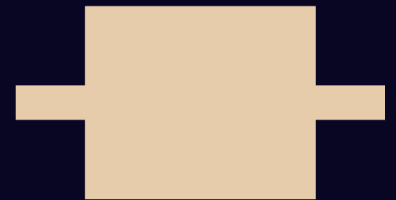
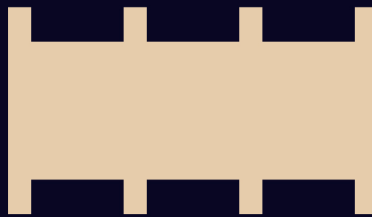
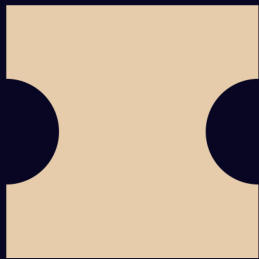
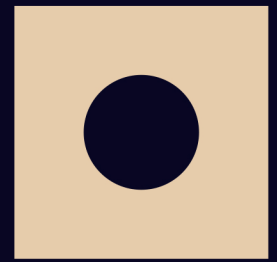
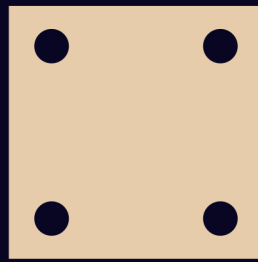
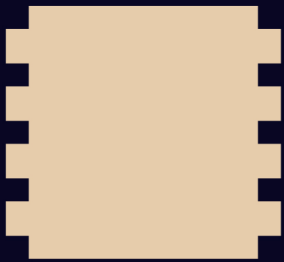
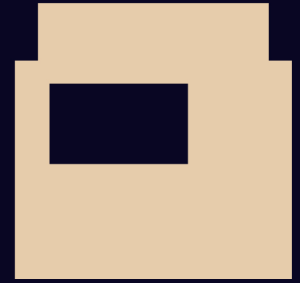
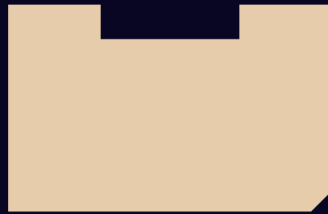
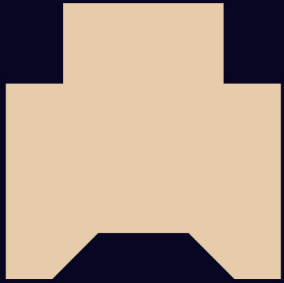


DigiKey



Club knowledge transfer guide

by Ramaditya Kotha, DigiKey Ambassador

Club Knowledge Transfer Guide

Educational clubs are often faced with a difficult dilemma - members leave after they graduate their institution and take their knowledge with them.

The solution to this problem is simple: implementing a knowledge transfer system. However, in many clubs I've been a part of, knowledge transfer has either been word of mouth, or so poorly organized that past knowledge is buried deep in a digital cave, making it unappealing and unwieldy to access or use.

In order to resolve this problem, I've created this guide to developing a feature rich, accessible and easily implementable knowledge transfer system. In developing this workflow, I had 4 main requirements:

- Simple but rich syntax with Markdown formatting
- Wikipedia-style formatting for usability
- An easy to implement website to make the system truly accessible
- And most importantly, easy to use even without a computer science background

Ultimately, I settled on a simple solution: storing files on GitHub, editing them with the Obsidian markdown editor, and publishing them using a tool called [quartz](#). The website is hosted on GitHub pages, for the ultimate ease of use.

Why is Knowledge Transfer Important

Typical members of college clubs are only present for 4 years. In that time they amass a huge amount of knowledge about club operation, technical procedures, and even club history. As these members graduate, all this knowledge is lost, leaving new members of the club to start from ground zero.

