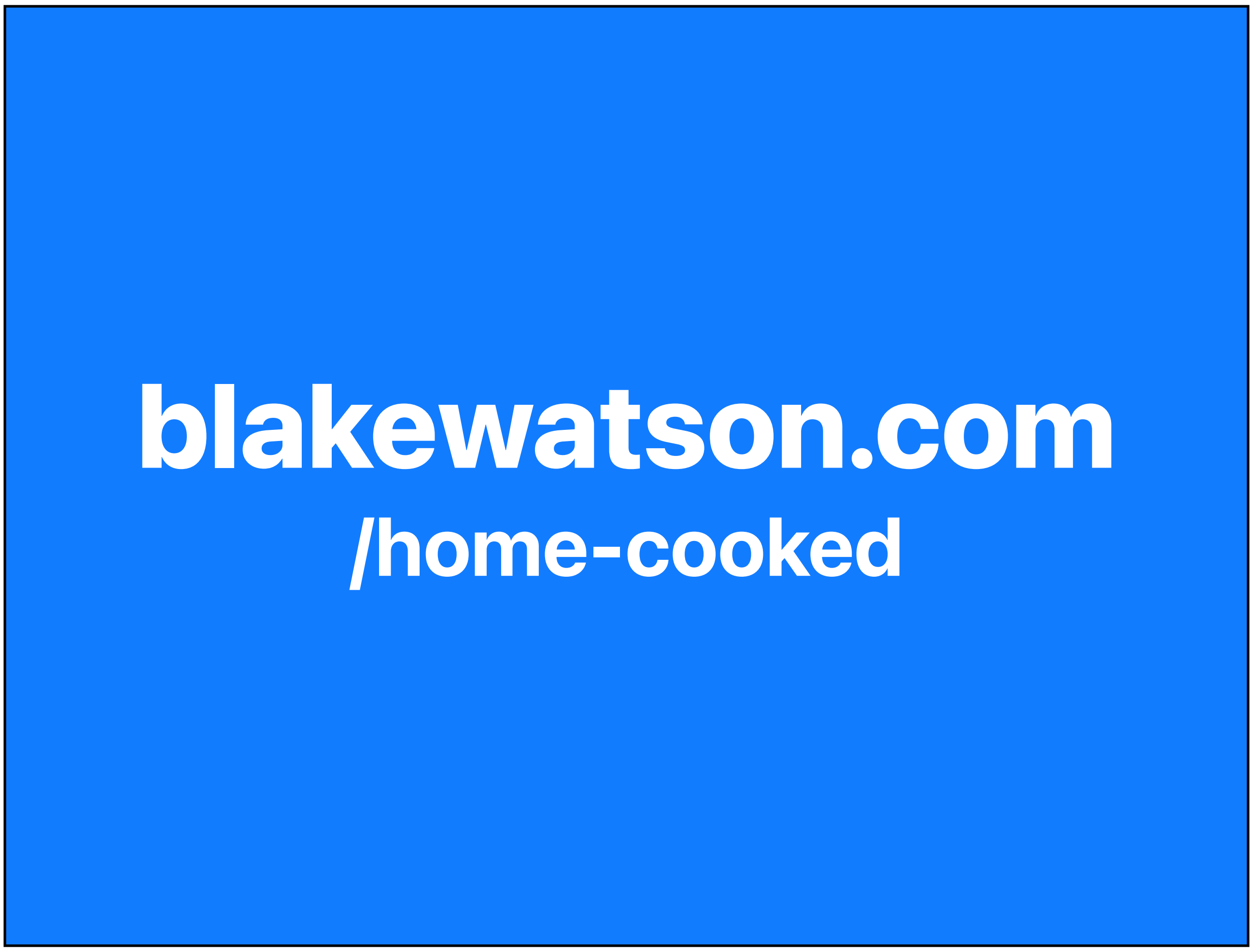
The joys of home-cooked apps

Blake Watson

Hey everyone. I’m Blake Watson. I’m a frontend developer working remotely with MRI Technologies out of Houston, TX. We’re a contractor for NASA and commercial space companies. I primarily work on a suite of internal applications called COSMIC, which NASA uses for tracking and managing hardware related to the spacesuit.

I’m also a Mad Genius alum so I always enjoy coming back to visit. Mad Genius definitely has the coolest offices of anywhere. And also the best-looking stuff I ever created was when I worked here. So thanks to Mad Genius for hosting.

