

note-taking  
with the power of  
graph theory

trevor nelson - mmxxii

if you've ever used a computer before,  
you've probably seen one of these:

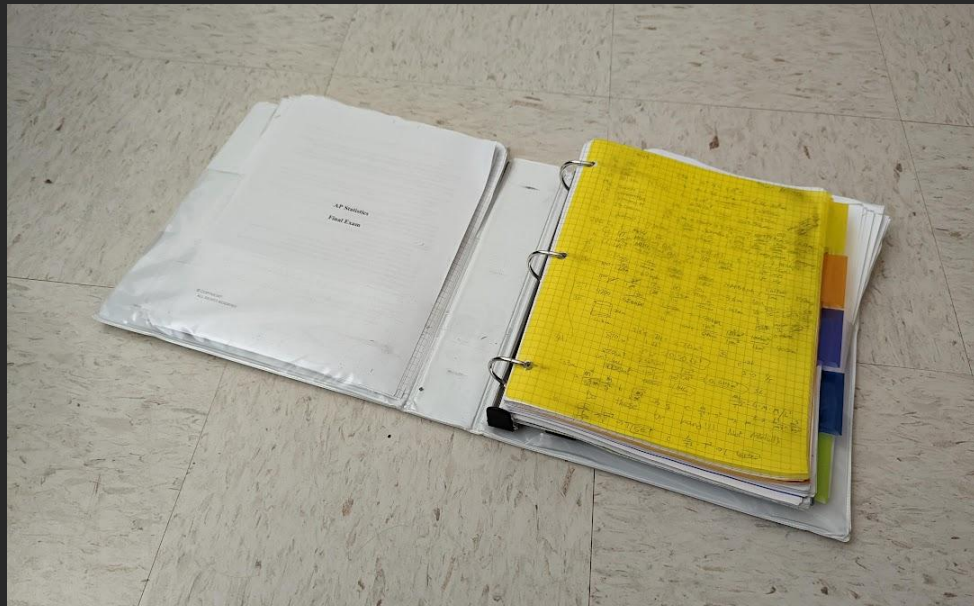


## folders are really cool

- let you categorize information
- can be nested to form a hierarchy
- we've been categorizing things like this since before computers even existed!

# unfortunately, they have a few issues.

- strict categorization
  - some things belong in more than one section
- information recall
  - good luck finding anything in here!
- digitizing folders and files helps alleviate the worst of these issues
  - *but doesn't fully solve them*
  - let's discuss



one of the many binders that got me through high school.  
it should probably be declared a superfund site at this point.

# strict categorization

- a recent example - malloc
  - this first showed up in CSC 230
  - came back to ~~haunt me~~ in CSC 246
- if i use class folders, where do notes on malloc go?
  - keep it in CSC 230?
  - move it to CSC 246?
  - split the file in two?
  - put it somewhere else entirely?
  - use a symlink?
- #tags can be helpful for this predicament
  - can have multiple in one file
  - independent of physical location

