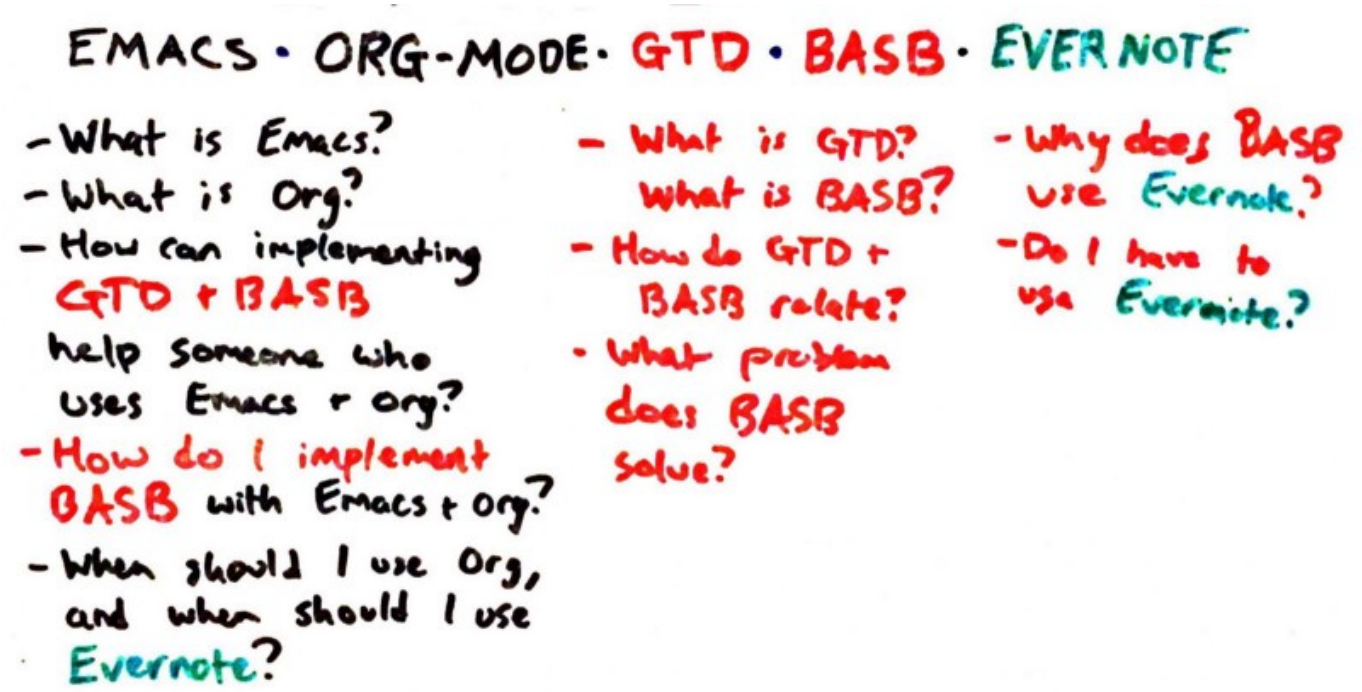
Tasshin

December 15, 2017

Implementing A Second Brain in Emacs and Org-Mode



This post is a continuation of "Building A Second Brain with Emacs and Org-Mode." If you haven't read that yet, read that post first.

It gave a high level overview of how BASB extends GTD, what Emacs and Org-Mode are (and why I wanted to implement BASB with them), and what principles emerge from using these tools. This post will go into detail about how I've implemented BASB using Emacs and Org-Mode. If you're already into Emacs and Org-Mode, I hope this gives you some interesting new possibilities for your Emacs configuration. If you're interested in BASB but are not familiar with Emacs and Org-Mode, I hope this post gives you a better sense of the power that Emacs and Org give you when implementing BASB.

Should I use Emacs and Org-Mode?