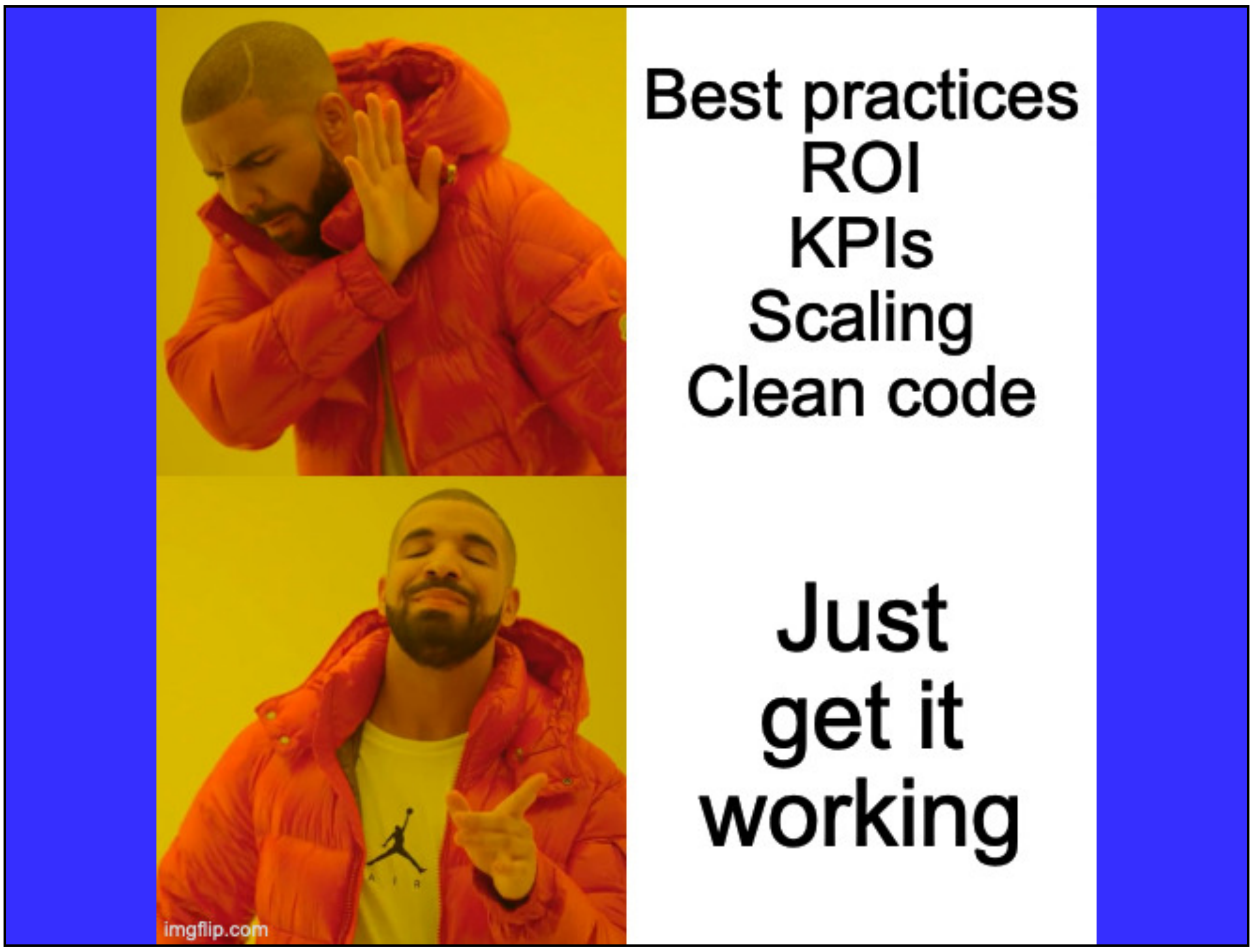


blakewatson.com
/home-cooked

So I’m going to be talking about *the joys of home-cooked apps*. If you want to follow along or reference it later there is a PDF and web version of this talk available here and it will have all of the links to the stuff I mention during the talk.

Home-cooked apps

So what am I talking about?
What is a home-cooked app?
It's the kind of app that you make for yourself, to solve your own problem or for your own entertainment. Much like a home-cooked meal, they can be shared with friends or family. But they aren't designed for mass consumption.



They aren't optimized for shareholders. There are no sales funnels. No user stories. And they don't scale. They don't have to. They aren't designed for thousands or millions of users. These are the kinds of apps that you make for a handful of people. Or maybe just for you.

Robin Sloan

author & home cook

Now I didn’t invent this term. A couple of years ago I read an **article by Robin Sloan**. He’s a New York Times best-selling author and calls himself the "programming equivalent of a home cook." In this article he described making this sort of short form video messaging service just for use in his household.

It was super simple. You basically hit record, send the video, and the receiver views it, after which it disappears.

That’s it. He built it in a moment of frustration when an existing app his family had been using was going away. The important bit here is that he didn’t make this app to be a product for the general public. He didn’t even design it to be a shared open source project. It's literally an app built just for their own use. And that’s what makes it home-cooked.

I’ve been making these kinds of apps for myself for years. It’s one of the things that drew me into programming. But I didn’t have a term for them or even a context for thinking about them until I read Sloan’s article. I was thrilled when read it because it spoke to ideas I held but hadn’t thought about actively—the idea that you can make useful things and they don’t need to be packaged for mass adoption to be successful.