Rather than hitting the reset switch for each new generation of club members, it's much more productive to have new members stand on the shoulders of the people that came before. This allows them to spend their time improving and developing the club, rather than repeating the actions and mistakes of previous club members.

Some clubs rely heavily on word of mouth knowledge transfer, where older club members teach younger club members how to run the club by directly showing them how to do things, or by having knowledge transfer meeting with them. This approach by itself has its pitfalls. The quality of knowledge transfer is entirely dependent on the individual doing the knowledge transfer. It's easy to forget some of the things you've done in a year, and even easier to forget what you've done in your 4 years at a club.

Additionally, a very large lesson learned from the pandemic lockdowns is that word of mouth knowledge transfer is a very fragile system. Almost all of the knowledge not explicitly written down before the pandemic was lost to club members after the pandemic, requiring us to start from square one.

The solution to all this is a robust approach and *club culture* to documenting things meticulously. In the moment, it may seem tedious, but documentation can save someone down the line hours on troubleshooting the same issues you may have ran into.

In the following sections, I'll walk you through documentation guidelines for some common things, as well as how to set up this documentation system. At the end, you should end up with a rich website, ready to be filled with articles and information from your members!

Knowledge Transfer Guidelines

Generally, it's best to separate your year to year and general documentation. This aids in keeping your knowledge transfer material easy to navigate, increasing the chance people actually end up using it.

General documentation is stuff that's referenced by your club each year, and generally doesn't change much on a year to year basis. It should cover broad topics that are widely applicable like downloading software, contact lists, and general knowledge transfer topics. I'm being vague here because what qualifies as "general documentation" is truly up to the individual. General documentation should be clean and polished, as many generations of club members will ideally be referencing it. It should also be reviewed often to keep it up to date.

Year to year documentation topics should include all information specific to just a single year. This information is useful to go back and look at changes implemented and solutions relevant from year to year. This information also tends to be more cluttered, and can serve as a convenient location to hold less polished documents such as meeting notes, planning, etc.

General Documentation

Sponsorships

One of the common difficulties with having a poor knowledge database is keeping track of sponsorships. While clubs churn through members on a year by year basis, contacts and terms with sponsors often don't. It can be incredibly frustrating for a sponsor to reorient and educate a new point of contact each year. In the worst case, this can lead to losing your club sponsorship.

To prevent this from happening, it's best to store the terms of a contract and write down the terms explicitly, for all members to see.


Whenever you do negotiate a sponsorship agreement, no matter how simple it is, be absolutely sure to create a statement of work document. A statement of work document outlines the services/goods/money a sponsor will provide you, and what they expect in return. This document should be agreed upon between you and your sponsor, and saved in your club knowledge transfer system for future reference. A statement of work serves as an easy point of reference for the details of your sponsorship.

On top of this, you should put together a document outlining not only relevant sponsorship information, but also a list of to-dos for your sponsorship obligations. Here's an example of

such a document for a fake company called BearBox. Note the hyperlink to the statement of work document:

BearBox Sponsorship

Contact:

- Big Bear
 - (xxx)-xxx-xxxx
- Sponsorship Duration:
- YYYY - Present
- [Statement of Work](#) 

