

Participant P08

00:04:21 **Participant**
So I'm gonna check first what files I have and especially the package dot JSON,
00:04:21 > **Viewing file** (package.json)
00:04:26 **Participant**
because this is where mostly where you run JavaScript programs.
00:04:32 So you can see that it has script dev and it does webpack serve,
00:04:35 so I guess this is just where npm.
00:04:39 First checking if there's no yarn lock file, but.
00:04:45 Webpack is not found, guess I'll.
00:04:50 Install webpack then.
00:04:53 Should I run, maybe?
00:05:02 Oh, oh, no.
00:05:04 Yes.
00:05:10 First I need to install. Of course my dependencies which is.
00:05:25 OK.
00:05:30 **Researcher**
Perfect, so feel free to get started on the on the
00:05:33 tasks.
00:05:34 **Participant**
OK.
00:05:37 There are no todos, a red box, no todos yet. Change the color blue. OK,
00:05:41 copying the blue color.
00:05:46 And guessing it's in either in the. Oh yeah, it's in todo, I guess.
00:05:48 > **Viewing file** (src/todo/app.css)
00:05:50 **Participant**
And then.
00:05:53 There should be empty message.
00:05:55 > **Editing file** (src/todo/app.css)
00:05:57 **Participant**
And it should recompile automatically and if I refresh it's in blue.
00:06:08 Yep.
00:06:11 Then the placeholder text, enter a task here.
00:06:18 And.
00:06:23 This is a lot of files.
00:06:24 So I'll just do.
00:06:24 Maybe a global search?
00:06:28 > **Viewing file** (src/todo/components/header.jsx) through file search
00:06:29 **Participant**
Yep. And here we can see the placeholder text.
00:06:33 > **Editing file** (src/todo/components/header.jsx)
- 00:06:40
00:06:45 **Participant**
OK.
00:06:49 Yeah, that looks all good to me and I guess.
00:06:54 This is also done then and then the last one sorts on.
00:07:01 Alphabetical order, OK.
00:07:05 Then I see that this is.
00:07:08 A React codebase and then if I add an item.
00:07:14 Callback title. Type add item, payload.
00:07:22 Oh, on submit add item.
00:07:26 > **Viewing file** (src/todo/constants.js)

00:07:28 **Participant**
And I can go to the. Oh, no, no, no.

00:07:32 > **Editing file** (src/todo/components/header.jsx)

00:07:36 **Participant**
Hopefully my progress is saved.

00:07:37 I guess so. OK.

00:07:38 **Researcher**
Yes, yeah, everything's saved.

00:07:39 So no worries.

00:07:41 > **Editing file** (src/todo/app.css)

00:07:42 **Participant**
OK.

00:07:43 I guess I have to wait again.

00:07:48 I think I'm going to be pressing that button a lot, but.

00:07:50 > **Viewing file** (src/todo/components/item.jsx)

00:07:53 **Participant**
OK.

00:07:53 So it adds new item. So I guess I'll just look in items.

00:08:01 No, this is only a single item, so not in here.

00:08:02 > **Viewing file** (src/todo/components/input.jsx)

00:08:05 **Participant**
Like.

00:08:07 Yes.

00:08:11 These are all the inputs, but not still not the list.

00:08:17 > **Viewing file** (src/todo/components/main.jsx)

00:08:21 **Participant**
Ah, here I can see visible todos.

00:08:29 This is where it filters the todos.

00:08:31 And then I'll guess I can also just make sure that it dot sort.

00:08:34 > **Editing file** (src/todo/components/main.jsx)
- 00:08:39

00:08:39 **Participant**
OK, hello.

00:08:46 **Researcher**
Are you being interrupted when typing?

00:08:48 **Participant**
Yes, it needs to.

00:08:50 It can't access my clam board, clipboard.

00:08:54 **Researcher**
And and when you press close on this pop up?

00:09:01 > **Editing file** (src/todo/components/main.jsx)

00:09:07 **Researcher**
Is it still not working?

00:09:10 > **Editing file** (src/todo/components/main.jsx)

00:09:13 **Participant**
I can only press again and more information.

00:09:19 Maybe.

00:09:21 **Researcher**
So is it actually interrupting you while typing or trying to play something?

00:09:24 **Participant**
Yeah, well, it can't.

00:09:28 I can't make it type.

00:09:37 > **Editing file** (src/todo/components/main.jsx)