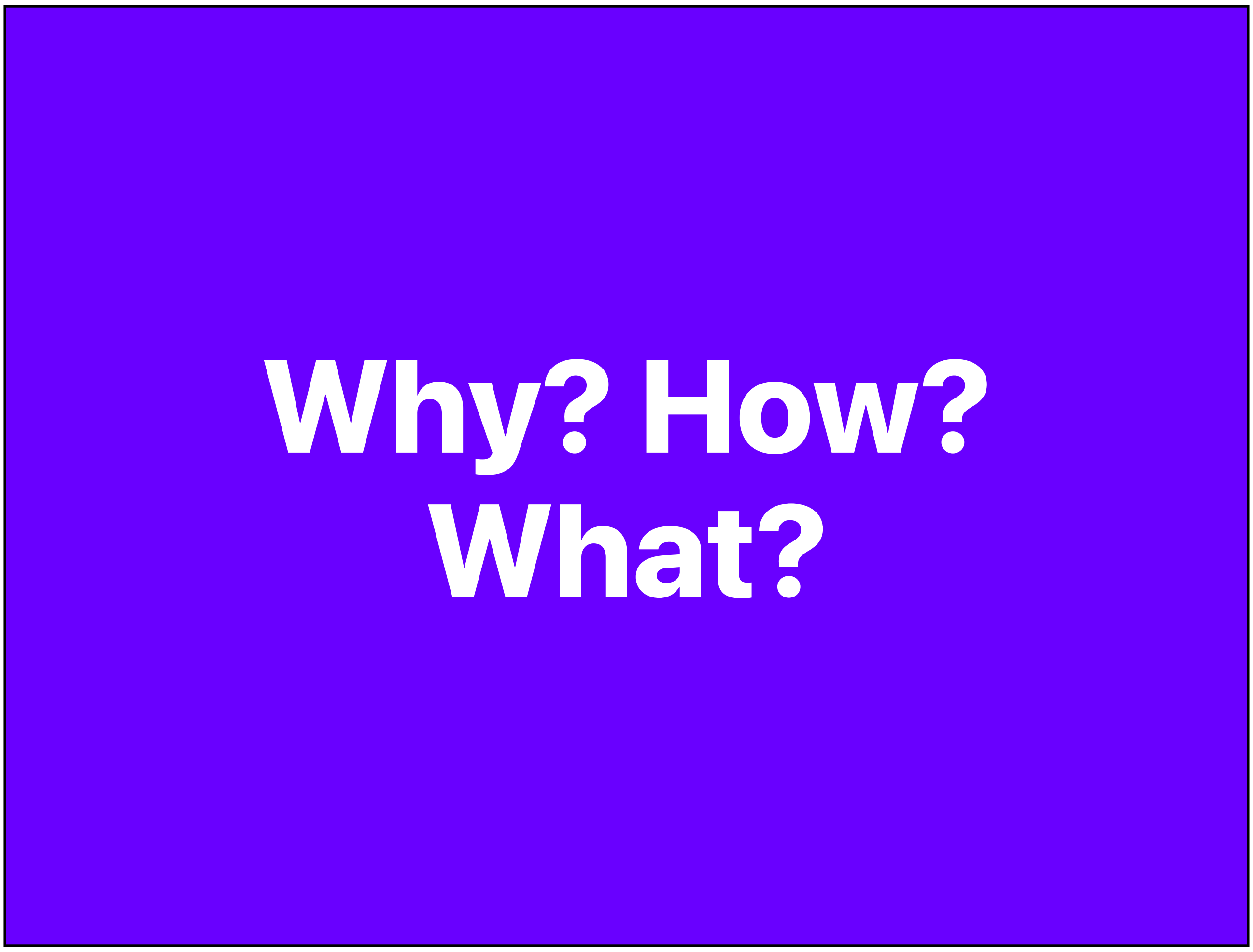
I love this. I think it’s a powerful idea. And maybe developers do this all this all the time. But this is the first time I heard it expressed this way and it smacked me right in the face—like of course we can make stuff just for ourselves. This is a thing we can do.

Now, that’s not to say you should *never* generalize a solution to make it more useful for more people. I’m just saying you don’t *have* to. Not every idea needs to be a product or a package.

Making a home-cooked app is about redefining what success looks like.

I think back to my late grandfather who would often take to his shop to make custom things for himself and his family—a cabinet for his wife, toys for the grandkids, and even a custom-made wheelchair lock down system designed specifically for the van and wheelchairs me and my brother had at the time. As a kid I was mesmerized with his craftsmanship. From watching him, the idea of building things for oneself was ingrained in me from a young age. I would go on to learn that having a disability often means coming up with creative solutions to your problems.

Okay, enough backstory. Let's get into it.



Why? How? What?

My job here tonight is to:

1. Convince you that you should make your own home-cooked apps
2. Show you examples of home-cooked apps
3. Give you a little recipe to help you get started

The goal is to get your wheels turning on the sorts of things *you* could make to improve your own life and those close to you.

but why tho?

There are several good reasons.

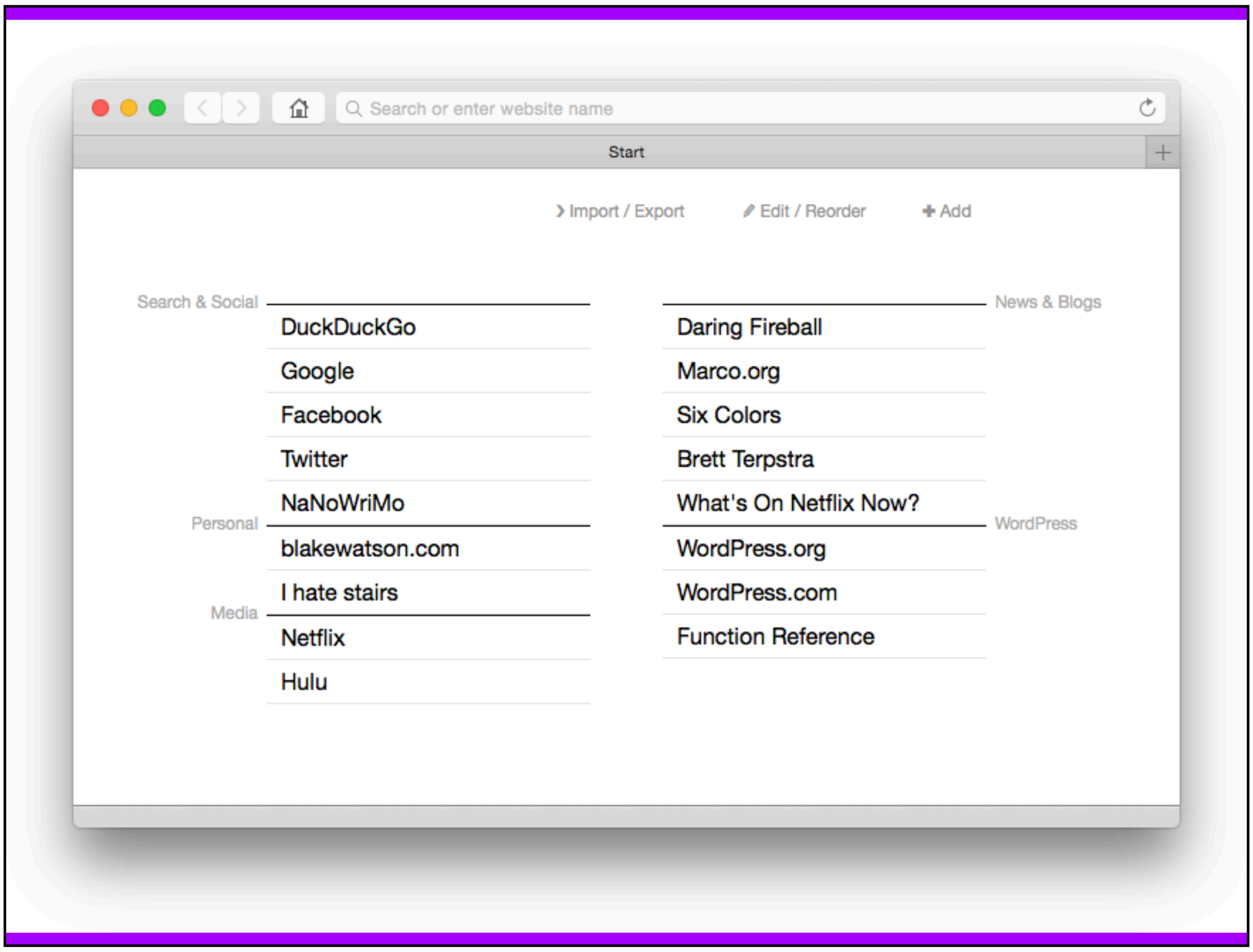
- You can make exactly what you want. Have you ever used an app and thought to yourself, "Man, this app would be dope if it just did this *one thing*"? *Guess I'll have to make a feature request and wait around for months until the developer says they're not going to do it.* No, guess what? When you make it yourself, you can make it do that thing.
- If you build with the idea that you are only serving yourself or a few people, you can skip over many of the challenging problems of software development: compatibility, extensibility, standards, best practices, scaling, etc. In land of home-cooked apps, the only important benchmark is, **does it work**.
- Another reason is privacy. You have some control over your data, depending on how you build your app. You have an opportunity to keep it out of the hands of big companies who will harvest it and use it for its own purposes.
- Lastly, it's fun. Building what you want and then using it is a rewarding experience. It's magic to me. It's what pulled me into web dev and why I still get excited about it.

