

Participant P28

00:05:24 **Participant**
Well, I guess the index dot JSON is the.

00:05:30 Sorry, index dot JS is the main file, so I probably need to run that.

00:05:38 I think, but the docs are lost.

00:05:43 Yeah. So probably I just need to run the index

00:05:46 dot JS that kind of like compiles.

00:05:53 No, wait.

00:05:55 > **Viewing file** (webpack.common.js)

00:05:59 > **Viewing file** (webpack.dev.js)

00:06:01 **Participant**
I'm not very familiar with JavaScript.

00:06:05 So.

00:06:06 Uh, it's a bit confusing, but also I think that this uses.

00:06:09 > **Viewing file** (webpack.common.js)

00:06:11 **Participant**
I guess React maybe.

00:06:14 It's like probably the React web application or something like that,

00:06:19 so I assume there is some kind of a command to compile this whole like all

00:06:24 the packages that are related to the.

00:06:29 How is it called.

00:06:31 To the whole program. Basically all the packages, all the HTML,

00:06:35 JS, JavaScript code and so on, and just compiles into like one web page

00:06:39 or something like that.

00:06:43 I don't know.

00:06:43 I would in general search web for that, but because I don't have this option I

00:06:47 guess I just need to ask the Copilot in this case because I have no idea how to

00:06:51 compile that.

00:06:55 That's my thought process.

00:06:58 **Researcher**
Cool. Well then, please go ahead with trying to run it.

00:07:08 **Participant**
OK.

00:07:12 > **Viewing file** (webpack.dev.js)

00:07:19 **Participant**
That's a bit awkward, but yeah, I see what you mean.

00:07:22 > **Viewing file** (webpack.common.js)

00:07:54 **Participant**
So. Yeah, I will just ask that I need to compile

00:07:57 and run the code that I have never seen before or like the project that I

00:08:01 have never seen before and probably based from entry to output from the webpack

00:08:05 common dot JS.

00:08:08 Like paste the code in and see if it can suggest the command,

00:08:13 how to compile and run the code.

00:08:16 Or the project.

00:09:06 **Chat** (window: 0)
> **Agent**
github.copilot.workspace
> **Context**
workspace, selection (webpack.common.js)
> **Message**

I need to compile and run the project that I have never worked with before. I will input a part of the webpack.common.js file content below. Can you provide with the command prompt command to run project with this structure? entry: { app: path.resolve(__dirname, "src", "index.js"), }, plugins: [new HtmlWebpackPlugin({ title: "TodoMVC: React", template: path.resolve(__dirname, "public", "index.html"), }),], output: { filename: "[name].bundle.js", path: path.resolve(__dirname, "dist"), clean: true, },

> Response

To compile and run this project in your workspace, use the following command in your terminal:

```
npm run dev
```

This will start the development server using the configuration in [webpack.dev.js](#). Open the local server URL shown in the terminal to view your app.

> Response rendering

Participant

Do I need to run it in the development server?

> Viewing file (webpack.dev.js)

Researcher

Yeah, since you'll be since you'll be working on it on the projects, you don't need to fully build it. Just being able to run it is fine.

Participant

OK.

OK, I'm already running into problems.

Researcher

So what are you thinking now?

Participant

Yeah, I will just copy the error that I got into the Copilot and say that this is the error that I got. Actually, maybe sharing the, so now that I remember, that sharing the using the at workspace or yeah, it might be smart. I think this is how you're supposed to use it.

Chat (window: 0)

> Agent

```
github.copilot.workspace
```

> Context

```
workspace, viewport (webpack.dev.js)
```

> Message

```
@workspace Trying the command I got the following error: /workspaces/copilot-study-  
todo (main) $ npm run dev > todomvc-react@1.0.0 dev > webpack serve --open --  
config webpack.dev.js sh: 1: webpack: not found
```

> Response

You need to install the project dependencies before running the dev server. Run this command in your terminal:

```
npm install
```

After installation completes, run:

```
npm run dev
```

This will start the development server as defined in your [package.json](#).

> Response rendering

Participant

Oh, OK, makes sense.

00:09:08
- 00:09:09

00:09:25

00:09:29

00:09:32

00:09:35

00:09:39

00:09:42

00:10:01

00:10:17

00:10:19

00:10:22

00:10:29

00:10:33

00:10:43

00:11:04

00:11:06
- 00:11:07

00:11:13

00:11:14 So I firstly need to install all the dependencies and then run the.
00:11:23 Run the project I guess.
00:11:25 Makes sense.
00:11:56 OK, it also suggests that there is several
00:11:58 vulnerabilities and to run the NPM audit, but I don't think that I have time for
00:12:02 that.
00:12:02 So I guess I'll skip for now.
00:12:13 Hey, it compiled and ran.
00:12:18 **Researcher**
Perfect. All right.
00:12:20 Thanks for thinking aloud the way you're doing, by the way, that's helping a ton.
00:12:24 **Participant**
Yeah.
00:12:25 **Researcher**
So feel free to get started on the on the tasks.
00:12:29 **Participant**
So the first one is practice task A.
00:12:33 OK, cool.
00:12:57 > **Viewing file** (src/todo/components/header.jsx)
00:13:07 **Participant**
So.
00:13:10 > **Viewing file** (src/todo/components/input.jsx)
00:13:10 **Participant**
I'm a person who personally thinks that I will firstly try before I use the helper,
00:13:15 so I think that I'm just gonna look a bit into the.
00:13:21 Different components. So I see that there is like a footer,
00:13:24 header, input and so on.
00:13:26 So I think that I'll just find try to find the text for no todos here yet
00:13:31 is displayed and there is probably some like.
00:13:35 HTML tag next to it or something that I can change?
00:13:41 Yep, so that's what I'm trying to do.
00:13:52 > **Viewing file** (src/todo/components/item.jsx)
00:14:04 > **Viewing file** (src/todo/components/main.jsx)
00:14:15 > **Editing file** (src/todo/components/main.jsx)
- 00:14:18
00:14:23 **Participant**
And because I think that there might be a way to maybe recompile the code without
00:14:27 actually stopping and running it again, I will ask Copilot because I don't have a
00:14:32 web for that.
00:14:34 And I will use workspace because it then knows the structure of the whole project.
00:14:57 **Chat** (window: 0)
> **Agent**
github.copilot.workspace
> **Context**
workspace, viewport (src/todo/components/main.jsx)
> **Message**
@workspace is it possible to recompile the code without fully stopping the project and
running it again?
> **Response**
Yes, it is possible. When you run `npm run dev`, [webpack-dev-server](#) is used. It watches
your files for changes and automatically recompiles and reloads the app in the browser
when you save changes. You do not need to stop and restart the server—just save your
files, and the changes will be reflected automatically.
00:14:59 > **Response rendering**
- 00:15:00
00:15:13 **Participant**
OK.

