

Participant P02

00:07:00 **Researcher**
So what are you doing right now?

00:07:03 **Participant**
Yes, right now I'm navigating the structure of
00:07:08 the workspace files to
00:07:12 figure out the files that should open to complete the tasks.
00:07:22 So I'm going back to the task description.
00:07:41 > **Viewing file** (src/todo/app.css)
00:07:42 **Researcher**
So is there anything specific that you're looking for now?

00:07:47 **Participant**
Right now I'm looking for the file that would contain
00:07:53 the functionality that the practice task A requires you to change.
00:07:59 **Researcher**
OK.
00:07:59 **Participant**
So trying to see the file where they define the
00:08:04 color.
00:08:19 So I'm trying here before I use Copilot, just to kind of go over the files and
00:08:25 then if I do manage to find the
00:08:29 functionality on my own then
00:08:34 I'll I'll change it myself.
00:08:35 But yeah, otherwise I'll go to Copilot.
00:09:28 > **Viewing file** (src/todo/app.jsx)
00:09:35 > **Viewing file** (src/todo/app.css)
00:10:05 **Participant**
So now I feel like I found the correct file but
00:10:13 I'm.
00:10:18 Not sure if.
00:10:20 Oh, I think I found it. I think I'll try
00:10:27 changing that value.
00:10:32 I'm, I haven't
00:10:36 practiced with
00:10:41 > **Editing file** (src/todo/app.css)
00:10:41 **Participant**
CSS for quite a while so
00:10:44 it's taking me.
00:10:48 Yes.
00:10:54 And then it asks
00:10:58 to refresh.
00:11:09 **Researcher**
Yes, could you? Sorry, this was that's on me.

00:11:12 **Participant**
Yeah, yeah.
00:11:13 **Researcher**
Could you first try running the project?
00:11:17 And we ask you to figure out how to do this by yourself as well.
00:11:21 **Participant**
Yep.
00:11:28 I think in this case I will.
00:11:32 Umm.

00:11:34 Definitely need help of Copilot, so I.

00:11:39 Do we need to? From what they understand instructions we

00:11:43 need to start it with the @workspace before I ask it?

00:11:47 Or can I just directly ask it a question?

00:11:49 **Researcher**

There's some different commands and yeah,

00:11:52 **Participant**

Mm hmm.

00:11:53 **Researcher**

please feel free to experiment a bit or see what they do or see what works for you.

00:11:58 **Participant**

Mm hmm.

00:12:25 **Researcher**

So what are you thinking right now?

00:12:27 **Participant**

I am.

00:12:28 I'm inspecting the file structure.

00:12:33 Again to see.

00:12:39 Because normally when you run a command there is like a main file that you have

00:12:43 to run.

00:12:44 So I'm trying to see which one that would be.

00:12:54 **> Viewing file** (src/todo/app.jsx)

00:12:56 **Participant**

I would assume.

00:13:00 This one.

00:13:04 And.

00:13:06 Now that you found the file, I only see the.

00:13:10 Run in debug option.

00:13:18 So then I assume that I would need to perhaps input the run command in the

00:13:26 terminal instead.

00:13:52 **Chat** (window: 0)

> Agent

github.copilot.vscod

> Context

viewport (src/todo/app.jsx)

> Message

How do you run a .jsx file in terminal?

> Response

To run a .jsx file, you need a JavaScript runtime like Node.js and a transpiler like Babel to handle JSX syntax. Install Babel CLI and the necessary preset by running npm install --save-dev @babel/core @babel/cli @babel/preset-react. Create a Babel configuration file (babel.config.json) in your project root with the following content: { "presets": ["@babel/preset-react"] }. Transpile the .jsx file to plain JavaScript using Babel by running npx babel src/todo/app.jsx --out-file dist/app.js. Run the transpiled JavaScript file using Node.js with the command node dist/app.js.

00:14:03 **> Response rendering**

- 00:14:03

00:14:45 **Researcher**

So what do you think now of this response?

00:14:51 **Participant**

Now to see the response.

00:14:56 I'm thinking of.

00:15:03 Whether I would need to do as they say and compile.

00:15:10 A JS file.

00:15:14 Or if it's already here, just under.

00:15:19 A different name, perhaps?

00:16:16 Great.

00:16:33 And now I'm just interested in what's in these files because they have a JS

00:16:35 > **Viewing file** (webpack.common.js)

00:16:40 **Participant**
prepended so.

00:16:46 I inspect these files.

00:16:59 > **Viewing file** (src/todo/app.jsx)

00:17:00 > **Viewing file** (webpack.common.js)

00:17:01 > **Viewing file** (webpack.dev.js)

00:17:08 > **Viewing file** (webpack.common.js)

00:17:27 **Participant**
So this file it contains some information about the loader.
And also the Babel which the Copilot instructions talked about.
But I'm unfamiliar with this code structure, so I'm thinking of asking
Copilot, if it could potentially explain what this
file is for.

00:17:55 **Researcher**
OK.

00:18:02 **Participant**
So from what I
understood

00:18:05 I can just try and ask it directly, giving it the name of the file.

00:18:13 **Chat** (window: 0)

00:18:43 > **Agent**
github.copilot.workspace

> **Context**
workspace, selection (webpack.common.js)

> **Message**
Could you summarize what the "webpack.common.js" file does?

> **Response**
The [webpack.common.js](#) file defines the shared configuration for Webpack, which is used to bundle the project's assets. Here's a summary of its key components:

1. **Entry Point:**
 - Specifies the main file to start bundling from: [src/index.js](#).
2. **Plugins:**
 - Uses [HtmlWebpackPlugin \(node_modules/@types/html-webpack-plugin/index.d.ts\)](#) to generate an HTML file (public/index.html) and inject the bundled scripts into it.
3. **Output:**
 - Configures the output directory (dist) and file naming pattern ([name].bundle.js).
 - Ensures the output directory is cleaned before each build.
4. **Resolve:**
 - Allows importing files with .js and .jsx extensions without specifying them explicitly.
5. **Module Rules:**
 - Processes JavaScript and JSX files using babel-loader with presets for modern JavaScript (@babel/preset-env) and React (@babel/preset-react).
 - Handles image assets (png, svg, jpg, jpeg, gif) as resources.

This configuration is designed to handle React applications and is intended to be merged with environment-specific configurations (e.g., development or production).

> Response rendering

Participant

Now they see the explanation.

I'm trying to.

[unintelligible] because it doesn't appear to me now that I do need to follow

similar steps as provided by the Copilot in order to compile the app.jsx file.

Researcher

And so what are you thinking right now?

Participant

I'm paying attention to the module rules part because it talks about.

Processing JavaScript and JSX files.

So.

I'm trying to.

As I understand then from this description from the way this looks this only.

Predefines.

How the files should be processed, but they wouldn't be able to use the webpack common.js as an actual loader.

So then it appears to me that I do need.

To.

Compile.

OK.

I'm going back to these instructions.

Just to check on the tasks.

So.

[unintelligible], I'm checking to see if potentially I missed out on there is already

a Babel

configuration file.

Umm.

But it appears

not, so I will add a new.

Oops, I didn't do it here.

No.

Create a new file.

Here.

> Viewing file (babel.config.json)

Participant

OK.

And.

I'll try.

Instructions.

> Implementing Copilot code by copying full snippet (babel.config.json)

Participant

And then try.

So it requires me to stop that also.

So there is an.

An apparent syntax error in.

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

Participant

The.

App file.

00:26:42 **Researcher**
Right.

00:26:43 So.

00:26:46 I'll I'll quickly help you because it seems like this is becoming a bit of.

00:26:48 **Participant**
Yep.

00:26:52 Yeah.

00:26:54 **Researcher**
A A weird error, so I'll I'll. I'll just tell you what to or how to start the the application.

00:26:59 **Participant**
Yeah.

00:27:01 **Researcher**
No worries at all, by the way, it's.

00:27:02 **Participant**
Mm H.

00:27:06 **Researcher**
Like this is completely normal, right?
So there's nothing going wrong.
So in the in the terminal, if you type npm install.

00:27:08

00:27:10 **Participant**
Mm hmm.

00:27:12

00:27:19 **Researcher**
Yes. And then run this.
Now you can type npm space run space dev.
With a v, yes.

00:27:40

00:27:47 **Participant**
Yeah.

00:27:50 **Researcher**
And now any changes you make.
Yes, you can run this.

00:27:52

00:27:53 **Participant**
Mm hmm.

00:27:55 **Researcher**
So now it opens any changes you make will automatically show on this page.
Or maybe you'll need to refresh, but you won't need to run the commands again.

00:27:58

00:28:03 **Participant**
Mm hmm.

00:28:04 **Researcher**
So again, no worries at all.
And take your time.

00:28:07

00:28:12 **Participant**
Yeah. Thank you very much for helping me out with this.
But it appears you know the first task is done.
So now move on to practice task B, yeah.

00:28:16

00:28:19

00:28:26 **Researcher**
Yep.

00:28:31 **Participant**
All right.

00:28:33

00:28:36 OK.

00:28:40 All right.

00:28:42 **> Viewing file** (src/todo/app.css)

00:28:45 **Participant**
So.

00:28:47 Yeah thinking that I found the how to change the button color in the CSS file.

00:28:54 I just quickly go through it, but.
00:28:57 > **Viewing file** (src/todo/app.jsx)
00:29:00 **Participant**
Yeah, I don't see the needed field there.
00:29:04 > **Viewing file** (babel.config.json)
00:29:05 **Participant**
And I don't see it in the JSX file either.
00:29:08 > **Viewing file** (webpack.common.js)
00:29:11 **Participant**
This.
00:29:15 Either.
00:29:19 So I'm thinking that the fields that I need to change to be somewhere in the
00:29:27 components subfolder.
00:29:32 > **Viewing file** (src/todo/components/main.jsx)
00:29:33 **Participant**
Umm.
00:29:35 And I kind of.
00:29:38 Go through them.
00:29:41 Look, we need to find some.
00:29:52 > **Viewing file** (src/todo/components/input.jsx)
00:30:09 **Participant**
It appears it should be
00:30:13 here, see placeholder.
00:30:20 OK so.
00:30:23 It's a.
00:30:26 > **Viewing file** (src/todo/components/main.jsx)
00:30:26 **Participant**
Function input.
00:30:32 Which makes me think the other function is used somewhere, probably in the.
00:30:38 Main.
00:30:41 File.
00:30:43 So that should after all be in the main.jsx file.
00:31:31 So here. Yeah, for now, I'll just.
00:31:33 > **Viewing file** (src/todo/components/footer.jsx)
00:31:37 **Participant**
Check all the files, but I I do already kind of feel
00:31:42 comfortable with using.
00:31:46 Copilot. So it did cross my mind that that could
00:31:50 also just try and ask it in which file.
00:31:55 This functionality is.
00:31:59 Defined.
00:32:09 > **Viewing file** (src/todo/components/item.jsx)
00:32:33 > **Viewing file** (src/todo/components/input.jsx)
00:32:51 > **Viewing file** (src/todo/components/main.jsx)
00:32:59 **Researcher**
So what exactly are you looking for right now?
00:33:04 **Participant**
Yeah I'm looking for the.
00:33:06 Place where the text, the placeholder text is kept.
00:33:15 Yeah, because they found that in the input.jsx
00:33:19 file.
00:33:19 Only the function is defined that takes the placeholders.
00:33:26 As a parameter. So I'm trying to find in which file.
00:33:30 It is used and the placeholder is defined.
00:33:35 And.

00:33:39 But they do not seem to find it in the components.
00:33:41 > **Viewing file** (src/todo/app.jsx)
00:33:44 **Participant**
And it's not in the.
00:33:51 Main app.jsx file.
00:33:55 So thinking where it could be defined.
00:34:11 > **Viewing file** (src/todo/components/footer.jsx)
00:34:11 **Participant**
Yeah, I was pretty certain it would be
00:34:14 somewhere in the components and now I'm wondering if I'm just missing it
00:34:18 somewhere.
00:34:21 And I think.
00:34:25 I might.
00:34:28 Probably, yeah, to save time.
00:34:30 Ask Copilot.
00:35:03 And now I'm thinking how to pose a question.
00:35:09 To get optimal answer.
00:35:45 So this question does feel a bit big to me,
00:35:49 **Chat** (window: 0)
> **Agent**
github.copilot.default
> **Context**
viewport (src/todo/components/footer.jsx)
> **Message**
In which file in the "components" folder is the "placeholder" parameter for the Input()
function specified?
> **Response**
The placeholder parameter for the Input() function is likely specified in the
input.jsx file within the components folder. If such a file exists, it would typically
define the Input component and include the placeholder as a prop. You can search
for placeholder in the components folder to confirm its location using the following
terminal command:

grep -r "placeholder" src/todo/components/

00:35:49 **Participant**
but I'm gonna try and ask Copilot and see if it works.
00:35:50
- 00:35:52 > **Response rendering**
00:36:02 **Participant**
Umm.
00:36:15 Yes to.
00:36:18 It, as I thought, because of the way I posed the question
00:36:22 it does.
00:36:25 Answer it differently.
00:36:27 So now I'm thinking if I should try again to ask Copilot or.
00:36:34 Try and look for it myself again.
00:36:39 Because.
00:36:41 I do feel I was close to finding it, but.
00:36:47 I also don't know if it might just take me more time if I continue doing it on my
00:36:52 own.
00:37:03 So I'm gonna ask instead.
00:37:14 Which files used?
00:37:19 Input function.
00:37:24 To see where it's located and.
00:37:27 **Chat** (window: 0)
> **Agent**

github.copilot.default

> Context

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

> Message

Which files use the Input() function?

> Response

To determine which files use the Input() function, you can search for the term Input across the project. Run the following command in the terminal:

```
grep -r "Input(" src/
```

This will list all occurrences of the Input() function in the [src/](#) directory, including the files where it is used.

> Response rendering

Participant

Potentially in one of them.

The placeholder value specified.

So I'm gonna try.

Following.

This advice. I see.

That.

The app is still running, so I'll try opening.

A new terminal for this.

All right, so it only.

Appears to use it in the input.

Which leaves me.

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

Participant

A bit.

Confused.

So now I'm thinking of how to.

We approach this.

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

Participant

OK.

So wait.

Check this file constants.js because I thought yeah possibly if like the placeholder text is treated like a constant kind of parameter it could be there.

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

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

Participant

Oh oh wait. I I found it.

Right, and then I change it to "Enter task here".

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

Participant

And then.

Yes.

Researcher

Great. Thanks for thinking out loud. The way you do by the way.

It's really helping a ton.

Participant

And the final practice task.

Umm.

All right.

[unintelligible].

00:37:30
- 00:37:33

00:37:31

00:37:35

00:37:48

00:37:51

00:37:53

00:37:58

00:38:02

00:38:07

00:38:19

00:38:23

00:38:29

00:38:35

00:38:35

00:38:40

00:38:57

00:39:02

00:39:03

00:39:07

00:39:07

00:39:10

00:39:18

00:39:21

00:39:27

00:39:31

00:39:36

00:39:42

- 00:39:45

00:39:55

00:39:57

00:40:01

00:40:05

00:40:11

00:40:14

00:40:23

00:40:24

00:40:30 Yes. So yeah having already looked
00:40:33 through the
00:40:36 file structure before, I'm pretty certain.
00:40:41 The aspect that they need to change is again somewhere in the components
00:40:48 subfolder.
00:40:51 **> Viewing file** (src/todo/components/item.jsx)
00:40:52 **Participant**
I'm gonna try going to item.
00:40:59 Yeah, because I assume that's where kind of the
00:41:02 todo items
00:41:05 are regulated.
00:41:19 Yeah, just just looking through the file
00:41:21 structure to see what's in it.
00:41:45 Yes. So it appears to do mostly with the
00:41:48 functionalities of the items
00:41:52 themselves and it's just removing updating them but not adding them.
00:42:02 **> Viewing file** (src/todo/components/header.jsx)
00:42:03 **Participant**
I'm going to try checking just the rest of the files.
00:42:04 **> Viewing file** (src/todo/components/main.jsx)
00:42:11 **Participant**
OK, here it appears that it
00:42:14 does handle editing the files.
00:42:16 Because that's the last edit item.
00:42:31 Yeah.
00:42:50 So I'm trying to see.
00:42:53 Umm.
00:42:59 How it incorporates the toggle items into the code and where it might state the way
00:43:06 they are.
00:43:08 Displayed in which order?
00:43:18 OK.
00:43:19 So it's
00:43:23 just a todo
00:43:28 variable, and each variable has an index.
00:43:39 Umm.
00:43:52 So now I'm trying to understand.
00:43:59 If somewhere maybe there is like a thinking of other programming languages
00:44:05 like.
00:44:07 A list of all to do items because it for now appears like it just kind of.
00:44:13 Goes through.
00:44:16 OK, visibletodos, yes.
00:44:20 Where's the visibletodos
00:44:26 list, and?
00:44:46 OK.
00:44:46 So now I understand the structure so.
00:44:51 The main function gets the todos and then they're filtered and then
00:44:55 they're just kind of displayed in the order that they're provided in.
00:45:00 So then I assume that in order to display them alphabetically, I need to change
00:45:12 the code either
00:45:14 here.
00:45:15 In the starting line 15 under use memo,
00:45:22 or.
00:45:28 Potentially in here where they're mapped.
00:45:33 And so now I need to decide where it would be most efficient.
00:45:39 To do so.
00:45:44 And yeah, I think it'll be in the

