

Participant P12

00:07:16 > **Viewing file** (webpack.common.js)

00:07:16 **Participant**

Let me search for something on copilot.

00:07:21 I don't remember how I view the page

00:07:33 **Chat** (window: 0)

> **Agent**

github.copilot.default

> **Context**

viewport (webpack.common.js)

> **Message**

@codebase how to visualize a page

> **Response**

To visualize the project page, you can follow the steps below:

1. **Install the project dependencies:** Make sure you have installed all the dependencies listed in the [package.json](#) file. Run the following command in the integrated terminal of Visual Studio Code:


```
npm install
```
2. **Configure the development environment:** The [webpack.common.js](#) file suggests that you are using Webpack. Make sure there is a script in the [package.json](#) to start the development server, such as:


```
"scripts": {  
  "start": "webpack serve --config webpack.dev.js"  
}
```


If there isn't, add it.
3. **Start the development server:** Run the following command to start the server:


```
npm run start
```
4. **Access the application in the browser:** After starting the server, open the browser and access <http://localhost:8080> (or the port configured in the Webpack Dev Server).
If you encounter any issues, check the [webpack.dev.js](#) file to confirm the development server settings.

00:07:46 **Participant**

since JavaScript is not a language I use,

00:07:51 almost at all. So I don't remember much.

00:08:04 I'll run the recommendations he gave

00:08:06 in the chat.

00:08:39 Ah, did I write it wrong?

00:08:44 Ah, there's no start script.

00:08:54 Well, if copilot said that...

00:08:58 > **Viewing file** (package.json)

00:08:59 **Participant**

the package.json has to have a start script,

00:09:03 then I will

00:09:08 copy this script and put it here

00:09:11 > **Editing file** (package.json)

00:09:12 > **Implementing Copilot code** by copying part of snippet (package.json)

00:09:24 **Participant**

00:09:27 in the code
00:09:35 **> Editing file** (package.json)
00:09:35 **Participant**
And now, I'll try npm start.
00:09:45 Open the browser.
00:09:57 **Researcher**
I can't see, right? It would be nice if I could see the project
00:10:05 **Participant**
00:10:05 Ah, I can click here, the tab one.
00:10:12 To-dos, let me test and see what this does.
00:10:22 Yeah, okay, let's go back to the code. The task says
00:10:32 "When there are no to-dos, a red box appears with the message
00:10:36 no to-dos here yet. Change the color of this box
00:10:39 to blue."
00:10:45 Let me go back here to to-dos. I already did a test, I'll clear it.
00:10:53 I'll do what was there, a box really let me check
00:10:58 here. Blue is blue.
00:11:01 It changes things in the no, so, but a blue box already appears.
00:11:09 It's that he says to change the.
00:11:17 in the PDF it says, a red box appears with the message
00:11:21 no tasks here yet change the color of this box to blue. It's not red,
00:11:27 but he is specifying a specific blue here.
00:11:30 Maybe it's not this blue, so I'll try to change to this specific blue
00:11:35 Specific
00:11:41 Let me update this here for me.
00:11:47 It really is blue, isn't it?
00:11:53 Yeah, let's go back to the code.
00:12:04 **Researcher**
[omitted], sorry, I think... forgive me, if you can...
00:12:10 I sent you the wrong link.
00:12:12 That's why the box isn't blue. Sorry.
00:12:18 Hold on, just a minute while I change the link here,
00:12:27 Yeah, hold on, just a second, just a second.
00:12:36 I'm confirming here. You can
00:12:40 click on this one here.
00:12:45 Click on this one here, look, on this other one you can exit everything
00:12:48 completely from this workspace. You can click on this other one, sorry,
00:12:52 you can do the same thing just because then the box is blue.
00:13:03 Then you can start over and execute.
00:13:09 **Participant**
But here I'm already logged in.
00:13:11 **Researcher**
You're logged in, it just changed the location. Sorry.
00:13:20 I can't see.
00:13:44 **> Viewing file** (package.json)
00:13:49 **Participant**
So I will repeat the steps.
00:14:12 **Chat** (window: 1)
> Agent
github.copilot.default
> Context
viewport (package.json)
> Message
@codebase how to visualize a page
> Response

To visualize a page in your React project, you can follow the steps below:

1. Ensure the development server is running:

- In the terminal, run the command defined in the dev script of your [package.json](#):

```
npm run dev
```

- This will start the development server configured by webpack-dev-server.