00:09:37 **Participant**
Yeah. So I if I press one button then it just

00:09:40 opens this.
00:09:43 Paste window.
00:09:44 **Researcher**
And this is with.
00:09:47 > **Editing file** (src/todo/components/main.jsx)
00:09:49 **Researcher**
OK.
00:09:49 So if you're trying to copy and paste, you might need to use control shift C and
00:09:54 control shift shift V.
00:09:56 > **Editing file** (src/todo/components/main.jsx)
- 00:09:59
00:09:56 **Participant**
Yeah, but I'm just. Oh, but that does work.
00:09:58 But I can't just type anymore.
00:10:03 **Researcher**
Wow.
00:10:05 **Participant**
Maybe you just need to refresh again.
00:10:14 Hmm.
00:10:27 **Researcher**
I'm trying to look into how to solve this issue now, sorry for this.
00:10:31 **Participant**
No, no.
00:10:34 Maybe it's the keyboard layout thing?
00:10:37 > **Editing file** (src/todo/components/main.jsx)
- 00:10:45
00:10:40 **Participant**
Oh, now it's works again, I guess.
00:10:41 **Researcher**
Now it works.
00:10:42 Let's hope it, let's hope it keeps working.
00:10:46 **Participant**
Umm.
00:10:50 And technically, I think because this has filter,
00:10:53 I guess this just an array or array like so I guess then dot sort should work.
00:11:03 Guess I'll restart it again to see if it works.
00:11:08 Now if I do this.
00:11:20 Uh.
00:11:33 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:11:33 **Participant**
Is this one working?
00:11:35 Sorry, I'm seeing all kinds of errors in here in
00:11:38 > **Viewing file** (src/todo/components/main.jsx)
00:11:38 **Participant**
this code that it doesn't know.
00:11:42 Pagination controls, but it should see it.
00:11:49 Uh.
00:11:51 Is defined but never used and then on the bottom here it is used.
00:11:59 Hmm.
00:12:04 > **Editing file** (src/todo/components/main.jsx)
00:12:09 > **Editing file** (src/todo/components/main.jsx)
- 00:12:16
00:12:26 **Participant**
It is not a problem. But why.
00:12:29 Is this then a problem?
00:12:32 Maybe it added a bracket too much because this one this one.
00:12:39 > **Viewing file** (src/index.js)

00:12:43 > **Viewing file** (src/todo/components/main.jsx)
00:12:44 > **Editing file** (src/todo/components/main.jsx)
- 00:12:51
00:13:00 > **Viewing file** (src/todo/components/paginationControls.jsx)
00:13:01 **Participant**
OK, so the re-, the, this is kind of strange.
00:13:05 It does have the.
00:13:08 Correct inputs, the props.
00:13:11 > **Viewing file** (src/todo/components/main.jsx)
00:13:12 **Participant**
So why does this?
00:13:18 > **Editing file** (src/todo/components/main.jsx)
- 00:13:29
00:13:27 **Participant**
No, I don't think there's anything wrong with the
00:13:30 input statement, but I can make it auto import maybe?
00:13:35 > **Editing file** (src/todo/components/main.jsx)
- 00:13:56
00:13:35 **Participant**
Oh, isn't this a React thing?
00:13:38 That everything should start with capital if it's.
00:13:48 Hmm.
00:13:51 Going back and then pagination controls. Yep.
00:13:55 And then capital P as well.
00:13:57 No idea why that was changed, but sure.
00:14:02 So now it should work again.
00:14:07 Yeah, and they're not sorted alphabetically yet,
00:14:11 so.
00:14:13 Hmm.
00:14:16 I'll do it now.
00:14:17 > **Editing file** (src/todo/components/main.jsx)
- 00:14:19
00:14:19 **Participant**
Dot sort.
00:14:22 And I guess the normal one is alphabetical, but I'll gonna see now.
00:14:34 It is not, but it's also not sorting them.
00:14:42 Because probably todo is yeah, OK. I need to sort on.
00:14:49 How does that work again.
00:14:53 > **Editing file** (src/todo/components/main.jsx)
- 00:14:58
00:15:01 **Participant**
Hmm.
00:15:07 Gues I'm going to ask Copilot now.
00:15:18 I'm going to ask it to sort a struct.
00:15:24 Or object.
00:15:28 Want one.
00:15:53 And then.
00:15:56 Checking.
00:15:58 This is fine, Does it have anything else? No.
00:16:02 **Chat** (window: 0)
> **Agent**
github.copilot.default
> **Context**
viewport (src/todo/components/main.jsx)
> **Message**
I want to sort an array in js. It is a struct or object. I want to sort on one field of the
struct. It should be alphabetically.
> **Response**

To sort an array of objects alphabetically by a specific field, you can use the `Array.prototype.sort()` method. Here's an example:

```
const todos = [
  { id: 1, title: "Write code" },
  { id: 2, title: "Review PRs" },
  { id: 3, title: "Test application" }
];

// Sort alphabetically by the "title"
todos.sort((a, b) => a.title.localeCompare(b.title));

console.log(todos);
```

Explanation:

- `localeCompare()` is used to compare strings in a case-sensitive and locale-aware manner.
- The `sort()` method modifies the original array.

If you want to ensure the sorting doesn't mutate the original array, you can create a new sorted array like this:

```
const sortedTodos = [...todos].sort((a, b) =>
a.title.localeCompare(b.title));
```

00:16:10

Participant

Yes, but then I need to define. Oh.

00:16:13

That's how it works, yeah.

00:16:16
- 00:16:25

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

00:16:18

Participant

So when I sort, I also need to have a function from which sorts these two.

00:16:22

Then A dot.

00:16:26

Checking how the todos are shaped.

00:16:29

They are.

00:16:37

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

00:16:42

Participant

I guess defined in the app dot JSX, todos.

00:16:43

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

00:16:48

Participant

Todo reducer from reducer.

00:16:48

Umm.

00:16:51

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

00:17:01

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

00:17:03

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

00:17:03

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

00:17:05

Participant

Uh.