To begin, I want to clarify that I don't recommend Emacs and Org-Mode unconditionally. They're incredible software, but they can be a time sink, albeit an interesting one. I learned to program by playing with my Emacs configuration—it's a classic example of a breakable toy, a project that you're allowed to fail at. It is also notorious for having a steep learning curve. This is true, although I prefer to describe it as infinite, which is to say, you never master it. Every Emacs user I know is constantly learning new features, and even the best Emacs users don't know it all.

If you've got something that works for you, keep using it. I hope these posts help you learn something about how to iterate and improve on your current system. Still, some people might become interested in using Emacs and Org as a

result of these posts, so I want to give clear guidelines about when that might or might not be a good idea.

Under what circumstances would I recommend using Emacs and Org-Mode?

- **If text is the lingua franca of your notes.** Emacs has an unparalleled ability to edit text. You can't knock it till you try it. The thing that blew me away was C-t (that's Emacs notation for Control and t pressed simultaneously), which transposes characters. So if you have typed "cta" rather than "cat", you can place your cursor (with your keyboard, not your mouse) between the t and the a and transpose them with that command. There are similar commands available for transposing words, sentences, lines, and paragraphs
- **If you want to learn to program**, or you want to learn a Lisp, or literate programming.
- If being able to **create your own workflow** is essential to your success
- **If you program, touch source code, or Git repositories**: check out Magit, possibly the greatest Git user interface of all time, certainly the best I've used

When would I not recommend learning or using Emacs and Org-Mode?

- **If none of the above apply**
- **If you already use Vim**, which is often seen as a foil to Emacs. If you use Vim, I expect a lot of the ideas here will apply to you, too, even if the details don't. Of course, if some of the features mentioned here seem interesting, you might want to look into emulating Vim in Emacs, with Evil or Spacemacs. As they say in Spacemacs, the best editor isn't Vim or Emacs, it's Vim and Emacs. Of course, since I don't use Vim myself, I can't speak to that, so I come in peace
- **If you have a note-taking and/or task management system** that works for you
- **If you hate fiddling with computers**

Well, I want to try it. How do I install Emacs and Org-Mode?

If you're on Windows, use the official GNU web page. If you're on Mac OS X, download a binary and install it, or use a package manager like Homebrew. If you're on Linux or another UNIX-like, use your distribution's software or package manager.

Once you've installed Emacs, you can start climbing the Emacs learning curve with the built-in tutorial. Hit Control and h simultaneously, and then hit t. (In Emacs notation, which you'll learn soon, this is notated as C-h t). I think the tutorial took me 30–60 minutes, but it might take a little less or a little longer. You'll also want to complete a basic tutorial of Org-Mode. (You may also find this series of visual notes on Emacs useful.)

Between installing Emacs and completing two tutorials, it should take you something like two to five hours to decide if it's worth investing even more time to make Emacs and Org Mode a central part of your digital workflow. That's a lot of time, compared to a lot of things, but there's a lot of power in there. If you decide it's worth it, you never really stop learning Emacs. Feel free to reach out to me if you get stuck or have a question.