And with that let's take a look at some examples. By their very nature, they are going to be weirdly specific. But that's the whole point.

Start

I cheated. This became an actual product.

Okay I’m going to start (heh) the tour of examples by breaking one of my own rules. I made this new tab page for myself but ended up packaging it up for other people to use. It ultimately became my main side project, A Fine Start.



Search & Social
DuckDuckGo
Google
Facebook
Twitter
NaNoWriMo

Personal
blakewatson.com
I hate stairs

Media
Netflix
Hulu

News & Blogs
Daring Fireball
Marco.org
Six Colors
Brett Terpstra
What's On Netflix Now?

WordPress
WordPress.org
WordPress.com
Function Reference

The idea here was to have a single HTML file as a new tab page that would allow you to add and organize text links directly on the page. This might seem trivial but at the time I created it, built-in new tab pages were using gratuitous visual effects, ugly screenshots of webpages, and very poor user experience in my opinion. I just wanted a new tab page with plain text links that I could add, edit, and arrange.

I made it for me but ended up packaging it up into a product and found out others liked it too. That’s cool, but it’s not required in the world of home-cooked apps.

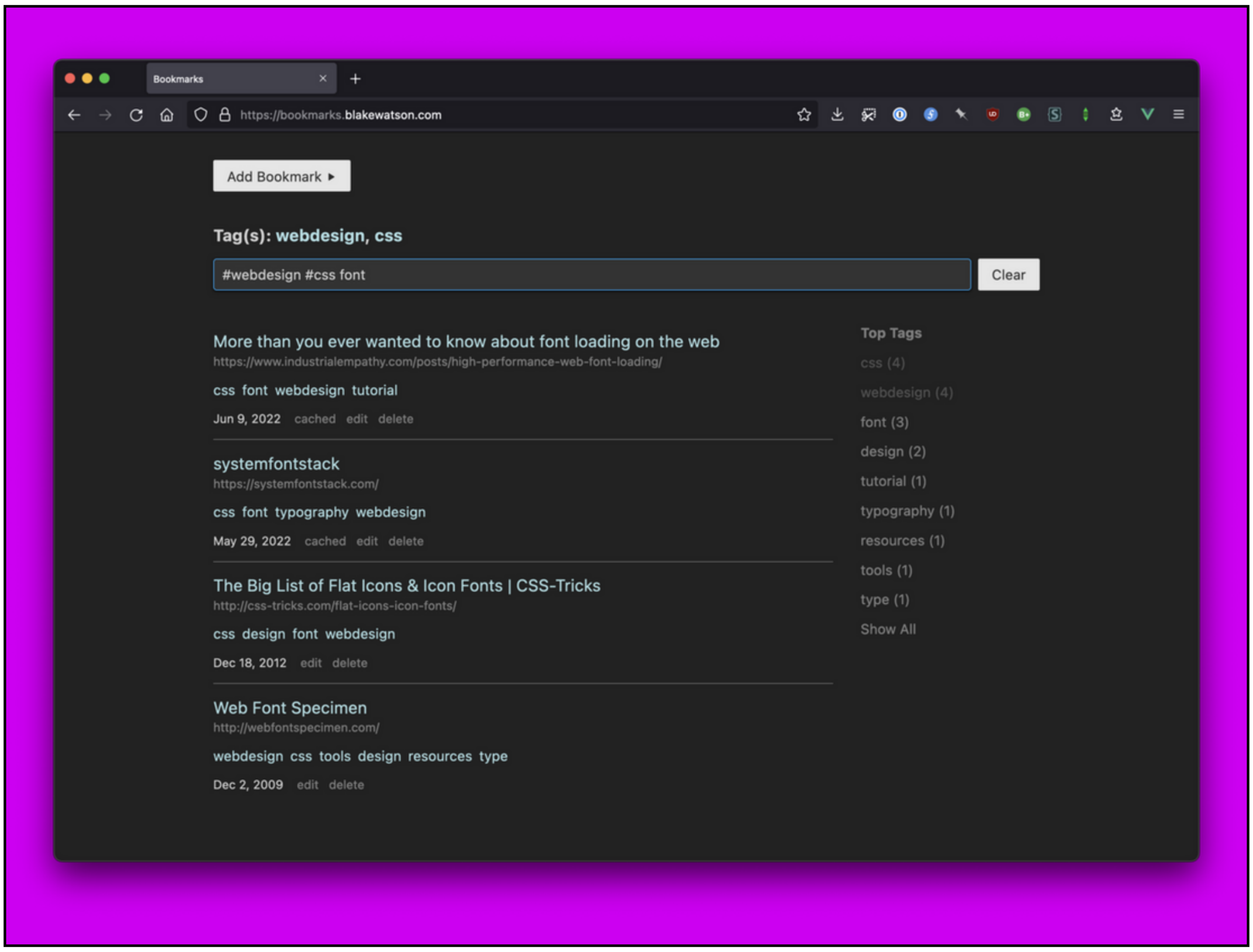
My DIY Pinboard replacement