2. Open the browser:

- The command `webpack serve --open` should automatically open the browser at the correct URL (usually `http://localhost:8080`).
- If it doesn't open, you can manually access the default development server URL.

3. Check the initial page code:

- Ensure that the entry point of your React application is correctly configured in Webpack (usually in the [webpack.dev.js](#) file).
- The main React file (like `index.js` or `App.js`) should render the root component.

If you need help configuring or debugging something, share more details about the configuration files or the React code.

00:14:14
- 00:14:21

> Response rendering

00:14:36

Participant

The response came a little different, but

00:14:47

I will run what he said.

00:14:56

It didn't work, then.

00:15:01

I will ask him again to see if the same response comes.

00:15:16

Chat (window: 1)

> Agent

github.copilot.default

> Context

codebase

> Message

#codebase how to visualize the page?

> Response

To visualize your React project page, follow the steps below:

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

```
npm install
```

- 2. Start the development server:** Use the script configured in [package.json](#) to start the server:

```
npm run dev
```

This will start the development server configured in the [webpack.dev.js](#) file. By default, it will be available at `http://localhost:8080`.

- 3. Open the browser:** The `npm run dev` command should automatically open the browser at the correct URL. If it doesn't, manually access `http://localhost:8080`.

- 4. Check the entry point:** The entry point of your application is defined in the `webpack.common.js` file:

```
entry: {  
  app: path.resolve(__dirname, "src", "index.js"),  
},
```

Make sure the `src/index.js` file is correctly configured to render the root component [App\(src/index.js\)](#).

5. **HTML structure:** The `public/index.html` file contains the `<div id="root">`
`</div>` element where React renders the application.

If you follow these steps and still can't visualize the page, share any error messages that appear in the terminal or browser console so I can help debug the problem.

00:15:27
- 00:15:34

> Response rendering

00:15:36

Participant

Ah, now it came npm install.

00:16:36

Now it's red.

00:16:39

a red box appears.

00:16:43

Change the color of this box to blue.

00:16:55

Done.

00:17:00

How

00:17:03

Could it be

00:17:07

No...

00:17:16

No, but well, if I ask him here how to change

00:17:20

the color

00:17:22

Researcher

Hmm, Hmm.

00:17:24

Participant

I can do that, but then I have to find a way to describe... if I want a specific answer, like...

00:17:28

Like "change the color of this box here". I don't trust my, like...

00:17:33

In finding a way to describe, that I have a guarantee that he will

00:17:40

understand which button I'm talking about,

00:17:50

so I prefer to look for the button in the code, because for me it's easier.

00:17:53

Researcher

Hmm, Hmm.

00:17:54

00:18:00

Participant

The mental effort is less for me to look for the button in the code than to try to describe here which button it is.

00:18:05

00:18:10

Researcher

Hum, OK.

00:18:14

Participant

00:18:16 So I'm going to look for this button in the code.
00:18:27 > **Viewing file** (package-lock.json)
00:18:27 **Participant**
00:18:31 Ah.
00:18:31 I know that normally color is in the CSS file.
00:18:33 > **Viewing file** (src/todo/app.css)
00:18:37 **Researcher**
00:18:38 Um.
00:18:38 **Participant**
00:18:40 But I need to find the button
00:18:40 > **Viewing file** (src/index.js)
00:18:42 **Participant**
00:18:48 first to find the
00:18:48 identifier.
00:18:53 > **Viewing file** (src/todo/app.jsx)
00:19:00 **Participant**
00:19:03 Components.
00:19:03 > **Viewing file** (src/todo/components/main.jsx)
00:19:08 **Participant**
00:19:25 What is written on the button again?
00:19:25 Here.
00:19:28 It's todo-list-empty-container.
00:19:33 It's the class name.
00:19:34 > **Viewing file** (src/todo/app.css)
00:19:36 **Participant**
00:19:41 It must have the name here in the CSS
00:19:41 but the color is not defined here.
00:19:49 Or is it? Ah, no, here are the colors.
00:19:57 todo-list-empty-message, here is the red color.
00:20:01 it's probably this color here, I'm going to change this and check
00:20:07 if it changes there.
00:20:12 > **Editing file** (src/todo/app.css)
00:20:13 **Participant**
00:20:23 This way is easier, in my opinion.
00:20:23 than finding a way to describe the button,
00:20:26 that I guarantee will be the right button.
00:20:35 I don't know if this has hot reload. It does. Changed to blue.
00:20:43 That was the first task.
00:20:51 Task B, change placeholder.
00:20:55 The placeholder text in the input field currently displays the question,
00:20:59 what needs to be done? Change the text to,
00:21:01 type a task here.
00:21:34 Is it written "what needs to be done"? it is written "what needs to be done".
00:21:43 So...
00:21:45 **Researcher**
00:21:47 Um.
00:21:47 **Participant**
00:21:51 Here I can use... I think it would be as easy to use the
00:21:51 copilot as a code search. Because as
00:21:55 I know it's a specific text, that's in a specific place.
00:22:03 If I search for this text, I will find it in all instances of this
00:22:08 text in the code.
00:22:09 So both, if I search with the copilot, search a file search,
00:22:14 I'll find it there using vscode all instances of it.
00:22:19 As I have the impression that if I search directly the files,