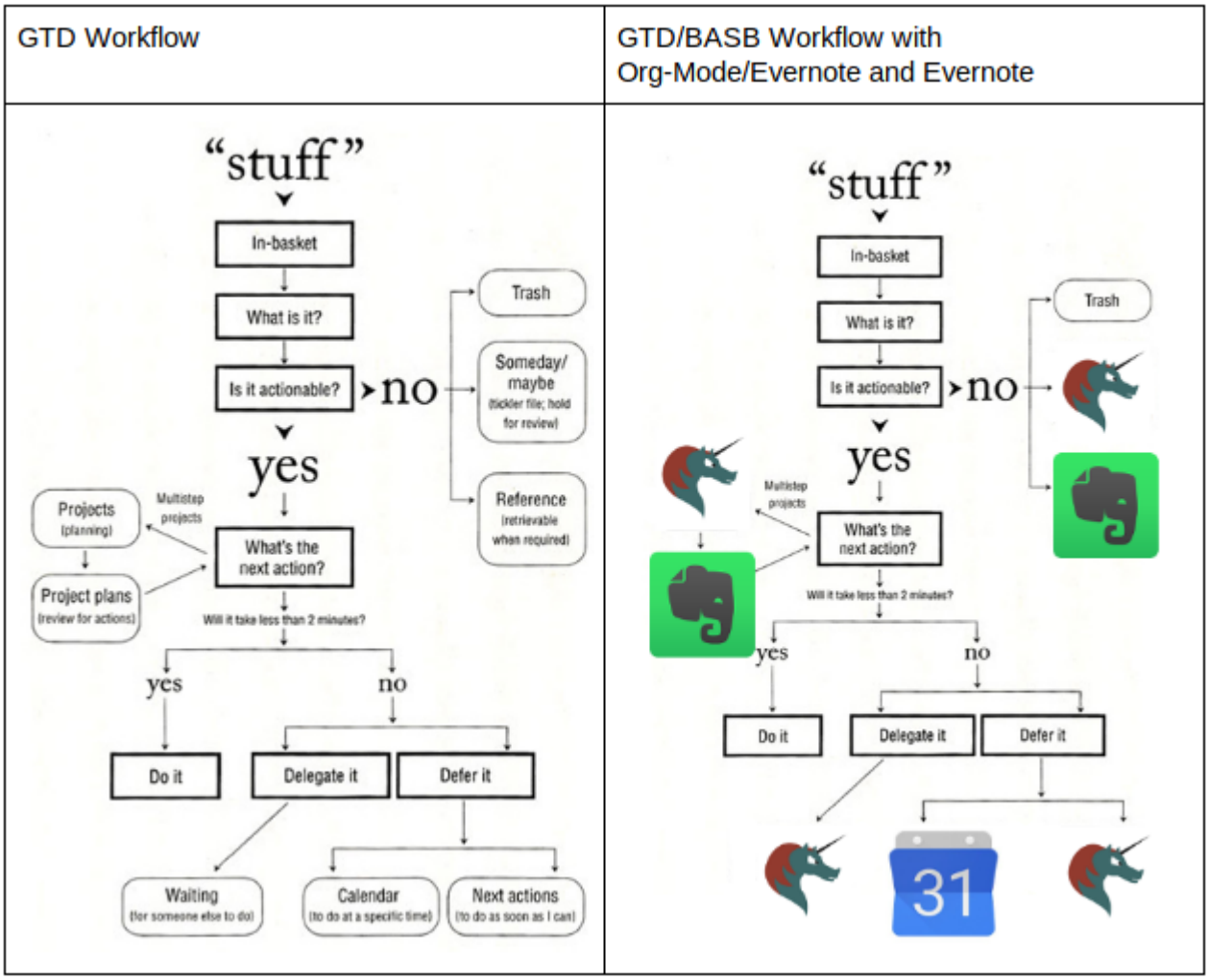
How would Emacs and Org-Mode users implement BASB?

I can see two ways to implement these ideas in Emacs and Org-Mode. I call them the *minimal* and *maximal* approaches:

1. The minimal approach: Use Emacs to edit text files, program, use shells and version control, and so on. Use Org Mode as a task manager, and to create text-heavy notes. When the notes stabilize, export those notes to Evernote or another note-taking program for reference
2. The maximal approach: Do everything in Emacs and Org-Mode, hard stop

It isn't immediately obvious how to implement the maximal approach in Emacs. Org-Mode is excellent at dealing with actionable information, and generating notes. It's also good at maintaining structured notes that include checklists, executable source code snippets, etc. But it's not good at everything. A good reference system should be able to easily store all kinds of materials: not just text (which Emacs excels at), but also handwritten notes, photos, screenshots, voice memos, videos, and more. Emacs can do most of that, with attachments, but the interface is ugly compared to more user-friendly software. Moreover, since it's designed for computers with physical keyboards, it's not well suited to capturing or searching for information on a mobile device. Evernote, on the other hand, is designed to store, sync, search, and share mixed media notes across platforms.