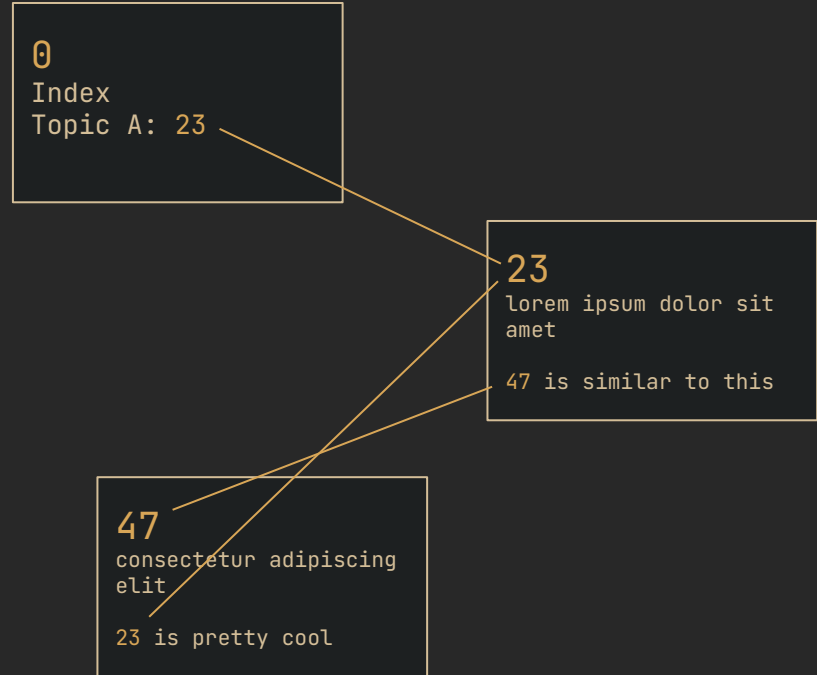
# information recall

- the entire point of taking notes, right?
- computers *can* make searching through folders easier
  - still a pain with large volumes of information
  - difficult to find related topics
  - difficult to make connections between topics

your brain doesn't put knowledge into folders,  
so why should your notes?

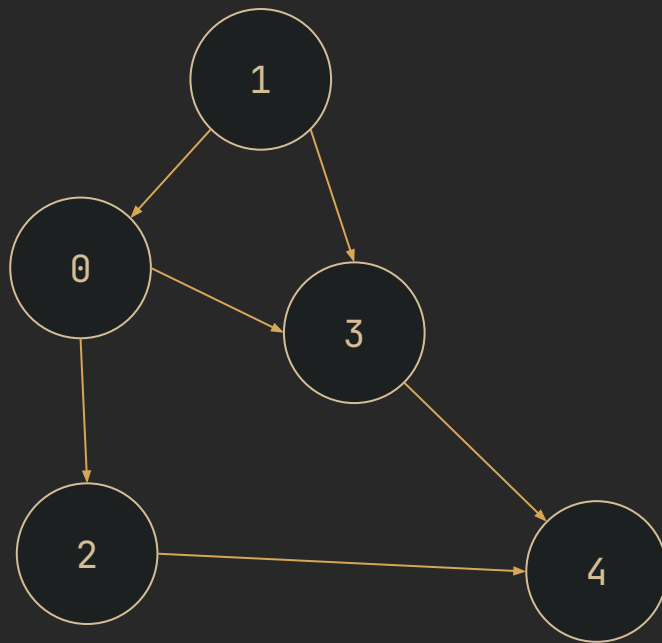
# enter zettelkasten ("slip box")

- been around since the 1600s
- put knowledge on notecards
- number them
- build a network of references using these numbers
- organize to facilitate lookup
- can create an index to jump to specific topics
- like links or pointers