00:22:23 it will take less time than in the copilot I will search directly in the files.
00:22:35 > **Viewing file** (src/todo/components/header.jsx) through file search
00:22:39 **Participant**
Then it already shows me where it is written "what needs to be done" and here it says to
00:22:42 **Researcher**
Hum?
00:22:44 **Participant**
replace with, type a task here.
00:22:48 > **Editing file** (src/todo/components/header.jsx)
00:22:56 **Participant**
Let's update and check if this was changed.
00:23:01 It was changed, I will do a test.
00:23:10 OK, task b done and verified.
00:23:14 Task c. Currently, the tasks are displayed in the order they
00:23:17 were added. Modify the application so that the
00:23:22 tasks are sorted alphabetically.
00:23:27 Ah, this is more complicated.
00:23:40 I personally don't remember
00:23:44 if in the... well I don't know react and I don't remember if in the
00:23:56 frameworks it already comes... probably there must be...
00:24:02 tools that automatically sort the lists
00:24:07 There must be. So we would have to use something
00:24:14 of these already sorted.
00:24:19 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:24:21 **Participant**
Let me understand the code.
00:24:25 Probably each task is a component
00:24:34 React inserts into the page and organizes. Then each task being a
00:24:40 component replicates an entire internal
00:24:45 structure, so it uses some way
00:24:48 to sort and show in what order this is displayed. If we see which
00:24:54 > **Viewing file** (src/todo/components/main.jsx)
00:24:59 **Participant**
algorithm it was using to sort or where this description is
00:25:07 It's here... Oh...
00:25:11 todo-list, it must be this here. Let's find it.
00:25:19 > **Viewing file** (src/todo/components/item.jsx)
00:25:23 **Participant**
This must be an item.
00:25:35 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:25:47 **Participant**
This is something I wouldn't know how to do right away,
00:25:50 I would have to look on the internet how to do it or now, since I have copilot
00:25:55 open here I will search, ask it.
00:26:09 How do I know the identity of the thing
00:26:18 > **Viewing file** (src/todo/components/main.jsx)
00:26:26 **Participant**
the todo-list.
00:26:29 **Chat** (window: 1)
> **Agent**
github.copilot.default
> **Context**
codebase
> **Message**
#codebase how to reorder todo-list?

> Response

To reorder the to-do list ([todo-list \(src/todo/components/item.jsx\)](#)), you can implement a "drag-and-drop" feature or add buttons to move items up or down. Here is a simple approach using buttons to reorder:

1. Add reorder actions in the reducer

In the file `src/todo/reducer.js`, add two new action types: `MOVE_ITEM_UP` and `MOVE_ITEM_DOWN`.

```
File: src/todo/reducer.js
export const todoReducer = (state, action) => {
  let newState;
  switch (action.type) {
    // ...existing cases...
    case "MOVE_ITEM_UP":
      newState = [...state];
      const upIndex = action.payload.index;
      if (upIndex > 0) {
        [newState[upIndex - 1], newState[upIndex]] =
        [newState[upIndex], newState[upIndex - 1]];
      }
      break;
    case "MOVE_ITEM_DOWN":
      newState = [...state];
      const downIndex = action.payload.index;
      if (downIndex < state.length - 1) {
        [newState[downIndex], newState[downIndex + 1]] =
        [newState[downIndex + 1], newState[downIndex]];
      }
      break;
    default:
      throw Error(`Unknown action: ${action.type}`);
  }

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

2. Add reorder buttons in the component [Item \(src/todo/components/item.jsx\)](#)

In the file `src/todo/components/item.jsx`, add buttons to move the items up or down.

