Ignore the code

- Toggles Considered Harmful (2026/08/09 09:21)
  I just read this post from Marcin Wichary's Unsung, where he points out a peculiar macOS dialog box that shows boolean toggles in four different shades of grey: Marcin offers some improvements, but the actual problem is fundamental to macOS: these types of toggles are bad UI controls and should simply not exist. I think toggles originated on iOS, where they replaced checkboxes. I suspect the reason is that they feel more tactile on a touchscreen, have a larger touch target, and resemble physical switches - Apple's original UI design approach for iOS valued this kind of skeuomorphism. Toggles make sense in the real, physical world because of their mechanical properties. But even there, they need additional affordances to work. For example, this power strip has a toggle switch to turn it on and off: Note that there are labels for each state, and a light to indicate the current state. That's because the toggle itself is symmetrical. There is no obvious way to tell whether it is on or off. Now let's go back to the digital toggle: Is this toggle on or off? If you don't already know that "left" means "off" and "right" means "on", you have absolutely no way to tell. Apple's attempts to make toggle states more obvious ultimately created the confusing situation in the original screenshot. Therefore, the most expedient solution to the original UI problem is simply to use a control whose state is obvious. Update: Discussion on lobste.rs, including this point I wasn't aware of: There is an accessibility setting on macOS named "differentiate without color", which does help by showing little on/off indicators on the toggles. If your System Settings search field also doesn't work, it's under Display. Enabled. If you require a short url to link to this article, please use https://ignco.de/818But wait, there's more!Want to read more like this? Buy my book's second edition! Designed for Use: Create Usable Interfaces for Applications and the Web is now available DRM-free directly from The Pragmatic Programmers. Or you can get it on Amazon, where it's also available in Chinese and Japanese.
- A Bunch of Random Updates (2026/06/23 16:31)
  I've updated some things in the last few years. Now's as good a time as any to link to them. bookfeed.io bookfeed.io is an RSS feed for book releases. I'm not sure why, but Google Books' search results have become increasingly unreliable over the last year. I've added openlibrary as a second source, and I combine the results from both. I've also added an .ics renderer, so you can now show book releases for authors you care about in your calendar. flicross.com Nonogram puzzles based on Flickr photos. Added a daily puzzle and fixed support for Firefox mobile. qrat.love QR code generator. Added more QR code types. Added STL and OBJ exporters so you can now 3D-print your QR codes. Other Stuff I've always written a lot of Mac programs for my personal use, but I rarely polished them for a public release.[^mac] With LLMs, it's become easier to invest some additional time to get things to a state where they might be useful for others, so here are a few apps I made recently: [^mac]: Mainly because I find writing code for Macs less and less enjoyable. Zap.app is an alternative app switcher that allows you to exclude apps you never switch to. I've missed DragThing ever since it stopped working, so I made Top Drawer to satisfy that need. Lantenna is a LAN scanner meant to be used for one's own local network and supports renaming, hiding, and favoriting entries. I understand that Alan was meant as a joke, but I find it genuinely helpful, so here is the version I'm running. After Mozilla killed Pocket, I've grown increasingly uncomfortable with storing my stuff on other people's servers. Karakeep is a great bookmarking tool, but the Android app isn't made for reading. I've made my own Firefox extension and Android app focused on reading. There's also a Firefox extension that shows Karakeep search results on Kagi's search results

page. RSS Spy is a tiny Firefox extension that tells you if the site you're looking at has an RSS feed, and lets you copy it. Gans, a more convenient client for Ente Auth. Lucerne, an experimental Word processor. I've built these mainly for my own convenience, but perhaps somebody else finds them helpful as well. If you require a short url to link to this article, please use https://ignco.de/814But wait, there's more!Want to read more like this? Buy my book's second edition! Designed for Use: Create Usable Interfaces for Applications and the Web is now available DRM-free directly from The Pragmatic Programmers. Or you can get it on Amazon, where it's also available in Chinese and Japanese.