00:15:14 So I guess if I just control. Oh, there are errors.
00:15:58 OK.
00:15:58 This didn't help.
00:16:01 > **Editing file** (src/todo/components/main.jsx)
00:16:09 **Participant**
Although if I remove my changes then there are still errors here,
00:16:14 so I guess it's it was there before me, but also the color didn't change so.
00:16:21 > **Editing file** (src/todo/components/main.jsx)
00:16:21 **Participant**
Wait actually let me try this, fully stopping it and.
00:16:28 Saving it and running it again just to see if I'm doing everything correctly
00:16:32 with, just like assuming that the tool will
00:16:34 recompile itself.
00:16:37 Well, I guess so, but the color should not be here.
00:16:40 > **Editing file** (src/todo/components/main.jsx)
- 00:16:42
00:16:41 **Participant**
I will try here.
00:16:42 If it doesn't work, then I'll ask Copilot and see how it
00:16:46 works.
00:16:47 If it works, it doesn't work OK.
00:17:10 Umm.
00:17:12 The screen sharing is in my face.
00:17:14 I'll move it somewhere on the other screen, probably because I cannot switch.
00:17:20 The windows otherwise.
00:17:57 Oh.
00:18:10 > **Viewing file** (src/todo/app.css)
00:18:21 **Participant**
I just had a revelation.
00:18:24 **Researcher**
What gave you this revelation?
00:18:26 **Participant**
I looked that there is like a class.
00:18:27 > **Editing file** (src/todo/app.css)
00:18:30 > **Viewing file** (src/todo/components/main.jsx)
00:18:32 **Participant**
And.
00:18:32 I just remembered from the times when I actually had to use HTML and CSS that
00:18:36 > **Editing file** (src/todo/components/main.jsx)
00:18:36 **Participant**
actually it's all defined in the CSS.
00:18:38 And then I was like, OK.
00:18:39 Yeah, fair.
00:18:39 I can just go to the where the class name is and probably there is there are all
00:18:44 the properties so I don't have to overwrite them here,
00:18:47 but I can just go there and change the CSS.
00:18:51 This is kind of like a things that I remember I did at some point long ago,
00:18:55 but.
00:18:57 Yeah, well, it works now.
00:19:00 But yeah, I didn't remember them instantly.
00:19:07 I think that's done.
00:19:29 So I think that with this task I will just use the search across.
00:19:37 All the files.
00:19:38 Or maybe because I already investigated a bit.
00:19:40 > **Viewing file** (src/todo/components/header.jsx)

00:19:40 > **Viewing file** (src/todo/components/input.jsx)

00:19:42 **Participant**
Different locations and I probably can.

00:19:46 Find where this text is. Although there is a lot of like

00:19:50 placeholders and stuff like that.

00:19:51 > **Viewing file** (src/todo/components/main.jsx)

00:19:52 **Participant**
So I may be.

00:19:57 Would just use like the built in search. If I can use that for this task.

00:20:07 **Researcher**
Sorry, was that a question?

00:20:10 **Participant**
Yeah. So basically, can I just use the Visual Studio Code

00:20:14 search functionality to find the specific place, phrase?

00:20:17 > **Viewing file** (src/todo/components/header.jsx) through file search

00:20:18 **Researcher**
Yeah, use anything you want within the within

00:20:20 **Participant**
Oh, OK. Within the yeah. OK.

00:20:20 **Researcher**
the IDE.

00:20:22 **Participant**
So basically I just search for the text that I need to just replace here.

00:20:27 - 00:20:31 > **Editing file** (src/todo/components/header.jsx)

00:20:29 **Participant**
And then I can just.

00:20:33 Paste it here and then.

00:20:39 Yeah, enter a task here.

00:20:40 So that's done.

00:21:02 OK, so I need to enter the todos.

00:21:07 The quick 123 brown fox, OK.

00:21:27 So yeah, I need to change the ordering of the todos

00:21:30 in the list, which should be how do I get back to files.

00:21:34 Oh yeah, it was here.

00:21:35 > **Viewing file** (src/todo/components/input.jsx)

00:21:37 **Participant**
So currently they are sorted by the time when they were added,

00:21:41 but I need to sort them alphabetically.

00:21:45 And to be fair.

00:21:50 I think I'm gonna use the.

00:21:54 Copilot option for that, because I feel like I can spend a lot of

00:21:58 time on trying to find a way to do it.

00:22:01 Meanwhile, Copilot might provide the version really

00:22:04 quick because I don't know enough about.

00:22:09 Well, it's either.

00:22:11 Done in the JavaScript or yeah, probably in JavaScript or HTML or I don't

00:22:17 know.

00:22:18 So basically I think that I will ask Copilot to just help me with that because.

00:22:24 I will spend way too much time on trying to figure it out myself.

00:22:29 I just need to think where it actually sorts them so I can like narrow down the

00:22:37 area where to like how to. Well, I will narrow down the area in order to

00:22:44 ask the correct question.

00:22:46 Let's say it like that.

00:22:50 So I'll find the place where.

00:22:54 I need to search for.

00:22:55 > **Viewing file** (src/todo/app.css)

00:22:57 **Participant**
The change in the order.

00:22:58 > **Viewing file** (src/todo/constants.js)

00:22:59 > **Viewing file** (src/todo/reducer.js)

00:23:04 **Participant**
Oh.

00:23:17 > **Viewing file** (src/index.js)

00:23:20 > **Viewing file** (src/todo/reducer.js)

00:23:21 > **Viewing file** (src/todo/components/input.jsx)

00:23:27 **Participant**
So.

00:23:36 > **Viewing file** (src/todo/components/item.jsx)

00:23:37 **Participant**
This is the input.

00:23:38 This is the items, yes, so it should be somewhere here.

00:23:43 Umm.

00:25:22 **Chat** (window: 0)

> **Agent**
github.copilot.default

> **Context**
viewport (src/todo/components/item.jsx)

> **Message**
I am trying to order my to-dos by the alphabetical order, compared to the current order where todo-s are ordered by the time they were added. What part of the current file is responsible for the order of items in the list?

> **Response**
The order of the to-dos in the list is not determined in the [src/todo/components/item.jsx](#) file. This file ([Item \(src/todo/components/item.jsx\)](#) component) is only responsible for rendering a single to-do item.

The order is determined in the parent component that renders the list of items, which is likely the Main component (main.jsx). In that file, the following code is responsible for rendering the list:

```
File: src/todo/components/main.jsx
<ul className={classnames("todo-list")} data-testid="todo-list">
  {visibleTodos.map((todo, index) => (
    <Item todo={todo} key={todo.id} dispatch={dispatch} index=
{index} />
  ))}
</ul>
```

Here, visibleTodos is an array of to-dos, and its order determines how the items are displayed. By default, this array preserves the order from the original todos array (which is usually by time added).

To order your to-dos alphabetically, you should sort the visibleTodos array before mapping over it in main.jsx.

> **Response rendering**

00:25:23 **Participant**
Yeah. So firstly I want to like find the exact

00:25:26 place.

00:25:29 Which is, or well the function. I guess that is responsible for ordering

00:25:34 the todos.

00:25:36 And I ask which part of the current file is responsible for the order of the items

00:25:42
00:25:46
00:25:55
00:25:55
00:25:58
00:26:11
00:26:17
00:26:36

in the list?

Todos in the list is not determined in the. OK so it says it's not determined here.

OK.

So it's only responsible for one todo item in the list,
but it's not really responsible for the whole list.

And now I will ask, can you search the workspace?

And suggest a solution to this problem.

Chat (window: 0)

> **Agent**

github.copilot.workspace

> **Context**

workspace, viewport (src/todo/components/item.jsx)

> **Message**

@workspace can you search the workspace and suggest the solution for my problem?

> **Response**

Based on your workspace and the code in [src/todo/components/item.jsx](#), your code for editing, toggling, and removing todo items looks correct. However, if you are experiencing a specific problem (such as editing not working, checkboxes not toggling, or items not being removed), please clarify the exact issue.

Here are some common issues and solutions based on your code:

1. Editing a Todo does not update the title or remove the item when empty:

The [handleUpdate\(src/todo/components/item.jsx\)](#) function calls [removeItem\(id\)\(src/todo/components/item.jsx\)](#) if the title is empty, and [updateItem\(id, title\)\(src/todo/components/item.jsx\)](#) otherwise. Both functions dispatch the correct actions. Make sure your reducer in [src/todo/reducer.js](#) handles [UPDATEITEM\(src/todo/constants.js\)](#) and [REMOVEITEM\(src/todo/constants.js\)](#) correctly, which it does.

2. Checkbox does not toggle completion:

The [toggleItem\(src/todo/components/item.jsx\)](#) function dispatches the [TOGGLE_ITEM\(src/todo/constants.js\)](#) action, which is handled in your reducer.

3. UI does not update after actions:

Your components use React state and reducer correctly. If the UI does not update, ensure that the reducer returns a new array/object and not a mutated one. Your reducer does this correctly.

4. Component not rendering or error in console:

Make sure you are importing and using the [Item\(src/todo/components/item.jsx\)](#) component correctly in [src/todo/components/main.jsx](#).