```
File: src/todo/components/item.jsx
export const Item = memo(function Item({ todo, dispatch, index }) {
  const moveUp = useCallback(() => dispatch({ type: "MOVE_ITEM_UP",
    payload: { index } })), [dispatch, index]);
  const moveDown = useCallback(() => dispatch({ type:
    "MOVE_ITEM_DOWN", payload: { index } })), [dispatch, index]);

  return (
    <li className={classnames({ completed: todo.completed })} data-
    testid="todo-item">
      <div className="view">
        {/* Existing code */}
        <button className="move-up" onClick={moveUp} data-
        testid="move-up-button">
          ↑
        </button>
      </div>
    </li>
  );
});
```

```
        <button className="move-down" onClick={moveDown} data-  
testid="move-down-button">  
          ↓  
        </button>  
      </div>  
    </li>  
  );  
});
```

3. Update the CSS for the buttons

In the file `src/todo/app.css`, add styles for the reorder buttons.

```
File: src/todo/app.css  
.move-up,  
.move-down {  
  background: #007bff;  
  color: white;  
  border: none;  
  padding: 5px;  
  margin: 0 5px;  
  cursor: pointer;  
  border-radius: 3px;  
}  
  
.move-up:hover,
```

```
.move-down:hover {  
  background: #0056b3;  
}
```

4. Test the functionality

- Start the development server:

```
npm run dev
```

- Add some items to the list and use the buttons to reorder them.

This approach allows you to reorder the list items simply and efficiently. If you need something more advanced, like "drag-and-drop," I can help implement it!

> Response rendering

Participant

Yeah, actually, the question wasn't good because I should have put how to sort alphabetically.

I'll ask again.

It's still running....

Chat (window: 1)

> Agent

github.copilot.default

> Context

codebase

> Message

#codebase how to sort todo-list alphabetically?

> Response

To sort the to-do list ([todo-list \(src/todo/components/item.jsx\)](#)) alphabetically, you can use the sort method on the task array before rendering them. Here's how you can implement this:

1. Add a sorting function in the component [Main \(src/todo/components/main.jsx\)](#)

In the file `src/todo/components/main.jsx`, modify the code to sort the items before rendering them.

```
File: src/todo/components/main.jsx  
export function Main({ todos, dispatch }) {  
  const { pathname: route } = useLocation();  
  const [currentPage, setCurrentPage] = useState(1);  
  const [lastAddedItem, setLastAddedItem] = useState(null);  
  
  const itemsPerPage = 5;  
  
  // Filter items based on the route and sort alphabetically  
  const visibleTodos = useMemo(() => {  
    const filteredTodos = todos.filter((todo) => {  
      if (route === "/active") return !todo.completed;  
      if (route === "/completed") return todo.completed;  
      return todo;  
    });  
  
    // Sort alphabetically by title  
    return filteredTodos.sort((a, b) =>  
      a.title.localeCompare(b.title));  
  }, [todos, route]);
```

00:26:41
- 00:27:10

00:26:47

00:26:51

00:26:54

00:27:23

00:27:24

```

    const toggleAll = useCallback((e) => dispatch({ type: TOGGLE_ALL,
payload: { completed: e.target.checked } })), [dispatch]);

    useEffect(() => {
      if (todos.length > 0) {
        const lastItem = todos[todos.length - 1];
        setLastAddedItem(lastItem);
      }
    }, [todos, lastAddedItem]);

    return (
      <main className="main" data-testid="main">
        {visibleTodos.length > 0 ? (
          <>
            <div className="toggle-all-container">
              <input
                className="toggle-all"
                type="checkbox"
                id="toggle-all"
                data-testid="toggle-all"
                checked={visibleTodos.every((todo) =>
todo.completed)}
                onChange={toggleAll}
              />
              <label className="toggle-all-label"
htmlFor="toggle-all">
                Toggle All Input
              </label>
            </div>
            <ul className={classnames("todo-list")} data-
testid="todo-list">
              {visibleTodos.map((todo, index) => (
                <Item todo={todo} key={todo.id} dispatch=
{dispatch} index={index} />
              ))}
            </ul>
          </>
        ) : (
          <div className="todo-list-empty-container">
            <span className="todo-list-empty-message">No tasks
here yet</span>
          </div>
        )}