00:17:08

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

00:17:11

Participant

Maybe they are.

00:17:12

I'm trying to see how the todo items actually look like.

00:17:15

I see that they have an ID, a payload.

00:17:19

00:17:23 I'm guessing that they should change.
00:17:28 On the titles.
00:17:31 So.
00:17:32 I'm just gonna go.
00:17:34 With.
00:17:43 They have an ID and a title, so I guess it's just the title.
00:17:47 **> Viewing file** (src/todo/components/main.jsx)
00:17:48 **Participant**
I'm gonna go back here and sort on A dot title.
00:17:49 **> Editing file** (src/todo/components/main.jsx)
- 00:18:03
00:18:07 **Participant**
And.
00:18:15 Still says there's one error, but it's not really up to date.
00:18:20 Oh.
00:18:22 It's trying to.
00:18:24 Write.
00:18:27 I'm just going to copy it to be sure.
00:18:34 **> Viewing file** (src/todo/reducer.js)
00:18:39 **Participant**
Did I accidentally remove load state?
00:18:44 **> Editing file** (src/todo/reducer.js)
- 00:18:45
00:18:44 **Participant**
Huh.
00:19:39 Oh no, now it's back.
00:19:42 Uh.
00:19:44 OK, this file looks fine.
00:19:45 **> Viewing file** (src/todo/components/main.jsx)
00:19:46 **Participant**
Looks fine again.
00:19:47 This one's still fine so.
00:19:51 Everything's saved, everything working.
00:20:32 We can close, stop this for now.
00:20:39 OK, no?
00:20:44 OK.
00:20:51 OK.
00:20:51 Here's my todo app.
00:20:57 And it's finally.
00:21:00 Sorting as it should.
00:21:04 Should it?
00:21:08 Yes.
00:21:12 OK.
00:21:12 I'm guess these are all.
00:21:15 Oh, I need to do these.
00:21:44 123 brown fox quick the. Yep, that's it.
00:21:47 **Researcher**
Perfect. Thanks.
00:21:50 I'll send you the actual tasks now.
00:21:53 Thanks for thinking aloud the way you're doing, by the way.
00:21:56 That's really helping a lot.
00:22:00 **Participant**
Yeah, I know.
00:22:00 That I should do it a little bit more but.
00:22:03 **Researcher**
No, you're you're doing fine.
If.

00:22:06

00:22:08

00:22:11

Should I notice that you're not doing it, I'll I'll remind you.

Participant

Hmm.

00:22:11

Researcher

So no worries about that.

00:22:13

Participant

Sure. OK.

00:22:17

Ensure that state.

00:22:25

OK so.

00:22:28

My first instinct would be to save all these todos.

00:22:32

In.

00:22:35

In my session storage so.

00:22:40

I'll.

00:22:41

Maybe first I'll go to the reducer because this is where all these things

00:22:42

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

00:22:45

Participant

end up.

00:22:48

But here there's load state.

00:22:52

And save state OK.

00:22:54

So save state is a function which I want to click. Oh, it's on top here.

00:22:58

It is local storage.

00:23:00

It does item todos and here it loads it again.

00:23:04

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

- 00:23:06

00:23:04

Participant

But I can see that there's a discrepancy in the string that it stores it under and this should.

00:23:09

Hopefully fix it.

00:23:10

Maybe it already did it. No, but when I test, then refresh again.

00:23:13

Still no.

00:23:20

Todos, todos, stringify state.

00:23:31

When does it call save state?

00:23:36

Every time we add something again.

00:23:40

So I'm just gonna.

00:23:42

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

00:23:48

- 00:23:56

00:23:50

Participant

Put some console log that I know that it's actually saving it or.

00:23:58

And when does it call load state, it does it when we have to call the initial state.

00:24:02

But we don't.

00:24:05

Use initial state anywhere I guess.

00:24:08

Just to make sure that it is at least saving I'll I'll make sure to put it here so that you can see it.

00:24:16

So when I add a task.

00:24:22

I.

00:24:23

Uh.

00:24:27

OK, it's not.

00:24:30

It's not saving it and also is there anything in my local storage?

00:24:36

It is, at least in here.

00:24:42

OK, but this one didn't work.

00:24:50

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

00:25:01

Participant

But sure.

00:25:03

Then.