If you’ve never heard of Pinboard, it’s a web app for keeping bookmarks. So it keeps the URL, title, description, and you can add tags for organization. I’ve been a longtime user and this app met my needs adequately for a long time.

But it was also becoming the place where my bookmarks went to die. I felt like retrieving things from it wasn’t very easy. I also paid for an optional archival account. This means that every link I save gets cached by Pinboard so that if that link goes bad (which so many of them do) I would still have a copy.

But I noticed my archival account was often failing to capture pages. And the whole thing was becoming unenjoyable.

What to do?



Tag(s): webdesign, css

#webdesign #css font

More than you ever wanted to know about font loading on the web
https://www.industrialempathy.com/posts/high-performance-web-font-loading/
css font webdesign tutorial
Jun 9, 2022 cached edit delete

systemfontstack
https://systemfontstack.com/
css font typography webdesign
May 29, 2022 cached edit delete

The Big List of Flat Icons & Icon Fonts | CSS-Tricks
http://css-tricks.com/flat-icons-icon-fonts/
css design font webdesign
Dec 18, 2012 edit delete

Web Font Specimen
http://webfontspecimen.com/
webdesign css tools design resources type
Dec 2, 2009 edit delete

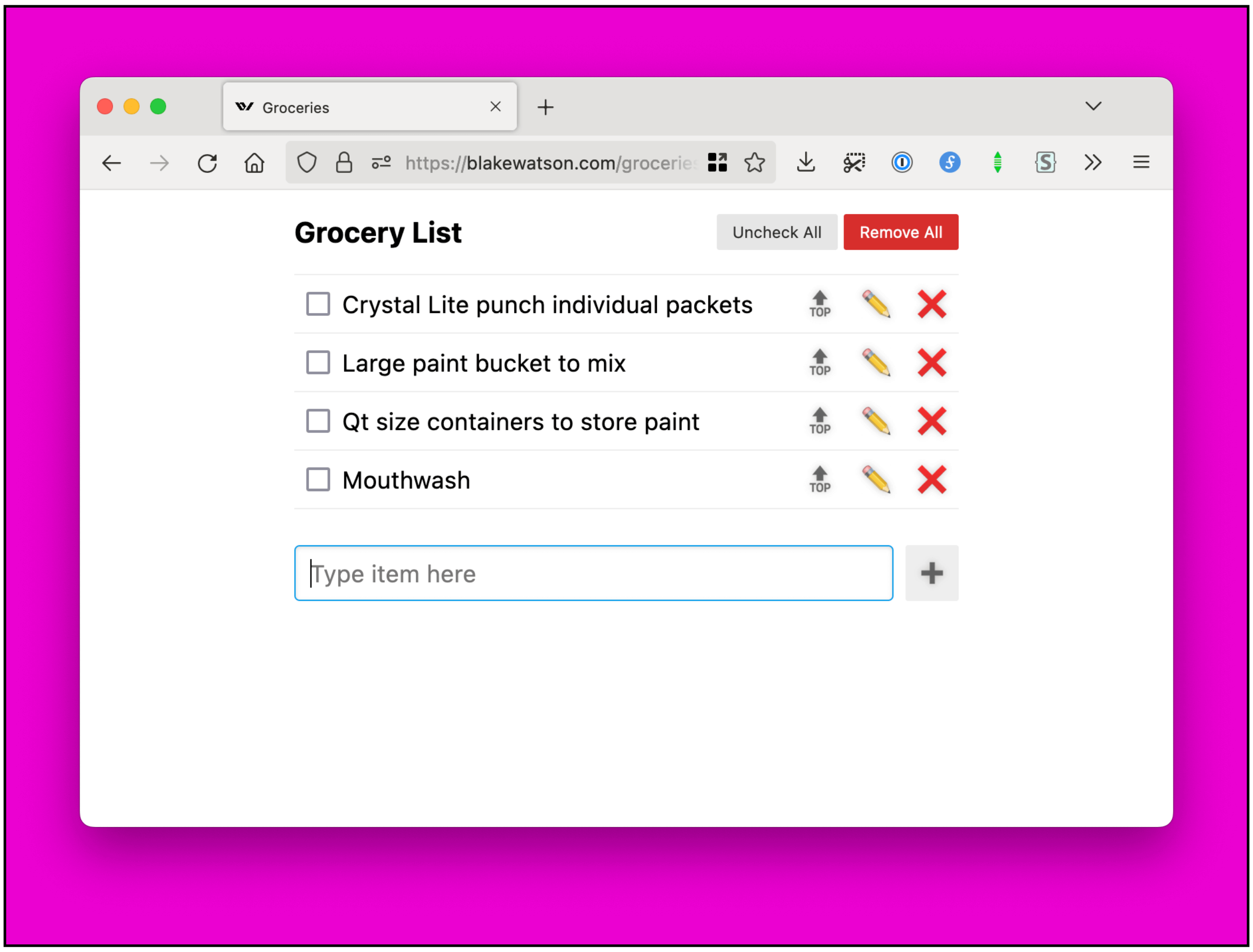
Top Tags
css (4)
webdesign (4)
font (3)
design (2)
tutorial (1)
typography (1)
resources (1)
tools (1)
type (1)
Show All