5. Typo in reducer:

In [src/todo/reducer.js](#), there is a typo:

```
case L00D_STATE:
```

It should be:

```
case LOAD_STATE:
```

This could cause issues if you ever dispatch the [LOAD_STATE\(src/todo/constants.js\)](#) action.

6. Incorrect import of PaginationControls:

In [src/todo/components/main.jsx](#), you import [paginationControls\(src/todo/components/main.jsx\)](#) with a lowercase p, but the component is exported as [PaginationControls\(src/todo/components/paginationControls.jsx\)](#).

Change:

```
import { paginationControls } from "../paginationControls";
```

to:

```
import { PaginationControls } from "../paginationControls";
```

And update the usage:

```
<PaginationControls  
  currentPage={currentPage}  
  itemsPerPage={itemsPerPage}  
  setCurrentPage={setCurrentPage}  
  totalItems={visibleTodos.length}  
>
```

If you clarify the specific problem, I can give a more targeted solution. For now, please fix the typo in your reducer and the import/usage of [PaginationControls](#) ([src/todo/components/paginationControls.jsx](#)) as described above.

00:26:38
- 00:26:47

> Response rendering

00:27:04

Participant

OK.

00:27:05

I assumed that it kind of kept track kept track of my previous question,

00:27:09 but I assume it's not happening because it's currently suggesting me a bunch of.
00:27:13 Well, from the first look a bunch of other
00:27:15 solutions that I didn't ask for.
00:27:19 So I guess I'll just copy the previous.
00:27:25 Request that I already wrote and just add the workspace tag or what's it called
00:27:31 like the command like at workspace.
00:27:35 For it to search for exact place in my workspace for the specific problem that I
00:27:40 need to solve right now, not all the problems in the world
00:27:43 basically.
00:27:59 And then I need to change the last question.
00:28:15 **Chat** (window: 0)

> Agent

github.copilot.workspace

> Context

workspace, viewport (src/todo/components/item.jsx)

> Message

@workspace I am trying to order my to-dos by the alphabetical order, compared to the current order where todo-s are ordered by the time they were added. What script is responsible for the order of todo-s that I have to change?

> Response

The order of your todos is determined by the array returned from your reducer in [src/todo/reducer.js](#) and how it is rendered in the [Main \(src/todo/components/main.jsx\)](#) component.

Currently, todos are rendered in the order they appear in the [todos \(src/todo/components/main.jsx\)](#) array, which is ordered by the time they were added (new items are appended).

To order your todos alphabetically, you should sort the [visibleTodos \(src/todo/components/main.jsx\)](#) array in [Main \(src/todo/components/main.jsx\)](#) before rendering. This is the best place to apply the alphabetical sort.

You can update the [visibleTodos \(src/todo/components/main.jsx\)](#) calculation in [Main \(src/todo/components/main.jsx\)](#) to sort by title:

```
File: src/todo/components/main.jsx
const visibleTodos = useMemo(
  () =>
    todos
      .filter((todo) => {
        if (route === "/active")
          return !todo.completed;

        if (route === "/completed")
          return todo.completed;

        return todo;
      })
      .slice() // make a copy before sorting
      .sort((a, b) => a.title.localeCompare(b.title)),
  [todos, route]
);
```

This will display your todos in alphabetical order by their title.

> Response rendering

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

Participant

Oh, so I can add a sort function that I guess.
It sorts.

00:28:18
- 00:28:21

00:28:53

00:29:05

00:29:14

00:29:15 Well, locale compare locale compare.
00:29:18 I guess it's the alphabetical sorting, so just try to.
00:29:24 Oh, and slice is also required, so I make a copy of it before sorting.
00:29:29 I'm not sure if it's needed, but I guess I'll just try out and see if
00:29:32 it works or not.
00:29:34 > **Editing file** (src/todo/components/main.jsx)
- 00:29:36
00:29:34 **Participant**
Umm.
00:29:43 > **Implementing Copilot code** by copying part of snippet
(src/todo/components/main.jsx)
00:29:46 > **Editing file** (src/todo/components/main.jsx)
- 00:29:48
00:30:01 **Participant**
What was the phrase again?
00:30:03 Uh, the quick 123.
00:30:05 Quick.
00:30:09 Uh.
00:30:21 Oh, no. Actually it does sort.
00:30:24 **Researcher**
Perfect. Thanks.
00:30:26 **Participant**
OK, one more thing, if I don't do slice, I'm just curious what happens.
00:30:28 **Researcher**
Yeah.
00:30:30 Sure.
00:30:34 **Participant**
Again, I'm not entirely sure why it's needed so.
00:30:38 > **Editing file** (src/todo/components/main.jsx)
00:30:45 **Participant**
Actually it doesn't change anything, but I'm not sure why we need this slice
00:30:50 option.
00:30:51 So I will probably just Google that if it's needed or not or other than that
00:30:56 it's done I guess.
00:30:59 **Researcher**
All right, then. I'll send you the other tasks.
00:31:04 Let's see.
00:31:05 We have 15 minutes left, but again, no rush. You're doing great.
00:31:07 **Participant**
OK.
00:31:11 **Researcher**
If I can find the teams chat, that would be great.
00:31:15 There we go.
00:31:17 And again, really helpful how you're thinking aloud.
00:31:20 This is exactly what we're looking for.
00:31:22 So you're doing great.
00:31:23 **Participant**
OK.
00:31:23 That's good.
00:32:03 OK.
00:32:03 This is the area which I'm not really familiar with.
00:32:29 > **Viewing file** (src/todo/app.css)
00:32:31 > **Viewing file** (src/todo/components/main.jsx)
00:32:33 > **Viewing file** (src/todo/components/footer.jsx)
00:32:36 > **Viewing file** (src/todo/components/main.jsx)
00:32:36 **Participant**

00:32:40 Yep. So I think in this case I'll just ask
00:32:44 Copilot on.
00:32:47 Well, which part of the files are responsible
00:32:52 for the page refresh? Because I don't have that much time to
00:32:58 actually go and search for it myself due to the time constraints I guess. So yeah,
00:33:01 I'll ask.
00:33:04 And again the workspace tab because, sorry
00:33:06 the command in the beginning because it needs to access all the files.
Chat (window: 0)
> Agent
github.copilot.workspace
> Context
workspace, viewport (src/todo/components/main.jsx)
> Message
@workspace Which part of the project is responsible for saving the state of todos-
when page is refreshed?
> Response
The logic for saving the state of todos when the page is refreshed is handled in
[src/todo/reducer.js](#):

- The function [saveState \(src/todo/reducer.js\)](#) saves the todos array to [localStorage \(node_modules/@types/node/globals.d.ts\)](#) under the key "todos" every time the reducer runs.
- The function [loadState \(src/todo/reducer.js\)](#) loads the todos from [localStorage \(node_modules/@types/node/globals.d.ts\)](#) (although it uses the key "todoS", which is likely a typo and should match "todos").

These functions are used in the [todoReducer \(src/todo/reducer.js\)](#) to persist and restore the todo list state across page refreshes.
> Response rendering
Participant
OK.
00:33:46 So it's a reducer.
00:33:49 > Viewing file (src/todo/reducer.js)
Participant
That I have seen a little bit before.
00:33:52 So initial state is load state.
00:33:57 State.
00:33:58 Save state. Oh, I see.
00:34:04 > Editing file (src/todo/reducer.js)
Participant
I see.
00:34:05 So there was a typo here.
00:34:07 So I think that if the.
00:34:12 If the set item and get item are with the same name then in this case.
00:34:18 It should theoretically work.
00:34:21 And I see that there is like a load state function and.
00:34:25 Actually I didn't finish reading the response and now I see that it says that
00:34:29 there was a typo.
00:34:33 Yeah. OK.
00:34:33 So theoretically, now it should no, no.
00:34:39 The.
00:34:40 Brown.
00:34:42 I wonder if it will work.
00:34:45 OK.
00:34:45 Maybe I should read the response then.
00:34:50 The key todos every time.

00:34:59
00:35:09
00:35:26
00:35:30
00:35:36
00:35:36
00:35:39
00:36:16
00:36:21
00:36:23
00:36:27
00:36:30
00:36:37
00:36:41
00:37:16

OK.