Accordingly, I've mostly adopted the first, more minimal approach for myself. I try to use Emacs for what it's good at (editing text files, using shells, programming, and creating notes with Org), and use Evernote for what it's good at (storing reference materials of multiple mediums, not limited to plain text). This visual diagram, based on the GTD workflow diagram, gives a broad visual overview of what this workflow looks like:



I'd like to be *capable of* implementing the maximal approach, so I've paid close attention to certain pieces of functionality that might be necessary for doing so. The remainder of this post will be about specific implementation details for this workflow, as well as the context for why I would want to implement each feature.

Capture / Progressive Summarization

I want it to be easy to create valuable notes in Org Mode. I also wanted to have a method for implementing the principle behind capturing that is taught in BASB: Progressive Summarization.

Progressive Summarization

The general idea of progressive summarization is that you want to create those notes so that, when you see them in the future, it's easy to quickly comprehend the purpose. You do that by moving through the following process:

- **Layer 1:** When you encounter something interesting, capture it
- **Layer 2:** Bold the most interesting parts
- **Layer 3:** Highlight the most interesting bolded sections
- **Layer 4:** Summarize the bolded portions and the note in your own words
- **Layer 5:** Turn your notes into something new: a tweet, a blog post, even a book

You don't do each step for every note. These are optional layers, added as needed. You add the layers **opportunistically,** that is, when you encounter your notes in your day to day work. This last bit should give you a sense of what Just in Time Project Management is all about.

Layers 1, 4, and 5 are easy to do in Org-Mode: just write text. To bold in Org Mode for Layer 2, you can wrap text in asterisks. I use a feature of Smartparens, sp-local-pairs, to make this very easy: just highlight the text, hit asterisk, and the text is wrapped with asterisks and hence bolded.

```
(sp-with-modes '(org-mode)
  (sp-local-pair "*" "*"))
```

Layer 3, highlighting, is the tricky part. For me, what's most important is to be able to export highlights to Evernote when they're ready. You can highlight HTML with the mark tag. Additionally, Org-mode can have inline literal HTML tags, like this:

@@html:@@bold text@@html:@@

I can use this to highlight text on export with the **<mark>** tag:

```
(use-package wrap-region
  :init
  (wrap-region-mode t)
  :config
  (wrap-region-add-wrapper "@@html:<mark>@@" "@@html:</mark>@@" "~" 'org-mode))
```

This method is similar to the Smartparens feature above, but allows for more complicated wrappers. It is sufficient for note creation (Layer 1 and possibly 2–3) in Org-Mode and exporting them to Evernote. However, it is also messy and cluttered, and not ideal for long-term storage in a file. I just use them when creating a note that I'm planning to export to Evernote once I've completed it.

Ideally, I want to easily highlight a region so that it is always highlighted in Emacs and HTML exports. My goal is to be able to do all of this in Emacs. To do that, I'd like to:

- Hide or clean this markup when displayed in Emacs, à la **prettify-symbols-mode**
- Automatically highlight the wrapped region in Emacs

John Kitchin's ov-highlight package aims to show persistent highlights in Emacs, but unfortunately it's not ready for prime time yet.

Exporting Notes from Org-Mode

Here is the workflow that I've established. I create notes in Org-Mode, process notes in Org-Mode, and store notes for reference in Evernote. Org-Mode simply works better than Evernote for me when creating text-heavy notes. Evernote is better at storing and accessing those notes. I want to make it as easy as possible to get the best from both tools. This means I need to make it easy to export notes from Org-Mode—whether they're whole files, headers, or even just specific sections of text—into Evernote.

The central part of this workflow is John Kitchin's ox-clip, which copies content from org-mode as formatted HTML to your system's clipboard. With it you can select a portion of a buffer and copy formatted HTML of that content to your clipboard for easy import into other programs.

Here are a couple of other tools that might be useful when exporting content from Org into note taking programs like Evernote:

- set **org-export-with-toc** and **org-export-with-section-numbers** to *nil*
- **org-preview-html**: creates a workflow for previewing org files as you edit them, popularized by Markdown-centric tools like Ghost
- **pandoc-mode**: easy access to pandoc ("a universal document converter") for a variety of formats, including PDF formatted LaTeX-style.