# a quick little graph theory primer

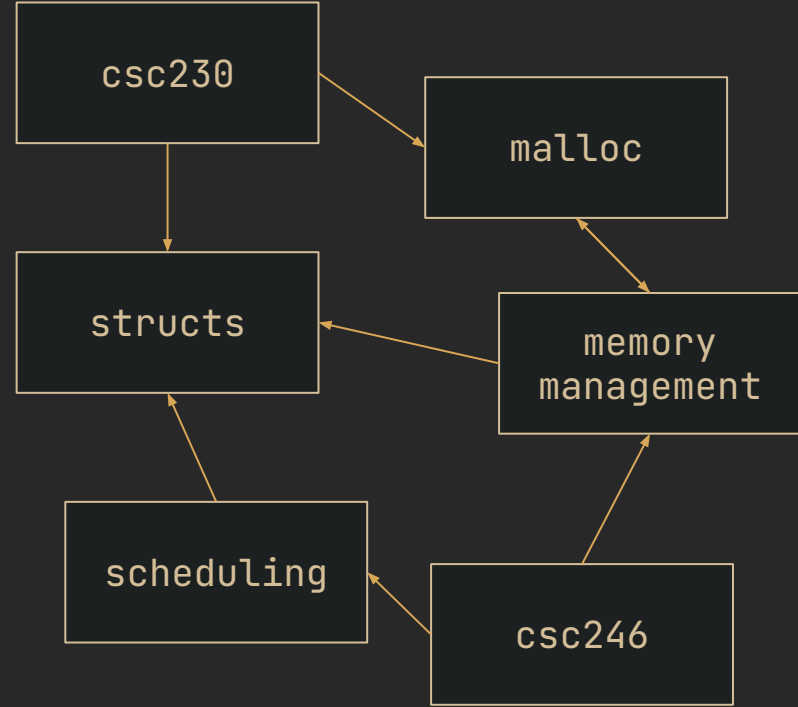
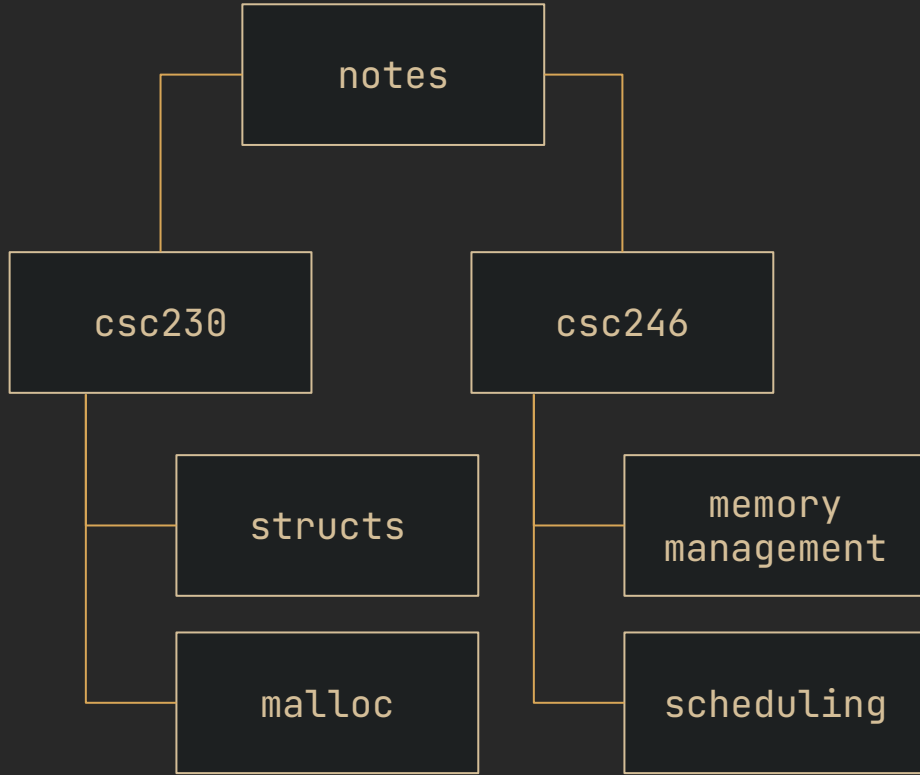
- a graph is a collection of vertices, with edges that link them together
- for the astute observer, zettelkasten seems a lot like a *directed graph*
  - each notecard is a vertex
  - links are the edges, which are directional (directed)
- graphs give more flexibility than a file tree
- easy to work with in software





# zettelkasten for the digital age

- don't need numbers anymore
  - just reference files by name
  - some people still use them
- can link with more specificity than the file level
  - many tools support links to subheadings and specific lines
  - `[[wikilink]]` syntax is common
- indexes are still used, sometimes called MOCs
  - "map of content"
  - one might have an MOC for computer science
  - that could then point to an MOC on the C programming language
  - don't have to use them at all



## evergreen notes

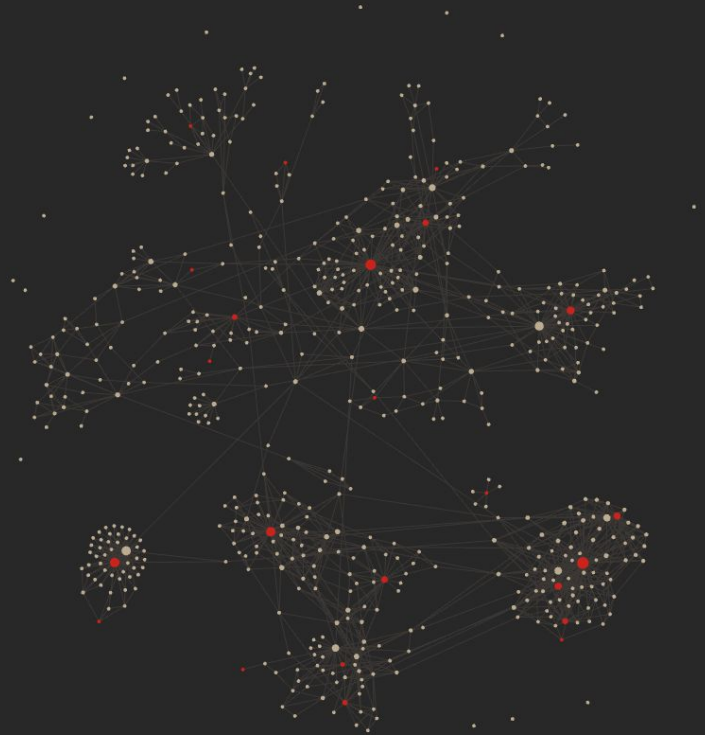
- evergreen notes should be atomic
- evergreen notes should be concept-oriented
- evergreen notes should be densely linked
- prefer associative ontologies to hierarchical taxonomies
- write notes for yourself by default, disregarding audience
- *notes start as transient but should solidify over time*