For some reason it didn't really work how I hoped it will.

Yeah. So in this case, I think I will need a little bit more assistant assistance.

> **Editing file** (src/todo/reducer.js)

Participant

Oh.

I found more typos.

So I think that I will just ask.

In this case, chat.

Sorry not ChatGPT, Copilot.

What exactly I can change?

For loading and saving the state to work.

With the current state, because I changed a couple of things here and there, like a couple of typos.

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

viewport (src/todo/reducer.js)

> **Message**

Saving the state on page update still doesn't work. What could be the possible remaining issue?

> **Response**

Your code for saving and loading todos from [localStorage \(node_modules/@types/node/globals.d.ts\)](#) is correct. However, a common issue is **server-side rendering or running in an environment where [localStorage \(node_modules/@types/node/globals.d.ts\)](#) is not available**, such as during the initial render in some React setups, or if your code is running outside the browser (e.g., in Node.js or a test environment).

Possible issues to check:

1. **Are you using React's `useReducer` with [initialState \(src/todo/reducer.js\)](#) from [loadState\(\) \(src/todo/reducer.js\)](#)?**

Make sure your main component initializes state like this:

```
const [todos, dispatch] = useReducer(todoReducer, initialState);
```

If you use `useReducer(todoReducer, [])`, it will always start empty.

2. **Is your app running in the browser?**
[localStorage \(node_modules/@types/node/globals.d.ts\)](#) only works in the browser. If you run your code in a Node.js environment, it will fail silently.
3. **Are you clearing [localStorage \(node_modules/@types/node/globals.d.ts\)](#) elsewhere?**
Make sure no other code is removing or overwriting the "todos" key.
4. **Are you using the correct key?**
Both [loadState \(src/todo/reducer.js\)](#) and [saveState \(src/todo/reducer.js\)](#) use "todos", which is correct.
5. **Are you using strict mode or hot reloading?**
In React Strict Mode, reducers may run twice in development, but this should not affect persistence.

How to debug:

- Open your browser's DevTools → Application → Local Storage and check if the "todos" key is being set and updated as you add/remove todos.

- Add a `console.log` in [saveState \(src/todo/reducer.js\)](#) and [loadState \(src/todo/reducer.js\)](#) to verify they are called as expected.
-

Example of correct usage in your main component:

```
File: src/index.js
import React, { useReducer } from "react";
import { todoReducer, initialState } from "../todo/reducer";

function App() {
  const [todos, dispatch] = useReducer(todoReducer, initialState);
  // ...
}
```

If you are already doing this and it still doesn't work, please check for errors in the browser console related to [localStorage \(node_modules/@types/node/globals.d.ts\)](#).

00:37:17
- 00:37:26

> Response rendering

00:37:50

> Viewing file (src/todo/components/main.jsx)

00:38:09

Participant

00:38:11

OK, in this case because I didn't use the workspace tab, it doesn't know what exactly is going out

00:38:14

> Viewing file (src/todo/reducer.js)

00:38:15

Participant

outside of the reducer dot JS. So I think in this case I should just re

00:38:16

> Viewing file (src/todo/components/main.jsx)

00:38:20

Participant

ask a question and ask, add the workspace.

00:38:25

Whatever this is.

00:38:26

Chat (window: 0)

> Agent

github.copilot.workspace

> Context

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

> Message

@workspace Saving the state on page update still doesn't work. What could be the possible remaining issue?

> Response

The issue is that in [src/todo/app.jsx](#), you are not initializing the [useReducer \(node_modules/@types/react/index.d.ts\)](#) hook with the persisted state from [localStorage \(node_modules/@types/node/globals.d.ts\)](#). You are using:

```
const [todos, dispatch] = useReducer(todoReducer, []);
```

This always starts with an empty array, so even though your reducer saves and loads state, the initial state is not loaded from storage.

Solution:

Initialize [useReducer \(node_modules/@types/react/index.d.ts\)](#) with the [initialState \(src/todo/reducer.js\)](#) exported from [src/todo/reducer.js](#):

```
File: src/todo/app.jsx
import { todoReducer, initialState } from "../reducer";
// ...existing code...
export function App() {
  const [todos, dispatch] = useReducer(todoReducer, initialState);
  // ...existing code...
}
```

This will ensure your todos are loaded from [localStorage \(node_modules/@types/node/globals.d.ts\)](#) on page refresh.

00:38:27

Participant

At workspace.

00:38:28
- 00:38:30

> Response rendering

00:38:30

Participant

So I think I'll just repost this question basically.

00:38:40

> Viewing file (src/todo/app.jsx)

00:39:04

Participant

Yeah. So it says that this always starts with

00:39:07

an empty array. The way we currently initialize todos.

00:39:10

So instead of that, I just need to use something different.

00:39:14

Which yeah, I guess we need to add this initial state

00:39:20

command as an import to this file as well from reducer and then use this.

00:39:20

> Editing file (src/todo/app.jsx)

00:39:28

Participant

Initial state instead of.

00:39:30

> Editing file (src/todo/app.jsx)

00:39:32

Participant

Empty array, empty list. So in this case, yeah it will.

00:39:39

Load the initial state which we saved before the page update.

00:39:42

I'm sorry. Yeah, which we save every time

00:39:44

I guess the list changes or something. I'm not sure how exactly this code works,

00:39:48

but I think it just like saves the state every now and then.

00:39:53

Yeah, cool.

00:39:54

Now there are my items.

00:39:57

So I think this task is done.

00:40:00

Do we still have time for the next one?

00:40:04 **Researcher**
Let's see, we have 6 minutes.

00:40:07 **Participant**
OK.

00:40:07 **Researcher**
So yeah, if you want you can just start trying to do it at least.

00:40:11

00:40:13 **Participant**
Mm hmm.

00:40:18 OK.

00:40:18 So basically we need to do a similar thing,

00:40:21 but now we need to save whatever the user was typing in the input field.

00:40:24 **> Viewing file** (src/todo/components/input.jsx)

00:40:25 **Participant**
So I guess it should be somehow connected to the input dot JSX and there's like this input there's label.

00:40:33 I'm not sure where it's saved, but maybe it's also.

00:40:35

00:40:39 **> Viewing file** (src/todo/constants.js)

00:40:40 **> Viewing file** (src/todo/reducer.js)

00:40:41 **Participant**
It might be something to do with the reducer.

00:40:47 Maybe not.

00:40:49 But I'll just see.

00:40:52 OK.

00:40:53 Probably not, because reducer is responsible for all

00:40:55 **> Viewing file** (src/index.js)

00:40:56 **Participant**
the todos in.

00:40:57 **> Viewing file** (src/todo/reducer.js)

00:40:58 **Participant**
The list, it seems like it.

00:41:00 **> Viewing file** (src/todo/constants.js)

00:41:00 **Participant**
Constants are.

00:41:07 Yeah.

00:41:08 I don't know why.

00:41:11 **> Viewing file** (src/todo/app.jsx)

00:41:11 **Participant**
Add item.

00:41:12 **> Viewing file** (src/todo/reducer.js)

00:41:15 **Participant**
Add item.

00:41:24 **> Viewing file** (src/todo/components/main.jsx)

00:41:24 **> Viewing file** (src/todo/components/item.jsx)

00:41:28 **Participant**
OK.

00:41:29 **> Viewing file** (src/todo/components/input.jsx)

00:41:31 **Participant**
In this case, I'll just.

00:41:37 Ask I guess I.

00:41:40 Enter task here.

00:41:44 Yeah. Well, I guess I'll just use some help if I have this opportunity.

00:41:48

00:42:11 When the page is.

Chat (window: 0)**> Agent**

github.copilot.workspace

> Context

workspace, viewport (src/todo/components/input.jsx)

> Message

@workspace I would like to preserve the state of the input field when the page is reloaded. How would it be possible to do?

> Response

To preserve the state of the input field when the page is reloaded, you need to:

1. Store the input value in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) on every change.
2. Initialize the input field with the value from [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) when the component mounts.