Exporting Notes from Evernote

If you start with the minimal approach and decide to migrate back to using Org-Mode for everything (as Sacha Chua, the pioneer of integrating Org-Mode with Evernote, did), then you'll want to know how to export Evernote Notes into Org-Mode. This project, enex2org, looks promising for converting Evernote's file format, enex, to org files. I haven't tried it myself yet, though. Alternatively, you can hyperlink to an Evernote with a shared URL. This really doesn't work all that well, but it makes a first step toward escaping the Evernote elephant.

Easy Narrowing and Widening

If you're writing a note in context in Org Mode—perhaps several levels deep in an Org file—you'll want to be able to pay attention to just the contents of that note, while keeping it stored at that location. Some people like to use indirect-buffers for this, but I prefer a feature called "narrowing" and "widening," which lets you limit what's shown (and what's editable) in an Emacs buffer to a particular region. Here's are two screenshots of my Emacs configuration, which is currently ~2500 lines. The first shows multiple sections, with all of the lines in the file accessible (some visible, all editable)—everything is "widened":

And here is a screenshot where I've narrowed to one sub-section of the configuration, one piece of functionality with explanation and the relevant Emacs Lisp:

```
**** org-autosort
     I wanted to be able to add a sort property to files or subtrees and get automatic autosorting. A user of /r/orgmode/ delivered a solution.

     #+BEGIN_SRC emacs-lisp
       (defun yant/org-entry-has-subentries ()
         "Any entry with subheadings."
         (let ((subtree-end (save-excursion (org-end-of-subtree t))))
           (save-excursion
             (org-back-to-heading)
             (forward-line 1)
             (when (< (point) subtree-end)
               (re-search-forward "^\*+ " subtree-end t)))))

       (defun yant/org-entry-sort-by-property nil
         "Apply property sort on current entry. The sorting is done using property with the name from value of :SORT: property.
            For example, :SORT: DEADLINE will apply org-sort-entries by DEADLINE property on current entry."
         (let ((property (org-entry-get (point) "SORT" 'INHERIT)))
           (when (and (not (seq-empty-p property))
                      (yant/org-entry-has-subentries))
             (funcall #'org-sort-entries nil ?r nil nil property))))

       (defun yant/org-buffer-sort-by-property (&optional MATCH)
         "Sort all subtrees in buffer by the property, which is the value of their :SORT: property.
              Only subtrees, matching MATCH are selected"
         (org-map-entries #'yant/org-entry-sort-by-property MATCH 'file))

       (add-hook 'org-mode-hook #'yant/org-buffer-sort-by-property)
     #+END_SRC
```

There are many good commands for narrowing or widening to specific parts of specific Emacs buffers. As with many Emacs commands, by default you need to learn many different keyboard shortcuts to make use of them, which can be an obstacle. Some time ago I wrote a function called **narrow-or-widen-dwim**. Artur Malabarba profiled and augmented this function in his blog, Endless Parentheses. I have combined it with his toggle-map, so for me, it's bound to **C-x t n**. It's easy to remember: "toggle narrowing."

Using Emacs and Org-Mode on the Go

If you're making Org-Mode a central part of your workflow whether for just tasks, or actionable and non-actionable information, you'll want to be able to capture information into Org from anywhere. This is a bit of a challenge, as Emacs and Org-Mode are really designed for use with a full keyboard, not a phone's screen. (This is a big part of why I'm embracing my Evernote overlords for reference materials.)

Still, there are some solutions here. On Android, check out Orgzly; on, iPhone, look into MobileOrg or beorg. I prefer beorg. It doesn't do everything that you can do in Org-Mode, but it's great for looking at your files, and looking at your agenda. Org agenda views are a quick and flexible way to collect and display related information from across multiple files. In this case, Beorg can replicate this functionality to show you any events or tasks that you have for a given period of time, such as "today" or "this week." Beorg can access events through your phone's calendar. On my computer, I access Google Calendar events with org-gcal. Beorg works great for capturing tasks and quick notes too, although you might want to capture notes with something else, such as Evernote.

