

Participant P05

00:05:18 > **Viewing file** (src/todo/app.jsx)
00:05:28 **Participant**
Let me [unintelligible].
00:05:30 > **Viewing file** (public/index.html)
00:05:43 > **Viewing file** (src/index.js)
00:05:56 > **Viewing file** (src/todo/reducer.js)
00:05:59 > **Viewing file** (src/todo/constants.js)
00:05:59 > **Viewing file** (src/todo/app.css)
00:06:07 > **Viewing file** (public/index.html)
00:06:07 **Participant**
I do believe.
00:06:09 Like in this one but is has to be.
00:06:25 **Researcher**
So what are you thinking right now?
00:06:28 **Participant**
What am I thinking right now that there should be.
00:06:32 Something like this?
00:06:34 So, since there is nothing I have to run, probably the index.
00:06:35 > **Viewing file** (src/index.js)
00:06:39 **Participant**
Mm hmm.
00:06:42 I'm looking for something, but I'm not very sure what I'm looking
00:06:45 for.
00:06:48 If that makes any sense.
00:07:09 OK, let's not.
00:07:11 > **Viewing file** (.eslintrc)
00:07:13 > **Viewing file** (src/index.js)
00:07:26 **Participant**
Put as a node.
00:07:51 I do need to switch on the node, but actually I have no idea how to do
00:07:54 that so.
00:08:02 **Chat** (window: 0)
> **Agent**
github.copilot.workspace
> **Context**
workspace, viewport (src/index.js)
> **Message**
how do I run the app?
> **Response**
To run the app in development mode, use the dev script defined in the [package.json](#) file. Open the integrated terminal in Visual Studio Code and run:

npm run dev

This will start the development server using webpack-dev-server and open the app in your default browser.
00:08:03 **Participant**
First question then.
00:08:14 > **Response rendering**
- 00:08:15
00:08:22 **Participant**
I had I at least I thought about npm install but I don't have to do that.

00:08:56 > **Viewing file** (package.json)
00:09:19 **Researcher**
So what are you trying to do now?
00:09:21 **Participant**
What I'm trying to do now is because we said that we should use Copilot,
00:09:22 **Chat** (window: 0)
> **Agent**
github.copilot.default
> **Context**
viewport (package.json)
> **Message**
> todomvc-react@1.0.0 dev > webpack serve --open --config webpack.dev.js sh: 1:
webpack: not found
> **Response**
The error indicates that the webpack command is not available. This usually happens if the node_modules folder is missing or the dependencies are not installed. To fix this:

1. **Install Dependencies:** Run the following command in the terminal to install all required dependencies:

npm install
2. **Run the App:** After the installation is complete, try running the app again:

npm run dev

If the issue persists, ensure that node and npm versions meet the requirements specified in the [package.json](#) file (node >= 18.13.0 and npm >= 8.19.3). You can check your versions with: ``sh node -v npm -v

00:09:23 > **Response rendering**
- 00:09:31
00:09:25 **Participant**
I want to actually ask the error in Copilot, not Google it.
00:09:29 **Researcher**
Mm hmm.
00:09:31 **Participant**
Yeah OK.
00:09:38 Hey I was right actually.
00:09:41 As I told you, I was thinking about npm install.
00:09:46 And he said to do the run dev.
00:09:49 So I did the run dev, but yeah.
00:10:18 I think I'm. Yeah, I wrote it wrong.
00:10:29 Hey, good.
00:10:33 **Researcher**
So now you can get started on the tasks.
00:10:40 **Participant**
Yes, [unintelligible] this works.
00:10:54 Practice A, change the color when there's no todo.
00:10:57 There's a red box with a message no todos here, change the color. OK.
00:11:02 Actually, I'll run this again.
00:11:11 > **Viewing file** (src/index.js)
00:11:17 **Participant**
Index CSS where are you?
00:11:40 **Researcher**
What are you thinking now?
00:11:42 **Participant**
I am thinking that I want to look for index CSS or I want to look exactly where.

00:11:47 > **Viewing file** (src/todo/components/header.jsx)
00:11:48 > **Viewing file** (src/todo/components/item.jsx)
00:11:51 **Participant**
Yeah, I'm looking for some sort of.
00:11:52 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:11:55 > **Viewing file** (src/todo/components/main.jsx)
00:11:57 **Participant**
Variable that has.
00:12:00 Wait, I'm going here.
00:12:03 A color code or a I'm just looking for a color code or for red somewhere.
00:12:08 > **Viewing file** (public/index.html)
00:12:08 **Researcher**
Right.
00:12:10 **Participant**
Umm.
00:12:14 But I'm not very sure where.
00:12:15 > **Viewing file** (src/index.js)
00:12:17 > **Viewing file** (src/todo/reducer.js)
00:12:20 > **Viewing file** (src/index.js)
00:12:46 **Participant**
Open everything.
00:12:51 That's it.
00:12:53 Yeah.
00:12:54 > **Viewing file** (src/todo/app.css)
00:12:56 **Participant**
App CSS and.
00:13:18 Am I allowed to use this?
00:13:20 **Researcher**
Yes you are.
00:13:22 **Participant**
Great.
00:13:27 Todo list empty message.
00:13:44 Here you are.
00:13:48 OK, I did find this one.
00:13:51 > **Editing file** (src/todo/app.css)
00:14:02 **Participant**
Change placeholder.
00:14:08 The text.
00:14:08 What needs to be done?
00:14:22 > **Viewing file** (public/index.html)
00:14:29 > **Viewing file** (src/index.js)
00:14:37 > **Viewing file** (src/todo/reducer.js)
00:14:38 > **Viewing file** (src/todo/constants.js)
00:14:43 > **Viewing file** (src/todo/reducer.js)
00:14:43 **Participant**
OK.
00:14:44 **Researcher**
So how are you approaching this now?
00:14:49 **Participant**
Yes, how exactly am I approaching this now?
00:14:55 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:14:58 **Participant**
I am still looking for some sort of an input at this point.
00:15:04 So I want to see.

00:15:07 I want to see.
00:15:11 > **Viewing file** (src/todo/components/main.jsx)
00:15:11 **Participant**
Not this one.
00:15:17 Now I assume this is sort of a form.
00:15:20 I see.
00:15:21 This is a yeah, div with an input and there is.
00:15:29 OK.
00:15:30 I assume this is sort of a form.
00:15:31 I'm looking for something that could.
00:15:34 Point.
00:15:36 To placeholder.
00:15:41 Or again.
00:15:44 A constant or something. So I'm not entire sure.
00:15:47 > **Viewing file** (src/todo/constants.js)
00:15:49 **Participant**
Not constant.
00:15:51 > **Viewing file** (src/todo/components/input.jsx)
00:15:58 **Participant**
Placeholder here, for instance placeholder.
00:16:03 Placeholder.
00:16:07 Function on submit placeholder.
00:16:11 Great.
00:16:14 > **Viewing file** (src/index.js)
00:16:18 **Participant**
There, that should be the.
00:16:23 > **Viewing file** (src/todo/app.jsx)
00:16:35 > **Editing file** (src/todo/app.css)
00:16:37 > **Viewing file** (src/todo/reducer.js)
00:16:38 > **Viewing file** (src/todo/constants.js)
00:16:41 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:16:42 **Participant**
Yeah. OK.
00:16:44 I will read again so.
00:16:51 Hashtag code, codebase to actually search all the files.
00:17:09 Never used this so I have no idea.
00:17:20 **Chat** (window: 0)
> **Agent**
github.copilot.default
> **Context**
viewport (src/todo/components/paginationControls.jsx)
> **Message**
#codbase where is the placeholder set, or where is the input function called?
> **Response**
In the provided [PaginationControls \(src/todo/components/paginationControls.jsx\)](#) component, there is no placeholder being set or input function being called. The component is focused on rendering pagination buttons (Previous and Next) and handling page changes through the [pageChange \(src/todo/components/paginationControls.jsx\)](#) function.