00:45:48 Use memo.
00:45:56 Is, now I'm thinking that.
00:45:59 There could be a, since there is a filter like I see, some kind of common list
00:46:06 manipulation function that maybe there is
00:46:11 a sort
00:46:16 function, so I'm gonna try and.
00:46:23 Either look it up or.
00:46:34 I think we may try just type it in todos and the comment and see what autocomplete
00:46:40 options pop up. If there is this functionality.
00:46:45 But I have another question. Now is how to add
00:46:50 that function into here because of this structure is a bit unfamiliar to me.
00:47:26 Filter to this parenthesis and here it says to do the filter.
00:47:34 Umm.
00:47:44 OK, so I think we figure out general
00:47:48 structure
00:47:51 of the code. So I see that.
00:47:55 This and this are kind of like the two parameters of
00:48:00 use memo.
00:48:05 So now I'm thinking maybe I just.
00:48:05 **> Editing file** (src/todo/components/main.jsx)
- 00:48:09
00:48:11 **Participant**
No, there is no autocomplete offer
00:48:13 so I think I'll just ask Copilot.
00:48:17 [unintelligible]
00:48:37 **Chat** (window: 0)
> Agent
github.copilot.default
> Context
viewport (src/todo/components/main.jsx)
> Message
JavaScript list sort function
> Response
In JavaScript, you can use the `Array.prototype.sort()` method to sort a list. Here's
an example of how it works:

```
// Example: Sorting numbers in ascending order
const numbers = [5, 2, 9, 1, 5, 6];
numbers.sort((a, b) => a - b);
console.log(numbers); // Output: [1, 2, 5, 5, 6, 9]

// Example: Sorting strings alphabetically
const fruits = ["banana", "apple", "cherry"];
fruits.sort();
console.log(fruits); // Output: ["apple", "banana", "cherry"]

// Example: Sorting objects by a property
const todos = [
  { id: 1, title: "Do laundry", completed: false },
  { id: 2, title: "Write code", completed: true },
  { id: 3, title: "Read book", completed: false },
];
todos.sort((a, b) => a.title.localeCompare(b.title));
console.log(todos);
// Output: Sorted by title alphabetically
```

Notes:

- For numbers, use `(a, b) => a - b` for ascending order and `(a, b) => b - a` for descending order.
- For strings, you can use `localeCompare()` for proper alphabetical sorting.
- Sorting is **in-place**, meaning it modifies the original array.

00:48:38

Participant

Try to pose a question,

00:48:39
- 00:48:43

> Response rendering

00:48:41

Participant

a kind of non question format, more like querying.

00:48:45

See if it understands what I'm asking about.

00:48:52

Yes.

00:48:57

Oh, it's nice because it already gave some

00:49:04

examples, even being quite close to what I'm looking for.

00:49:16

I'm gonna try.

00:49:18

Just kind of.

00:49:21

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

00:49:21

Participant

Copying this part.

00:49:22
- 00:49:27

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

00:49:26

Participant

Filter.

00:49:30

OK, then I'm just gonna adapt this.

00:49:35

Uh.

00:49:41

Property OK.

00:49:42

Oh yeah, this is already alphabetic, so I think.