00:25:04

So this function is called every time we get a new state and action and then we

00:25:08

00:25:27

00:25:33 create this new state.
00:25:37 But we should only do this once.
00:25:41 Um.
00:25:49 No, actually this should just be initial
00:25:52 state.
00:25:54 Because.
00:25:55 > **Editing file** (src/todo/reducer.js)
- 00:26:00
00:25:57 **Participant**
And hopefully this should work because every time we're saving it.
00:26:02 And.
00:26:06 Yes.
00:26:10 Maybe we'll see because what can also happen is that every time we get a new
00:26:14 state, we'll use the original state.
00:26:19 But that should be fine, yes.
00:26:34 That's a nope.
00:26:35 Umm.
00:26:42 But it is.
00:26:47 So we have all these objects that I just put in here.
00:26:50 It's under the keys todos and now.
00:26:55 It's not loading it.
00:26:56 > **Editing file** (src/todo/reducer.js)
- 00:27:05
00:26:57 **Participant**
Can I try to console log it again?
00:27:10 Maybe I need to restart the dev server again because these ones are not.
00:27:16 Regenerated like the other ones.
00:27:20 Hmm.
00:27:24 OK, it is calling the loading one.
00:27:27 Oh.
00:27:30 Just overwriting this state.
00:27:34 With this one.
00:27:42 > **Editing file** (src/todo/reducer.js)
- 00:27:43
00:27:42 **Participant**
Then uh.
00:27:50 Todo reducer.
00:27:56 Oh, my God. OK, this is dumb.
00:28:00 You can just do this.
00:28:01 Load state here.
00:28:10 > **Viewing file** (src/index.js)
00:28:11 **Participant**
I'm just going to go to where this one is called and I guess it was.
00:28:12 > **Viewing file** (src/todo/reducer.js)
00:28:13 > **Viewing file** (src/todo/app.jsx)
00:28:17 > **Viewing file** (src/todo/components/main.jsx)
00:28:22 **Participant**
In this main function.
00:28:29 And in here we can.
00:28:31 > **Editing file** (src/todo/components/main.jsx)
00:28:32 **Participant**
So I think use memo is used only.
00:28:37 Done only once.
00:28:39 But it's use effect. No, it's use effect.
00:28:41 > **Editing file** (src/todo/components/main.jsx)
00:28:43 **Participant**

00:28:46
- 00:28:57

00:28:49

00:29:00

00:29:09

00:29:11

00:29:14

00:29:17
- 00:29:28

00:29:17

00:29:24

00:29:38

00:29:46

00:29:49

00:29:53

00:29:59

00:30:01

00:30:03
- 00:30:06

00:30:03

00:30:07

00:30:08

00:30:10

00:30:13

00:30:17

00:30:17

00:30:19
- 00:30:25

00:30:24

00:30:24

00:30:27

00:30:29

00:30:33

00:30:33

00:30:35

00:30:40

00:30:41

00:30:42

00:30:43

00:30:45

00:30:47

00:30:49

00:30:54

00:30:55

Umm.

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

Participant
In here we can probably do.

Hmm.

Use memo. we get the todos.

We get to do from here, OK.

In dispatch.

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

Participant
OK. And then probably we can just do dispatch.

And then I guess type is load.

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

Participant
Wait, is this load state used anywhere else?

Because then.

Mm.

No, because then I'm just going to change this one to instead of use action dot payload.

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

Participant
I'm just going to say new state is initial state.

And then I'm going to just call.

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

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

Participant
Does it then need to have some kind of action?

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

Participant
Sure. Then I'm just going to go load state.

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

Participant
With payload being empty.

Researcher
By the way.

Sorry, I see there's a pop up on the bottom right that your extensions.

Have crashed.

Could you try to restart these by pressing on the button?

Participant
OK, sure.

Researcher
Not sure if this is actually doing anything, but thanks. Go on.

> **Viewing file** (src/todo/constants.js) through file search

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

Participant
Me neither, but.

OK.

So this one should do this now and this one gives me an error because.

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

Participant
It's not load state. It's oh, I need to import it.

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

00:30:58
00:31:01
- 00:31:03
00:31:03

00:31:11
00:31:15
00:31:22
00:31:23
00:31:32
00:31:34
00:31:35
- 00:32:03

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

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

Participant

Load state.

And then payload and then.

Yeah.

OK.

There we go. It does not enter a task anymore, OK?

Oh, this one. OK. No, never mind.

This one needs to be gone because it updates my, this use effect

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

Participant

has some dependencies, so it runs all the time,

but I want it to only run once, so I gave it no dependencies.

Will this work?

Maybe if dispatch is not defined, but it is because it's already in here,
yeah.

So now it should all work and then when I refresh I have the same task, yes.

Oh, this is the getting started. So this is the task.

Add some todos, refresh page.

Marks some todos completed, refresh page.

OK.

Yep, everything's still there. Edit.

Hmm.

I need to press enter first.

These are still here and then delete some.

So I've done those. Yep.

First task done.

User refreshes the page while typing a new todo, text input disappears.

OK.

So I'm unsure what it means by submitting and I guess submitting means pressing
enter.

So I'm just going to check it that when I do this now, yeah.

That's not submitting.

OK.

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

Participant

So I'm gonna go look at the input because this is where you submit it.

And then we have to handle key down.

Which is?

Yeah.

So instead of as you can check when it's enter and else it should also store it.

Am I going to do this in here, sure.

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

Participant

So here I should say that.

I'm gonna make a, Am I gonna do it in here?

Researcher

Sorry, could you maybe restart the extensions
again?

Participant

Oh.

Researcher

I'm not exactly sure what's going on, sorry for that.

00:31:38

00:31:41
00:32:01
00:32:05
00:32:09
00:32:15
00:32:25
00:32:28
00:32:30
00:32:33
00:32:36
00:32:45
00:32:48
00:32:51
00:32:56
00:32:58
00:33:04
00:33:15
00:33:15
00:33:21
00:33:21
00:33:26
00:33:35
00:33:37
00:33:38

00:33:42
00:33:47
00:33:52
00:33:54
00:34:03
00:34:07
- 00:34:21
00:34:07

00:34:21
00:34:25

00:34:27
00:34:28

00:34:29

00:34:31

Participant

No, no, no.