If you're referring to an input field or placeholder functionality, it is not present in this code. You might want to check other components in your codebase where an input field or placeholder might be implemented. Let me know if you'd like help searching for it!
> **Response rendering**
00:17:23
- 00:17:24
00:17:32 **Participant**

00:17:36

Oh.

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

codebase

> **Message**

#codebase where is the placeholder set, or where is the input function called?

> **Response**

The placeholder is set in the [Input\(src/todo/components/input.jsx\)](#) component, which is located in the file [src/todo/components/input.jsx](#). Specifically, the [placeholder\(node_modules/@types/react/index.d.ts\)](#) prop is passed to the `<input>` element:

```
<input
  className="new-todo"
  id="todo-input"
  type="text"
  data-testid="text-input"
  autoFocus
  placeholder={placeholder}
  defaultValue={defaultValue}
  onBlur={handleBlur}
  onKeyDown={handleKeyDown}
/>
```

The [placeholder\(node_modules/@types/react/index.d.ts\)](#) value is passed to the [Input\(src/todo/components/input.jsx\)](#) component from the [Header\(src/todo/components/header.jsx\)](#) component in [src/todo/components/header.jsx](#). Here's the relevant code:

```
<Input onSubmit={addItem} label="New Todo Input" placeholder="What
needs to be done?" />
```

So, the placeholder text "What needs to be done?" is set in the [Header\(src/todo/components/header.jsx\)](#) component and passed down to the [Input\(src/todo/components/input.jsx\)](#) component.

00:17:37

Participant

Woops.

00:17:39

Umm.

00:17:43

Researcher

What make you realize your typo?

00:17:46

Participant

Yeah. I didn't actually see something like this.

00:17:49

- 00:17:51

> **Response rendering**

00:17:49

Participant

And then there was just a redundant thing, and he was looking only into pagination controls, which was the open one.

00:17:53

00:17:54

But I need you to.

00:17:55

I need the the Copilot to look everywhere you know.

00:17:59

So oh input jsx.

00:18:00

> **Viewing file** (src/todo/components/header.jsx)

00:18:01

> **Viewing file** (src/todo/components/input.jsx)

00:18:14

Participant

From the header component in the we have already established that.

00:18:32

Input on submit.

00:18:45

> **Viewing file** (src/index.js)

00:18:46

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

00:19:00

Participant

Yeah, I I'm assuming it could be under header.

00:19:02

> **Viewing file** (src/todo/components/header.jsx)

00:19:04

Participant

Oh here.

00:19:09

Yeah, so basically it did help me at this point

00:19:12

just by the fact that.

00:19:20

It showed me this, but I actually did not read it well

00:19:24

enough to actually see this is there already.

00:19:26

I don't know what.

00:19:27

I just found it a bit redundant I guess.

00:19:38

Enter a task here, right.

00:19:40

I don't think I'm allowed.

00:19:42

I don't think I can.

00:19:44
- 00:19:47

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

00:19:45

Participant

Get an error.

00:19:53

OK. And the next sort todos, the todos are currently displayed in the order they were added.

00:19:59

00:20:08

Most of the application OK add the following todos in the order the

00:20:12

quick one two three brown.

00:20:48

OK.

00:20:49

Now I just added to see what's going on.

00:20:51

And now where should I look for this?

00:20:54

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

00:20:54

> **Viewing file** (src/todo/components/item.jsx)

00:20:55

> **Viewing file** (src/todo/components/footer.jsx)

00:20:57

Participant

Let's see what is footer.

00:21:25

> **Viewing file** (src/todo/components/item.jsx)

00:21:26

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

00:21:27

> **Viewing file** (src/todo/components/item.jsx)

00:21:41

Researcher

Is there anything specific you're looking for now?

00:21:44

Participant

Hmm.

00:21:50

Not.

00:21:51

Really, I guess I think I'm just trying to

00:21:54

understand.

00:21:55

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

00:21:55

> **Viewing file** (src/todo/components/paginationControls.jsx)

00:21:57

> **Viewing file** (src/todo/components/item.jsx)

00:21:59

Participant

How is this handled?

00:22:02

But I could just.

00:22:10

Ask the Copilot, how are the.

00:22:15

Todos handled.

00:22:16

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

codebase

> **Message**

#codebase how are the todos handled?

> Response

The todos in your codebase are managed using the [useReducer \(node_modules/@types/react/index.d.ts\)](#) hook in the [App \(src/todo/app.jsx\)](#) component. Here's a breakdown of how they are handled:

1. State Management

The [useReducer \(node_modules/@types/react/index.d.ts\)](#) hook is used to manage the state of the todos. The reducer function ([todoReducer \(src/todo/reducer.js\)](#)) is defined in [src/todo/reducer.js](#). It handles various actions to update the state of the todos.

```
File: src/todo/app.jsx
const [todos, dispatch] = useReducer(todoReducer, []);
```

2. Reducer Logic

The [todoReducer \(src/todo/reducer.js\)](#) function processes actions to modify the todos. It supports the following actions (defined in [src/todo/constants.js](#)):

- **ADD_ITEM**: Adds a new todo.
- **UPDATE_ITEM**: Updates the title of an existing todo.
- **REMOVE_ITEM**: Removes a todo by its ID.
- **TOGGLE_ITEM**: Toggles the [completed \(src/todo/components/item.jsx\)](#) status of a todo.
- **REMOVEALLITEMS**: Clears all todos.
- **TOGGLE_ALL**: Toggles the [completed \(src/todo/components/item.jsx\)](#) status of all todos.
- **REMOVECOMPLETEDITEMS**: Removes all completed todos.
- **LOAD_STATE**: Loads todos from local storage.

3. Local Storage

The todos are persisted in local storage using the [loadState \(src/todo/reducer.js\)](#) and [saveState \(src/todo/reducer.js\)](#) functions in [src/todo/reducer.js](#).

- [loadState \(src/todo/reducer.js\)](#): Loads the todos from local storage when the app initializes.
- [saveState \(src/todo/reducer.js\)](#): Saves the todos to local storage whenever the state changes.