00:49:47

This could potentially be it. I don't think there are any errors even

00:49:52 though it's.
00:49:56 You wanted a different.
00:50:02 So I'm gonna to check if it works.
00:50:05 Yeah, the following todos.
00:50:16 Double click.
00:50:20 The.
00:50:32 Oh.
00:50:33 And oops I.
00:50:49 Yeah, it does appear to alphabetically.
00:50:55 **Researcher**
Perfect. Thanks.
00:50:58 I see that we're right on the 40 minute mark.
00:51:01 **Participant**
Mm hmm.
00:51:01 **Researcher**
So that was the tasks.
00:51:04 Thanks a lot.
00:57:25 All right then. I have a few questions for you about.
00:57:28 **Participant**
Mm hmm.
00:57:29 **Researcher**
Well, about the session and what you thought of
00:57:32 Copilot.
00:57:33 **Participant**
Yep.
00:57:34 **Researcher**
So yeah, first of all, could you find your way through the
00:57:37 codebase, navigating through files, and did Copilot help with this?
00:57:44 **Participant**
Yeah, I think.
00:57:46 I don't think I asked.
00:57:50 Copilot much about the except for the
00:57:57 components
00:57:59 file folder and the
00:58:03 web dot JS files.
00:58:06 I think it was definitely useful in understanding what some of the files do.
00:58:12 And yeah, it was kind of a good combination of me
00:58:15 being able to more or less navigate the files and using Copilot to understand
00:58:19 kind of what they do to even better see what the hierarchy is.
00:58:25 **Researcher**
Oh, all right.
00:58:25 So you felt like it was mostly you navigating through the files using
00:58:31 Copilot to help for specific files then?
00:58:36 **Participant**
I think so.
00:58:37 **Researcher**
All right.
00:58:38 And do you feel like you understood the codebase well?
00:58:43 **Participant**
Yeah, I I feel like.
00:58:47 For the tasks that I had to do.
00:58:53 Like they were limited to specific files in the workspace,
00:58:56 so I did not explore all of the files so I don't have like a complete overview of
00:59:01 exactly how everything works together, but I I think I have more or less
00:59:06 understood like the part that I worked with.
00:59:09 All that functions.

00:59:11

Researcher

Did Copilot help in getting this understanding or did it work against you?

00:59:19

Participant

I think.

00:59:24

It did help here with.

00:59:28

When I was in the beginning of the session,

00:59:31

especially when I asked it about some of the dot JS files and JSX files,

00:59:36

to see what files do what.

00:59:39

Researcher

All right. I think at some point, you said

00:59:44

"Oh, I'm thinking of using Copilot to save

00:59:47

some time."

00:59:48

Participant

Mm hmm.

00:59:48

Researcher

So in that case, were there also reasons why you would not

00:59:52

want to use Copilot?

00:59:55

Participant

Umm.

00:59:56

I think one of the bigger reasons would be kind of to challenge myself more, so

01:00:05

yeah, I was curious just to see if I would be

01:00:08

able to figure it out. And normally that's what I try to do,

01:00:12

but yeah.

01:00:13

Researcher

That makes a lot of sense, actually.

01:00:17

Also I noticed

01:00:20

you spent quite a lot of detail in formulating your questions to Copilot.

01:00:27

Is there a reason for this?

01:00:32

Participant

Yeah, it would be mostly to try and get kind of

01:00:35

the correct answer right away, sort of you know,

01:00:38

to avoid having kind of a long exchange where I try to one by one kind of tune

01:00:44

the prompt.

01:00:46

Researcher

Right. Is this generally your experience when

01:00:50

working with LLMs or with chatbots?

01:00:54

Participant

Yeah. Yeah. Let's see. So.

01:00:58

Think in the past when I used them.

01:01:03

They do respond more or less on point with the first questions,

01:01:07

but then you either have to kind of either correct them or you know.

01:01:13

Researcher

Yeah.

01:01:17

All right.

01:01:17

Few more.

01:01:20

So.

01:01:22

Did Copilot or using Copilot or being able to use Copilot change how you would

01:01:27

normally approach a task like this?

01:01:32

Participant

Yeah, I think so because.

01:01:36

You have very often, especially in the past,

01:01:39

I would instead of using Copilot, go on the web and try look up the answer

01:01:44

there, which it takes a lot more time because

01:01:47

you need to find the correct websites and I need to go back sometimes all the way

01:01:52

to the documentation, which can be a bit more difficult to

01:01:56

navigate.

