it to say, how it was working,
00:56:34 what the code was doing. But then, I.
00:56:37 Realized that no, it wouldn't help much
00:56:41 I started asking the copilot to try to solve the problem.
00:56:45 **Researcher**
Cool.
00:56:46 **Participant**
See if it had any suggestions to solve the problem.
00:56:48 The best way to solve it.
00:56:49 **Researcher**
ok. Thank you very much for participating in our study.