Summary

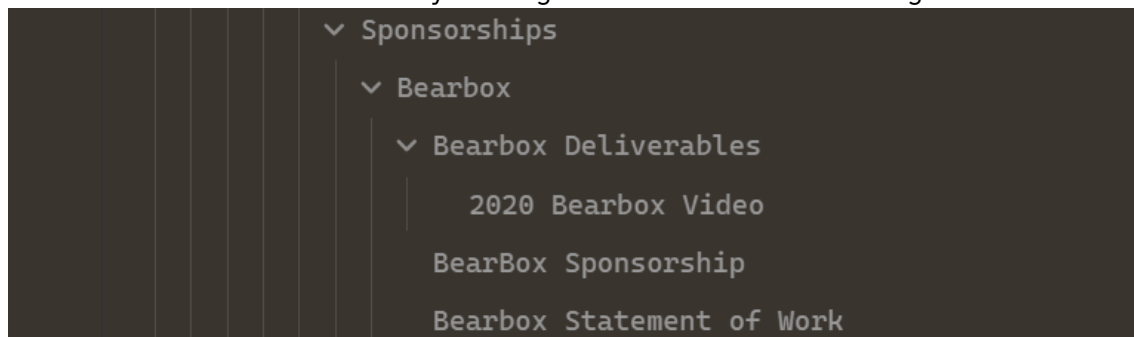
To-Do

- ☐ Promo Video
- ☐ BearBox Ad
- ☐ Bearbox Flag

Additional Information

If you have a sponsorship that's been running for a few years, it's very worth it to keep record of what you've done for your sponsor in the past. Doing this gives a valuable resource to future members of your team to see the type of work you've done, as well as develop new and appealing incentives for your long standing sponsors!

All this information can easily be organized with the following file structure:



Contact Sheets

Contact sheets are critical to maintain your clubs connections with people outside your club. They're a great way to document alumni contact, company contact, or event contacts. As far as documentation goes, contact sheets are very simple. Here's a basic example:

Name	Title/Position	Company/Organization	Phone	Email	Graduation Year (If Alumni)
John Doe	Software Engineer	NimbusTech			2022
Jane Smith	Marketing Director	Frostbyte Inc.			
Alan Chen	Mechanical Engineer	BajaCore Dynamics			2025
Maria Lopez	Product Manager	SkyBridge Systems			2021
Derek Johnson	CTO	BlueHelix Labs			
Emily Zhang	UX Designer	Lunaris Creative			2020
Carlos Martinez	Business Analyst	QuantumForge			2023
Sarah Kim	Legal Advisor	AeonTrust Group			
Liam Patel	Operations Manager	Vertex Logistics			2019
Olivia Green	Data Scientist	NovaMetrics			2024
Nathan Wright	DevOps Engineer	CloudNest Solutions			
Ava Brooks	HR Coordinator	PinnaclePoint			2023
Ethan Rivera	Systems Architect	StratosNet			2020
Zoe Bennett	Graphic Designer	PixelForge Studios			
Mason Young	Finance Officer	IronClad Capital			2018
Sophia Hill	Research Scientist	Ecliptica Labs			
Jackson Lee	QA Engineer	NeoNova Tech			2022
Lily Adams	Content Strategist	BrightMatter Media			2021
Benjamin Torres	Field Engineer	VulcanEdge Systems			
Grace Nguyen	Customer Success Lead	CascadeCloud			2023

Of course, you should modify this sheet to fit your needs. I also recommend separating your company/event contacts, and your alumni contacts. Separating the two makes it much easier to plan alumni events, or find relevant alumni.

General Procedures

It's absolutely critical that you clearly document your general procedures. Oftentimes, the nuances of these procedures can be lost during training, or simply just skipped over. If I had a nickel for each time I've heard "who do we contact at the school for xxxx?", I'd be rich.

When documenting general procedures, you want to aim for a Wikipedia style structure. One of the powerful parts of Obsidian and the system I'll be showing you in the next section, is the support for Wikipedia style links and the search engine built in. With this in mind, meticulously document your procedures, naming them in a detailed manner and using as many keywords as possible. This will make it easier to search for them in the future.

Here's an example of documenting a general procedure:

Taking out the Trash

```
1 The trash can in the middle room uses these trash bags. Please tie the trash bags tightly and throw them away in the large south  
2 dumpster.
```

```
3 ⚠ Warning
```

```
Do not use the north side dumpster, it is reserved for the art department
```

```
</>
```



This is also a great procedure to follow when documenting club history! Write about significant events in your club! For clubs that run for a long time, it's always interesting to see where the club's come from and how its evolved.