4. Adding Todos

The [Header \(src/todo/components/header.jsx\)](#) component contains an [Input \(src/todo/components/input.jsx\)](#) component where users can add new todos. When the user submits a new todo, the [ADD_ITEM \(src/todo/constants.js\)](#) action is dispatched.

```
File: src/todo/components/header.jsx
const addItem = useCallback((title) => dispatch({ type: ADD_ITEM,
payload: { title } })), [dispatch]);
<Input onSubmit={addItem} label="New Todo Input" placeholder="Enter a
task here." />
```

5. Displaying Todos

The [Main](#) ([src/todo/components/main.jsx](#)) component filters and displays todos based on the current route (`/`, `/active`, or `/completed`). It uses the [Item](#) ([src/todo/components/item.jsx](#)) component to render individual todos.

```
File: src/todo/components/main.jsx
const visibleTodos = useMemo(() => todos.filter(...), [todos, route]);
<Item todo={todo} key={todo.id} dispatch={dispatch} />
```

6. Editing and Removing Todos

The [Item](#) ([src/todo/components/item.jsx](#)) component allows users to edit or remove todos. It dispatches the [UPDATEITEM](#) ([src/todo/constants.js](#)) or [REMOVEITEM](#) ([src/todo/constants.js](#)) actions based on user interactions.

```
File: src/todo/components/item.jsx
const updateItem = useCallback((id, title) => dispatch({ type:
UPDATE_ITEM, payload: { id, title } })), [dispatch]);
const removeItem = useCallback(() => dispatch({ type: REMOVE_ITEM,
payload: { id } })), [dispatch]);
```

7. Footer Actions

The [Footer](#) ([src/todo/components/footer.jsx](#)) component provides options to clear completed todos or filter todos by their status. It dispatches actions like [REMOVECOMPLETEDITEMS](#) ([src/todo/constants.js](#)).

```
File: src/todo/components/footer.jsx
const removeCompleted = useCallback(() => dispatch({ type:
REMOVE_COMPLETED_ITEMS })), [dispatch];
```

Summary

- Todos are stored in the state managed by [useReducer](#) ([node_modules/@types/react/index.d.ts](#)).
- Actions are dispatched to modify the state.
- The state is persisted in local storage.

- Components like [Header \(src/todo/components/header.jsx\)](#), [Main \(src/todo/components/main.jsx\)](#), [Item \(src/todo/components/item.jsx\)](#), and [Footer \(src/todo/components/footer.jsx\)](#) interact with the todos through the [dispatch \(src/todo/app.jsx\)](#) function.

00:22:17

Participant

Maybe it makes it shorter.

00:22:36

- 00:22:53

> Response rendering

00:22:55

Participant

OK.

00:22:55

The use reducer hook is used to manage the state of the todos, the reducer function todo reducer.

00:23:02

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

00:23:03

Participant

No, I just. Oh, OK.

00:23:05

Well this is useful.

00:23:07

Um.

00:24:04

Researcher

So what do you think of this response?

00:24:07

Participant

I think this is a very clear response as it explained really well.

00:24:13

Now.

00:24:17

No in ChatGPT.

00:24:20

I am not entirely sure if it is correct, but makes sense you know.

00:24:27

So.

00:24:35

How does it?

00:24:38

They just want to be sorted.

00:24:46

So probably.

00:24:48

When they are loaded.

00:24:56

Loads the todo from the storage when the app initializes.

00:24:59

So probably somewhere in the function of the load state.

00:25:10

I want to.

00:25:13

Sort them but.

00:25:17

Sort them, after I'm loaded.

00:25:27

Chat (window: 0)

> Agent

github.copilot.default

> Context

viewport (src/todo/reducer.js)

> Message

When would be the best way to sort the todos?

> Response

The best way to sort the todos depends on the desired sorting criteria (e.g., by title, completion status, or creation date). Since the todos are managed by the [todoReducer \(src/todo/reducer.js\)](#), sorting should ideally be done in a way that doesn't mutate the state directly. Here are a few approaches:

1. Sort When Displaying Todos

The simplest and most flexible way is to sort the todos when they are displayed in the Main component. This avoids modifying the state and allows dynamic sorting based on user preferences.

```
File: src/todo/components/main.jsx
const sortedTodos = useMemo(() => {
  return [...todos].sort((a, b) => a.title.localeCompare(b.title));
```

```
// Example: Sort alphabetically by title
}, [todos]);

// Use `sortedTodos` instead of `todos` for rendering
sortedTodos.map((todo) => <Item todo={todo} key={todo.id} dispatch=
{dispatch} />);
```

2. Sort in the Reducer

If sorting is a core feature (e.g., todos should always be sorted by creation date or another criterion), you can sort the todos in the [todoReducer \(src/todo/reducer.js\)](#) whenever the state is updated. For example:

```
File: src/todo/reducer.js
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 cases...
  }

  // Sort todos by title (or any other criterion)
  newState = newState.sort((a, b) => a.title.localeCompare(b.title));

  saveState(newState);
  return newState;
};
```

3. Allow User-Controlled Sorting

You can add a sorting feature in the UI, allowing users to choose how to sort the todos (e.g., by title, completion status, or creation date). This can be implemented using a `useState` hook in the Main component to track the selected sorting method.

```
File: src/todo/components/main.jsx
const [sortMethod, setSortMethod] = useState("title");

const sortedTodos = useMemo(() => {
  switch (sortMethod) {
    case "title":
      return [...todos].sort((a, b) =>
a.title.localeCompare(b.title));
    case "completed":
      return [...todos].sort((a, b) => a.completed -
b.completed);
    case "date":
      return [...todos].sort((a, b) => a.id - b.id); // Assuming
`id` reflects creation time
    default:
      return todos;
  }
}, [todos, sortMethod]);

// Dropdown or buttons to change sorting method
<select value={sortMethod} onChange={(e) =>
setSortMethod(e.target.value)}>
  <option value="title">Sort by Title</option>
  <option value="completed">Sort by Completion</option>
```

```
<option value="date">Sort by Date</option>
</select>

// Use `sortedTodos` for rendering
sortedTodos.map((todo) => <Item todo={todo} key={todo.id} dispatch=
{dispatch} />);
```

Recommendation

- **Sort When Displaying:** If sorting is optional or user-controlled, sort in the Main component or wherever the todos are displayed.
- **Sort in the Reducer:** If todos must always follow a specific order (e.g., by creation date), sort them in the [todoReducer \(src/todo/reducer.js\)](#).

Let me know if you'd like help implementing any of these approaches!

00:25:27

Participant

So I will keep asking because.

00:25:28
- 00:26:02

> Response rendering

00:25:31

Participant

Yeah, no idea.

00:25:49

I hope it did not 'cause I was getting a call so I hope it did not stop the recording or anything.

00:25:53

00:25:56

Researcher

No, that's.

00:25:58

That went all right. No worries.

00:26:02

Participant

OK.

00:26:19

So when displaying todos.

00:26:27

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

00:26:30

Participant

OK.

00:26:31

So basically I just asked some ideas about.

00:26:39

I just asked some ideas about when to do the sort so he said to sort when displaying sort in the reducer.

00:26:44

Allow user control sorting.

00:26:48
00:26:50
00:27:11
00:27:13
00:27:16
00:27:26

Yeah. Last one is the complicated.
Is it already?
Does it already has any kind of sorting?
I think so.
Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

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

> **Message**

does it already have any kind of sorting based on date?

> **Response**

No, the current implementation of the [Main \(src/todo/components/main.jsx\)](#) component does not include any sorting based on date or any other criteria. The [visibleTodos \(src/todo/components/main.jsx\)](#) array is filtered based on the current route (/, /active, or /completed), but the order of the todos remains the same as they are in the [todos \(src/todo/components/main.jsx\)](#) array.