- Please, Support Books (2025/11/22 13:21)
  The Pragmatic Bookshelf has all of their books at half price[^except] until December 1st using coupon code "save50". They're DRM-free and compatible with any ebook reader. [^except]: All of their books except for The Pragmatic Programmer. Here are some of their books I've read recently that I enjoyed a lot: Hands-on Rust and Advanced Hands-on Rust. Rust is a difficult language to wrap your head around, particularly if you've been programming for a long time and have some things to unlearn before Rust starts making sense. It needs an up-front investment until it all snaps into place. These two books are a huge help for getting to that "Oh, I get it now!" point. Cruising Along with Java. As a Java programmer, it's easy to stick with what you know. This book provides a succinct explanation of newer Java features and how and when to use them. Kotlin Brain Teasers. I very much enjoy all the Brain Teasers books, but since I've recently started using Kotlin, I'm mentioning this one in particular. Server-Driven Web Apps with htmx is also a great book to read. HTMX is an interesting idea and a good alternative to much larger frameworks for many projects. Write Better with Vale. Admittedly, I haven't read this book yet, but it's on my "to read" list. The topic is super interesting, and since it's from PragProg, I know it's going to be good. Some other books that might be of interest include The Healthy Programmer, a whole-ass book just on tmux, the eye-opening Seven Obscure Languages in Seven Weeks, or Dave Thomas' new book Simplicity.[^mine] [^mine]: There's also my book. LLMs have had a significant impact on book sales, particularly technical books. These systems are killing the companies that created much of the content that made LLMs useful in the first place. Not just PragProg, but also other publishers. In related news, Manning has Black Friday deals right now, too. If you want companies like The Pragmatic Bookshelf to continue making high-quality, trustworthy content, please support them. If you require a short url to link to this article, please use https://ignco.de/811But wait, there's more!Want to read more like this? Buy my book's second edition! Designed for Use: Create Usable Interfaces for Applications and the Web is now available DRM-free directly from The Pragmatic Programmers. Or you can get it on Amazon, where it's also available in Chinese and Japanese.
- Stop Uploading Your Data to Google (2025/06/11 18:00)
  Yesterday, I saw this video on YouTube by DJ Slope, a videogame youtuber. The very short version is that YouTube is deactivating all of his services and deleting his channel. He suspects they are doing that because he uploaded a file to Google Drive that triggered an automatic account deletion. This isn't the first time something like this has happened. Here is a case from 2022 where CSAM detection deleted a man's account when he sent pictures of his child to his doctor. DJ Slope will likely be fine because he has a large enough YouTube channel, which should help him find someone who can connect him with a real person at Google.[^1] The same applies to the gentleman who got The New York Times to report on his problem. [^1]: Also, please note that I'm not blaming him for using Google's services. Only Google is to blame for how it treats its customers. But it probably won't apply to you when it happens to you. A few years ago, I realized that I had photos and emails dating back to the mid-90s on my Google account. I had auto-upload enabled on my phone's Google Photos account. What are the chances that one of these hundreds of thousands of pieces of data would trigger some automatic action at some point? What are the chances that I could get in touch with somebody who could fix this for me? At that point, all of my email went through Google, my calendar ran on Google, my phone required a Google

account to work, my files were on Google Drive, and I had logged into probably hundreds of third-party services using my Google account. And that's just off the top of my head. Losing access to my Google account would have been devastating. So I set some rules for myself: Do not upload any data to Google. My Google account is too important to risk it. Now, no services are tied to it, except for those that must be tied to it. Self-host as much as possible. If self-hosting is not possible, use end-to-end encrypted services whenever possible. Use one service for one thing, so that when it gets disabled, only that one thing is affected. Here are my suggestions. These are not exhaustive, but they are services I trust. If I don't include a service you love, it's because I'm not familiar with it and haven't used it, not because it's not good. Email Use Proton Mail or Tuta. Do not use the email address they provide. Use your own domain name so that you can keep your email address when you switch providers. Cloud Storage Use Filen or self-host. