

QuickAdd_NewNoteWith_MOC_Relate

Problem

I kept creating dangling orphan sad-faced notes, and losing those all the time.

Idea

In general

I want to instantiate a new note, and I want Obsidian to prompt me to link it to another note.

More specifically

I want to create a new note, and I want to put links to other notes in the properties of the new note.

I first learned about this in [No Boilerplate's Obsidian video](#). (The shown syntax is a little off: `[[` and `"` should be in reverse order, like this: `"[[linked_note_name]]"`).

I want to do 3 things:

1. I want to create a new note.
2. I want to be prompted to put a link to a 'vertical / higher' level note (called a 'MOC', for 'Map of Content'), in the properties of the new note.
3. I want to be prompted for a 'horizontal' level note (called 'related'), and have that also be in the properties.

Solution

[QuickAdd](#) will be used to instantiate a simple template from the template folder. It will then show me a list of all my current notes, of which I can select one for the MOC-linked-note, and then again for the related-linked-note.

The Template

The template looks like this:

new_note_template

date_created:

date_modified:

MOC:

related:

Important

1. The words (exact, inc capitalisation) of the properties 'MOC: ' and 'related: '. If you use other words, amend them as well in the corresponding 'Capture' sections later.
2. *One blank line under each of these properties!*

(When using Linter, make sure to disable automatic linting of this file / or all the files in the Templates folder).

Script to get all the existing notes

Stored somewhere in the Obsidian vault. Put it in a 'Scripts' folder for all I care. Save it with a useful name, ending in `.js`.

Mine is called `selectSourceFile.js`.

```
module.exports = async function(params) {
  // Debug notification to confirm script is running
  // new Notice("Script started");

  try {
    // Get all markdown files
    const allFiles = params.app.vault.getMarkdownFiles();

    // Debug
    // new Notice(`Found ${allFiles.length} files`);

    // Show the suggester
    const selectedFile = await
params.quickAddApi.suggester(
  (file) => file.basename,
  allFiles
);

    if (!selectedFile) {
      new Notice("No file selected");
      return;
    }

    // This is where you create the variable
    'selectedNote' (it could've been called anything).
    // It is also setting it as the output of this script
    (to be used later in QuickAdd) by assigning it to the
params.variables
    // In QuickAdd, you use this output as it's input by
    using `{{value:selectedNote}}` in the block after where
    this file runs. Note the lack of the full path
    (params.variables...) when using in QuickAdd!
    params.variables = { selectedNote:
selectedFile.basename };
  }
}
```

```
// This is how you use that same variable in this
same script. Note the full 'path' (params... etc).
// new Notice(`Selected:
${params.variables.selectedNote}`);

} catch (error) {
  new Notice(`Error: ${error.message}`);
  console.error(error);
}
}
```

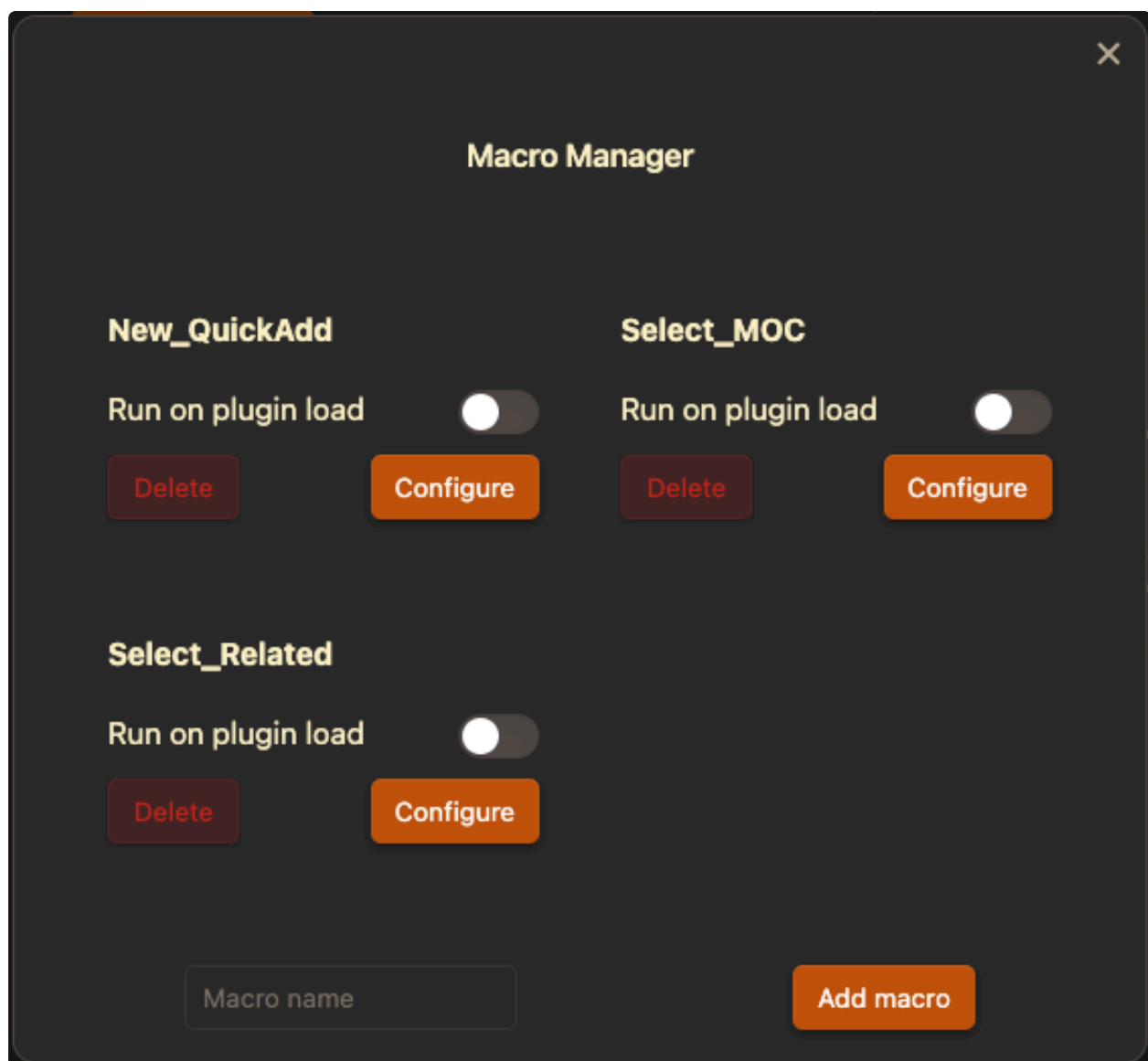


This script will get all the files in your vault, and display them as a dropdown list from which you can type / search / choose.

Keep an eye out for the `params.variables` and then specifically the `selectedNote` part. That will be important later.

Setting it up

Below is the 'Macro Manager' window in QuickAdd.

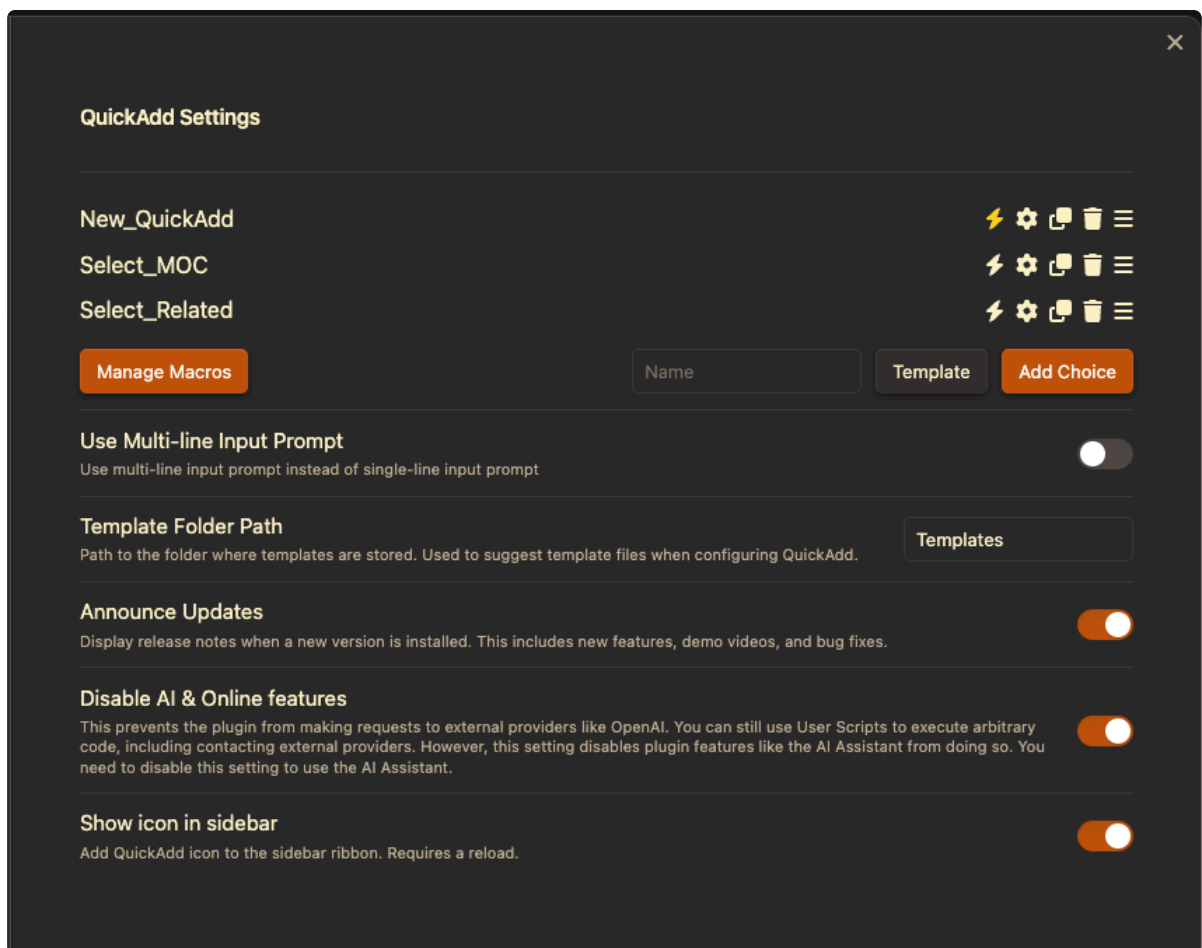


It is the macro 'New_QuickAdd' that I want to use. QuickAdd has a seemingly 'double' feature where you create a macro, and expose that separately. Very confusing.

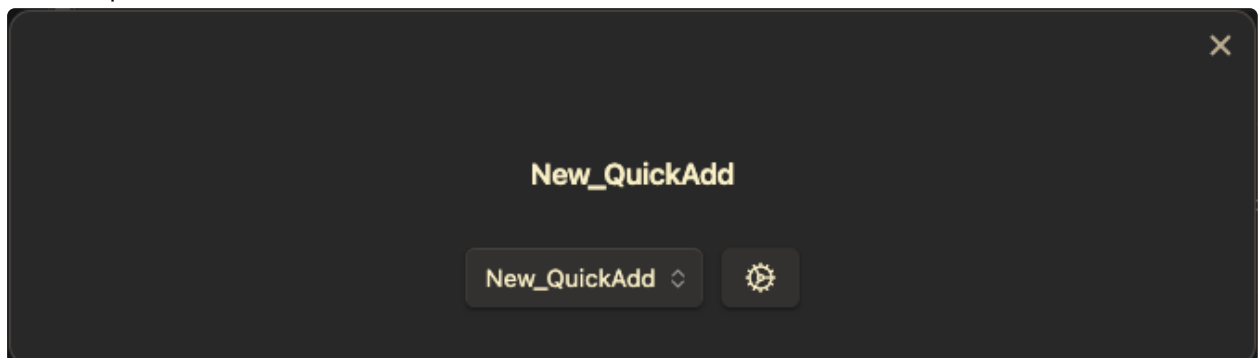
The macro is exposed below in the QuickAdd Settings. You do this by typing a name (in this case the same), selecting Macro, and hitting Add Choice.

Exposing it here allows two things:

1. The macro can be used by other macros / templates / captures in QuickAdd
2. The rest of Obsidian can use it.
 1. It is always possible with Command Palette (Run QuickAdd > New_QuickAdd)
 2. The yellow lightning bolt also makes it possible to assign a hotkey.



The exposed New_QuickAdd is using the macro New_QuickAdd.



The macro New_QuickAdd

When you hit the gear icon in the exposed thing (or via the Macro Manager), you can change the settings of the macro.

The macro New_QuickAdd uses a ('local') Template option to create a new file (hit the 'Template' button right there). I say 'local' because it is also perfectly happy to use an already created QuickAdd template or Capture. However, then you'd end up with even more separate things in these menus here.

Our macro also uses two other macros (Select_MOC and Select_Related). The latter two create the linked note in the properties of our new note. The order is important: create file first, add properties later.

The screenshot shows a dark-themed window titled "New_QuickAdd" with a close button (X) in the top right corner. On the left, a list of steps is shown: "1. New_File", "2. Select_MOC", and "3. Select_Related". To the right of this list are icons for settings (gear), deletion (trash), and a menu (three lines). Below the list, there are three buttons: "Capture", "Template", and a clock icon. The main area of the window is divided into four sections, each with a title, a description, a text input field, and an "Add" button:

- Obsidian command**: "Add an Obsidian command". The input field contains "Obsidian command".
- Editor commands**: "Add editor command". The input field contains "Copy".
- User Scripts**: "Add user script". The input field contains "User script".
- Choices**: "Add existing choice". The input field contains "Choice".

The NewFile creates a file using the template selected from the Templates folder. The template I want to use is the one seen earlier. The other thing to see here is the `{{value: New File Name}}`. This prompts you to type in a textbox, with a little header above saying 'New File Name'. *Important is the space!* `{{value: New File Name}}` and not `{{value:New File Name}}`.

×

New_File

Template Path

Path to the Template.

Templates/new_note_ter

File Name Format

Set the file name format.

File Name: New File Name

{{value: New File Name}}

Create in folder

Create the file in the specified folder. If multiple folders are specified, you will be prompted for which folder to create the file in.

Append link

Append link to created file to current file.

Set default behavior if file already exists

Set default behavior rather than prompting user on what to do if a file already exists.

Increment the file name

Open

Open the created file.

Default

New split

Split your editor and open file in new split.

Vertical

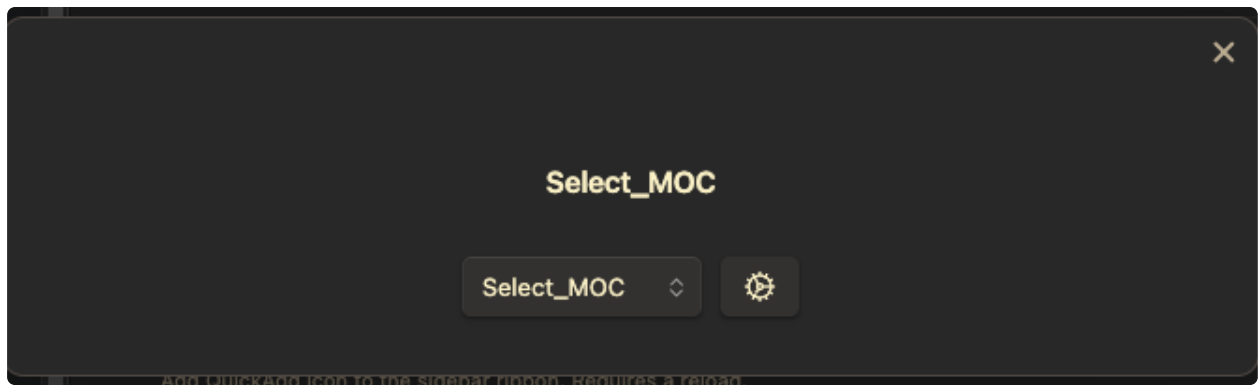
Focus new pane

Focus the opened tab immediately after opening

(A little foreshadowing, later we do something similar when getting input via the `.js`. However then the space is **not** allowed, and it is this one fucking space that took me 5 hours to realise why I was messing this up).

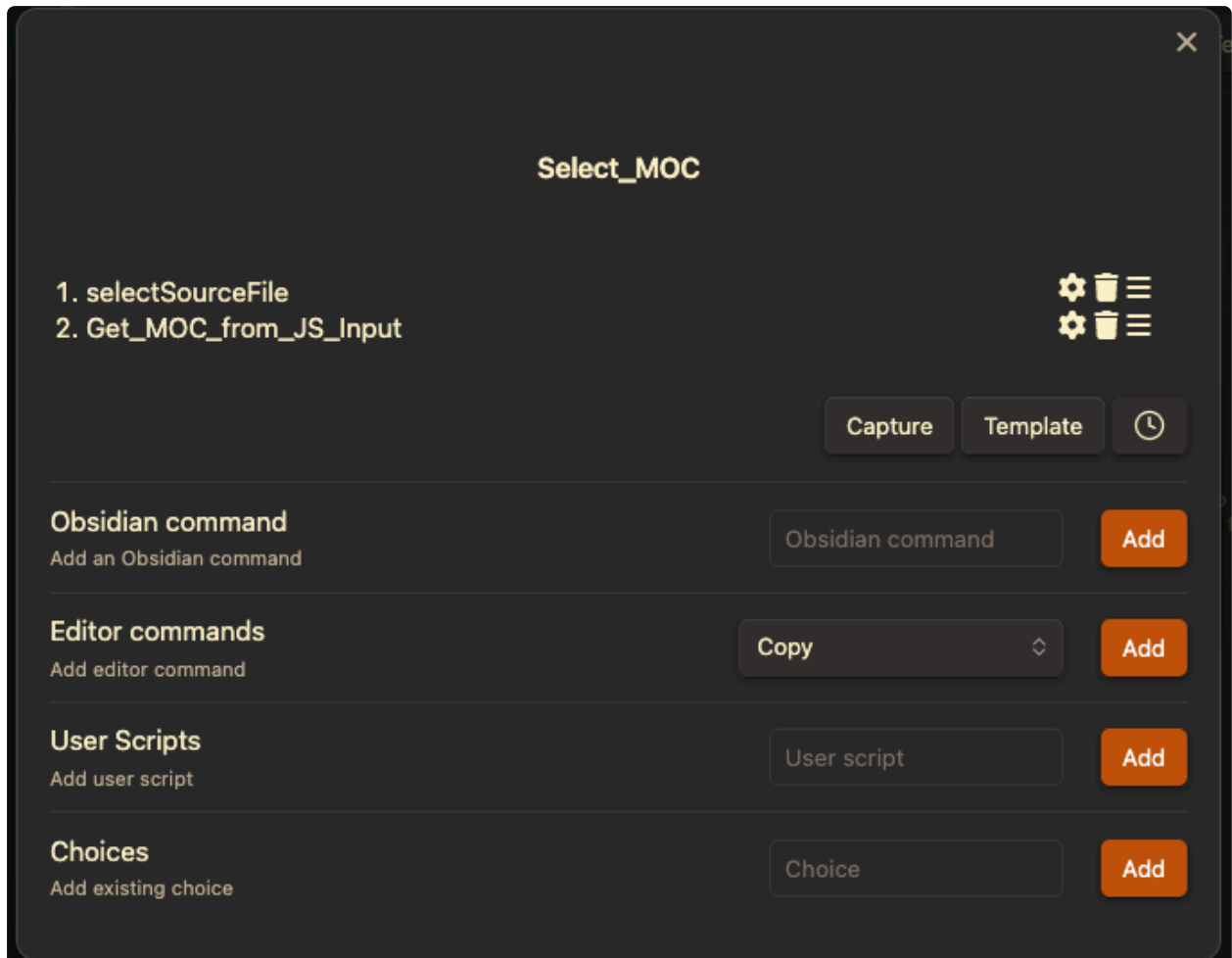
The exposed Select_MOC

Below you see the other macro being exposed.



The Select_MOC macro

Very important is the order! The first thing it uses is the 'User Script' `selectSourceFile.js` we saw before. The second thing is a 'local' Capture where it uses the script's output as it's own input. You get this 'local' Capture by hitting the big Capture button right there.



The Capture

This gets a selected file from the `.js` script, and puts it as a link into the properties / frontmatter of the newly created note.

Capture to active file (the newly created file):

×

Get_MOC_from_JS_Input

Capture To

File to capture to. Supports some format syntax.

Capture to active file

Task

Formats the value as a task.

Write to bottom of file

Put value at the bottom of the file - otherwise at the active cursor location.

Append link

Add a link on your current cursor position, linking to the file you're capturing to.

Insert after

Insert capture after specified line. Accepts format syntax.

MOC:

MOC:

Insert at end of section

Insert the text at the end of the section, rather than at the top.