If you want to sort the todos by date (e.g., by their creation time), you would need to add a sorting step to the [visibleTodos \(src/todo/components/main.jsx\)](#) computation. For example, if the todos have a `createdAt` property, you could sort them like this:

```
const visibleTodos = useMemo(() => {  
  return todos  
    .filter((todo) => {  
      if (route === "/active") return !todo.completed;  
      if (route === "/completed") return todo.completed;  
      return todo;  
    })  
    .sort((a, b) => new Date(a.createdAt) - new Date(b.createdAt));  
  // Sort by creation date (oldest first)  
}, [todos, route]);
```

If the todos do not currently have a `createdAt` property, you would need to add it when creating a new todo in the `todoReducer`. Let me know if you'd like help implementing this!

00:27:27

Participant

What I'm asking right now is because, well, I could just do this.

00:27:30
- 00:27:34

> **Response rendering**

00:27:32

Participant

But first I want to see if there is any.

00:27:38

Already kind of sorted.

00:27:50

If they're already sorted based on, I don't know time or whatever.

00:28:29

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

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

> **Message**

let's sort them alphabetically

> **Response**

To sort the todos alphabetically by their title, you can modify the [visibleTodos \(src/todo/components/main.jsx\)](#) computation in the [Main \(src/todo/components/main.jsx\)](#) component by adding a `.sort()` step. Here's how you can do it:

Updated Code

```
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;
      })
      .sort((a, b) => a.title.localeCompare(b.title)), // Sort
    [todos, route]
  );

```

Explanation

1. **Filtering:** The [filter\(src/todo/components/main.jsx\)](#) function ensures that only the todos relevant to the current route (/ , /active, or /completed) are included.
2. **Sorting:** The sort function sorts the filtered todos alphabetically by their title property using localeCompare.

Full Updated Code

Here's how the relevant part of the [Main \(src/todo/components/main.jsx\)](#) component looks after the change:

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

```

Now, the todos will be displayed in alphabetical order based on their titles. Let me know if you need further assistance!

00:28:30

Participant

So I will just ask to sort them alphabetically here.

00:28:30
- 00:28:35

> Response rendering

00:28:37

Participant

And I do remember this.

00:28:51
- 00:29:34

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

00:29:01

Participant

One thing when I'm not copying because if I'm copying that I will just.

00:29:04

I know I will just.

00:29:08

Try to run and see if it works and then I will never understand what what I actually do.

00:29:11

00:29:12 **Researcher**
OK.

00:29:14 **Participant**
So.

00:29:16 Local compare even in that.

00:29:35 **Researcher**
I see an error message popping up all the time though.

00:29:39 **Participant**
This one.

00:29:39 **Researcher**
Is this yes?

00:29:41 Could you maybe press retry?

00:29:44 Just to make it a.

00:29:45 **Participant**
It says that unable to read from the browser clipboard.

00:29:50 **Researcher**
OK, what if in the URL bar in the top of the screen to the right there's a clipboard icon?

00:29:54 **Participant**
Always allow?

00:29:59 **Researcher**
Yes. If you do that, then at least it will stop popping up.

00:30:02 And if you want to copy and paste then you can at least.

00:30:06 **Participant**
Well.

00:30:07 **Researcher**
Thanks.

00:30:25 So what are you doing now?

00:30:25 **Participant**
OK.

00:30:27 I was just trying to understand what he did.

00:30:30 **Researcher**
OK.

00:30:32 **Participant**
And that's just it. Oh, wait.

00:30:37 Should I just introduce them again?

00:30:42 What was the order?

00:30:53 Yeah.

00:31:32 Great, now it doesn't work at all.

00:31:52 > **Editing file** (src/todo/components/main.jsx)

00:31:59 > **Editing file** (src/todo/components/main.jsx)

00:32:02 **Participant**
Just checking if it's because of this.

00:32:51 OK.

00:33:09 **Chat** (window: 0)
> **Agent**
github.copilot.default
> **Context**
viewport (src/todo/components/main.jsx)
> **Message**
is my main.jsx correct then?
> **Response**
Your [src/todo/components/main.jsx](#) file has a small typo in the sorting function. The method [localCompare \(src/todo/components/main.jsx\)](#) is incorrect; it should be localeCompare. Here's the corrected code:

Corrected Code

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

Explanation of the Fix

- **localeCompare**: This is the correct method for comparing strings alphabetically. It ensures proper sorting, especially for strings with special characters or different locales.
- **localCompare** ([src/todo/components/main.jsx](#)): This is not a valid JavaScript method, so it would throw an error.

Final Check

With this fix, your [src/todo/components/main.jsx](#) file is correct, and the todos will now be sorted alphabetically by their [title \(src/todo/components/main.jsx\)](#). Let me know if you need further assistance!

00:33:10

Participant

So yeah, now it for some reason does not work by

00:33:13
- 00:33:18

> Response rendering

00:33:13

Participant

the sorting thing so.

00:33:16

The one two three. It doesn't work, so I will just.

00:33:24

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

00:33:25

Participant

Never mind.

00:33:32

Quick.

00:33:37

Brown and fox.

00:33:43

Yeah. OK.

00:33:46

Researcher

Great. So then I'll.

00:33:48

Participant

Typo here.

00:33:50

Researcher

I'll pass you the the tasks.

00:33:53

Participant

Uh huh.

00:33:54

Researcher

Let me see you have.

00:33:58

12 minutes left, but no worries. There's no rush.

00:34:03

In getting all the tasks done, just see how far you can get.

00:34:08

Participant

Well, let's see.

00:34:12

The getting started was.

00:34:47

Oh yeah.

00:34:57

Researcher

So what are you thinking now?

00:34:59

What will you do?

00:35:05

Participant

I do remember that they were saying that that's being saved somewhere, but.

00:35:28

> **Viewing file** (src/todo/components/item.jsx)

00:35:35

Participant

Again, now I will just try to see.

00:36:26

Yeah. Now I'll just ask ChatGPT to look into the

00:36:28

code for me.

00:36:29

Where exactly this attempt has been made.

00:36:39

Save.

00:36:50

Chat (window: 0)

> **Agent**

github.copilot.default

> **Context**

codebase

> **Message**

#codebase where was the attempt to save the todo list with the status has been made?

> **Response**

The attempt to save the todo list with its status is made in the [saveState](#) ([src/todo/reducer.js](#)) function within the [todoReducer](#) ([src/todo/reducer.js](#)) in the file [src/todo/reducer.js](#). Here's the relevant code:

```
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;
    case UPDATE_ITEM:
      newState = state.map((todo) => (todo.id ===
action.payload.id ? { ...todo, title: action.payload.title } : todo));
      break;
    case REMOVE_ITEM:
      newState = state.filter((todo) => todo.id !==
action.payload.id);
      break;
    case TOGGLE_ITEM:
      newState = state.map((todo) => (todo.id ===
action.payload.id ? { ...todo, completed: !todo.completed } : todo));
      break;
    case REMOVE_ALL_ITEMS:
      newState = [];
      break;
    case TOGGLE_ALL:
      newState = state.map((todo) => (todo.completed !==
action.payload.completed ? { ...todo, completed:
```

```

    action.payload.completed } : todo));
    break;
    case REMOVE_COMPLETED_ITEMS:
        newState = state.filter((todo) => !todo.completed);
        break;
    case LOAD_STATE:
        newState = action.payload;
        break;
    default:
        throw Error(`Unknown action: ${action.type}`);
  }