You will be shocked to hear that I created my own solution. It's a NodeJS app that saves URLs along with metadata like descriptions and tags. It automatically sends the URL to Wayback Machine and it saves the cached URL that Wayback sends back, so that I always have access to a cached copy.

I did break my own rule and open source this app with instructions for setting it up. But it comes with the *huge* caveat that I designed this for me and didn't take other users into account.

Groceries

Here’s a fun one that was made for family. We needed a way to have a shared grocery list between me, my brother, and our mother. There are numerous existing apps for list sharing but they all have cruft that a non-tech-savvy user (ie, my mom) would find confusing.


Groceries

https://blakewatson.com/grocerie

Grocery List

Uncheck All Remove All

- [ ] Crystal Lite punch individual packets
- [ ] Large paint bucket to mix
- [ ] Qt size containers to store paint
- [ ] Mouthwash

Type item here

So my brother and I teamed up to build this web app. It's optimized to be a home screen app. So Mom is able to use it on her phone as an app and Matt and I can use it on a desktop web browser (since mobile devices are a bit difficult for us to handle).

The design is as dead simple as possible. It uses large emoji buttons for basic actions. You can add things, check them off, edit them, or delete them. We decided that drag-and-drop sort was too much UI complication, so we added this move-to-top button, which will shoot an item to the top of the list.

This inadvertently triggered a war between me and Matt, trying to make sure the stuff we want is at the top of the list. Like one day I checked the list and I noticed Matt's bottle of Jack Daniels was at the top of the list.

And that's the best part of the app—the list is synced so that any of us can add things to the list. There isn't even a concept of users. We all use the same token which we put in once and it basically saves it forever. It's super secure obviously.

You can't make multiple lists. That would introduce more UI. Instead, the same list is reused for every grocery trip. There's a button that unchecks everything and one that deletes everything.

That's it. That's the app. It's silly, really, but works great for us.

The time I accidentally created a domain-specific language

The previous apps we looked at, while *technically* being home cooked just for me, were still fairly relatable examples that a lot of people might find useful. A new tab page; bookmarks organizer; a grocery list.

But now we're getting into the real obscure stuff. My brother and I have a team of caregivers that work various day and night shifts. At the end of each pay period they need to turn in detailed timesheets. Sometimes these timesheets are difficult and error-prone.

I wanted a way to keep my own records and also help my caregivers complete the timesheets accurately. I toyed around briefly with a spreadsheet. I know they are powerful, but man I hate them. After being annoyed with that, I decided that what I really needed was something that worked like Markdown. I wanted a plaintext entry system. It works like this:

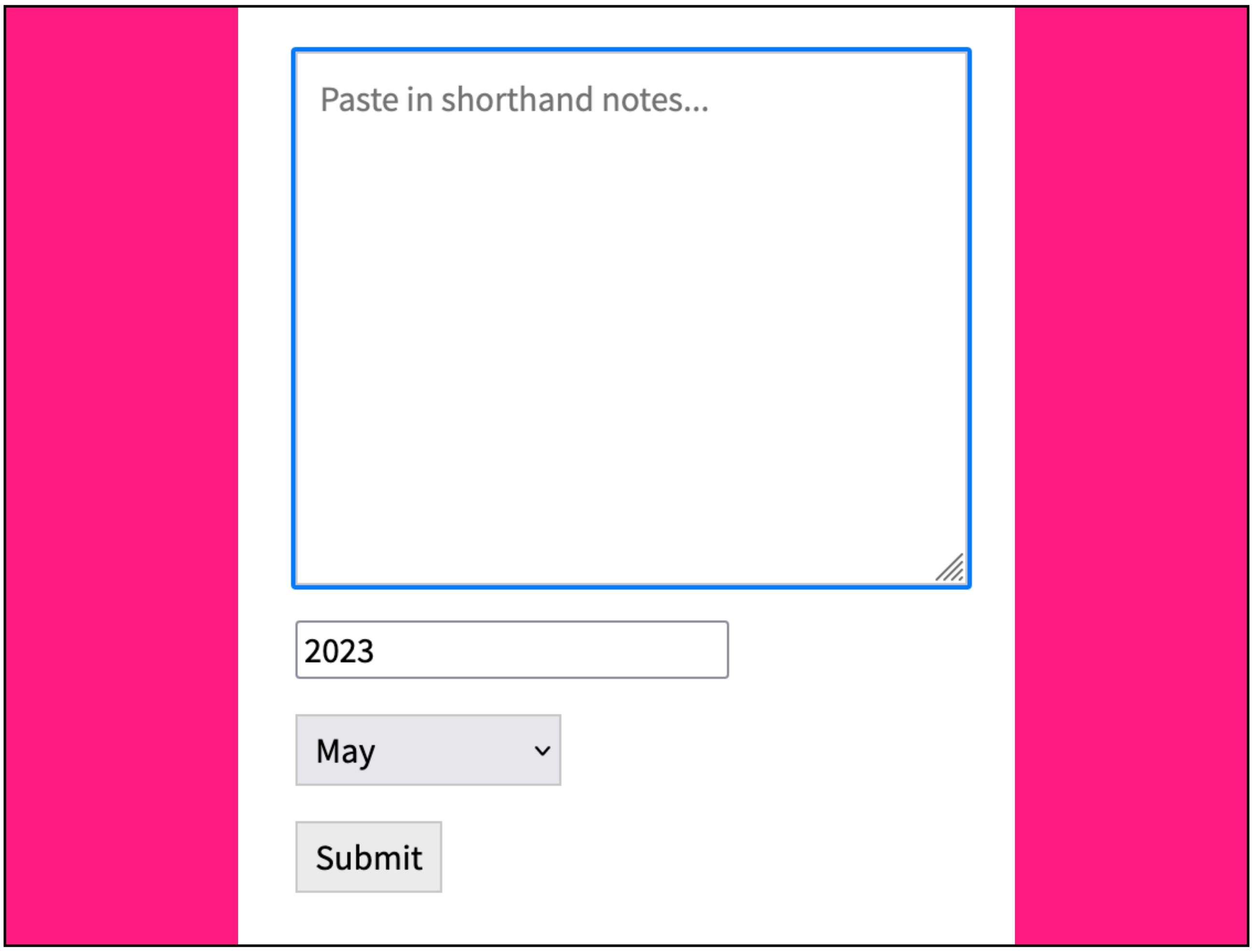
```
# This is a comment. It is ignored.

1 Day
Johnny Appleseed

1-2 Night
Jane Doe

# Johnny left early
2 Day (7:00am-3:00pm)
Johnny Appleseed

2-3 Night *
Jane Doe
```