[https://notes.andymatuschak.org/Evergreen\\_notes](https://notes.andymatuschak.org/Evergreen_notes)

# pkm

personal knowledge management

- lots of tools available
- most tools use files in a directory, some don't
  - usually markdown
- facilities for including images, websites, pdf, etc.
- backlinks help find relevant connections
- visualizing the graph reveals emergent patterns
  - helps to resurface old notes



# obsidian

- proprietary
- local-first markdown
- electron-based
- lots of community extensions and themes
- great graph visualization

[obsidian.md](https://obsidian.md)

The screenshot shows the Obsidian Live Preview interface for a document titled "Finite Automata". The document content includes:

## Finite Automata

A type of `Finite State Machine`

- \* Finite automata process `Strings`
- \* A recognizer, whereas a `Finite State Machines` do more control and modeling
- \* Useful for text matching, other problems involving sequences or state transitions
- \* Pretty efficient to implement

### Deterministic

#### Deterministic Finite Automaton

$(Q, \Sigma, \delta, q_0, F)$

$Q$  - set of states  
 $\Sigma$  - alphabet  
 $\delta$  - transition function:  $(Q \times \Sigma \rightarrow Q)$   
 $q_0$  - start state  $q_0 \in Q$   
 $F$  - set of final states  $F \subseteq Q$

- Finite automata diagrams:
  - Circles as states, with a double circle indicating *Final States*
  - Arrows are labeled with one or more symbols
  - A start state shown using an arrow from no other state
  - E.g.

```
graph LR; start(( )) --> q1((q1)); q1 -- a --> q1; q1 -- b --> q2(((q2))); q2 -- "a,b" --> q2;
```

$Q = \{s_1, s_2\}$   
 $\Sigma = \{a, b\}$   
 $\delta = \{((s_1, a), s_1), ((s_1, b), s_2), ((s_2, a), s_2), ((s_2, b), s_2)\}$

8 backlinks Live Preview 683 words 3342 characters

# Logseq

- AGPL-3.0
- supports markdown or org-mode
- local-first
- browser or electron app
- graph view
- journals

[logseq.com](https://logseq.com)

The screenshot displays the Logseq application interface, which is a local-first, browser-based or Electron-based note-taking application. The main window is divided into two primary sections: a document editor on the left and a graph view on the right.

**Document Editor (Left Panel):**

- Title:** "Queries"
- Content:**
  - A heading: "What are 'Queries'?"
  - A bullet point: "Queries are for asking questions from your knowledge base and the outside world (in the coming weeks)." Below this text is a large image of a person silhouetted against a starry night sky with the Milky Way galaxy.
  - A byline: "By Greg Rakozy"
  - A list of links: "How to write queries?", "Simple Queries filters", and "More query examples".
  - A section titled "5 Linked References" with a sub-heading "Changelog 2020".
  - A breadcrumb trail: "Dec 29th, 2020 > | Features".
  - A bullet point: "Simple | Queries | support #experiment".
  - A note: "For example: Query: (and (todo now later done) [[tag]])".

**Graph View (Right Panel):**

- Navigation:** "Contents", "Recent", "Page graph", "Help".
- Panel Title:** "Page graph"
- Graph Structure:** A network graph with "Queries" as the central node. Other nodes include "Advanced Queries", "Changelog 2020", "Changelog", "Content", "Block properties", "block timestamps", "questions", "Dec 5th, 2020", and "Dec 7th, 2020".
- Search/Filter:** A search bar at the top right of the graph panel.
- Term Properties:** A panel at the bottom right showing "term/properties" and "examples: > let's add two more books:". It lists properties: "author: sönke ahrens", "publication-date: february 21, 2017", and "type: book".