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

```

Key Points:

1. [saveState \(src/todo/reducer.js\)](#) Function:

- This function saves the current state of the todos to [localStorage \(node_modules/@types/node/globals.d.ts\)](#) under the key "todos".
- It is called after every action that modifies the state.

2. When It Happens:

- After any action (e.g., [ADDITEM \(src/todo/constants.js\)](#), [UPDATEITEM \(src/todo/constants.js\)](#), [TOGGLE_ITEM \(src/todo/constants.js\)](#), etc.) that changes the state, the [saveState \(src/todo/reducer.js\)](#) function is invoked to persist the updated state.

3. Where It Is Stored:

- The todos are stored in the browser's [localStorage \(node_modules/@types/node/globals.d.ts\)](#) as a JSON string.

Let me know if you need help debugging or modifying this functionality!

00:36:54

Participant

I do expect it to be here but.

00:37:01

Searching codebase for save state.

00:37:03

- 00:37:13

> Response rendering

00:37:15

Participant

Save state within the todo reducer.

00:37:19

Which is in reducer dot here.

00:37:21

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

00:37:49

Participant

Oh, this one I guess.

00:38:05

- 00:38:08

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

00:38:13

Participant

I highly doubt this would be the issue though.

00:38:18

Yeah, it is not.

00:38:42

- 00:38:44

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

00:38:43

Participant

But I thought the issue would be this.

00:38:47

Just this would get item todo S and get item todo.

00:39:00

After an action that changes the state, the save state function is invoked to persist

00:39:05

the updated storage the todos are stored in the browser local storage I

00:39:09

JSON string.

00:39:20

Was it even here I don't even remember where it was, so I would just.

00:39:40

Chat (window: 0)

> Agent

github.copilot.default

> Context

viewport (src/todo/reducer.js)

> Message

could the mistake be because of todoS and todo from setItem and getItem?

> Response

Yes, the mismatch between "todoS" in [localStorage.getItem \(node_modules/@types/node/globals.d.ts\)](#) and "todos" in [localStorage.setItem \(node_modules/@types/node/globals.d.ts\)](#) is likely causing the issue. The [loadState \(src/todo/reducer.js\)](#) function is trying to retrieve the state using the key "todoS", but the [saveState \(src/todo/reducer.js\)](#) function is saving it under the key "todos". This inconsistency means that the app will not correctly load the saved todos from [localStorage \(node_modules/@types/node/globals.d.ts\)](#).

Fix

To resolve this, ensure that both [loadState \(src/todo/reducer.js\)](#) and [saveState \(src/todo/reducer.js\)](#) use the same key. For example, change "todoS" in [loadState \(src/todo/reducer.js\)](#) to "todos".

Corrected Code

```
File: src/todo/reducer.js
export const loadState = () => {
  const state = localStorage.getItem("todos"); // Corrected key
  return state ? JSON.parse(state) : [];
};
```

```
const saveState = (state) => {
  localStorage.setItem("todos", JSON.stringify(state)); // Key
  remains consistent
};
```

Explanation

- **Key Consistency:** Both [loadState \(src/todo/reducer.js\)](#) and [saveState \(src/todo/reducer.js\)](#) now use the key "todos".
- **Impact:** This ensures that the app can correctly load the saved todos from [localStorage \(node_modules/@types/node/globals.d.ts\)](#) when it initializes.

Let me know if you need further assistance!

00:39:41
- 00:39:47

> Response rendering

00:39:57

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

00:39:59

Participant

Yeah, let's try this.

00:40:34

Researcher

What are you trying to do now?

00:40:37

Participant

I am just trying to understand what the todo reducer is and how exactly.

00:40:43

It works.

00:40:48

Save state new state.

00:40:53

Save state in a state local storage.

00:41:27

Chat (window: 0)

> Agent

github.copilot.default

> Context

viewport (src/todo/reducer.js)

> Message

still does not solve the issue

> Response

The issue might persist due to a typo in your [todoReducer \(src/todo/reducer.js\)](#). Specifically, the case for [LOADSTATE \(src/todo/constants.js\)](#) is misspelled as [LOODSTATE \(src/todo/reducer.js\)](#). This would prevent the reducer from correctly

handling the [LOAD_STATE\(src/todo/constants.js\)](#) action, which is responsible for loading the todos from [localStorage\(node_modules/@types/node/globals.d.ts\)](#).

Fix the Typo

Update the [todoReducer\(src/todo/reducer.js\)](#) to use the correct [LOAD_STATE\(src/todo/constants.js\)](#) action type:

```
File: src/todo/reducer.js
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;
    case UPDATE_ITEM:
      newState = state.map((todo) => (todo.id ===
action.payload.id ? { ...todo, title: action.payload.title } : todo));
      break;
    case REMOVE_ITEM:
      newState = state.filter((todo) => todo.id !==
action.payload.id);
      break;
    case TOGGLE_ITEM:
      newState = state.map((todo) => (todo.id ===
action.payload.id ? { ...todo, completed: !todo.completed } : todo));
      break;
    case REMOVE_ALL_ITEMS:
      newState = [];
      break;
    case TOGGLE_ALL:
      newState = state.map((todo) => (todo.completed !==
action.payload.completed ? { ...todo, completed:
action.payload.completed } : todo));
      break;
    case REMOVE_COMPLETED_ITEMS:
      newState = state.filter((todo) => !todo.completed);
      break;
    case LOAD_STATE: // Corrected from LOOD_STATE
      newState = action.payload;
      break;
    default:
      throw Error(`Unknown action: ${action.type}`);
  }

  saveState(newState);
  return newState;
};
```

Explanation

- **Typo in [LOADSTATE\(src/todo/constants.js\)](#):** *The reducer was looking for [LOODSTATE\(src/todo/reducer.js\)](#), which does not match the [LOAD_STATE\(src/todo/constants.js\)](#) constant. As a result, the action to load the state from [localStorage\(node_modules/@types/node/globals.d.ts\)](#) was not being handled.*
- **Impact:** This would prevent the app from initializing the todos correctly from [localStorage\(node_modules/@types/node/globals.d.ts\)](#).

Verify the Fix

1. Ensure that the [LOAD_STATE \(src/todo/constants.js\)](#) constant is correctly defined in [src/todo/constants.js](#):

```
export const LOAD_STATE = "LOAD_STATE";
```

2. Check where the [LOAD_STATE \(src/todo/constants.js\)](#) action is dispatched in your app. For example, it might be dispatched during app initialization.
3. After fixing the typo, restart your app and verify that the todos are correctly loaded from [localStorage \(node_modules/@types/node/globals.d.ts\)](#).

Let me know if the issue persists!

00:41:28

Participant

Yep, still does not solve the issue. So I will keep just.

00:41:30
- 00:41:44

> Response rendering

00:41:33

Participant

Asking.

00:41:48

Oh, I would have.

00:41:50

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

00:41:52

Participant

Never thought of that.

00:42:00

Even though I had an error here, I didn't really look into that.

00:42:26

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

00:42:33

> Viewing file (src/index.js)

00:42:42

Chat (window: 0)

> Agent

github.copilot.default

> Context

codebase

> Message

#codebase can you verify it for me?

> Response

After reviewing the provided codebase, here are the key points and potential issues related to saving and loading the todo list:

1. [loadState \(src/todo/reducer.js\)](#) and [saveState \(src/todo/reducer.js\)](#) Functions

- **Code:**