Software Usage

Software usage is the last example of documentation I'll talk about. When training new members, it's absolutely critical to teach them all the relevant software they'll need to succeed in your club. You should aim to create detailed documentation about not only how to install a software, but also how to use the software itself.

While it's helpful to learn from someone lecturing or teaching you directly, it's often useful to have a resource for that software to reference later. Furthermore, it's useful to have a quick summary of your club's tech stack so that prospective or new members can come prepared to their first meeting with all the software they'll need installed.

Here's a great example of how you could set documentation for SolidWorks:

	Solidworks
1	Installation
2	
3	How to Use SolidWorks
4	1. Intro
5	2. Sketches
6	3. Part Modelling
7	4. Composite Part Modelling
8	5. Assemblies
9	6. File Management
10	7. Analysis
11	8. Exporting

All of the links don't necessarily have to be made by you, I often link to external resources that I myself found useful.

Year to Year Documentation

While year to year documentation can get very cluttered, it's good to maintain some structure.

BOMs and Planning Sheets

It's always a great idea to write down your bill of materials (BOM) for future years to reference. I have made great use of prior years' BOMs to inspire my designs, as well as understand design decisions they made.

You can use a separate BOM software, a spreadsheet, or store it directly into an Obsidian file, whatever works best for your club.

Year to Year Design Revisions

If you're a project club, a very useful thing to record is your year to year design decisions. I've been on many teams where the reasons we've designed things a certain way has been forgotten, leading us to make the same mistakes as prior alumni.

You should also record lessons learned for the same reason: to prevent people coming to the club after you from making the same mistakes.

Creating a Website

Now that you have an idea of the general structure for efficient knowledge transfer, let's set up the infrastructure to hold all of your knowledge transfer material. By the end of this section, you should have your own Obsidian setup, with backup to GitHub. You should also be able to publish your knowledge transfer database as an easy to access website!

Technology Overview

Markdown

The Markdown language is a common typesetting language used by many platforms. In a sense it allows you to "code" the format you'd like your text to be in. Once you get used to Markdown, you can quickly and seamlessly write beautifully formatted text documents. While I won't be covering Markdown's syntax here, please reference this [awesome guide by Obsidian on Markdown](#).

Obsidian

Obsidian is a notes/knowledge base application. It provides an interpreter for markdown formatted files, as well as support for code blocks, LaTeX, and mermaid.js flow charts. The full feature list of Obsidian can be [found at their website](#).

One of the most valuable aspects of using Obsidian is that it provides support for Wikipedia style links. Wikipedia style links allow you to hyper link to other documents that you've already written in your knowledge database. The result is a connected web of knowledge. Say you write an article on bees. You can easily create hyperlinks within that article to an article you've already written about honey. This keeps your knowledge base strongly connected and clean, as you can avoid repeating information.

Quartz

In order to create a website from our Markdown knowledge base in Obsidian, we'll have to convert all our files to HTML files. Quartz does exactly that. [Quartz](#) is a project by Jacky Zhao and is widely used in the Obsidian community to publish entire Obsidian knowledge bases to the web. Quartz provides a lot of support for styling your website like a blog. You can redact certain articles from being posted, add authorship credits and author pages, even *search within your own website in a search bar*!

Additionally, while there's no guarantee that Quartz or Obsidian will be supported far into the future, markdown files are just text documents with special formatting rules. This means that anyone who opens these files in their notepad will be able to view and edit them, albeit in raw, uncompiled text.

We will host the website using GitHub pages, with the option of buying our own domain if desired. This is because GitHub pages is easy to manage and free to use.