Here's how you can update your [Input](#) ([src/todo/components/input.jsx](#)) component:

```
File: src/todo/components/input.jsx
import { useCallback, useState, useEffect } from "react";

const hasValidMin = (value, min) => {
  return value.length >= min;
};

export function Input({ onSubmit, placeholder, label, defaultValue,
onBlur }) {
  const [inputValue, setInputValue] = useState(() => {
    return localStorage.getItem("inputValue") || defaultValue ||
    "";
  });

  useEffect(() => {
    localStorage.setItem("inputValue", inputValue);
  }, [inputValue]);

  const handleBlur = useCallback(() => {
    if (onBlur)
      onBlur();
  }, [onBlur]);

  const handleKeyDown = useCallback(
    (e) => {
      if (e.key === "Enter") {
        const value = inputValue.trim();

        if (!hasValidMin(value, 2))
          return;

        onSubmit(value);
        setInputValue("");
        localStorage.removeItem("inputValue");
      }
    },
    [onSubmit, inputValue]
  );

  const handleChange = useCallback((e) => {
    setInputValue(e.target.value);
  }, []);

  return (
```

```

    <div className="input-container">
      <input
        className="new-todo"
        id="todo-input"
        type="text"
        data-testid="text-input"
        autoFocus
        placeholder={placeholder}
        value={inputValue}
        onChange={handleChange}
        onBlur={handleBlur}
        onKeyDown={handleKeyDown}
      />
      <label className="visually-hidden" htmlFor="todo-input">
        {label}
      </label>
    </div>
  );
}

```

This will keep the input value in sync with [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) and restore it after a page reload.

00:42:36
- 00:42:49

> Response rendering

00:42:56

Participant

Yep. So it suggests to use the similar thing to the reducer where I use the local storage with I guess some input value as an ID.

00:42:59

00:43:05

00:43:06

00:43:11

00:43:15

00:43:19

00:43:22

00:43:25

00:43:25

00:43:26

It can be anything else I guess as well. So it just suggests to change the input component which is exactly here, to do that.

Umm.

And I need to add use state and use effect.

I don't.

I'm not sure about the effect, but I guess the state is like exactly

> Implementing Copilot code by copying part of snippet (src/todo/components/input.jsx)

00:43:29

Participant

what I'm going to be using for saving it.

00:43:40

00:43:43

00:43:46

Yep, so.

I'll use this constant.

> Implementing Copilot code by copying part of snippet (src/todo/components/input.jsx)

00:43:52

> Implementing Copilot code by copying part of snippet (src/todo/components/input.jsx)

00:43:56

Participant

Handle blur, don't need that.

00:44:04

> Implementing Copilot code by copying part of snippet (src/todo/components/input.jsx)

00:44:06

Participant

I need to change this thing and then.

00:44:12

00:44:13

Yeah, I guess I'm running low on time.

> Implementing Copilot code by copying part of snippet (src/todo/components/input.jsx)

00:44:17

> Implementing Copilot code by copying part of snippet (src/todo/components/input.jsx)

00:44:25

Researcher

If you want to continue until you're done, then that's fine as well.

00:44:26

> Implementing Copilot code by copying part of snippet (src/todo/components/input.jsx)

00:44:28

Participant

I want to finish this part at least to see if it actually works.

00:44:29

Researcher

OK.

00:44:32

Great. So no hurry.

00:45:12

> Implementing Copilot code by copying part of snippet (src/todo/components/input.jsx)

00:45:39

Participant

Damn it.

00:45:56

OK.

00:45:56

No, I just needed to save the file.

00:45:59

I forgot to save the file.

00:46:00

That was the problem. So yeah.

00:46:03

I don't have time to actually go over it, but also what I will do now,

00:46:07

just try to understand what exactly I did because currently because of lack of time

00:46:13

I just copy pasted it, but I usually also like myself try to

00:46:16

understand what exactly I copy pasted.

00:46:19

I just don't blindly copy paste the code because all this neural networks they can

00:46:24

hallucinate sometimes.

00:46:26

And just like give insecure code as in like something that is working but might

00:46:34

like also add some threats.

00:46:38

To the code, or I would.

00:46:39

Researcher

Alright.

00:46:40

Participant

Yeah, just double check that everything works

00:46:43

fine and that there is no some like weird security issues because of that.

00:46:51

Researcher

That makes sense. Thanks a lot.

00:48:19

Participant

OK.

00:48:21

Uh.

00:48:33

I don't know how to evaluate the length of the response.

00:48:38

It was fine.

00:48:43

Actually, half of the times I just went for the

00:48:45

solution. I didn't really read the chat that much.

00:49:02

OK.

00:49:29

I guess it was good.

00:49:31

I think on average it was pretty good and some things are I really enjoyed,

00:49:35

like all technical terms, I just looked at some things that like

00:49:40

server side rendering.

00:49:43

And like environment and just, yeah, I guess the terms were good.

00:49:48

Oh, sorry.

00:49:50

And the formatting is like quite visible. So you kind of see different parts of the

00:49:55

code which was also like pretty good. The other things, yeah.

00:49:58

Just.

00:50:00

Good responses.

00:50:01

The correctness I think it was good.

00:50:04

I put neutral just because for like the correctness,

00:50:07

because sometimes I ask the question and I kind of expect it to remember the

00:50:12

previous question I asked, but the memory was just gone.

00:50:16

So I asked the question how to solve the previous problem,

00:50:20

but I just said to me how to solve all the problems with the whole workspace and

00:50:25

that was kind of annoying.

00:50:28

Researcher

Yeah.

00:50:30

Participant

Because, yeah, I'm kind of like I'm using mostly ChatGPT

00:50:33

and it kind of remembers the previous conversation like at least to some degree.

00:50:40

And here it just didn't remember.

00:50:41

So yeah, I expected it to remember, but it didn't.

00:50:43

So that was a bit of a bummer.

00:50:47

But yeah, I think that, oh, there's more.

00:50:53 **Researcher**
There, there's some more. Yeah, sorry.

00:50:56 **Participant**
Yeah, no worries.

00:51:33 Also, by performance in the second question,
00:51:36 do you mean like how fast I am able to finish the tasks?

00:51:42 **Researcher**
So this is in in real life, not persé in the in the in the study.
00:51:48 So yeah, performance.
00:51:50 Well, whatever, you would interpret performance as.

00:51:54 **Participant**
Yeah, because like, well, my personal opinion,
00:51:57 it's good to speed up the tasks.
00:51:59 But you don't really learn from it that much.
00:52:02 So I will just put it slightly disagree just simply because I think that you will
00:52:07 learn less by.
00:52:09 Doing things faster.
00:52:35 Hmm.
00:52:50 I'll do that actually.
00:54:36 I don't know.
00:54:37 I'll just put it into the neutral.
00:55:29 Yep, I guess something like that.

00:55:35 **Researcher**
All right, this is the very last question.

00:55:38 **Participant**
OK.
00:55:55 I didn't reach the task C, so I assume it's did not attempt.
00:56:16 I guess they were both moderate because I didn't know exactly how to do them,
00:56:20 but with the help I managed to do them.
00:56:22 So it's kind of like not very difficult, but also not very easy because I didn't
00:56:26 instantly come up with the.
00:56:28 Uh solution, but also had to like a little bit search
00:56:33 for it.
00:56:41 I think they're both slightly realistic because if you're just like in the
00:56:44 beginning stage of developing something like that and you probably will have to
00:56:47 do these tasks.
00:56:49 Like of sorting alphabetically or whatnot for the list.
00:56:54 And well, sorry saving.
00:56:57 The.
00:56:58 Todos, so it was realistic.
00:57:01 Yeah. OK.

00:57:03 **Researcher**
All right. Thanks.
00:57:05 So then I have some questions for you.

00:57:08 **Participant**
OK.

00:57:10 **Researcher**
First of all, could you find your way?
00:57:12 So navigate through the codebase, and how did Copilot influence this?