Capturing Emails As Todo Items

You'll also want to integrate Emacs and Org-Mode with your email client, since many important tasks come from email messages. mu4e is a mail client (mu) for Emacs (4e). It has great support for processing emails as todo items. This is a good introductory article.

Inserting Links into Notes

org-cliplink allows you to insert a link from the clipboard, and it will set the text of that link to be the title of the linked page. By default, copying the link for something, such as org-cliplink's project page, would simply have the unformatted URL:

https://github.com/rexim/org-cliplink

But when inserted with org-cliplink, the link becomes this:

GitHub—rexim/org-cliplink: Insert org-mode links from clipboard

I wish I could do this in Evernote, but since I'm creating most notes in Org-Mode first anyway, it's easy enough to just copy the output with ox-clip.

Editing and Annotating PDFs in Emacs and Org-Mode

When using Emacs, you often stumble on some feature that you had no idea it could do. For me, that was learning that Emacs can view PDFs with its built-in DocView functionality. Unfortunately, Emacs' built-in PDF viewer is slow, memory intensive, and read-only. An alternative, pdf-tools, is fast and has read/write support for in-PDF annotations in Emacs.

The only snag is that you can't install pdf-tools like a normal package. It has many dependencies which vary by operating system and local set-up. I found it a little intimidating and involved, but the documentation is helpful. Here's a screenshot of what it looks like:

You'll want to change the default configuration. There are quite a few tweaks posted out there! I like interleave, shown above, which automatically juxtaposes a PDF with the Org file you are using to take notes.

Drag and Drop Images into Org-Mode

If you're planning on implementing the maximal approach, org-download will let you drag and drop images into a note with minimal configuration. It also exposes a function, **org-download-screenshot**, that lets you take a screenshot and that is automatically inserted into your org-mode buffer at point.

Be sure to set **org-startup-with-inline-images** to *t* or *non-nil*, if you haven't already. Then images will automatically show up where you want them..

Organize / P.A.R.A.

P.A.R.A. provides a simple and consistent system for organizing. It stands for Projects, Areas, Resources, and Archives:

- **Project:** a group of related tasks, linked to a specific outcomes and deadlines

- **Areas:** on-going spheres of activity (no deadline), with a standard to be met. For example, “call repair company” is a task; “replace garage door” is the relevant project; and “house” or “Home Maintenance” is an area, as you require a certain standard for you to continue owning the house and living there
- **Resources:** materials that aren’t tied to a specific project or area, but have some kind of interest for you, such as “emacs,” “productivity,” or “board games”
- **Archives:** inactive projects, areas, or resources. Examples might include a home maintenance project you completed last year, an old job, and a hobby you used to have but aren’t actively interested in any more

You clearly define your projects and areas. Then you replicate those projects and areas your inboxes and storage spaces. Tiago recommends that you ensure that your projects and areas are consistently and completely synced between your task manager and your note-taking program. You can replicate them in your other inboxes, such as your documents folder and cloud storage services, on an as-needed basis. Whenever you download a new file, record a new task, or save a new web page, you already know where it should go. If it’s actionable, it goes in the appropriate project or area in your task manager; if it’s not, you might store it in the relevant project, area, resource, or archive on your file system or note-taking program.

If this sounds crazy, well, now that I’ve implemented this system, the old way seems crazy. Many Emacs users include a note in their configuration explaining what each of their most important files does, and why. With this method, that information is unnecessary.

Here’s another, related point. For most Org-mode users, the actionable information is very close to the non-actionable information. This is because if you want to look at the actionable information, you can do so very easily with the agenda views, even if notes are next to tasks in the original files. When I first heard Tiago say that he likes to keep a strict distinction and separation between actionable and non-actionable information, I thought, “that’s crazy, I don’t like it, I love that about Org-mode, you’re just justifying a really backwards workflow.” (Sorry Tiago!)

It turns out Tiago actually has really sound reasons for recommending this separation. We have different needs for our tasks than our notes. Tasks need to be well-organized, up to date, easy to find and act on. There’s a higher quantity of reference materials, they change less often, and messiness is not only acceptable but contributes to creativity.