Setup

Package Installation

For starters, you'll need to install the following things:

1. [Obsidian](#): A free to use markdown editor

2. [Git](#): We'll use this to sync our files with GitHub
3. [NodeJS](#) v20
4. [npm](#) v9.3.1

Note

If you haven't already, [connect your git installation to your GitHub account](#)

Using a terminal, navigate to a location on your desktop where you'd like to store the knowledge transfer files. Execute the following code snippet:

```
git clone https://github.com/jackyzha0/quartz.git
cd quartz
npm i
npx quartz create
```

Follow the process in the terminal to set up your Quartz, you should be able to get away with all the default options. If you then open up your folder structure, you should be able to see a folder called content. This is where all the content for your knowledge base will go. Any markdown files in this folder will be processed by Quartz.



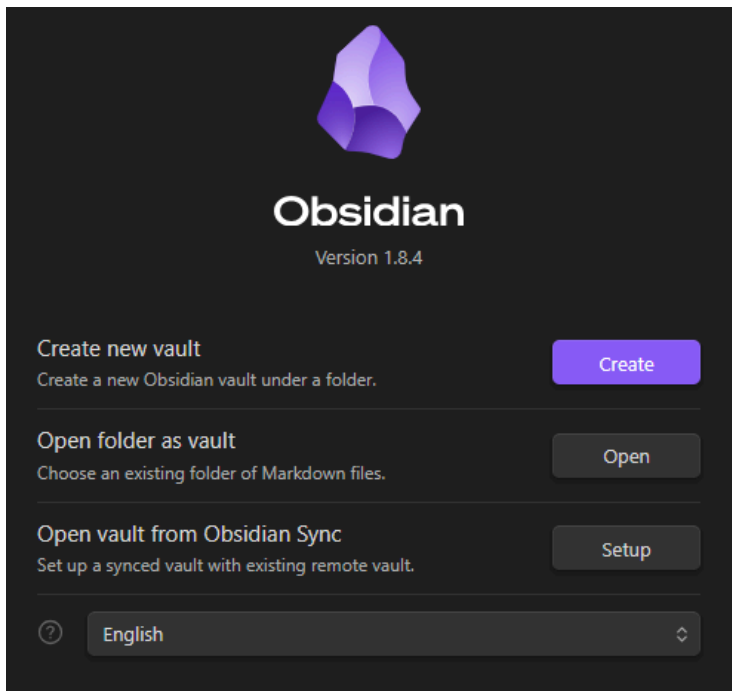
```
.github [DIR]
.quartz-cache [DIR]
content [DIR]
docs [DIR]
node_modules [DIR]
public [DIR]
quartz [DIR]
```