01:01:57 So it's definitely useful.
01:02:01 **Researcher**
And would you use web search sort of
01:02:03 in the in a similar way where you're doing the navigating and then only when
01:02:08 you don't understand something about the file use
01:02:11 web search or would this strategy look different.
01:02:14 Can you tell me a bit about this?
01:02:16 **Participant**
Umm.
01:02:19 I would.
01:02:22 So I think normally especially working with the language kind of with I'm less
01:02:27 familiar with, I might start with.
01:02:31 First, maybe looking over the files and looking
01:02:34 up some kind of simple tutorial or basic documentation to understand the basic
01:02:40 structures that are used and then I kind of go through the file on my own and look
01:02:46 up whatever is kind of missing that they haven't
01:02:52 gotten on the first read of the documentation.
01:02:56 **Researcher**
All right, that makes sense.
01:03:00 So in the survey you just did
01:03:04 I saw you gave all of the aspects
01:03:08 on which your rate Copilot's responses you gave them,
01:03:11 pretty high scores. Were there any of these aspects that stood out to you?
01:03:19 **Participant**
Maybe.
01:03:23 Can I reopen the survey to?
01:03:25 **Researcher**
Yes you can
01:03:26 just refresh it and you can see them again.
01:03:29 **Participant**
Right. Yeah. Sorry. Are you referring to these aspects or the
01:03:36 ones that are on the? Yeah.
01:03:37 **Researcher**
Yeah, these aspects.
01:03:40 But also any other aspect if you if anything stood out to you?
01:03:45 **Participant**
I thought, yeah I had these were all
01:03:50 pretty
01:03:52 good in general. Oh.
01:03:54 **Researcher**
You can just enter anything.
01:04:00 **Participant**
And then here.
01:04:04 Yeah, I think, so
01:04:05 the first one was
01:04:08 yeah, quite
01:04:10 resonated with me because yeah, I do feel working with code sometimes
01:04:14 a lot of the time goes into actually understanding
01:04:19 it when you're working with like a new project that has already some kind of a
01:04:25 code skeleton in it.
01:04:28 So I feel like it would be useful if I used something like Copilot in my work
01:04:34 more.
01:04:35 Umm.
01:04:38 Also, yeah, kind of connected to that would be a productivity.
01:04:42 So I would have more time.
01:04:48 Hmm.

01:04:51 Yeah, I think.
01:04:53 Those would be the most important aspects to me.
01:04:54 **Researcher**
Alright.
01:04:56 Are there any aspects of using Copilot, or of Copilot's responses that you didn't
01:05:03 like?
01:05:08 **Participant**
I.
01:05:10 Don't think so, at least out of the interactions that I
01:05:14 had with it. Like the final one was especially quite
01:05:18 nice where it gave concrete examples of how to sort lists,
01:05:22 and it was both nicely formatted
01:05:25 and quite clear.
01:05:29 Maybe the instruction that I got in the beginning about how to run and
01:05:35 compile JS files, maybe that was the least convenient one.
01:05:41 Because it was kind of formatted in a single kind of block of text.
01:05:45 So maybe if it was in bullet points, it would have been easier to work
01:05:47 **Researcher**
Alright.
01:05:48 **Participant**
with.
01:05:50 **Researcher**
That makes sense.
01:05:54 So.
01:05:56 Yeah, obviously you got some responses from
01:05:58 Copilot.
01:06:01 So did you change the way you ask your questions based on the responses that you
01:06:07 got?
01:06:08 So were your interactions that you had different from the interactions that you
01:06:13 would ideally have for example?
01:06:19 **Participant**
One thing I would say noticed is that in the beginning I think I paid more
01:06:24 attention to how I formulated the prompts.
01:06:27 So I would try to be more both grammatically correct and detailed.
01:06:33 And then I kind of saw that.
01:06:36 It has a good handle on the language.
01:06:39 So then I kind of made the prompts more abstract,
01:06:43 and sometimes it worked and sometimes it didn't.
01:06:48 So. Yeah.
01:06:49 **Researcher**
Right, so the the prompt you gave is for more
01:06:52 like you might choose a search engine for example.
01:06:56 **Participant**
Yeah, yeah.
01:06:58 **Researcher**
Right, great.
01:06:59 Thank you.