```
export const loadState = () => {  
  const state = localStorage.getItem("todos");  
  return state ? JSON.parse(state) : [];  
};  
  
const saveState = (state) => {  
  localStorage.setItem("todos", JSON.stringify(state));  
};
```

- **Verification:**

- The `loadState(src/todo/reducer.js)` function correctly retrieves the `todos(src/todo/components/main.jsx)` from `localStorage(node_modules/@types/node/globals.d.ts)` and parses them.
 - The `saveState(src/todo/reducer.js)` function correctly saves the `todos(src/todo/components/main.jsx)` to `localStorage(node_modules/@types/node/globals.d.ts)` as a JSON string.
 - Both functions use the same key ("todos"), so there is no mismatch here.
-

2. `todoReducer(src/todo/reducer.js)` Integration

- **Code:**

```
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;
    case UPDATE_ITEM:
      newState = state.map((todo) => (todo.id ===
action.payload.id ? { ...todo, title: action.payload.title } :
todo));
      break;
    case REMOVE_ITEM:
      newState = state.filter((todo) => todo.id !==
action.payload.id);
      break;
    case TOGGLE_ITEM:
      newState = state.map((todo) => (todo.id ===
action.payload.id ? { ...todo, completed: !todo.completed } :
todo));
      break;
    case REMOVE_ALL_ITEMS:
      newState = [];
      break;
    case TOGGLE_ALL:
      newState = state.map((todo) => (todo.completed !==
action.payload.completed ? { ...todo, completed:
action.payload.completed } : todo));
      break;
    case REMOVE_COMPLETED_ITEMS:
      newState = state.filter((todo) => !todo.completed);
      break;
    case LOAD_STATE:
      newState = action.payload;
      break;
    default:
      throw Error(`Unknown action: ${action.type}`);
  }

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

- **Verification:**

- The `saveState(src/todo/reducer.js)` function is called after every state update, ensuring that the todos are saved to `localStorage(node_modules/@types/node/globals.d.ts)`.
- The `LOAD_STATE(src/todo/constants.js)` action is correctly handled to load the state from an external source (e.g., `localStorage`).

[\(node_modules/@types/node/globals.d.ts\)](#).

3. [App \(src/todo/app.jsx\)](#) Component Initialization

- **Code:**

```
export function App() {  
  const [todos, dispatch] = useReducer(todoReducer, []);  
  return (  
    <>  
      <Header dispatch={dispatch} />  
      <Main todos={todos} dispatch={dispatch} />  
      <Footer todos={todos} dispatch={dispatch} />  
    </>  
  );  
}
```

```
    );  
  }  
}
```

- **Verification:**

- The [useReducer \(node_modules/@types/react/index.d.ts\)](#) hook initializes the [todos \(src/todo/components/main.jsx\)](#) state with an empty array ([]).
 - **Issue:** The [initialState \(src/todo/reducer.js\)](#) from [loadState\(\) \(src/todo/reducer.js\)](#) is not passed to [useReducer \(node_modules/@types/react/index.d.ts\)](#). This means the app does not load the saved todos from [localStorage \(node_modules/@types/node/globals.d.ts\)](#) on initialization.
-

4. Fix for Loading Todos on Initialization

To fix the issue, pass the [initialState \(src/todo/reducer.js\)](#) from [loadState\(\) \(src/todo/reducer.js\)](#) to [useReducer \(node_modules/@types/react/index.d.ts\)](#) in the [App \(src/todo/app.jsx\)](#) component:

- **Updated Code:**

```
import { loadState } from "../reducer";  
  
export function App() {  
  const [todos, dispatch] = useReducer(todoReducer,  
    loadState()); // Use loadState() here  
  return (  
    <>  
      <Header dispatch={dispatch} />  
      <Main todos={todos} dispatch={dispatch} />  
      <Footer todos={todos} dispatch={dispatch} />  
    </>  
  );  
}
```

5. Pagination Issue in [Main \(src/todo/components/main.jsx\)](#) Component

- **Code:**

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

- **Verification:**

- The [paginationControls \(src/todo/components/main.jsx\)](#) component is imported incorrectly. It should be imported as [PaginationControls \(src/todo/components/paginationControls.jsx\)](#) (capitalized) to follow React's naming conventions.
- **Fix:**

```
import { PaginationControls } from "../paginationControls";  
  
<PaginationControls  
  currentPage={currentPage}
```

```
itemsPerPage={itemsPerPage}
setCurrentPage={setCurrentPage}
totalItems={visibleTodos.length}
/>
```

6. General Observations

- The [Input \(src/todo/components/input.jsx\)](#) component in [src/todo/components/header.jsx](#) correctly handles adding new todos.
 - The [Item \(src/todo/components/item.jsx\)](#) component in [src/todo/components/item.jsx](#) correctly handles editing, toggling, and removing todos.
 - The [Footer \(src/todo/components/footer.jsx\)](#) component correctly handles filtering and clearing completed todos.
-

Summary of Fixes

1. Update the [App \(src/todo/app.jsx\)](#) component to initialize [todos \(src/todo/components/main.jsx\)](#) with [loadState\(\) \(src/todo/reducer.js\)](#):

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

2. Fix the import and usage of [PaginationControls \(src/todo/components/paginationControls.jsx\)](#) in [Main \(src/todo/components/main.jsx\)](#):

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

With these fixes, the app should correctly load the todos from [localStorage \(node_modules/@types/node/globals.d.ts\)](#) and handle pagination properly. Let me know if you encounter further issues!

00:42:45

Participant

He asked me also to do some verification, so I'll just ask him to verify them for me. 'cause it's.

00:42:49

00:42:56
- 00:43:19

> Response rendering

00:42:58

Participant

Looking for the load state.

00:43:08

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

00:43:25

Participant

OK.

00:43:28

Researcher

So what are you doing now?

00:43:30

Participant

Just reading the feedback for load state and save state and also for the todo reducer.

00:43:37

00:43:40

OK.

00:43:44

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

00:43:45

Participant

And in the app.

00:44:01

The use reducer initializes the todos with an empty array.

00:44:15

And the fix.

00:44:18

I can see.

00:44:21

Is.

00:44:22

To use the load state, which is the constant.

00:44:27
- 00:44:31

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

00:44:33

Participant

But load state is not defined.

00:44:34

> **Viewing file** (src/index.js)

00:44:36

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

00:44:36

Participant

Why is it not defined?

00:44:43

Import load state from the reducer, yeah.

00:44:45

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

00:44:49

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

00:44:50

Participant

No.

00:44:51
- 00:45:07

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

00:45:12

Participant

All right, see.

00:45:14

Oh, OK.

00:45:16

What do I have to check?

00:45:18

Edit and delete.

00:45:23

Delete and yes and edit.

00:45:42

How do I even edit this? Oh OK.

00:45:47

Yeah.

00:45:49

I guess that we are done with the first task.

00:45:52

Researcher

Perfect.

00:47:18

Participant

How you would you rate of length? Well, good.

00:47:25

Detail, well there were really good details. Use of technical terms.

00:47:38

I mean, I do believe it did use some technical technical terms, but yeah, so I think it's also good.