        <paginationControls
          currentPage={currentPage}
          itemsPerPage={itemsPerPage}
          setCurrentPage={setCurrentPage}
          totalItems={visibleTodos.length}
        />
      </main>
    );
  }
}

```

2. Test the functionality

1. Start the development server:

```
npm run dev
```

2. Add some items to the list with different titles.

3. Check if the items are displayed in alphabetical order.

3. Considerations

- The sorting is done only for display. If you want to save the sorted list in the global state, you will need to implement the logic in the [todoReducer \(src/todo/reducer.js\)](#) and dispatch an action to sort the items.
- If you want to switch between different types of sorting (alphabetical, by date, etc.), I can help implement a sorting selector.

> Response rendering

Participant

To automatically sort the to-do list you can use the sort method in the components/main file. That's what I have open.

Is that where the code is?

Yeah, he sent me the code here. I would have preferred if he had sent just

the part that changed. Now I'll have to see here what the difference is between this code and what is already written.

Pretty big, I think it didn't change everything.

Well, but he said there is sort, so is there some sort here?

I'll find it just by looking

copy

part

00:27:33
- 00:27:47

00:27:43

00:27:49

00:27:56

00:28:02

00:28:08

00:28:11

00:28:16

00:28:23

00:28:32

00:28:41

00:28:47

00:28:49

00:28:57 Here.

00:29:18 Is there a way for me to search for a word in this response?

00:29:25 I don't want to just copy the entire code and paste it.

00:29:29 I want to know that I'm changing little in the code that already existed.

00:29:35 **Researcher**

00:29:37 Hmm?

00:29:37 Are you trying to search now, did you try to copy?

00:29:41 **Participant**

00:29:41 No, not yet, not yet.

00:29:57 I don't know if control + F works here.

00:30:09 No.

00:30:09 Control + F only searches in the files.

00:30:13 I wanted to search in the copilot's response.

00:30:21 let me look better here... open the window more.

00:30:27 I'll find this sort... here! it sorts alphabetically by title

00:31:52 Ah, I can... It's returning this here,

00:31:56 so if I change this here for this,

00:32:03 it should probably work.

00:33:03 **> Editing file** (src/todo/components/main.jsx)

- 00:33:10

00:33:14 **Participant**

00:33:14 Then I'll take this here.

00:33:17 **> Implementing Copilot code** by copying part of snippet
(src/todo/components/main.jsx)

00:33:20 **Participant**

00:33:20 Probably there's already the only change. It gave an error because there's something extra.

00:33:40 That's right.

00:33:45 **> Editing file** (src/todo/components/main.jsx)

- 00:33:51

00:33:47 **Participant**

00:33:47 Here it opens a bracket that closes.

00:34:17 Apparently it's working.

00:34:20 I'll check if it's working.

00:34:33 It's not working, I broke the project.

00:34:38 Ah, my first option was to do this.

00:34:42 It was to try to change the code as little as possible.

00:34:45 **Researcher**

00:34:45 Hum, Hum.

00:35:01 **Participant**

00:35:01 I'll try to take this part here. I hadn't

00:35:05 defined

00:35:05 **> Implementing Copilot code** by copying part of snippet
(src/todo/components/main.jsx)

00:35:10 **Participant**

00:35:10 Save.

00:35:59 **Researcher**

00:35:59 No, but it's, it's all right, it's just as an example,

00:36:04 even for if you don't have ideas

00:36:10 **Participant**

00:36:10 It's that before it wasn't alphabetical, it was in order of description.

00:36:16 Since I didn't test before, I don't know if it was already like this or not

00:36:17 **Researcher**

00:36:17 That's it.

00:36:21 No, no, it wasn't.

00:36:31 I sent you one that we have

00:36:34 a few minutes still to finish the 40 minutes there that we talked about

00:36:41 doing the tasks. I sent you a second file and we

00:36:45 could start
00:36:49 on that file, until when it's all right for you,
00:36:52 I think we can look at one more. The first task from there.
00:36:52 **Participant**
Good.
00:36:57 **Researcher**
To me, a task from there can be the first,
00:37:00 but you will see there that you can choose.
00:37:46 **Participant**
So, if I receive a task like this before
00:37:50 asking probably, etc., as it is not making a distinction,
00:37:57 whether to persist in a database or in the browser.
00:38:03 I would clear this doubt. Because if it were to persist in the database, then...
00:38:11 first, if I don't know the database,
00:38:14 I would research which database to use and.
00:38:17 Research how to connect with this database, and which framework to use... if not
00:38:24 that, if it were just to persist in the session, persist
00:38:30 in the user's browser. Then yes, I would see if, like, it already has this
00:38:36 Implemented in react and maybe use probably to
00:38:42 see how to implement this without changing the code too much.
00:38:50 **Researcher**
Hmm, this is a question for me too, isn't it? It's to persist in the browser.
00:38:58 **Participant**
Persisting, browser.
00:39:05 I will assume that
00:39:12 react has something ready for this. Since I don't know react,
00:39:16 I will ask then, directly.
00:39:29 **Chat** (window: 1)

> **Agent**

github.copilot.default

> **Context**

codebase

> **Message**

#codebase how to persist tasks in the browser?

> **Response**

To persist tasks in the browser, you can use [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)). The code already has a partial implementation to load and save the state in [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)) in the file `src/todo/reducer.js`. Here's how you can ensure that tasks are saved and loaded correctly:

1. Check the load and save function in the reducer

In the file `src/todo/reducer.js`, the functions [loadState \(src/todo/reducer.js\)](#) and [saveState \(src/todo/reducer.js\)](#) are already implemented. Make sure the state is being saved and loaded correctly.

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

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

2. Ensure the initial state is loaded

In the file `src/todo/app.jsx`, the initial state is already being loaded from [localStorage](#) (`node_modules/@types/node/globals.d.ts`) through the function [loadState](#) (`src/todo/reducer.js`). Make sure the [initialState](#) (`src/todo/reducer.js`) is being passed correctly to the [useReducer](#) (`node_modules/@types/react/index.d.ts`).