00:34:33

Is there maybe?

00:34:37

Oh, this is just gonna open my.

00:34:43

I was like trying to see maybe if I have.

00:34:47

Some errors in here but.

00:34:59

Websocket errors.

00:35:07

But.

00:35:18

Yeah, this is fine. If I'm just gonna.

00:35:23

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

00:35:23

Participant

I guess.

00:35:31

Because I'm trying to see.

00:35:33

Because there are multiple input fields so.

00:35:38

Because it's a long. Can you change more than one at the same time?

00:35:43

No. OK then it's fine.

00:35:43

To just save this one, OK.

00:35:44

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

00:35:45

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

00:35:47
- 00:36:25

00:35:47

Participant

I'm just gonna add a function here.

00:35:50

Save draft.

00:35:56

Value and then.

00:36:01

I guess I'll just store the time with it as well.

00:36:07

And you're gonna do the local storage.

00:36:11

It was local storage dot set item.

00:36:32

Yeah, and then it should also shouldn't forget

00:36:36

to remove this when we submit it.

00:36:37
- 00:36:42

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

00:36:40

Participant

Uh.

00:36:44

Otherwise, we'll keep coming back.

00:36:47

So I'm just going to set item draft and then I'm going to add the.

00:36:48
- 00:37:03

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

00:36:55

Participant

Value.

00:36:55

This should work with shorthand notation and then also the timestamp because we should do it every second.

00:37:00

Just gonna check real quick how to.

00:37:05

Check GitHub Copilot how again we get the current time.

00:37:17

Should most of the time be really quick but.

00:37:25

OK.

00:37:38

Hmm.

00:38:04

Hello.

00:38:07

Researcher

00:38:15

Oh, this might be your extensions that have crashed again.

00:38:18

Could you, on the left.

00:38:20

Could you go to the extensions panel?

00:38:21

Hmm.

00:38:29

Participant

00:38:30

I could try and turn it off and on again, I guess.

00:38:32

Or is it chat?

00:38:33 **Researcher**
Yeah.

00:38:38 **Participant**
But it's so weird why it does all of this with.

00:38:42 **Researcher**
Yeah. We haven't had this happen before.

00:38:45 Hopefully it's not.

00:38:45 **Participant**
I can also try to go to like a Chromium or Safari or something.

00:38:51 **Researcher**
Umm.

00:38:55 **Participant**
I can just, yeah.

00:38:56 **Researcher**
Yeah, we might need to do this in this case.

00:38:59 Could you not close the Firefox window just yet?

00:39:03 **Participant**
Yeah, sure.

00:39:04 I'm just downloading it in the background I guess.

00:39:07 OK.

00:39:29 **> Editing file** (src/todo/components/input.jsx)
- 00:39:38

00:39:47 **Researcher**
OK.

00:39:47 That's unfortunate.

00:39:48 **Participant**
Yeah.

00:39:52 Then I'll just try to open Chromium.

00:40:07 Hey, maybe I need your 2FA code again, but.

00:40:10 **Researcher**
That's no problem, fortunately.

00:40:25 **Participant**
Oh, I don't.

00:40:35 **> Viewing file** (src/todo/constants.js)

00:40:36 **> Viewing file** (src/todo/app.jsx)

00:40:48 **Researcher**
Would it be that he signed in with your own accounts?

00:40:51 No. OK.

00:40:51 **Participant**
No, because this. Oh, no. Just need to click it.

00:40:52 **Researcher**
Now it's working.

00:40:54 **Participant**
It's like I just downloaded the browser, so I guess not.

00:41:05 **Chat** (window: 1)
> Agent
github.copilot.default
> Context
viewport (src/todo/app.jsx)
> Message
Get current time in js?
> Response
To get the current time in JavaScript, you can use the Date object. Here's an example:

```
const currentTime = new Date();
console.log(currentTime.toLocaleTimeString()); // Outputs the current
time in a readable format
```

This will display the current time in the format based on the user's locale. If you need the time in a specific format, you can use methods like `getHours()`, `getMinutes()`, and `getSeconds()`:

```
const now = new Date();
const hours = now.getHours();
const minutes = now.getMinutes();
const seconds = now.getSeconds();
console.log(`${hours}:${minutes}:${seconds}`); // Outputs time in
HH:MM:SS format
```

00:41:06
- 00:41:07

> Response rendering

00:41:09

Participant

OK.

00:41:09

Now it's working again. Yes, sure.

00:41:11

That was it again, yeah.

00:41:15

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

00:41:18

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

00:41:19

Participant

So, going back to input.

00:41:20

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

00:41:23

Researcher

You can allow this, yes.

00:41:24
- 00:41:26

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

00:41:24

Participant

Oh, now it actually, yeah.

00:41:27

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

00:41:27

Participant

OK, then I'm just gonna get the current time

00:41:30
- 00:41:50

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

00:41:30

Participant

and then timestamp current time dot.

00:41:41

Or is it dot also from.

00:41:44

Yeah. OK.

00:41:44

So I'm just going to get.

00:41:48

Get time I guess.