00:57:21 **Participant**
Well, I think that I just started with just
00:57:23 looking at different parts of the code.
00:57:25 What the codebase consists of, but with the Copilot it became.
00:57:31 Much more like easy because you can just at put the at workspace and then it
00:57:36 shows you the exact location of the exact thing that you need to change depending
00:57:42 on the task that you are doing. So in this case, well,

00:57:45 I didn't even have to think but.
00:57:47 Meanwhile, when I try to find some places myself and
00:57:51 I got a little bit lost.
00:57:54 **Researcher**
OK.
00:57:56 **Participant**
Because I'm not familiar with this code, I saw it the first time and I don't
00:58:00 really do the React application.
00:58:01 So in this case, yeah, I didn't exactly know where to search,
00:58:05 but with the Copilot it is easy.
00:58:08 **Researcher**
OK. And is that a good thing?
00:58:12 **Participant**
In a sense, as like I said before that.
00:58:20 The productivity increases, but like I don't really learn much from
00:58:23 it.
00:58:25 Because for example, for the last task I just changed the
00:58:28 necessary things.
00:58:29 So the solution works, but I don't know how it works.
00:58:32 So it really is your also responsibility to learn from it.
00:58:37 **Researcher**
All right.
00:58:38 And you feel like this, this learning is essential then.
00:58:43 **Participant**
I think so.
00:58:45 Because if I want to become a good specialist,
00:58:45 **Researcher**
OK.
00:58:48 **Participant**
I also need to know what I'm doing.
00:58:51 **Researcher**
That's fair.
00:58:51 That's fair.
00:58:53 So on a related note, do you feel like you understood the
00:58:56 codebase and how did Copilot influence that?
00:59:02 **Participant**
I think I understood it a little bit more. If I actually just try different things
00:59:08 myself, but because of limited time I didn't
00:59:11 really have had an opportunity to do that.
00:59:15 It helped me to like navigate initially quicker and maybe.
00:59:22 Yeah, it like really helped with.
00:59:25 Just reaching the solution.
00:59:27 But I think the additional work that I put I I basically.
00:59:31 Put initial, sorry.
00:59:37 Yeah, I try to put work in myself in order to
00:59:41 understand it, not just do the solution.
00:59:46 **Researcher**
And why would you want to.
00:59:46 **Participant**
What was the question again? Sorry.
00:59:50 **Researcher**
If Copilot helped understanding the code?
00:59:54 **Participant**
Uh.
00:59:55 Well, yeah. Also, if I read the answers then maybe I would
00:59:59 have understand it a bit better, but because I am really like seeing the

01:00:03 solution based in it and trying to understand myself then not really.
01:00:08 But if someone like me or someone else will read it and.
01:00:15 Yeah. Then it helps.
01:00:16 **Researcher**
And and so why did you prefer to try to understand the solution yourself over
01:00:22 reading the the answer?
01:00:27 **Participant**
I don't know, it's just the way I used to. Probably before, because yeah.
01:00:32 I'm in the computing science world since 2017.
01:00:36 And there was no such things as AI chatbots.
01:00:39 So in this case I'm just kind of like, I don't know, old school a bit, maybe.
01:00:42 So I'm trying to just figure out myself.
01:00:44 **Researcher**
Fair enough.
01:00:45 **Participant**
It's like a new thing that I'm not yet used to,
01:00:48 so that's why I'm still in the mindset of trying to just understand myself or maybe
01:00:52 like searching for the answer on Stack Overflow or something like that.
01:00:57 If I have an opportunity.
01:00:59 **Researcher**
So how would your strategy be to solve these tasks without using Copilot?
01:01:04 And how did using Copilot change your strategy?
01:01:10 **Participant**
I think I would just like search the web and see what other people did and try to.
01:01:17 Implement something similar.
01:01:20 In the code that I was given.
01:01:22 And I think Copilot just speeds up this process by a lot.
01:01:30 I still have like a doubt in my head about whether it's producing the correct
01:01:35 code because it's always also important to remember that.
01:01:42 But yeah, because it was the only tool available.
01:01:45 Then I would of course use that as my search search engine basically.
01:01:50 **Researcher**
And so you say.
01:01:52 You doubt if it was producing the correct code?
01:01:55 **Participant**
Well, for this easy task it probably did.
01:01:58 But like if I want to do some more complicated tasks then or maybe use the
01:02:05 language that?
01:02:08 Is not that commonly used. For example, I did my master's thesis with.
01:02:15 [omitted] and it's not very widely used,
01:02:18 and in this case I could not use the chatbots to help me with that,
01:02:23 because they just don't have enough resources to actually base their answer on.
01:02:28 In this case. Yeah, like it might not really help,
01:02:32 but reaching the.
01:02:34 Solution.
01:02:36 **Researcher**
And is it then not useful, just code generation or for other aspects
01:02:40 aspects as well?
01:02:43 **Participant**
No, it's useful for other aspects, but it might not be as useful for a code
01:02:47 generation in this cases that I gave example but.
01:02:53 Yeah.
01:02:56 **Researcher**
And so in in this, in this case in these experiments.
01:03:01 Did you feel like Copilot maybe was producing correct code?
01:03:05 Incorrect code or how did you verify that the code was correct?

01:03:07 **Participant**
Oh.

01:03:12 Well, I tried to just.

01:03:14 See if it works.

01:03:17 Kind of like test it out and in this case it produced the correct code of course,

01:03:21 because yeah, it's also.

01:03:24 Easy tasks that it is able to easily solve.

01:03:29 Yeah.

01:03:32 **Researcher**
Would you do the same with code that you.

01:03:34 For example, get from Stack Overflow?

01:03:38 **Participant**
Probably you mean the copy paste and check if it works?

01:03:41 **Researcher**
Yeah.

01:03:43 **Participant**
Yeah.

01:03:45 **Researcher**
Makes sense.

01:03:49 Yeah. So at some point in practice task C.

01:03:55 There was this slice function that you thought might not be necessary.

01:03:59 **Participant**
Yes.

01:04:01 **Researcher**
So how would you usually approach this?

01:04:04 See if it's is necessary or not.

01:04:09 **Participant**
I would probably check documentation.

01:04:12 **Researcher**
Mm hmm.

01:04:12 **Participant**
And like, see what exactly it does.

01:04:16 And just see if I need it or not in this case.

01:04:20 So just check the documentation probably.

01:04:24 **Researcher**
So try to understand.

01:04:24 **Participant**
Or maybe someone also asked the same question on Stack Overflow and people who know JavaScript better also answered if, like what exactly does and if it's needed or not.

01:04:29 Because I think Copilot if just gave me the solution and said that make a copy

01:04:34 before sorting.

01:04:35 But I didn't really explain and I also didn't ask for explanation.

01:04:41 So also a possible way is to ask what the slice function does.

01:04:44 But I don't because I am a bit old school. As I said,

01:04:48 I would also check documentation because it's probably documented in the yeah

01:04:53 JavaScript manual over there.

01:05:02 **Researcher**
So about these explanations you said you didn't ask for it,

01:05:05 so would you have preferred if the explanation wasn't there?

01:05:08 **Participant**
I think it would be nice, yeah, because currently well,

01:05:13 I understand what sort function does, but I'm not sure if slice is needed

01:05:17 because it also just added it to the code, but it didn't explain what it does.

01:05:21 And in this case I had this doubt whether I need it for my problem or not.

01:05:27 And it worked without it for some reason.

01:05:34 **Researcher**
Right.

01:05:38 So you would have liked an explanation on why it was included.

01:05:41 **Participant**
Yes.

01:05:43 **Researcher**
And then I think at some point you said.

01:05:48 You kind of only looked at the solutions and not so much at the explanations.

01:05:53 In this case. Would you have gone through the

01:05:56 explanation to see?

01:05:58 **Participant**
If it was there, probably because, yeah, this was just a questionable thing.

01:06:04 As in like I was not sure what exactly it does and in this case it would have been

01:06:10 nice to have it at least.

01:06:12 **Researcher**
Right. And then would you prefer if this was

01:06:15 already there or having to ask, ask again?