```
File: src/todo/app.jsx
import { todoReducer, initialState } from "../reducer";

export function App() {
  const [todos, dispatch] = useReducer(todoReducer, initialState);

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

3. Update the state in [localStorage](#) (`node_modules/@types/node/globals.d.ts`) after each action

In the file `src/todo/reducer.js`, the [saveState](#) (`src/todo/reducer.js`) is already being called after each action in the [todoReducer](#) (`src/todo/reducer.js`). Make sure all relevant actions (such as add, update, remove, etc.) are calling [saveState](#) (`src/todo/reducer.js`).

```
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:
      newState = action.payload;
      break;
    default:
      throw Error(`Unknown action: ${action.type}`);
  }

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

4. Test persistence

1. Start the development server:

```
npm run dev
```

2. Add some tasks.

3. Refresh the page and check if the tasks are still present.

5. Considerations

- Make sure the browser supports [localStorage](#) ([node_modules/@types/node/globals.d.ts](#)).
- If you need a more robust solution (like synchronization with a backend), I can help implement that.

> Response rendering

Participant

Ah, okay. He said he had already started, there's already a local storage, because he had already started to implement and then he gives steps here to do it.

I think I would follow these steps because,

since he didn't give any code page here,

I think I would just try to copy this

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

Participant

Do you already have the old state?

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

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

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

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

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

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

Participant

no, it's not the same, okay? Not only is it wrong...

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

Participant

I think it saved everything.

I think you have to run

again.

Close.

Open again.

It didn't disappear, so,

I think if I were running locally.

Researcher

You can try just today just update to see if it stays

add some and update, I think that's the test that's there.

Participant

Ah, okay.

Yeah, it didn't disappear, it didn't work. Maybe I...

Then I'll see if I had something here that I didn't.

Didn't do.

But that's it. I think time's up.

Researcher

Hmm, Hmm.

Participant

Detailed? good.

Length? most of the answers were good, but there was one that was bad,

00:39:39
- 00:39:53

00:40:07

00:40:12

00:40:26

00:40:29

00:40:36

00:40:53

00:41:00

00:41:22

00:41:41

00:41:42

00:41:52

00:41:54

00:42:14

00:42:24

00:42:28

00:42:31

00:42:43

00:42:46

00:43:02

00:43:09

00:43:15

00:43:18

00:43:20

00:43:25

00:43:31

00:43:39

00:43:44

00:43:53

00:43:56

00:43:59

00:46:04

00:46:07

00:46:11 so I'll put neutral.
00:46:12 **Researcher**
Hmm
00:46:15 **Participant**
The technical terms? good, it was great, it was successful.
00:46:21 Correct?
00:46:23 Well, the first ones worked. The last one didn't work,
00:46:27 but I doubt more my...
00:46:30 I doubt more, like, that I didn't understand what to do
00:46:35 than that he didn't say it right.
00:46:37 What to do then? I'll put good.
00:46:38 **Researcher**
OK?
00:46:42 **Participant**
Following my instructions? Yes, followed my instructions,
00:46:47 but I was also very succinct in the questions. So, I'll put good,
00:46:54 because I wanted a way to
00:47:00 write things more easily.
00:47:07 But I think it doesn't depend on
