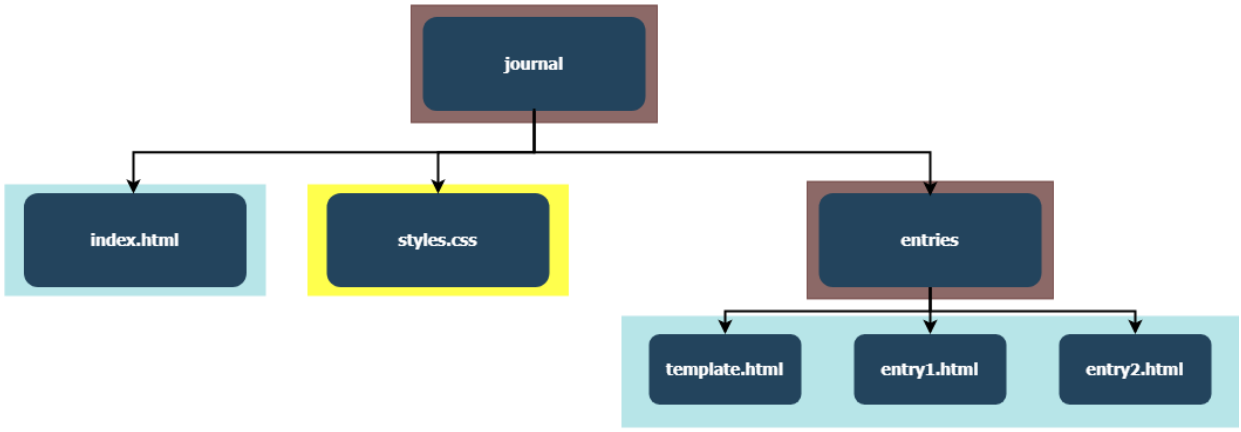
Refactor Journal to Multiple Pages

Refactor: To edit your code to meet new standards of design and/or efficiency.

Part 1: File Management

1. In the tree map diagram below, the brown highlights are on the folders/directories you will create. The blue highlights are on the HTML files you will create. The yellow highlight is on the CSS file you will create. In an appropriate location, start by creating a new folder called "journal" and also an "entries" folder inside it.



2. Copy your HTML template file (not your old journal) to create index.html. Give it appropriate `<meta>` information and a heading like "My Journal" or similar. Save and test. We will come back to this file later to add links to all of your journal entries.
3. Create a new styles.css file, and extract your existing CSS code from your old journal. Make sure your index.html file has an appropriate `<link>` tag to apply these styles. Save and test. Your index.html homepage should now have the basic styles from your previous work on the journal.
4. Make a new copy of your index.html file called template.html and save it inside your "entries" folder. Edit the template.html file to reflect the new `href` path to the style.css file (this probably means changing `href="styles.css"` to `href="../styles.css"`). Save and test to be sure your styles are applied properly. Next, consider what elements are needed for every journal entry that you have written. Look at your old journal file to see what every journal entry had in common. Set up your template.html file so that future journal entries can get their structure by simply copying the template.html file. Save and test.
5. Since we are going to have multiple pages, it's important for all of these sub-pages to have a link back to the homepage. Somewhere in the template.html file's `<body>`, include such a link by creating an anchor tag that opens with `<a href="../index.html">` and has content that describes the link as "Home" or "Back" or something similar. Save and test.
6. Finally, extract each individual journal entry to its own file. In the diagram above, the files names are entry1.html, entry2.html, and so on. You may choose any filename structure that works for you. It might include the names of the journal entries, the dates they were written, or any other information that makes sense for the file name.

Part 2: Homepage Navigation

1. In the index.html file, create `<a>` tags for each journal entry. The content between the opening and closing tags should be the names of the entries. The `href` attribute should look something like `href="entries/entry1.html"` for each link.

2. Optional: You may wish to wrap each anchor link in its own <p> tag so that you can include other information. Perhaps the date of the entry or the number/name of the lesson for which the entry was written.
3. Save and test to be sure that every link actually goes to the proper entry.
4. Wrap these navigation links in one `<nav>` tag, to provide better semantic structure and options for styling.
5. Style the navigation as you see fit, adding the CSS code to the styles.css file. Be sure that all of your styles on `<a>` tags use descendent selectors like `nav a{}` so that your navigation styles do not override the inline `<a>` tag styles within individual entries. Save and test.

Part 3: Cleanup & New Entry

1. Continue working until you are satisfied with your HTML and CSS code, and you are certain that all of your code is working as expected.
2. Be sure all of your files are saved in the proper locations.
3. Create a new entry in your journal, at least 3 complete sentences about anything related to hyperlinks. Copy your template.html to get started with the new entry, and be sure to add a link to the new entry inside the index.html file navigation.
4. Extra Time: Add "Previous" and "Next" navigation to each journal entry so that visitors do not have to go back to the homepage to continue reading your journal entries.