# dendron

- AGPL-3.0
- built on vscode
- schema system
  - acts "object oriented"
- integrates with other services via pods

[dendron.so](https://dendron.so)

The screenshot displays the Dendron IDE interface. On the left, the Explorer pane shows a workspace with folders for 'python.d.boolean.md' and 'ruby.d.boolean.md'. The 'ruby.d.boolean.md' folder is expanded, showing a 'Node Graph' and an 'Outline' with nodes for 'True' and 'False'. The main editor shows the schema definition for 'Boolean' in a Yaml format:

```
1 ---
2 id: 884eb5f8-sh03-0r7f-0he0-7ce8f546abf
3 title: Boolean
4 desc: ''
5 updated: 1600019385816 (Sep 13, 2020, 18:49 AM)
6 created: 1600019385816 (Sep 13, 2020, 18:49 AM)
7 stub: false
8 ---
9 |
10 ## True
11 -- everything that is not mentioned under 'false'
12 ---
13 ## False
14 - nil
15 - false
16
```

On the right, the 'Dendron Preview' pane shows the rendered content for 'Boolean':

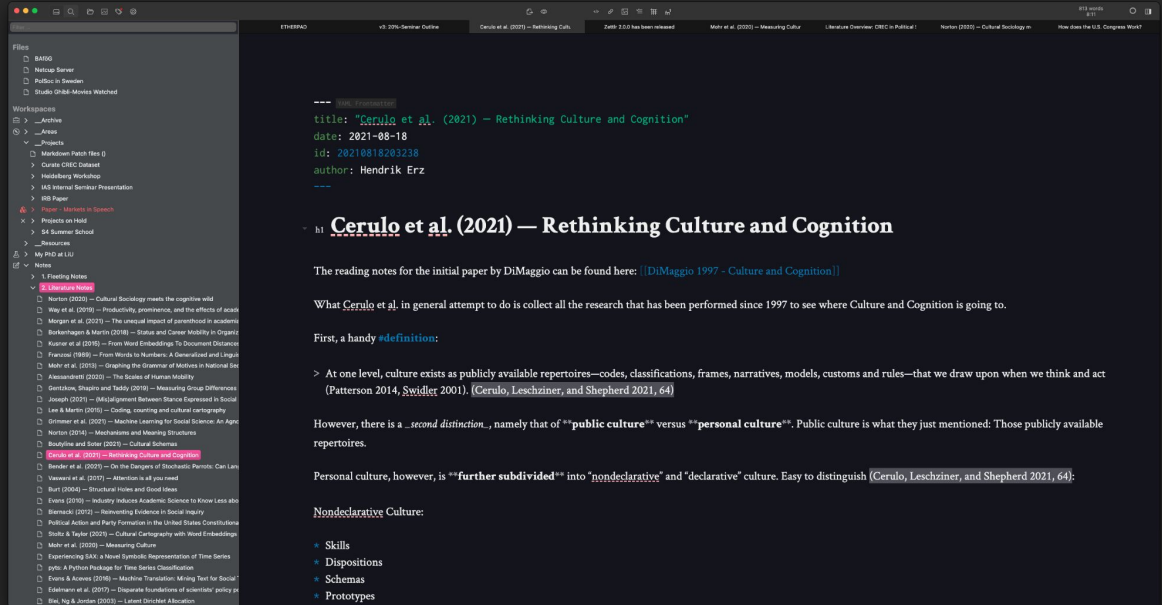
```
Boolean
True
- everything that is not mentioned under 'False'
False
- nil
- false
```

At the bottom, the 'Node Graph' pane shows a network diagram with nodes for 'Ruby', 'Lib', 'Dev', 'Data Structures', 'Boolean', 'Onboarding', 'Sample', 'Template', 'Byebug', and 'New-boolean', connected by lines representing relationships.