00:47:09 the copilot. It depends on me having more experience to
00:47:14 write these things or the tool itself. Like, react,
00:47:20 having terms that are easier to describe.
00:47:26 **Researcher**
To write more easily what do you mean by that?
00:47:30 **Participant**
The first task there, like, Ah, how to change the color
00:47:35 I don't know how to change the color. I found it easier to look in the code
00:47:38 than to ask the copilot.
00:47:40 how to do it.
00:47:41 Why? Because I don't know how to describe the box
00:47:44 that is, that needs to change color, like you would say "Ah, that box,
00:47:48 **Researcher**
that is
00:47:49 **Participant**
called X" then "how to change the X that is written?", like that.
00:47:54 **Researcher**
OK, I understand.
00:48:02 **Participant**
Use of formatting not formatting was very good, you know? I liked it.
00:48:13 I agree, using the copilot in my work
00:48:18 Allows me to do tasks faster? I think so.
00:48:22 If I worked with it, I agree, I think it depends on the task,
00:48:28 but probably yes.
00:48:31 using the copilot will improve my performance at
00:48:35 work?
00:48:38 It will help me understand the work better. I don't know if it will improve.
00:48:43 As a consequence, yes, so a little.
00:48:48 using the copilot at my work, will it increase productivity? Yes.
00:48:53 And then you ask people less. You ask the copilot more.
00:49:06 Will the copilot improve the effectiveness of my work? No,
00:49:11 it won't get worse, but it won't make you more effective.
00:49:18 Using the copilot will make the work easier? yes.
00:49:24 Mainly study.
00:49:26 Do I find using the copilot useful?
00:49:33 A little, but it depends on the area.
00:49:37 Would learning to use the copilot be easy? Yes, yes, I think even extremely.

00:49:47 There would be very few difficulties.
00:49:51 I Believe
00:49:54 it is easy to get the copilot to do what I want it to do? no, neutral,
00:50:02 neither agree nor disagree. There are some things that are easy.
00:50:07 There are some things that are difficult. My interaction with the copilot would be
00:50:14 Clear, understandable?
00:50:20 Neutral too, because I have to know exactly what
00:50:23 I want to search... the question I trust,
00:50:28 that I believe to be succinct, that I trust the answer.
00:50:32 **Researcher**
Hum, Hum.
00:50:32 **Participant**
I think if I were very.
00:50:36 If I wrote a lot
00:50:38 I don't trust the copilot's answer because
00:50:44 It would have to search for several things, and gather several things.
00:50:47 The more I described in detail,
00:50:50 the less I would trust the answers.
00:50:59 Would I find the copilot flexible to interact? yes.
00:51:03 yes, like I could use it for various
00:51:07 things.
00:51:08 Would it be easy to become good at interacting with the copilot? yes
00:51:14 Did I find the copilot easy to use? Yes, for some tasks, for most yes,
00:51:22 depending on the use, yes.
00:52:06 As I finished the first 3,
00:52:09 I'm going to put this here at the maximum, because I finished just didn't finish the extras.
00:52:12 **Researcher**
Hum, Hum.
00:53:59 **Participant**
These are very realistic things that people have to do.
00:54:09 **Researcher**
I just wanted to go through some points here, and some things
00:54:14 You even mentioned as you were answering the survey
00:54:24 But I wanted to understand if you managed to orient yourself with the code and how
00:54:29 did the copilot help you?
00:54:35 **Participant**
As the task was well written about like,
00:54:45 what needed to change... having a general notion of like,
00:54:50 how the code is organized, I managed to find things in the code
00:54:56 Several times without asking the copilot.
00:55:00 mainly because this code was small.