For each shift that is worked, I make an entry. Entries are separated by a line break. Each entry is two lines. The top line is the date followed by the type of shift that was worked (Day/Night). The next line is the name of the person who worked.

Most shifts follow a standard time range so I don’t specify the start and end time. But if, for some reason, someone works a different time schedule than normal, I can specify that in parentheses.

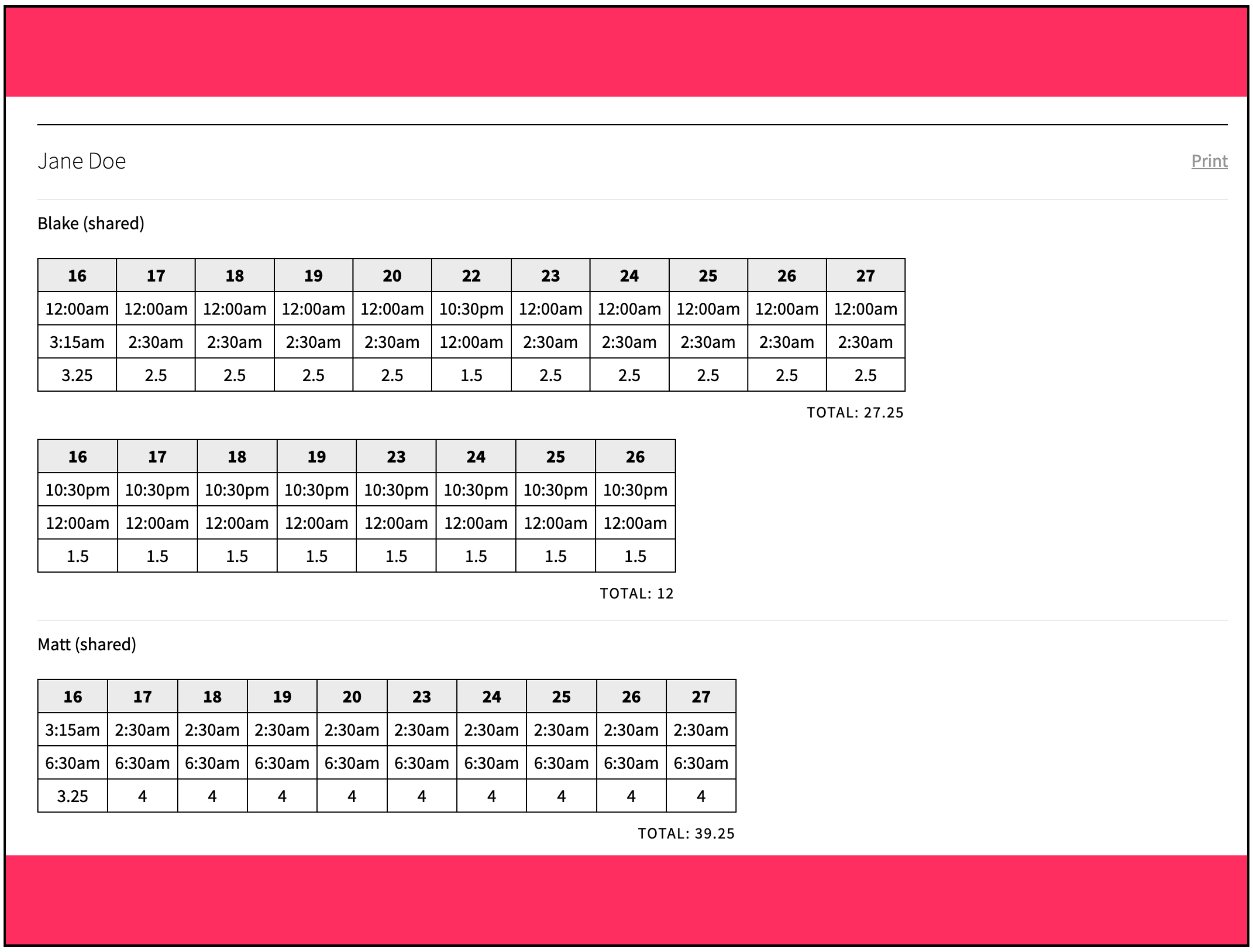
That last piece of metadata is represented by an asterisk and it means my bother and I were together for the whole shift and the caregiver worked with both of us. That means they'll need to do a timesheet for each of us and split the hours.