00:41:51

Yeah, I'm just gonna go compare the time stamp
then.

00:41:56

Then.

00:42:00

Every X seconds.

00:42:01

Yeah, I'm just gonna do every second, I guess so.

00:42:03
- 00:42:36

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

00:42:08

Participant

00:42:16 And I also need to load draft.
00:42:18 Umm.
00:42:33 Yeah.
00:42:41 I'm just going to simply.
00:42:47 Yeah, that's fine.
00:42:50 Yeah.
00:42:51 **> Editing file** (src/todo/components/input.jsx)
00:42:51 **Participant**
So if the key is.
00:43:05 I'm thinking about now if I remove it, what does it actually mean to be removed?
00:43:11 I guess it'll just.
00:43:16 Yeah, I'm not going to save none.
00:43:17 **> Editing file** (src/todo/components/input.jsx)
- 00:44:27
00:43:17 **Participant**
Just also going to make a function remove draft.
00:43:23 It's just.
00:43:27 I think there's a local storage dot.
00:43:30 Here, yeah.
00:43:33 But this.
00:43:34 Nah, it's nice to have different functions.
00:43:38 Remove the drafts.
00:43:58 Then I'm going to check if we actually have a save draft.
00:44:03 And if we have it, then we're going to check the timestamp.
00:44:08 I'm going to copy it here as well.
00:44:19 Get time, is.
00:44:29 Is larger or equal and get time is actually in.
00:44:30 **> Viewing file** (node_modules/typescript/lib/lib.es5.d.ts)
00:44:34 **Participant**
Oh no, I did it again.
00:44:40 This of course doesn't work in the browser, yeah.
00:44:41 **> Viewing file** (src/todo/components/input.jsx)
00:44:45 **Participant**
Hmm.
00:44:54 **> Editing file** (src/todo/components/input.jsx)
00:44:58 **> Viewing file** (node_modules/typescript/lib/lib.es5.d.ts)
00:45:00 **Participant**
And it, get time gets the.
00:45:05 Returns stored in milliseconds.
00:45:07 **> Viewing file** (src/todo/components/input.jsx)
00:45:08 **> Editing file** (src/todo/components/input.jsx)
- 00:46:14
00:45:09 **Participant**
So.
00:45:11 It should be two seconds times 1000.
00:45:16 And then the.
00:45:18 Saved draft.
00:45:25 Save draft dot.
00:45:28 Timestamp.
00:45:33 OK.
00:45:33 So if we have the current time.
00:45:41 If the current time is sort of this 1 + 2 seconds.
00:45:46 Also gonna put parentheses. I'm just gonna make it 2000 actually.
00:45:53 If that is the case, then we should.
00:45:57 Save.
00:45:58 Our.

00:45:59 New draft.

00:46:04 Umm.

00:46:12 We should save the e dot target dot value.

00:46:25 Should hopefully open up the window again, yeah.

00:46:28 **Researcher**

00:51:44 All right. I see we're at the 40 minute mark though.

00:51:48 So to start off with.

00:51:48 What was your strategy for navigating the codebase,

00:51:51 and did you manage to find your way through the code?

00:51:55 **Participant**

00:51:59 Yeah. For me, most of the time, I just really like to to double click

00:52:03 through and just really go in depth and just follow like a certain path.

00:52:04 But also everything was I used to work in React a lot.

00:52:04 **Researcher**

00:52:06 Mm hmm.

00:52:06 **Participant**

00:52:13 So for me it was sort of similar where everything sort of was.

00:52:16 So I guess that's mostly my strategy.

00:52:16 **Researcher**

00:52:17 All right.

00:52:19 And you.

00:52:23 Did you use Copilot to help you with your navigation or looking through the project?

00:52:24 **Participant**

00:52:26 No, no.

00:52:26 **Researcher**

00:52:30 OK. Do you also feel like you understood the codebase?

00:52:32 **Participant**

00:52:35 Yes, sort it off, yeah.

00:52:37 Yeah, there are some things that if I had more

00:52:39 time, I would look more into.

00:52:40 Why those changes?

00:52:45 Why those? Why it was the way that it was, I guess.

00:52:45 **Researcher**

00:52:49 And how would you usually do this?

00:52:49 **Participant**

00:52:52 So for me, I find it weird that there was sort of a

00:52:56 what? What do you mean? What you? What would you do this?

00:52:56 **Researcher**

00:53:01 So usually in your in your.

00:53:03 You work if you're trying to understand the project like this.

00:53:03 **Participant**

00:53:05 Yeah.

00:53:05 **Researcher**

00:53:07 How would you approach to trying to understand it?

00:53:07 **Participant**

00:53:11 Oh.

00:53:14 The, it really depends on project of course,

00:53:20 but one strategy that sort of works for me is to simply add a button and then see where everything I need to change.

00:53:26 So for instance, the first or the one where you need to let

00:53:30 the persistent storage. That was a good one to really get started,

00:53:34 because then you need to look at where the input is and then you keep on going

00:53:36 and keep on going and keep on going.

00:53:36 And then at some point you.