00:55:03 but even if the code were larger,
00:55:06 using text search in the code, I find it easy to find things in the code,
00:55:11 from my experience. What was the full question again?
00:55:17 **Researcher**
How did you manage to orient yourself. If you managed to orient yourself in the code
00:55:23 and how did the copilot influence you or if it influenced you?
00:55:28 **Participant**
The copilot, I think I would use it more not
00:55:31 to find things in the code that need to be changed, but more for
00:55:36 how to change things.
00:55:41 Code organization is something that is general, so
00:55:45 it depends on your level of experience and the copilot would be more for like.
00:55:49 "What do I need to do?"
00:55:50 "What do I need to change?"
00:55:52 And I would expect it to give me answers

00:55:56 more objective and short. Just like, "Oh, change in this place, this place,"
00:56:02 "this place."
00:56:05 **Researcher**
00:56:09 Hmm, Hmm, and if it were in the old times,
00:56:14 before all these new technologies, how would you do it?
00:56:19 without the copilot how would you solve these tasks?
00:56:25 what strategies, what exactly would you do?
Participant
00:56:29 I would search on the internet, how to do this
00:56:34 Probably some sites with quick tips about this.
00:56:36 Stack overflow, things like that.
00:56:42 Or those JavaScript tutorial sites,
00:56:48 react tutorial
00:56:51 about how to do these things.
Researcher
00:56:55 Uh-huh. And then, you also mentioned that you use
00:57:02 search a lot, through the code as well.
00:57:05 OK, do you feel that you managed to understand the
00:57:08 codebase, the code that was there which was,
00:57:11 it's a code
00:57:16 new, right? That you hadn't seen before.
Participant
00:57:22 Yes, a little bit like that I have like. I had a bit of experience
00:57:25 in frontend code, so I
00:57:26 could have an idea of what things were doing,
Researcher
00:57:30 Uh-huh.
Participant
00:57:34 but I couldn't explain what each term there does,
00:57:39 wouldn't know how to describe. I managed to have an idea from what I already
00:57:40 know of what it is.
00:57:41 What that code was doing.
00:57:48 But if it were to implement, the whole thing, I wouldn't know how to do it.
Researcher
00:57:52 There are some little things from that Survey you just
00:57:56 answered, especially there when you were talking
00:57:59 about the size of the response, the quality of the response
00:58:04 The correct was and everything else. You indicated there things you
00:58:06 liked and didn't like.
00:58:09 Could you tell me some aspects of the of the
00:58:16 response and the interaction with the copilot that stood out to you?
Participant
00:58:19 I liked that it comes well formatted. The response.
Researcher
00:58:25 Uh-huh.
Participant
00:58:30 I liked when it gave answers
00:58:33 about changing specific things. I intuitively,
00:58:38 believe that these changes are in very small parts of the code.
00:58:43 When it gave the answer to change just, like, the lines or the line,
00:58:48 I preferred it over when it gave an answer to change a whole page
00:58:53 entirely. And also I liked that he, when he gave the answer for each line,
00:58:56 he gave a
00:58:59 slight indication of why that line needed to be changed.
00:59:03 If I asked more, he would explain better why that
thing does. So, if I wanted to use the copilot to

00:59:06

learn, maybe I could do that.

00:59:13

Researcher

Do you feel that your interaction
with the copilot has changed over time?
There was one aspect that didn't change.

00:59:18

00:59:25

00:59:30

Participant

I still believe in asking
short questions and I think for now it's like this...
I think I'll keep doing that.

00:59:35

00:59:43