01:06:21 **Participant**
I think in this case.

01:06:24 I actually don't.

01:06:28 I cannot answer this question.

01:06:30 I'm not sure because then it makes the whole text much bigger.

01:06:31 **Researcher**
OK.

01:06:34 **Participant**
If it also explains every single function that is added.

01:06:38 And if I look at my request I just ask for.

01:06:45 Umm.

01:06:48 Yeah. I asked.

01:06:49 What script is responsible for the order of the todos that I have to change and

01:06:54 it just said that I need to go here and change this script and it answered

01:06:58 exactly what I asked.

01:07:02 So I guess in this case like extra prompt would be.

01:07:08 Fine.

01:07:09 **Researcher**
OK.

01:07:11 **Participant**
Because yeah, the chat just replies what you ask it and

01:07:14 it replied with exactly what I asked it.

01:07:18 **Researcher**
And I see in this case you asked specifically for the piece of code that's

01:07:22 responsible for ordering, and then it actually also gave.

01:07:24 **Participant**
Yep.

01:07:27 **Researcher**
A sort of already a solution to the to the task itself.

01:07:31 **Participant**
Yes.

01:07:33 **Researcher**
Is this what you intended for it to do or is this something that in hindsight you

01:07:38 like that it did?

01:07:41 **Participant**
I think it was in the hindsight, but I also kind of expect it as well.

01:07:47 I think I yeah.

01:07:50 Because I'm describing the problem and that I need a solution for this problem

01:07:54 and in this case it gave me the solution.

01:07:55 **Researcher**
Mm hmm.

01:07:57 Fair enough.

01:08:00 Well, that makes sense.

01:08:00 **Participant**
Because I said I'm trying to order my to do's by the alphabetical order,
01:08:05 so it also provided the well it said that this script is responsible and here is
01:08:10 the how you can change it in order to do the todos in the alphabetical order,
01:08:16 **Researcher**
So that's sort of a logical follow up for you.

01:08:16 **Participant**
yeah.

01:08:16 So I I guess yes, but it's very like hidden there.

01:08:23 **Researcher**
That's fair enough though.

01:08:28 So.

01:08:30 At some point I think for task this was for task B.

01:08:35 **Participant**
Mm hmm.

01:08:36 **Researcher**
Returned a rather large code snippet.

01:08:40 And I saw you.

01:08:43 Manually going through it.

01:08:45 So was it for you?

01:08:46 Easy to see.

01:08:49 What changes Copilot had made?

01:08:55 **Participant**
Well, I think so because it used the same
01:08:58 structure as the code, it just added extra lines.

01:09:07 Yeah, I think that like all these separate
01:09:09 functions that it added like use effect and.

01:09:14 I think it also added the handle change.

01:09:19 Yeah. So I could easily see them, but it also.

01:09:24 This input HTML tag.

01:09:28 It's changed it from being inline to like being multiline.

01:09:33 HTML tag in this case it was a bit hard to scroll through,
01:09:36 but at least it didn't change.

01:09:38 Any order so I could still find the locations where I need to add or remove
01:09:44 certain things.

01:09:48 But yeah, like this one was the only part where I
01:09:51 well struggled a little bit.

01:09:53 But because the order was still the same, it was still possible to change the
01:09:54 **Researcher**
Fair enough.

01:09:58 **Participant**
correct things.

01:10:00 **Researcher**
And what made you want to implement the individual changes rather than,
01:10:05 for example, just copying and pasting the entire thing?

01:10:11 **Participant**
Because then I kind of can control what exactly I'm changing.

01:10:16 So.

01:10:19 Even though I didn't, I barely had time to process the things
01:10:24 that it suggested to me.

01:10:25 I still in my head kind of looked at them at least,
01:10:29 like glanced at them and saw that, oh, yeah, this kind of does this probably.

01:10:35 Like this handle change sets the input value.
01:10:39 So I need this part of code and also for example in the input.
01:10:44 I noticed that I had to remove.
01:10:49 One.
01:10:51 Like keyword with the value it was like a default value or something like that.
01:10:56 That I had to remove and replace with.
01:11:00 The input value that we get from the top, so it's kind of like also helped me to
01:11:05 understand what exactly I'm doing, what exactly I'm changing,
01:11:08 **Researcher**
OK.
01:11:09 **Participant**
not just like blindly copy pasting the code without actually understanding what
01:11:14 I'm doing.
01:11:15 **Researcher**
That makes sense.
01:11:16 **Participant**
So it's just like kind of like a learning process.
01:11:19 **Researcher**
OK, so I think you said that using Copilot
01:11:23 will make you faster, but not.
01:11:28 Learn and understand as well.
01:11:30 Do you feel like it is possible?
01:11:32 Like you're doing now to sort of use Copilot in a way that still allows
01:11:37 for this learning and understanding.
01:11:39 Or will it always be less than doing it manually?
01:11:44 **Participant**
I think it still allows for learning and understanding,
01:11:48 but it's also an effort that I choose to do.
01:11:51 It's mainly because like I see some people use like ChatGPT or Copilot code.
01:11:57 Just like copy paste and then they don't learn from it.
01:12:01 And like I I I myself made a decision that even if I use I will still go over
01:12:07 the things that.
01:12:09 Generative AI returns to me.
01:12:13 So kind of like, yeah, it's really a personal choice.
01:12:17 I.
01:12:17 I I love learning so I try to understand the things that I'm doing.
01:12:25 But yeah, even in the small exercise I was kind of
01:12:26 **Researcher**
Yeah.
01:12:28 **Participant**
like, oh, I have not really worked with JavaScript
01:12:31 that much.
01:12:32 So let me see what I can learn from it.
01:12:35 **Researcher**
Fair enough.
01:12:37 So you talked about the terms that it used as well.
01:12:40 You mentioned for example server side rendering.
01:12:44 So were these terms familiar to you already or not yet?
01:12:48 **Participant**
Yes, they were familiar.
01:12:52 Yeah.
01:12:53 **Researcher**
So the fact that it used these words.
01:12:58 Was was nice in your case.
01:13:02 **Participant**
I guess so because like I'm very bad with the terms myself,

01:13:05 so it's nice to see how the things are actually properly called.

01:13:11 **Researcher**
OK.

01:13:12 **Participant**
Because I'm horrible with the terms, I even don't remember like how this at
01:13:17 workspace is called like I don't know like the tag, tab, whatever.

01:13:22 **Researcher**
Mm hmm.

01:13:22 **Participant**
So yeah, it's like nice to see how the things are
01:13:25 properly called.

01:13:26 **Researcher**
So for you, seeing these terms being used in context
01:13:29 allows you to sort of be like ah, this is what they mean.

01:13:33 **Participant**
Yes. Yeah.

01:13:37 **Researcher**
OK.

01:13:38 I have two more questions.

01:13:40 Sorry if this is taking so long.

01:13:41 Oh sorry, I have 3 actually.

01:13:42 **Participant**
No, no worries.

01:13:44 **Researcher**
First of all, did you change the way you interacted
01:13:48 with Copilot over time?

01:13:51 **Participant**
I think so because for the first task like the practice task that we did,
01:13:56 I kind of wanted to do everything myself.

01:13:59 But then when we reached the second PDF that you sent to me like the task A, B.
01:14:05 I realized that without internet it's very hard to do this.

01:14:10 Like, they're pretty simple, but because of my lack of knowledge of
01:14:14 JavaScript, they're very hard for me.

01:14:16 So in this case I started using chat more and more just simply because.
01:14:23 I wouldn't solve the problems without using.

01:14:29 The Copilot. Like, I would spend the whole session on,
01:14:32 just like one thing. And in this case I wanted to actually
01:14:35 just try a bit more.

01:14:37 **Researcher**
And for example, when searching the web, how what kind of things would you look
01:14:41 for when in a normal situation trying to solve this?

01:14:43 **Participant**
Umm.

01:14:50 Yes, I think that like I will firstly try to
01:14:53 identify the location where well. For example like this list sorting task.