00:53:38 Find every part of where the todo travels, so something like that would be a good
00:53:44 place to start.
00:53:46 **Researcher**
That makes sense.
00:53:48 And.
00:53:50 So you used Copilot for some some small things.
00:53:55 How would you have solved the tasks if Copilot hadn't hadn't been there?
00:54:02 **Participant**
I guess I would just have Googled it, but for some small questions.
00:54:07 And I was debating whether I should have used Copilot for it or not,
00:54:10 but the the questions were so small I was like,
00:54:12 I'll just get an immediate answer if you just type it in here and I don't need to
00:54:16 navigate click through, skip some ads, whatever.
00:54:19 **Researcher**
Right. Is this also how you use similar LLMs
00:54:24 usually?
00:54:26 Similar tools.
00:54:28 **Participant**
Yes. So what I I normally use the Copilot
00:54:30 inline for these kind of things, so I would just typed in dot sort and it
00:54:34 would sort of automatically give me a suggestion or something like that and
00:54:38 then.
00:54:40 And then I how I use LLMs. The larger ones mostly is as a sparring
00:54:45 partner, so more like I want to have this kind of
00:54:48 feature. What kind of benefits or or problems
00:54:51 would arise if I do it like this or if I do it like that?
00:54:57 **Researcher**
Right and using it as a sparring partner.
00:54:57 **Participant**
Yep.
00:55:02 Yeah.
00:55:04 **Researcher**
Did you do that during this during the session?
00:55:07 **Participant**
No, not the session, no.
00:55:09 **Researcher**
OK. And do you know why this why this was?
00:55:14 **Participant**
Yeah, the because I think the problems at hand
00:55:17 now were small enough or straightforward enough that it didn't.
00:55:21 It wasn't necessary, I guess.
00:55:23 **Researcher**
Fair enough.
00:55:25 And and then regarding the auto complete, do you see any advantages or
00:55:30 disadvantages of the chats compared to the auto complete?
00:55:35 **Participant**
Yeah, the the chat does give a lot of extra
00:55:38 text which if you are using a library which you are unfamiliar with,
00:55:43 that's fine.
00:55:44 But for the questions I asked, it was just to to get.
00:55:50 The get a sense of how.
00:55:53 The like how the language was sort of supposed to be,
00:55:55 or how the function is supposed to look, I guess.
00:56:00 **Researcher**
Right. Also in the survey indicated which
00:56:04 aspects of Copilot's responses you liked or disliked.

00:56:10 Explain if any of these stood out to you.

00:56:14 **Participant**
The responses, yeah.

00:56:16 **Researcher**
Yeah.

00:56:19 **Participant**
I'm not particularly.

00:56:20 **Researcher**
OK.

00:56:23 So you said that there was some more text.

00:56:26 Was this helpful to you or did it hinder you?

00:56:26 **Participant**
Mm.

00:56:35 Yeah, in some cases it's extremely helpful

00:56:37 because I've had also cases where it just has a oh, yeah, by the way,

00:56:42 this and this and this and that turned out to be the problem that I was looking

00:56:47 for. But most of the for now, it was really in.

00:56:50 A small window and I just needed a quick answer and then it sort of hindered

00:56:54 because it just cluttered the the rest of the information.

00:56:58 So.

00:57:01 It really depends on situation.

00:57:03 **Researcher**
So. So. So you mentioned the small window.

00:57:05 Would it have been a difference situation if it were in a different tab for example?

00:57:13 **Participant**
Yeah. For me that that's is how I use it.

00:57:16 So maybe it's also a preference thing.

00:57:20 Yeah, because I also use Claude and then they

00:57:22 have sort of a draft or if you want to wanted to generate code and it does it

00:57:26 on a like a different window and then it also has your chat in the left window.

00:57:31 I think that's very nice to to have the space and to see the code next to the

00:57:35 reasoning instead of it all being on the on the single line and you need to scroll

00:57:39 back and forth and those kind of things.

00:57:41 **Researcher**
That makes sense.

00:57:44 And how about the responses of such a system being tailored towards all the

00:57:48 code and the projects that you're working with?

00:57:52 Is this something that you want or expect out of an assistant?

00:57:55 **Participant**
Yes.

00:57:58 I do want it to be some fine control over it so.

00:58:03 I'm not sure how here.

00:58:05 I like that you can.

00:58:06 Sort of. Insert it yourself.

00:58:09 Which I think is the way it's supposed to.

00:58:10 So you can have some control of what data you share with your LLM.

00:58:14 So I've also set it up in Claude that every time it wants to access a file it

00:58:19 needs to request access first and I think that's a nice way to yeah,

00:58:23 where it to work.

00:58:23 **Researcher**
Alright.

00:58:25 And so this obviously this takes some more effort on your side,

00:58:30 but it gives you some control.

00:58:34 Is that a?

00:58:37 So towards this balance, would you say you're more towards

00:58:40 preferring this control aspect?

00:58:44

Participant

Yes, but that's also.

00:58:49

It is OK, I need to structure my thoughts I because

00:58:52

I on one hand I think it's important to have that as a data privacy control kind of that you choose what to disclose. For my own privacy, I guess, but also.

00:58:58

Researcher

OK.

00:59:00

Participant

LLMs now they can really sometimes get stuck in a way of doing things.

00:59:11

And then you keep trying to correct it and then it doesn't get sort of out of this downward spiral.