# zettlr

- GPL-3.0
- electron-based
- local-first markdown
- popular in academia
  - has tools for citations, etc

[zettlr.com](http://zettlr.com)

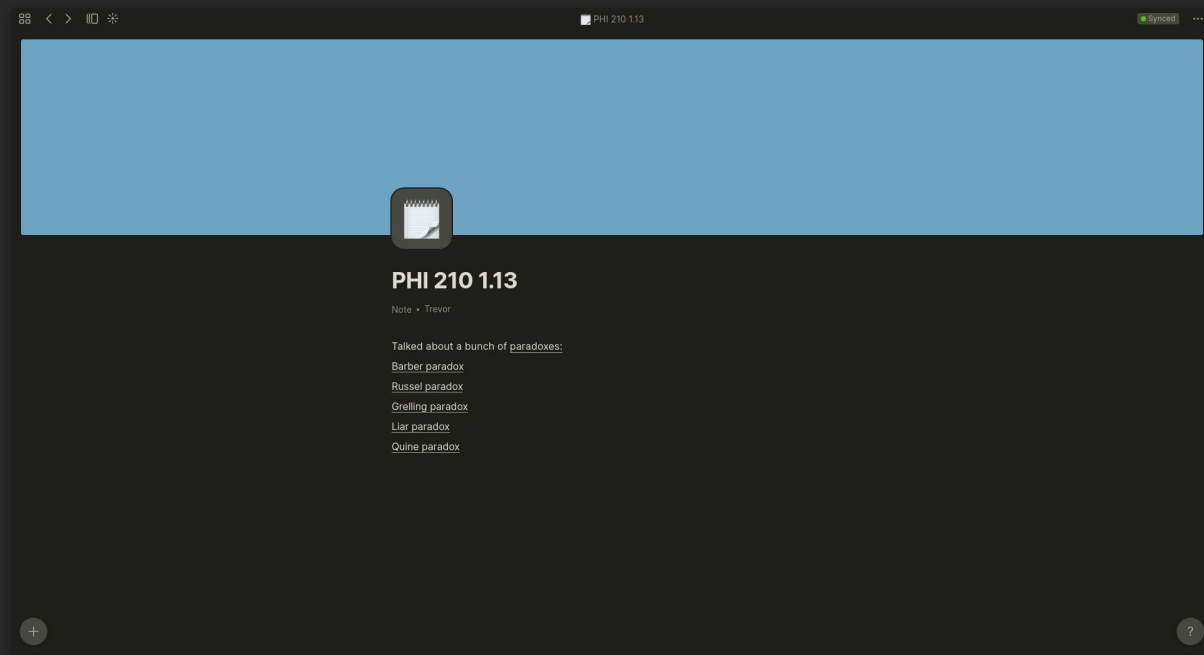




# anytype

- open source "soon"
- electron
- currently closed alpha
- uses objects and types instead of files
- stores and syncs data on ipfs
- all-in on web3

[anytype.io](https://anytype.io)



## vim/emacs

- vim and emacs can be configured for pkm use
  - add support for `[[wikilinks]]` or other syntax
    - emacs org mode has its own syntax
  - other functionality such as images
  - if you're a diehard vim/emacs user you can probably figure it out

# templates

- available in most tools
- preset content for new pages
  - i.e. insert certain headings automatically
- many tools support dynamic expressions
  - inserting today's date, for example
- some extend this to the idea of types, which can have additional properties in *frontmatter*
  - i.e. a watchlist
  - movies and shows can have properties for title, director, etc.
  - another page can query all the shows and display them in a table
  - obsidian's dataview plugin and anytype are the main examples

# dailies / journals

- many tools also use templates for journals
  - new file automatically created for each day
- typically used for task management
- also to create records of what you did on a day
- tracking mood, habits, etc

# syncing

- `git`
  - if you've got a collection of `.md` files, you can push and pull them with a `git` remote
- `syncthing`
  - fast and peer-to-peer directory syncing
  - a little more frictionless than `git` (no committing/pushing)
- or mount cloud storage
  - google drive, etc.
  - not the best for security, though

## tl;dr

- try organizing notes by concepts, not by date or course
- connecting notes together helps reinforce knowledge

## closing thoughts

- there is no universal "right way" to organize things
- the only right way is the right way for you
- experiment and see what works!
- lots of resources online to learn more

## sources

<https://en.wikipedia.org/wiki/Zettelkasten>

[https://notes.andymatuschak.org/Evergreen\\_notes](https://notes.andymatuschak.org/Evergreen_notes)

<https://obsidian.md>

<https://logseq.com>

<https://dendron.so>

<https://zettelr.com>

<https://anytype.io>