00:47:42

Wait, I can go back to that.

00:47:44

Correctness.

00:47:49

Well, was correct, following your instructions.

00:47:52

Also good. Tone.

00:47:56

Use of formatting.

00:48:02

Also really good, especially the.

00:48:04

Fact that you can click the.

00:48:08

Function and it just just brings it to the function.

00:48:12

So you don't have to actually look for it.

00:48:14

Debatable.

00:48:34

Well, how successful?

00:49:53

Does the preparatory tasks count?

00:49:57

Researcher

Yes, they count.

00:50:01

Participant

Then that's good for a bit more.

00:50:03

Let's go too four.

00:50:07

Oh, those are from the tasks.

00:50:50

Hmm.

00:50:52

Well, for those I did not attempt.

00:51:02

Researcher

Great.

00:51:05

So alright, we have some questions.

00:51:08

So let me find them.

00:51:14

Sorry, one second so.

00:51:17

How did you manage to find your way through the codebase?

00:51:21

Did you feel like you were successful in navigating the files and finding the

00:51:24

00:51:29 place where you wanted to work?

00:51:35 **Participant**
Yes, but I feel like I needed a bit of.
00:51:40 Time to actually get to know where everything is.
00:51:45 So I believe that.
00:51:48 It was really useful that I could.
00:51:52 Ask the Copilot.
00:51:53 **Researcher**
Yeah. Did Copilot help you in this?

00:51:56 **Participant**
Yeah, much more easier to find the stuff.
00:52:01 So I didn't have to look every single time for it.
00:52:03 **Researcher**
So what would your strategy look like normally without Copilot?

00:52:09 **Participant**
Well, normally if I would have to work on a
00:52:13 program as I'm doing now, for instance, I know.
00:52:19 I would know where I would find stuff probably,
00:52:22 **Researcher**
Mm hmm.

00:52:22 **Participant**
but since this was a new program I would not be so sure.
00:52:28 So.
00:52:31 Yeah, depends. I guess most of the times no,
00:52:34 but depends how big the program it is and if you make it yourself also,
00:52:38 no otherwise control find on files trying to make sense of something.
00:52:43 And as you saw trying to see where they tried to import it, for
00:52:47 instance I was looking for some base dot CSS or whatever at the very beginning,
00:52:51 **Researcher**
Mm hmm.

00:52:52 **Participant**
but it was actually app dot CSS.
00:52:58 **Researcher**
OK.
00:53:00 Then a slightly similar.
00:53:03 Question.
00:53:04 Did you feel like you understood the codebase and did Copilot have an effect
00:53:09 on on how well you understood this code base?
00:53:14 **Participant**
Oh yeah, for sure.
00:53:15 I I understood that codebase and I'm also sure that it actually helped me to
00:53:21 understand it because.
00:53:25 He did explain.
00:53:26 I also asked how they work because.
00:53:31 If he asked how they were, you have a bit of a logic.
00:53:32 But if you just try to do, does not really work.
00:53:37 So yeah, I think it really.
00:53:40 Helped me.
00:53:42 **Researcher**
OK. And do you think it helped you with speed
00:53:46 at which with you got used to the codebase or did it also help like in
00:53:52 getting you more comfortable with the codebase than you could without?
00:53:59 **Participant**
Both so speed and also to get it more comfortable with the codebase that I will
00:54:04 do without it.
00:54:06 **Researcher**

00:54:09 How would you? How would you approach a new code base
00:54:12 like this without Copilot?
Participant
00:54:15 Oh, I guess I just have to look into it and
00:54:16 try to understand what's going on.
00:54:19 Which function are calling which functions.
00:54:23 And if you just get a code that is the best approach to do,
00:54:26 you're going to a function trying to find the first one.
00:54:31 See what it calls and go from there and try to understand what's going on until
00:54:33 you see.
00:54:35 Yeah, until you understand what's going on.
00:54:39 And I also do use a lot of inspect.
Researcher
00:54:40 Yeah.
Participant
00:54:43 As you saw, because.
00:54:45 Yeah, at least you see something for the button.
00:54:48 I just looked for the button color button color, saw the class there, so yeah.
Researcher
00:54:52 Yeah.
00:54:57 And and and this this kind of strategy?
00:55:03 Would this usually also work for you for more complex tasks without using Copilot?
Participant
00:55:05 It would but it would take more time.
Researcher
00:55:06 OK.
Participant
00:55:10 I mean, if you're is very complex, I believe you expect to be a schema or
00:55:12 something.
Researcher
00:55:13 Sorry.
Participant
00:55:17 If it's very complex, you expect to have a schema or something,
00:55:21 or an explanation for how the application works.
Researcher
00:55:24 Right. So you would usually use some some form
00:55:27 of documentation for code base like this.
Participant
00:55:31 Yes, for a very complex one, you need the documentation to properly
00:55:32 understand it.
Researcher
00:55:42 And do you think you would still use this documentation even when using Copilot?
Participant
00:55:45 Depends a lot.
00:55:47 If I have to work.
00:55:51 Long time with a codebase or if I just have to fix something. You have to fix
00:55:52 something small maybe.
00:55:53 I would not.
00:55:57 Maybe I would be a bit more Copilot and less documentation,
00:56:03 but if I have to find to do something that is like a job or something that I
00:56:07 have to work with the codebase for like the next years,
00:56:11 then I would use the documentation more than the compil.
Researcher
00:56:14 And and why in that case?
Participant
Well, because then you actually need to

00:56:16 understand everything better.
00:56:18 Whereas in fixing one small thing you can get a bit of help and you fixed it and
00:56:23 you're done.
00:56:24 So it depends how big the task is I guess.
00:56:27 **Researcher**
So are you.
00:56:27 Are you then saying that using Copilot you'd get some more shallow understanding?
00:56:31 Maybe of the code base?
00:56:35 Or is that not what you're saying.
00:56:35 **Participant**
Yeah.
00:56:38 **Researcher**
OK.
00:56:42 And also you you indicated some aspects of Copilot's responses that you liked or
00:56:48 disliked.
00:56:51 Can you explain which which of these stood out to you and why. If you want,
00:56:55 you can reopen the survey and look at the list again.
00:56:59 Or maybe you can recall what was good or what was bad.
00:57:02 **Participant**
I will look in the chat a bit. That's fine.
00:57:06 **Researcher**
Right.
00:57:07 **Participant**
One of the best things was the fact that I could.
00:57:12 Go to the item.
00:57:12 So for instance, local storage I could just.
00:57:19 Click and go there.
00:57:20 So I think this is one of the best finding functions for me,
00:57:23 explaining how they work.
00:57:24 Also really good.
00:57:26 And also giving me advice on what to do next. For instance,
00:57:30 I got here an advise of.
00:57:33 What to verify next?
00:57:34 And I did ask him to verify for me because, well,
00:57:37 I could have done it myself. But.
00:57:40 Why not asking him to verify and point what's a mistake,
00:57:42 and then if there is a mistake then we check there.
00:57:46 **Researcher**
Yeah.
00:57:46 **Participant**
As it was in the app component.
00:57:49 He said it's a mistake in the app and I was like, yeah, OK, let's go to the app.
00:57:54 **Researcher**
So this is mostly about the the navigation then or is it also about the
00:57:59 the the bug detection that's appealing to you?
00:58:03 **Participant**
Navigatino and bug detection. I think those are the best.
00:58:05 Uh, things that you can.
00:58:08 **Researcher**
OK, I noticed that in this case when you
00:58:12 first asked about the about saving the state,
00:58:16 it's was only talking about the capitalization in the keys.
00:58:21 That was wrong.
00:58:23 Then you ask again.
00:58:24 And it was talking about the typo in the load item.
00:58:29 And then the third bug it found in after you asked it again to verify.