The proliferation of agenda files, the need to explain their purpose explicitly, and the constant juxtaposition and intermixing of actionable and reference information all demonstrate the need for a better organizational scheme. Since implementing P.A.R.A. and migrating reference materials to Evernote I have fewer and shorter agenda files, and each file has a clear purpose. Tasks and some notes remain. Org-Mode is excellent as a task manager, so it makes sense to keep tasks there; the remaining notes belong in Org-Mode, usually because they are directly relate to a specific action, or because they contain executable code. All other notes and files are nicely filed and organized in parallel containers on a separate service or system. Everything is neat and tidy in a completely unanticipated way.

These distinctions are already implicit in our organizational systems. For example, you probably already have a folder dedicated to your job or work. This becomes an area folder. I found that much of my data “self-organized” into the appropriate locations in a P.A.R.A.-based system. Because these distinctions are now explicit and consistent across systems, it’s easier to find everything, no matter what kind of information it is, or where it’s stored.

Project Overview with Org Column View

I wanted to get a quick view of which projects I have going at any given time. The column views feature works well to implement this.

I added several properties to my projects header in my tasks file. Properties are tags with values. For example, in this context, each project has a priority, A-C; a name (e.g., “Write post about Emacs”); a relevant area (“Business”); a deadline (a specific but flexible date); and an associated outcome (something concrete and specific—“post is published”—but also motivating—“generating new connections and ideas”).

```
* 1 Projects
  :PROPERTIES:
  :COLUMNS:  %1PRIORITY %18ITEM %13AREA %10DEADLINE %454OUTCOME
  :VISIBILITY: children
  :SORT: PRIORITY
  :END:
```

Here's what this dashboard looks like:

P	ITEM	AREA	DEADLINE	OUTCOME
B	* 1 Projects			
	:PROPERTIES:			
A	* Set up Bright..	Brightmind	<2017-12..	Brightmind has an
A	* Post Why Shin..	Brightmind	<2017-12..	We have a post ab
A	* Edit gift car..	Brightmind	<2017-12..	The correct gift
A	* Complete BASB	Business	<2017-12..	I have a function
A	* Write post ab..	Business	<2018-01..	I write and publi
B	* Plan Christma..	Relationships	<2017-12..	S and I have clea
B	* Network Coden..	Programming	<2017-12..	I have a working
B	* Write MBA pro..	MBA	<2018-02..	There is an essay
C	* Publish Miche..	MBA	<2018-01..	There is an inter

I can get this dashboard showing my current projects by hitting = (a speed command), **C-c C-x C-c** (**org-columns**), or by using a homemade function, **project-overview**, that quickly navigates to this view.

Stuck Projects

A stuck project is a project without any action steps or tasks associated with it. This is a concept that comes from David Allen's GTD.

Org-Mode can tell you which subtrees don't have tasks associated with them ("stuck projects"). You can also configure what it recognizes as a stuck project. Unfortunately, by default, this picks up a lot of noise: currently, it tells me I have ninety-five"stuck projects." I've clearly defined what my projects are, though, and I currently have eleven projects.

Moreover, those eleven projects each have an entry as a subheading of the "1 Projects" heading / subtree in my tasks file. That means that I can just get the stuck projects in that subtree. Voilà:

```
List of stuck projects:
  todo:       [#B] Write MBA profile with
```

Just one project is stuck. No noise, all signal!

Reviews in Org

I do weekly reviews, which David Allen calls the "Master Key to GTD." Implementing P.A.R.A. fine tuned this system. Reviews now happen easily, quickly, and consistently. I use a template in Org to facilitate this. It is essentially a checklist, but it has access to all of Org's power. I include text pulled from the Getting Things Done book and other sources to describe each step, but I also juxtapose that text with executable source blocks that help me accomplish each step (such as viewing my project overview, or stuck projects, featured above, and more).