I don’t specify the month and the year because it’ll be in the filename. That info will be used at compile time, which we'll see in a second.



I wrote a little frontend tool that can process this syntax and spit out a table of dates and times that I can print out that caregivers can reference when they are working on their timesheets.

This is where I paste in the plaintext data and provide the month and year.



Jane Doe

Print

Blake (shared)

16	17	18	19	20	22	23	24	25	26	27
12:00am	12:00am	12:00am	12:00am	12:00am	10:30pm	12:00am	12:00am	12:00am	12:00am	12:00am
3:15am	2:30am	2:30am	2:30am	2:30am	12:00am	2:30am	2:30am	2:30am	2:30am	2:30am
3.25	2.5	2.5	2.5	2.5	1.5	2.5	2.5	2.5	2.5	2.5

TOTAL: 27.25

16	17	18	19	23	24	25	26
10:30pm	10:30pm	10:30pm	10:30pm	10:30pm	10:30pm	10:30pm	10:30pm
12:00am	12:00am	12:00am	12:00am	12:00am	12:00am	12:00am	12:00am
1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5

TOTAL: 12

Matt (shared)

16	17	18	19	20	23	24	25	26	27
3:15am	2:30am	2:30am	2:30am	2:30am	2:30am	2:30am	2:30am	2:30am	2:30am
6:30am	6:30am	6:30am	6:30am	6:30am	6:30am	6:30am	6:30am	6:30am	6:30am
3.25	4	4	4	4	4	4	4	4	4

TOTAL: 39.25

This is an example printout. If it's hard to tell what's going on here, yeah I agree—and that's why I made this tool to figure it out. The timesheets we have to use were terribly designed.

The agency requires that the timesheets be completed by hand so what my tool does is output something that my caregivers can use as a reference while they are filling out the timesheets.

This example represents someone who worked the overnight shift every weekday. To record all the information it took three timesheets. Exactly half of the hours are assigned to me and the other half assigned to my brother.

Overnight shifts are particularly awkward to deal with because they span across two different dates, which means it takes multiple columns to record them.

A home-cooked language

I didn't know it at the time, but this kind of thing has a name—domain-specific language. What I created was essentially a human-readable computer language designed for tracking caregiver hours. My frontend tool is the compiler, which produces the final output—HTML tables I can print. I store all of my records in my note-taking app of choice as *source code*.

This year marks one decade since I first made this app. I'm sure I would do it much differently now, but even with its idiosyncrasies, it gets the job done.

Recipe

These apps were made for me so I’m not expecting you to think, “Oh I could use that.” But I *am* expecting you to think about how you could address some of your own specific wants and needs.

How to home-cook an app

The What

This is just a little recipe to get you started. There is really no right or wrong way to go about it. If it works for you, then do it.

Think of a problem of yours that you could solve. For example, some tedious work that you wish you had an app for but don’t. Or something you and your friends would like to use that isn’t readily available. It could be something a popular app does but in a way that is much simpler and straightforward.

How to home-cook an app

The What, *exactly*

Decide what, exactly, you want to build. Give yourself some rough requirements. Make a list. If it has UI, do little napkin sketch. I like to go into my notes app of choice and type out a list of the features I need and jot down a few notes about the behavior of the app.

It’s important at this stage to explore ideas but don’t be tempted to add a million features. You don’t want to get overwhelmed and kill the project on the vine.

You may have heard the term, MVP, or *minimum viable product*. It’s the idea that you define the least amount of features that accomplish the main goal of the app. That’s close to what we're doing here, but even easier! Because remember, we're not making a *product* per se. It’s something simpler and rougher. It’s home–cooked. Let’s call it a *minimum viable meal*. That’s your job as a home cook to figure out at this stage.

The main thing is to solve *your* problem, not some abstract user’s problem.

How to home-cook an app

The How

Determine the easiest way to get to your goal. No need to over-complicate things—unless you enjoy that sort of thing! We are developers after all.

My advice here is to use what you know. But really you can use whatever technology you want. I don’t care if it’s new and shiny or old and boring. Use jQuery if you want. Or no JavaScript at all. This thing you’re building just needs to work for you so feel free to adopt whatever technical debt you want.

What’s important is that all decisions are in service of getting the thing to work.

How to home-cook an app

Build it!

Even if it’s a larger project, just make small progress toward it. I wrote an article about finishing side projects. Check out the written version of this talk for a link to that.

But TL;DR, small progress adds up. Use your commit history for motivation—it’s an automatic list of everything you have accomplished. If you’ve kept things simple, then hopefully it won’t take too long to knock out the rough draft.

Once you start using it you can take note of any enhancements or bug fixes you need to make.

Go cook up your own apps

Most of the home-cooked apps I’ve made took me a weekend or two. When you strip an app down to its essentials and the only requirements are yours, you have the freedom to hack things together however you see fit.

I think the time investment on these projects was totally worth it. The last example of the timesheet system, in particular, has saved me probably *days* of time pouring over timesheets.

Hopefully this has you thinking of how to tackle a problem of your own or make a fun thing just for friends or family. Home-cooked apps are fun, useful, and a breath of fresh air.

So, go forth cook up some of your own.