Setting up Github

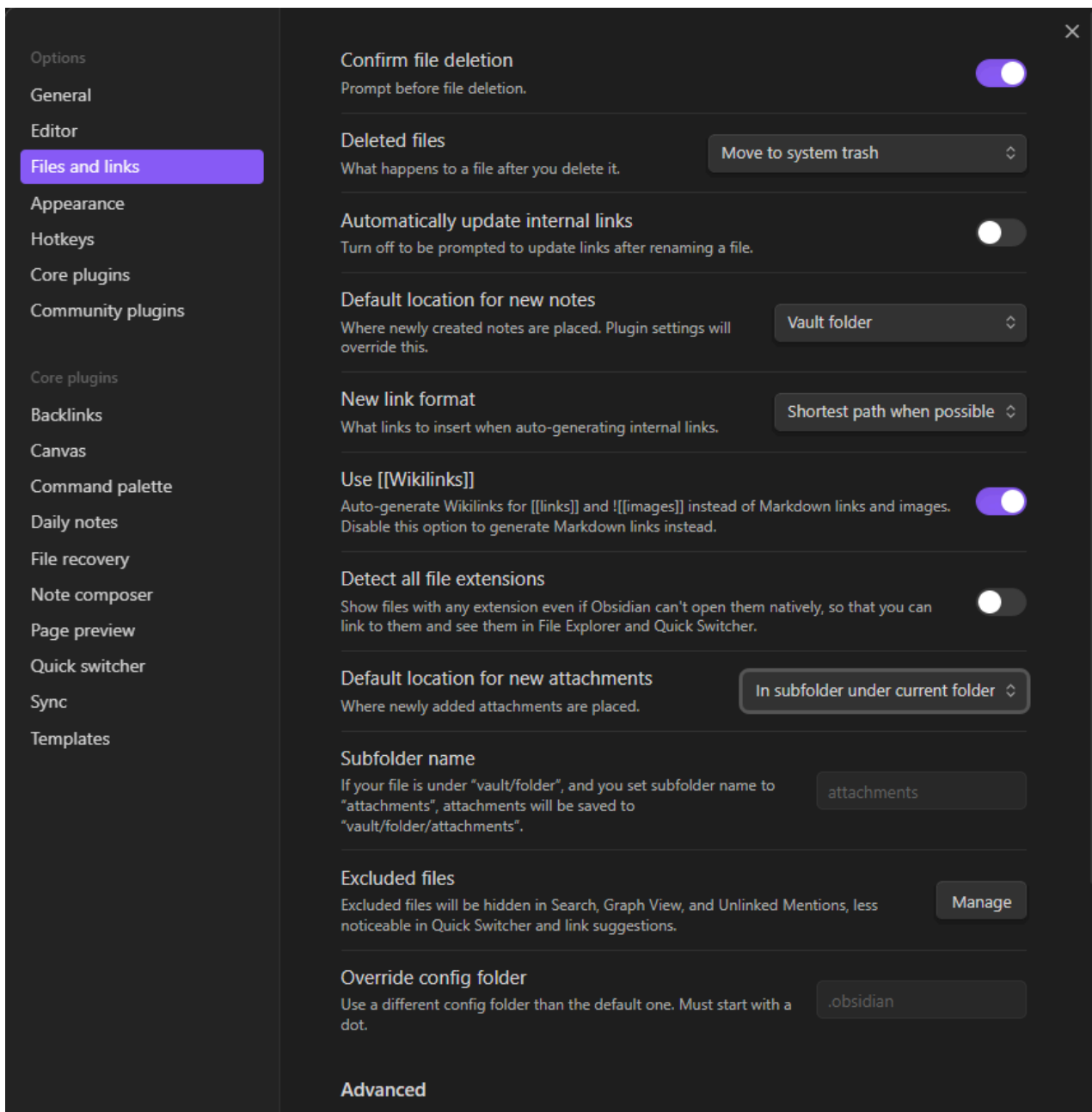
Now we can setup Github. We use Github to host a backup of your knowledge base, as well as eventually host your website. Follow [this guide](#) to complete the Github setup.

Setting up Obsidian

Setting up Obsidian is very straightforward. When you open Obsidian, you'll be greeted with the following screen:



Click on the "open folder as vault" option, and select the content vault from earlier. In your Obsidian settings, you'll want to go to "Files and Links" and change the "Default location for new attachments" to "In subfolder under current folder". This will keep all the images and attachments cleanly organized in our file system. You can leave the "subfolder name" option default. These folders won't appear in the final website.



You should now be all set to start writing!

Using Quartz

Now that you're able to write, let's talk about actually using Quartz. Like we talked about earlier, Quartz is a Markdown to HTML compiler. When you want to perform an action with Quartz, you'll want to use a terminal of your choice and navigate to the "quartz" top level

directory:



Building and Viewing Changes

Once you've started to make changes, you may want to check what your article looks like on your local computer before pushing your changes to the GitHub repo.

In order to build the website locally, you'll use the following command:

```
npx quartz build --serve
```

Note

If alerted with "Cannot find package 'yargs'" error, run the following command:

```
npm install yargs
```

Afterwards, you'll want to visit <http://localhost:8080/>, where the website will be locally hosted.

For more build options, please use the following command:

```
npx quartz build --help
```

You can actually make changes to the markdown pages and watch your website update in real time when viewing locally!