Having a checklist-based weekly review clearly defines what needs to be done. Augmenting it with executable code makes it even faster and more fun. If I discover an improvement to the process, altering the template improves all future reviews.

Here's how. Org has a feature called "capture templates." They let you quickly capture information in a variety of ways: a todo item, a quick note, and much more. Here's a simple one, that enters a todo item in a certain file at a certain location,

and puts the point (the cursor) at a specific location:

```
("t" "TODO" entry (file+headline "~/org/todo.org" "Inbox")
      "* TODO %?\n")
```

The actual capture template is the last part. The rest is metadata about how to access that template and what to do with it when it's captured.

The reason I mention this is that I used to think you could only have one line capture templates in Org-mode, like the last example. However, I wanted to have templates that were longer than one line: a form with multiple data points, for example. And ideally, it would be stored in a file. Karl Voit has written about a powerful approach that extends the yasnippet template system with a project called yankpad, which lets you manage snippets in Org Mode. Org also has native functionality that extends capture templates, which looks like this:

```
("w" "Review: Weekly Review" entry (file+datetree "~/org/reviews.org")
      (file "~/org/templates/weeklyreviewtemplate.org"))
```

This means that hitting "w" in the capture template screen creates a new entry in a file for reviews, following the template stored in a different file. I've also created some convenience and keybindings that you can find in my configuration.

Retrieve / JIT Project Management

Having captured and organized notes that are relevant to you, you'll have a lot of materials to make use of in your projects. How can you find and use these materials in an efficient and effective way?

Building A Second Brain's method for retrieval is called "just in time project management," which is about making use of your reference materials to create more deliverables, more frequently, through what Tiago calls "intermediate packets." Another way of describing it is creating the "conditions for rapidly executing and delivering projects using existing assets."

The BASB course covers sixteen workflow strategies that can help you employ Just in Time project management. These strategies aren't necessarily directly related to particular Emacs configurations or principles. I do want to discuss two points that emerged from this section for me.

Archipelago of Ideas

The first, Archipelago of Ideas, is about connecting ideas on a topic. Its title is taken from a quote from a BoingBoing article author Stephen Johnson wrote about his writing process:

> *I launch myself into the actual writing of the book... [by opening my notes in my word processor]. Instead of confronting a terrifying blank page, I'm looking at a document filled with quotes: from letters, from primary sources, from scholarly papers, sometimes even my own notes. It's a great technique for warding off the siren song of procrastination. Before I hit on this approach, I used to lose weeks stalling before each new chapter, because it was just a big empty sea of nothingness. Now each chapter starts life as a kind of archipelago of inspiring quotes, which makes it seem far less daunting. All I have to do is build bridges between the islands."*

When I read Tiago's explanation of this workflow strategy, I realized that this strategy was already implicit in the way I was using Org Mode. When writing something I'll simply list all of my ideas on the topic, re-order them, and start writing

transition sentences between the ideas I want to keep.

RandomNote

Having information stored in an organized fashion makes it easier to find something specific. But what if you don't know what you're looking for? What if you want something relevant, but not predictable? You need randomness. You need a random note. A random note in your reference materials will be relevant to you, and you will understand it, but it's not expected, either. This is a wonderful tool for exploring unexpected connections.

RandomNote opens a random note in your Evernote. The BASB developer crew has implemented it across Evernote platforms. I wrote an Org-mode version, org-randomnote. It's on MELPA, a package archive for Emacs, so you can easily try it, whether or not you implement the rest of BASB.

Conclusion

I hope these posts have shown you some interesting configurations for Emacs and Org-Mode, explained what Emacs and Org are and why you might use them; what BASB is, how it relates to GTD, and what problems it solves; and what principles emerge from using all of these tools and workflows. I'd love to hear if you found these posts useful. If you have questions, or if you have a different way of implementing these features or solving these workflow problems, please send your thoughts and questions to @tasshinfogleman on Twitter.

If you enjoyed this post, subscribe to my newsletter to get updates on my new blog posts and current projects. Or, follow me on Twitter.

Productivity

Basb, Emacs, Evernote, Org Mode, Pkm

Tasshin