00:59:13

So it's also sometimes nice to control like don't look at all of these files.

00:59:15

Don't get stuck into this paradigm.

00:59:18

Just look at this file and what do you think of this? Or?

00:59:21

Yeah, and it's more of a limitation of current

00:59:27

Researcher

That makes sense.

00:59:30

Participant

LLMs, because I'm sure in the future this will

00:59:33

be, when they have bigger context

00:59:34

windows for instance.

00:59:35

It should be better.

00:59:37

Researcher

And so you're talking about the limitations of LLMs now.

00:59:42

Suppose LLMs were a lot better.

00:59:47

And you could just type one of these tasks into the assistant and get it to solve.

00:59:51

Solve a task with, well, a high reliability.

00:59:54

What would you think of this?

00:59:58

Would you make use of such features?

00:59:59

01:00:02

Participant

Definitely, because to be fair, these these tasks, they are very, these.

01:00:06

For this to do app. It really upgrades the user experience

01:00:10

and they are really no brainers.

01:00:14

But everybody has done these before.

01:00:15

It's such a sort of an easy task actually sort of to do. Just for me, sort of.

01:00:20

I would say that it wastes my time on spending these kind of things,

01:00:24

while I can also do some really cool new feature in this todo app. So for.

01:00:29

For me, it's sort of like picking your battles

01:00:31

that if there's a really easy feature and it can just program it for you, then sure, let's go.

01:00:37

And then I'll do the harder or the more software, software architecture parts.

01:00:37

01:00:42

Researcher

That makes sense.

01:00:44

And in in in this session then.

01:00:48

Do you have any reason for not using the assistant in this way?

01:00:58

Participant

Yeah, that is a good point.

01:01:04

I guess for the first one, because there was already a half

01:01:08

implementation, I figured that the LLM would not

01:01:11

understand this half implementation because it would just probably think oh, this is how it should work.

01:01:16

And on the second one, it definitely could have done it actually.

01:01:19

But yeah, didn't come up yet.

01:01:22

01:01:25

Researcher

OK.

01:01:26 Do you usually?
01:01:29 Prefer to just do it by yourself.
01:01:37 **Participant**
In some sense, yeah.
01:01:40 I like to because LLMs also they really, if you just say fix this then it will do it
01:01:45 probably in a different programming style than the rest of your code is in.
01:01:52 And then I really like to have some sort of setup for the AI to to finish it later
01:01:57 so I can just steer it into this programming style and then it does that.
01:02:01 That's maybe how I would use it.
01:02:06 **Researcher**
So sorry. You, you mean some some setup for the AI to
01:02:09 finish it later?
01:02:11 What do you mean with this?
01:02:13 **Participant**
So that's so if if there's five different ways to solve a problem and everything in
01:02:18 my codebase is in a particular coding style,
01:02:21 then I would type the first couple of beginnings and I will ask the AI to to
01:02:27 finish that that feature with.
01:02:29 Sort of this starting point.
01:02:31 **Researcher**
Right. So to to steer it towards this this
01:02:35 programming style that you're expecting?
01:02:38 **Participant**
I guess, yeah.
01:02:39 So maybe here I would have put the comments down or maybe the functions sort
01:02:43 of and I would say like oh please finish this in a way that you would behave it.
01:02:46 **Researcher**
Yeah.
01:02:49 That makes sense.
01:02:51 So last but not least.
01:02:53 **Participant**
Yeah.
01:02:54 **Researcher**
You used Copilot a few times.
01:02:58 Maybe you notice some aspects that you liked or disliked about how it worked.
01:03:04 Did you change your interaction?
01:03:08 Depending on what you observed.
01:03:14 **Participant**
I got a little bit irritated that it just continuously loaded but that's not the Copilot's
01:03:18 fault.
01:03:18 But that's why I kept making my question shorter.
01:03:25 And I was also testing a little bit how good the responses were.
01:03:30 Like how?
01:03:30 How well it understood my my task so.
01:03:35 In Claude normally I just type one sentence and normally that's good enough.
01:03:39 And I was testing here a little bit if that also works and I think it is. For the
01:03:44 **Researcher**
OK.
01:03:44 **Participant**
questions I asked. I think it was fine.
01:03:47 **Researcher**
Right. Sorry I have one more.
01:03:50 One more question actually.
01:03:50 **Participant**
Yeah, yeah.
01:03:53 **Researcher**

01:03:56 So you indicated that you had.
01:04:00 To work quite hard and that the session was also quite mentally demanding.
Participant
Mm.
01:04:01 **Researcher**
What were few factors that played into this?
01:04:05 **Participant**
Yeah. Unfamiliar environment and unfamiliar
01:04:10 code base I guess.
01:04:11 I think that is the main thing.
01:04:14 So I don't use Visual Studio Code, so for me that's also somewhat of a
01:04:19 change.
01:04:20 And then yeah, the way this was all structured was just,
01:04:24 yeah, that was more of a problem.
01:04:28 **Researcher**
All right, that makes sense.
01:04:29 **Participant**
And yeah.
01:04:32 **Researcher**
Do you have anything to add?
01:04:34 **Participant**
No, that's it.
01:04:36 **Researcher**
Great.