Syncing with GitHub

For convenience, Quartz ships with a few commands that make managing updating the website with your local changes simpler. In order to push updates to the website, you can simply use:

```
npx quartz sync
```

For more options, please use:

```
npx quartz sync --help
```

⚠ Caution

Please ensure that your current working branch is up to date with main, as modifications could overwrite someone else's work or cause conflicts.

Voila! That's all you'll need to start contributing.

Quickstart

Here's a quick summarized workflow for when you want to make a contribution, but don't remember a few useful commands:

1. Navigate to the quartz folder with your terminal of choice (NOT the content folder, it's parent folder)
2. In the terminal, run `git pull`
3. Open the content folder in Obsidian and type out your contribution
4. View local changes with `npx quartz build --serve`
5. Push changes to main with `npx quartz sync` once satisfied

Contributing

Contribution guidelines are a key part of maintaining any knowledge base. Quartz and Obsidian have a few specific ones that I'll cover here.

A Quick Word On Writing For Knowledge Transfer

When you're writing with the intention of providing information for someone else, try to be cognizant of what they may or may not know. While you may have familiarity working on something, there's a good chance that the person taking your place has no experience whatsoever. You should avoid vague statements and put yourself in their shoes.

I highly recommend adding plenty of figures, and explaining things in as detailed and thorough manner as possible.

Frontmatter

Quartz and Obsidian both come with support for a special syntax called *frontmatter*.

Frontmatter is extra metadata at the top of a Markdown file that can be used to store key information about specific notes:

```
1 ---
2 title: Example Title
3 draft: false
4 tags:
5   - example-tag
6 ---
7
8 The rest of your content lives here. You can use Markdown here :)
```

Frontmatter is very useful for organizing your knowledge base. I highly recommend [reading the documentation](#) to get the full use out of frontmatter.

Authorship Pages

By using tags that correspond to the names of contributors in your club, you can easily attribute parts of your knowledge base to different people. This is incredibly useful for future generations to have a point of contact about prior contributions to the knowledge base. Quartz also lets you have a *tag page*, which in our case essentially functions as an author bio.

Simply make a folder in your Obsidian vault called "tags" and add notes in the folder with the same names as your tags. Each of those notes can contain a bio of the authors.

To-Do Lists

With larger clubs and teams, it can be a pain to keep track of who's writing what, or even to divvy up the writing work. To remedy this you can create a really simply to-do list. You can prevent the to-do list from showing up on the website by setting the "draft" flag in the frontmatter to "true". Here's an example of a to-do list I put together utilizing the author

tags we talked about earlier:

Software

- ☐ Center lock Software Overview
- ☐ Controls Software Overview

Research and Testing

- ☐ M24 Position Control System Testing [#RamadityaKotha](#)

Knowledge Transfer

History

- ☐ 2025 Trailer Restoration [#RamadityaKotha](#)