00:58:29 **Participant**
Yes.

00:58:34 Mm hmm.

00:58:35 **Researcher**
So you see.

00:58:38 It didn't find all of these at once, would you have wanted this to happen or.

00:58:43 **Participant**
No, absolutely not.

00:58:45 **Researcher**
Absolutely not.

00:58:45 **Participant**
I think it's better that it.

00:58:47 Yeah. First thing, the first thing I don't remember exactly.

00:58:52 Yeah. First I asked if there was.

00:58:56 It's not sort.

00:58:58 I think something first time was actually something quite redundant.

00:59:03 Yeah.

00:59:06 I said that it it works here and this is where it saves and what it happens.

00:59:12 And then I spotted the todos mistake at some point here.

00:59:18 So I ask him if that can be a mistake.

00:59:21 So yeah, he said.

00:59:22 It can be a mistake.

00:59:22 And then he spotted the next one.

00:59:24 So I think it's better that the spotted one by time rather than everything at once because.

00:59:28 If you fix everything at once, you just risk to destroy everything and

00:59:37 you don't know where you started from and how to go back.

00:59:42 **Researcher**
OK.

00:59:43 **Participant**
And then you would still have to start from step number one and go step by step again.

00:59:48

00:59:49 **Researcher**
OK.

00:59:51 So in this case you saw it.

00:59:54 Like only reply to what you were asking for that small step, right?

00:59:59 So this is for you preferred over.

00:59:59 **Participant**
Yeah.

01:00:03 Very big replies.

01:00:03 **Researcher**
Just asking fix this and then it's doing everything at once.

01:00:07 **Participant**
Yes, yes, it's much more preferred for me to.

01:00:11 Rather, take small mistake.

01:00:14 Fix it together with the Copilot and then go to the next one. If it doesn't work.

01:00:23 **Researcher**
Yeah, that makes sense.

01:00:27 Also, I saw you in the survey.

01:00:29 You rated a bit lower.

01:00:33 That you thought, Copilot wouldn't improve your performance.

01:00:38 But then you did explain that it allowed you to do things faster.

01:00:44 So can you explain why this is?

01:00:49 **Participant**
Umm.

01:00:53 In the same yeah, it would.

01:00:57 Does improve the performance.
01:00:58 Not entirely, sure.
01:00:59 Was it under the performance?
01:01:02 **Researcher**
Yes.
01:01:04 **Participant**
Yeah, it does improve the performance.
01:01:06 **Researcher**
It does, yeah?
01:01:07 **Participant**
Yeah.
01:01:09 **Researcher**
Do you think there are things that it doesn't improve?
01:01:14 **Participant**
Things that doesn't improve.
01:01:17 Actually, for the performance is a bit of a a mixed
01:01:20 because sometimes again tend to give a lot of.
01:01:26 Things you can do at the same time and you don't know which one to do, how to do.
01:01:30 Then you have to look over all of them.
01:01:31 **Researcher**
Mm hmm.
01:01:34 **Participant**
So sometimes it can give a a lot of things at the same time.
01:01:39 **Researcher**
OK.
01:01:40 **Participant**
It makes you more confused about what's going on rather than.
01:01:45 There. So sometimes you also have to lose a bit
01:01:47 of time of trying to understand what he says.
01:01:51 Then to implement but.
01:01:51 **Researcher**
And.
01:01:53 And do you think it in some cases gives information that's irrelevant or is the
01:01:58 information still all irrelevant, but it's just a lot?
01:02:03 **Participant**
Sometimes.
01:02:06 Honestly, I believe.
01:02:12 It doesn't. Most of the information is relevant,
01:02:14 but at some point it just drops in some irrelevant, for example pagination issue.
01:02:20 Which?
01:02:22 Yeah, I think it was. Not really.
01:02:26 Yeah, it's still the same.
01:02:28 So you have it's doing the verification and fixing and yeah.
01:02:33 **Researcher**
So maybe this is one of these things you were talking about where it was
01:02:37 addressing bugs you didn't want to talk about.
01:02:38 **Participant**
Yeah, yeah.
01:02:40 Yeah, it's mostly because now this is a bit of
01:02:43 a step further that I don't even was.
01:02:46 Not even plan on my list.
01:02:48 Point it out, but it's a bit too much.
01:02:50 **Researcher**
OK.
01:02:52 **Participant**
As in, I don't understand why would I.

01:02:56 Even though there's clearly.
01:02:56 **Researcher**
Yeah.
01:03:00 **Participant**
An issue?
01:03:01 **Researcher**
That makes sense.
01:03:03 So you interacted with Copilot a bit.
01:03:07 And you saw some things that you liked and that you didn't like about it.
01:03:11 Or maybe that Copilot could do or couldn't do.
01:03:14 So did you change the way you were interacting with Copilot?
01:03:17 Based on what you observed, how it works.
01:03:24 **Participant**
While interacting?
01:03:32 Yeah, I think.
01:03:33 I think I started to give him a big a bigger task. At the very beginning I was
01:03:38 asking more conceptual stuff to understand the flow,
01:03:41 just spot the mistakes and at some point I was just asking him actually,
01:03:45 can you verify this for me instead of telling me
01:03:48 to verify it.
01:03:51 So I think I, yeah, I did.
01:03:54 In the first time I used him more as to explain stuff and in the end I used him
01:03:57 to actually do some stuff for me.
01:03:59 **Researcher**
OK. And why do you think this changed?
01:04:03 **Participant**
Because I just realized it was much more easier to ask him to do the verification
01:04:08 rather than me having to go to that specific even though I did go after it to
01:04:13 go to that specific constant in the file constant Java and then check the constant
01:04:19 he could.
01:04:19 Understand himself. So I just ask him to do it.
01:04:23 **Researcher**
So why did you try to do this more yourself at the at the start?
01:04:29 Was this because you you like doing this or because you think Copilot
01:04:30 **Participant**
Because.
01:04:32 **Researcher**
couldn't do it? Maybe or.
01:04:34 **Participant**
I rather try to do it myself because I was trying to understand what's going on
01:04:38 in the code.
01:04:41 **Researcher**
So then eventually, did you?
01:04:44 Maybe not care us much about understanding the code or what was the
01:04:48 reason that it changed after a bit.
01:04:55 **Participant**
Because I understood the code, but he asks to do some specific
01:04:59 verification that I'd rather.
01:05:02 Wanted to skip so I jumped to the next step, then verifying those small things and
01:05:07 then checking if the constant load state is correctly defined.
01:05:11 **Researcher**
OK.
01:05:13 **Participant**
So I would rather ask him to do the small stuff, the small checking stuff.
01:05:18 So I could focus on other stuff.
01:05:21 **Researcher**

01:05:24

Yeah, that makes a lot of sense.

Participant

So I go faster.

01:05:27

Researcher

That makes sense.