Consider subsections

Enabling this will insert the text at the end of the section & its subsections, rather than just at the end of the target section. A section is defined by a heading, and its subsections are all the headings inside that section.

Create line if not found

Creates the 'insert after' line if it is not found.

Top ▾

Capture format

Set the format of the capture.

- "[{{{value:selectedNote}}}]"

Important: The thing you type below 'Insert after' must have the exact wording / capitalisation of one of the properties found in the note template (in this case 'MOC:').

Because of the 'Insert after' is also why we needed to have that **blank line** underneath this property. If you don't, then the properties section get's messed up.

Getting the selected note into the Capture

The way to get the output from the `selectSourceFile.js` to our Capture, is to use `{{value:selectedNote}}` in the Capture Format. (See *that spacing right there? 5 hour spacing, that is!*).

The `selectedNote` part comes from what the script outputs via it's `params.variables`. So because the script sets the `params.variables.selectedNote` as the name of the note that we selected, we import that value here this way. If the script had set the note name to `params.variables.buttsoup`, you would have used `{{value:buttsoup}}` here instead.

!!!IMPORTANT SPACING IN THE CAPTURE FORMAT!!! there is **no** space between `value` and `selectedNote`. So use `{{value:selectedNote}}` and not `{{value: selectedNote}}`. Five hours, ladies and gentlemen.

The `[]` and `]` brackets surrounds the note name to make it a link to our selected note, pretty standard Obsidian stuff.

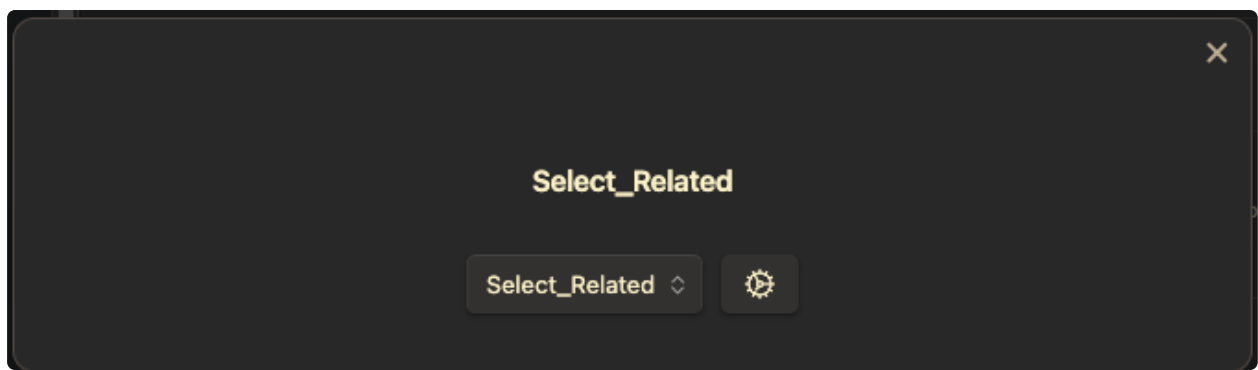
Now the `"` and `"` surround that yet again because... I don't quite know actually? It has to do with the note being used as a link inside of the *properties* of a new note. If you'd simply used the link in the body of the note, the `"` wouldn't have been needed.

Your Capture input should be `"[[{{value:selectedNote}}]]"` if you're using the earlier script exactly, or (as an example) `"[[{{value:pickedFilePath}}]]"` if you're using [danielo515's](#) fine writeup / script.