Manufacturers

- ☐ Add RUL file to JLC and update JLC

General

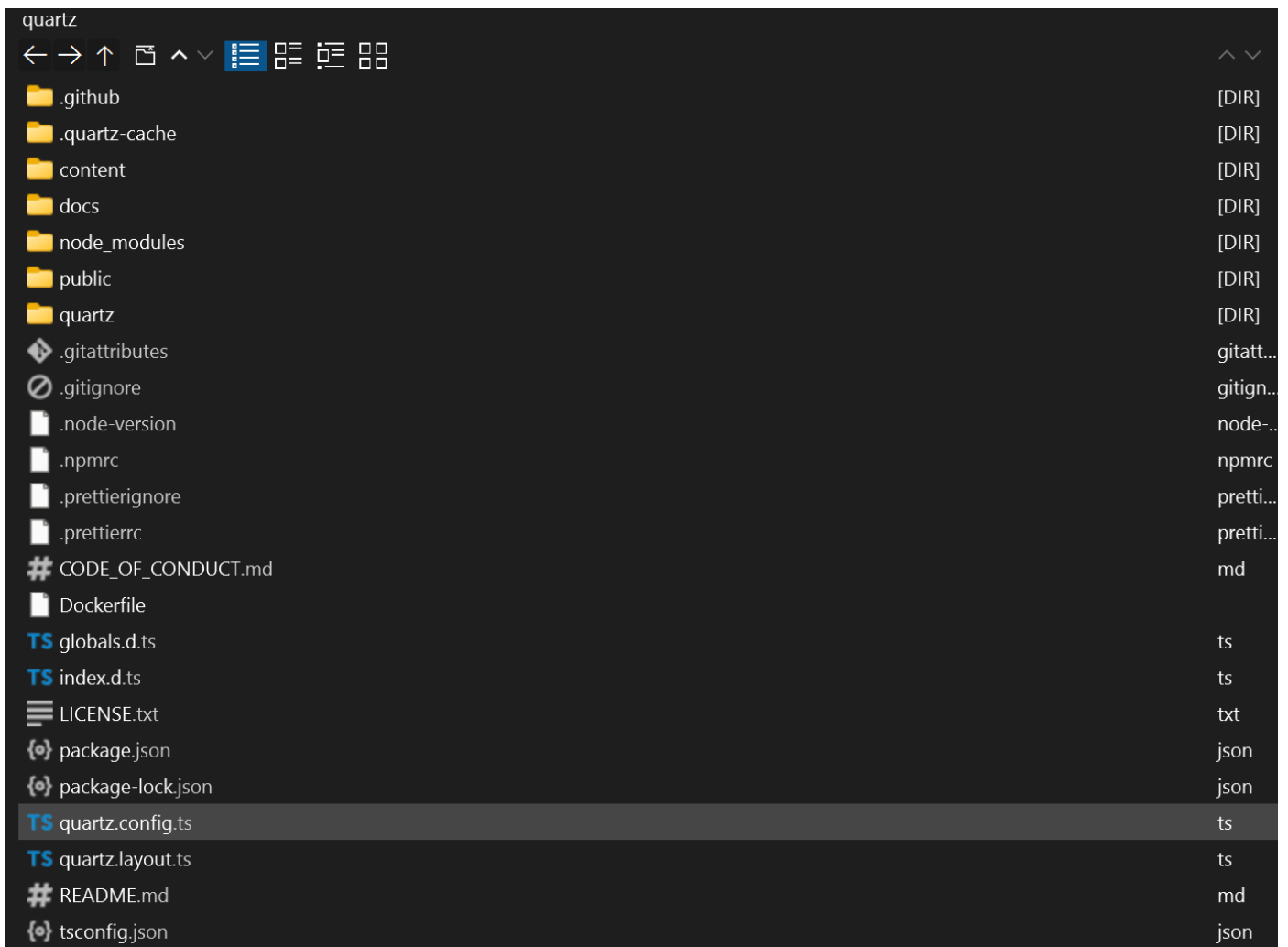
- ☐ Engine Wiring
- ☐ Brake Lights Wiring
- ☐ Fix Tuning Vehicle Parameters [#RamadityaKotha](#)

Website Customization

Quartz also comes with capability for a lot of customization. These are all optional, and some customizations require a small level of comfort with coding.

To change the logo of the website, you'll use a file explorer to navigate to the "quartz/quartz/static/" folder. In here, you'll replace "icon.png" with an icon of your choice!

To change the website colors is a little more involved. First use a text editor of your choice to open the "quartz.config.ts" file in the top level directory of your Quartz installation.



In the "colors" section, you'll change the hex codes of the various fields to change the colors of the website. You'll be able to find information about what each field changes in [this documentation](#). You can experiment with changing any of the other fields.

Going Further

If you'd like to go further, Quartz offers many more customization options and features than what I've shown here. You can read about these [in their feature list](#)!

And voila! You should now have your own knowledge transfer website up and running, and a decent idea of how to keep it running well for future generations of your club!

