Twitter Thread by Sam Julien



Sam Julien
@samjulien

Reducing the Pain of Context-Switching ■

Context switching is a huge pain as both a dev & dev advocate and makes it hard to be creative!

Three things that help:

■ Centralize your capture
■ Review regularly
■ Automate where you can

Let's dig into each of these!



■ Centralize your capture

"Cognitive setup" is the big drain of context-switching. Try @fortelabs' "slow burns ■ vs heavy lifts ■■■■■".

Trade 1 big block of cognitive setup + creative work for short sessions that keep the project context fresh. More: https://t.co/BW0XjTudAi

First "slow burn" strategy: centralize where you capture ideas, take notes, & track project status. You might gather:

- Links to relevant docs or websites
- Ideas
- Notes or caveats
- Dated status updates

You can put these in physical journals, 1 app, or multiple apps.

For me, no 1 app can do everything so I use:

- @NotionHQ for status updates
- @draftsapp for jotting down ideas/notes to categorize later (not having to think about where to put an idea as you're having it is ■)
- @obsdmd for archives/notes/research ("knowledge management")

You can go really deep into knowledge management; just google "Zettelkasten" and you'll see what I mean. If you want to check out Obsidian for yourself, here's a great intro tutorial: https://t.co/sl7AVEAYvF

■ Review regularly

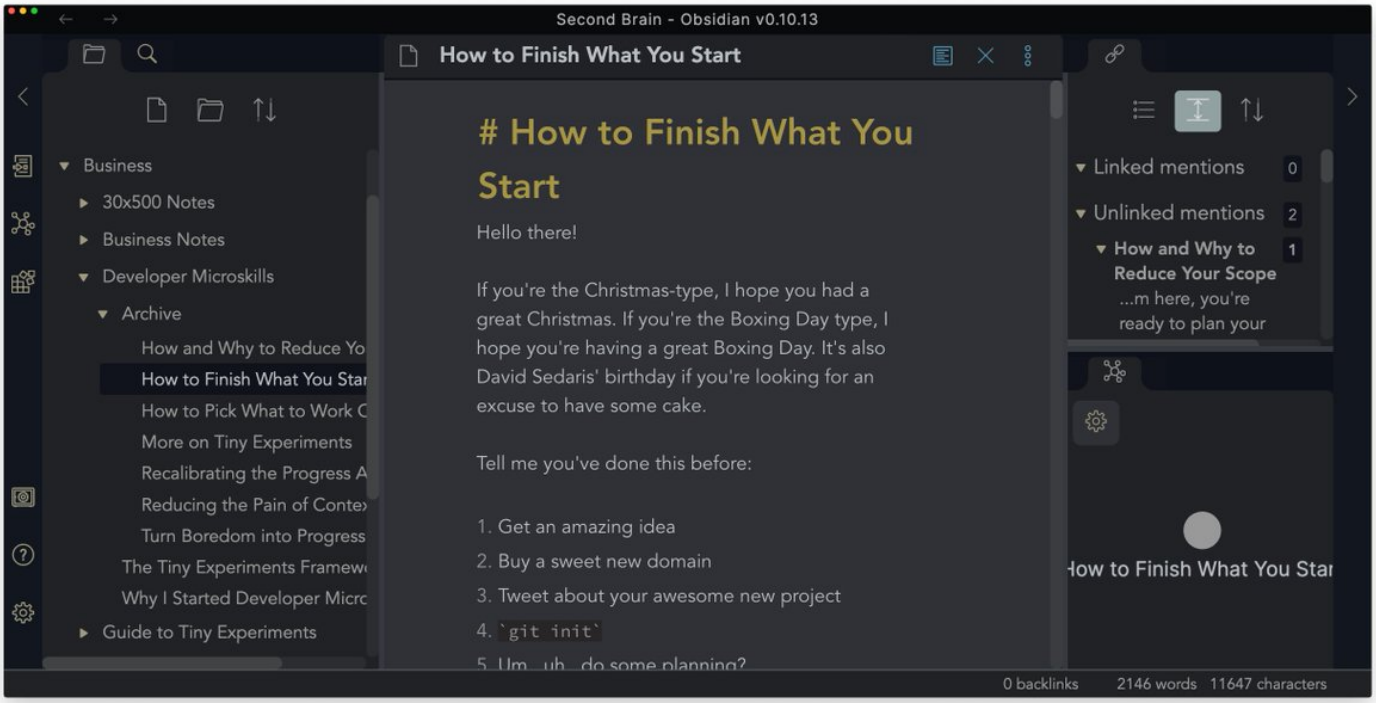
The most overlooked & underrated thing that helps with context-switching is reviewing projects & notes regularly. It:

- Helps us notice patterns
- Clears the junk from our heads
- Helps us understand our true capacity

More: https://t.co/9vE7lgVQyk

I review my projects in Notion about once a week & add comments about the last thing I was working on & my next action.

I do some "digital gardening" in Drafts & Obsidian each day; I'll review my Drafts inbox & move to Obsidian or clean up my notes on whatever I'm working on.



■ Automate where you can

Automation can be a rabbit hole, but the place to prioritize is reducing your likelihood of getting distracted. Like when you head to Chrome to Google something & then come to your senses an hour later while deep in the @dog_rates timeline ■.

Take advantage of all the ways technology can help you, whether shell scripts or tools like @zapier & @integromat. I rely on @keyboardmaestro & @TextExpander to insert snippets, launch collections of apps, etc. Here's how I use KM w/screencasting: https://t.co/ZUfhQx9Brq

I also learned a trick from @MacSparky on how to use KM to create "context palettes" of shortcuts. I hit "ctrl-cmd-opt-B" to bring up a bunch of business shortcuts to launch sites, open repos in @code, or open a Notion project. Here's how to do it: https://t.co/nGbvHMVIL8

Start slow with automation; notice where you're getting distracted & gradually build automations that save you steps.

It's easy to over-engineer something complicated & then ditch the whole thing out of frustration.

To really nerd out on this, check out @automatorsfm ■

I've found this combo of centralizing capture, regular review, & automation to be extremely powerful even when I only get it right 70% of the time. The slow burn ■ significantly reduces the amount of time I need to get creative work done.

If you found this helpful, you'd probably also like my Developer Microskills newsletter! Each week, I send out a practical, actionable way to improve as a dev & dev advocate. Last week's issue was an expanded version of this thread!

Join 1090+ other devs: https://t.co/V2TzZmX1F5