01:14:58 This is where I actually started to realize that I actually need some help.
01:15:04 I would firstly find the place where the list is created.

01:15:10 And then I will probably just Google how to sort this type of list in JavaScript.
01:15:18 And then just like, see if the suggested solutions work or
01:15:21 not.

01:15:22 **Researcher**
OK. And and.

01:15:27 Do you maybe know why?

01:15:27 Or could you explain why, for example, you didn't use Copilot exactly the same
01:15:32 as you would use Google for example.

01:15:41 **Participant**

I think because.

01:15:43 Copilot also has access to my code and that's the big difference because it can
01:15:49 actually suggest a solution based on the code that it sees. Meanwhile, in Google,
01:15:55 while there is like millions of projects or whatever written in JavaScript and in
01:16:01 this case it will not find the
01:16:03 exact, like nobody else did the exact same thing.
01:16:08 So it's impossible to find the exact solution that I need.
01:16:13 Meanwhile, with Copilot, it just has like this knowledge,
01:16:17 to actually understand what is wrong with my code and where exactly I should
01:16:22 change it if it makes sense.

01:16:24 **Researcher**

Yeah, sure.

01:16:25 So do you.

01:16:26 Do you prefer using Copilot then over Google?

01:16:30 Or does Google using Google still have some edge over it?

01:16:37 **Participant**

Well, it was my first time using Copilot, so.

01:16:42 I don't know.

01:16:42 Like in, in just everyday I just use Google.

01:16:46 **Researcher**

That's fair.

01:16:48 That's fair.

01:16:48 **Participant**

Also use ChatGPT, but that's as I answered to my like pre
research questionnaire that you sent out.

01:16:52 Is that?

01:16:56 I mainly use it for text editing or maybe for solving the issues with which I have
01:17:00 been stuck for awhile.

01:17:03 **Researcher**

So for more general guidance, maybe.

01:17:03 **Participant**

But I don't really.

01:17:06 Yes, probably.

01:17:08 **Researcher**

That makes sense.

01:17:09 **Participant**

Or like the text editing, because I write horribly in English.

01:17:15 **Researcher**

Yeah, I think we can all relate to that.

01:17:20 In the very first was this the very first?

01:17:26 Yes, in the very first question to Copilot,
01:17:30 you said.

01:17:39 I'm not sure what exactly you asked for, maybe asked to explain something and then
01:17:44 you included.

01:17:46 Oh, because I don't know much about React yet.

01:17:50 So what made you include that part?

01:17:56 **Participant**

Oh.

01:18:00 You mean in the question that I ask from chat or from our talk?

01:18:05 **Researcher**

In the yeah. In the chat. So maybe could you go to the very first

01:18:10 **Participant**

Yeah, I can scroll up.

01:18:10 **Researcher**

question?

01:18:17 So you say I've never worked with this project before.

01:18:23 So what?
01:18:23 What made you include this part?
01:18:24 **Participant**
I don't know.
01:18:24 Maybe it's just an excuse. Like, oh, I'm stupid.
01:18:28 I don't know how it works.
01:18:29 Can you please give me an answer?
01:18:30 I don't know. It's just.
01:18:31 **Researcher**
Yeah.
01:18:34 **Participant**
I know that these chatbots they're just machines basically,
01:18:38 but it's kind of like I guess nice to have just like a human conversation with
01:18:44 them.
01:18:45 **Researcher**
That's that's very fair.
01:18:45 **Participant**
As in like I don't know, it's just maybe like me imagining that
01:18:49 there is like some other like.
01:18:52 Being with feelings on the other side, but I understand it's a machine,
01:18:55 so it doesn't care if I'm being polite or just explaining why I cannot compile it
01:18:59 myself.
01:19:01 **Researcher**
So you just like personally to have sort of this, this conversation with it.
01:19:05 **Participant**
Yeah, I guess it's kind of like this is a
01:19:07 conversation I would have with someone who is like my mentor or teacher or
01:19:12 someone who can, like, help me to compile this code.
01:19:17 So I guess I used similar approach to.
01:19:22 Asking the question on how to run this project.
01:19:25 **Researcher**
And do you feel like that does anything with the answers?
01:19:30 **Participant**
No.
01:19:30 **Researcher**
No. All right, that makes sense.
01:19:33 **Participant**
It's the same thing as, for example, I do sometimes,
01:19:36 and I know some friends like if they receive an answer that works,
01:19:39 then they will just be like thanks.
01:19:42 Like you don't have to thank a machine.
01:19:44 It's it's just something maybe, like programmed inside that if you got a
01:19:47 **Researcher**
Yeah.
01:19:48 **Participant**
solution then you just say oh, thank you.
01:19:52 **Researcher**
Fair enough.
01:19:53 I I might have done this occasionally before as well.
01:19:58 OK.
01:19:58 Very last question, I promise. I think you said in the survey that.
01:20:05 Copilot wasn't very flexible or that you wouldn't find it easy to come become
01:20:10 skillful at using Copilot. Can you explain?
01:20:12 Sort of your thoughts behind this.
01:20:16 **Participant**
So it was not flexible.

01:20:22 I think it was mainly because of it not. Oh wait.
01:20:30 I think it was related to the Copilot not remembering. The previous conversation.
01:20:36 Was it?
01:20:37 And I think I also explained a bit that.
01:20:42 I kind of asked one of the question that yeah here can you search the workspace
01:20:47 and suggest the solution to for my problem and then it just went on
01:20:51 everything that is wrong with this project.
01:20:55 Meanwhile, I meant that.
01:20:58 Suggest a solution to the problem I mentioned in the previous prompt because
01:21:03 it was the last thing that we talked about,
01:21:05 but it kind of like didn't really remember.
01:21:09 What we talked about before, even though I described my problem in the
01:21:11 **Researcher**
Yeah.
01:21:13 **Participant**
previous prompt.
01:21:15 But it just, yeah, didn't remember.
01:21:17 And then started here, suggesting everything that can be changed.
01:21:22 **Researcher**
Yeah.
01:21:26 **Participant**
Yeah, it was just like, too much. In this case,
01:21:29 I kind of expected it to have some kind of like, a memory at least, like,
01:21:34 I don't know, like, some prompts behind.
01:21:39 Because as I mentioned, I used ChatGPT sometimes and for example
01:21:43 a couple of times I in the same conversation I asked can you create?
01:21:50 Like or. Can you give some ideas on how I can like?
01:21:55 Combine these paragraphs. Oh no.
01:22:01 How I can summarize the paragraphs that we talked before and then it kind of like
01:22:05 provides an answer like yeah, add this, add that, add this, mention that.
01:22:11 And well, here it just didn't remember.
01:22:14 And I think in this case it would be nice to like have at least some memory of the
01:22:20 previous conversation.
01:22:23 That would add a bit of flexibility that you can just ask.
01:22:26 Oh yeah, this solution still didn't work.
01:22:28 Can you suggest something else? And then it kind of like suggests another
01:22:32 solution to the same problem discussed two problems beforehand.
01:22:37 **Researcher**
That makes sense.
01:22:38 **Participant**
I guess this is like the flexibility that I meant.
01:22:42 **Researcher**
Yeah.
01:22:43 Yeah, that makes sense.
01:22:45 And becoming skillful at using Copilot.
01:22:50 **Participant**
I don't remember what I answered. Did I answer like no or yes.
01:22:54 I think I answered that.
01:22:54 **Researcher**
I I think you said.
01:22:56 Sorry, go ahead.
01:22:57 **Participant**
I think I answered that I will be able to.
01:23:00 **Researcher**
Oh, OK.
01:23:01 **Participant**

Or.

01:23:02

Researcher

So you do feel like you'd be like it'd be quite easy to become skillful at using it.

01:23:09

Participant

Yeah. Yeah, I think so.

01:23:11

Researcher

OK.

01:23:13

Great.

01:23:13

Participant

Because, well, yeah.

01:23:16

Researcher

Is there anything you want to add?

01:23:18

These were my questions, but maybe have something else I think is relevant.

01:23:22

Participant

Umm.

01:23:25

No, I don't think so.