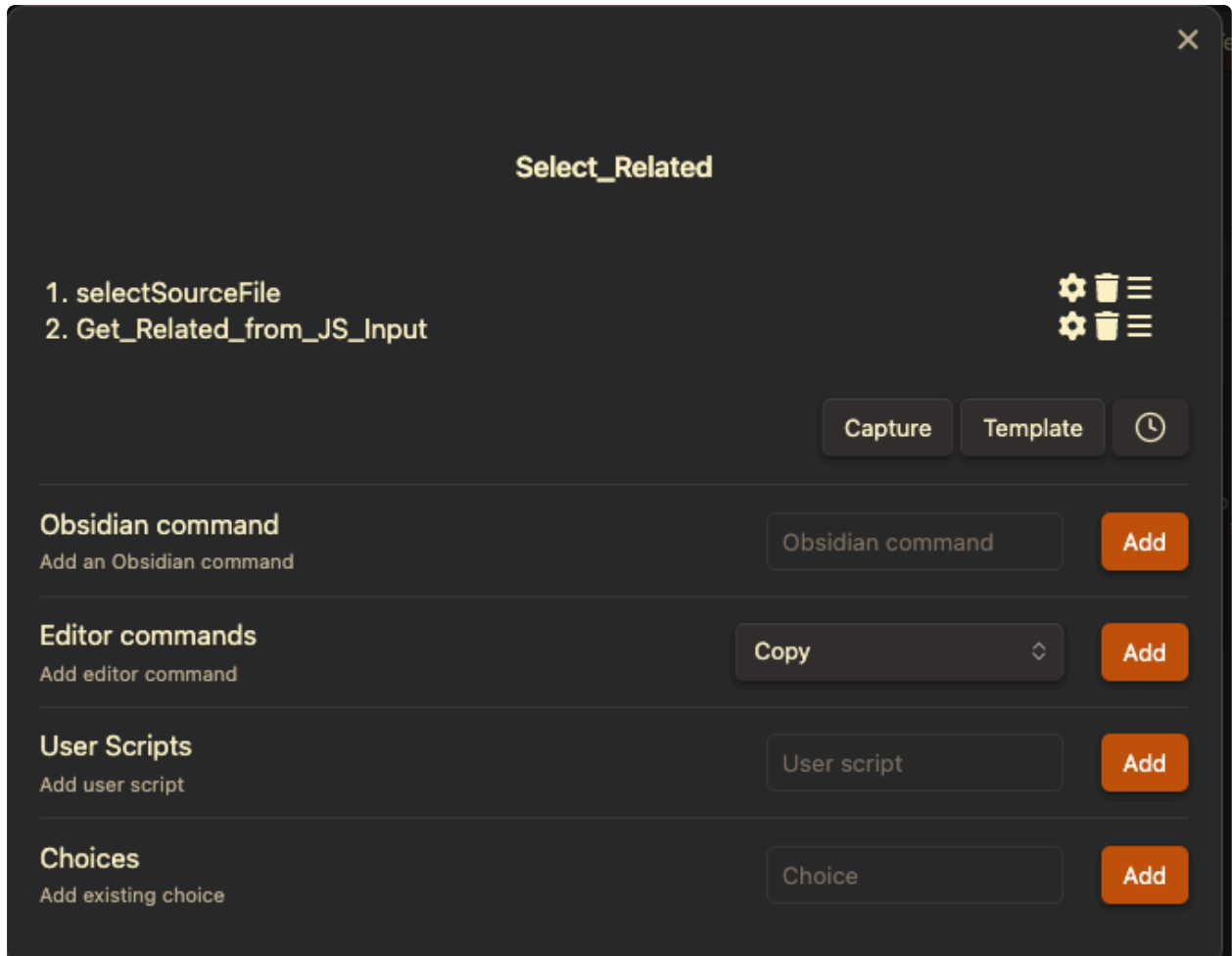
Now we do the same thing but now for 'related' note

Now getting the 'related' note as a link into the frontmatter.

This is the exposed macro:



This is the macro itself. See that we're using the `selectSourceFile` script here yet again.



This is the 'local' Capture. Again capture to active file. We're using the property name ('related:') from the template here exactly. And again, theres __no space between `value` and `selectedNote` .

×

Get_Related_from_JS_Input

Capture To

File to capture to. Supports some format syntax.

Capture to active file

Task

Formats the value as a task.

Write to bottom of file

Put value at the bottom of the file - otherwise at the active cursor location.

Append link

Add a link on your current cursor position, linking to the file you're capturing to.

Insert after

Insert capture after specified line. Accepts format syntax.

related:

related:

Insert at end of section

Insert the text at the end of the section, rather than at the top.

Consider subsections

Enabling this will insert the text at the end of the section & its subsections, rather than just at the end of the target section. A section is defined by a heading, and its subsections are all the headings inside that section.

Create line if not found

Creates the 'insert after' line if it is not found.

Top

Capture format

Set the format of the capture.

- "[{{{value:selectedNote}}}]"

Final words and possible improvement

This should work!

You can even duck out of adding either field (with escape). However, in future versions I'd want to make it a little nicer, because it throws some error messages when you do.

The option to add multiple 'related' notes could be nice as well.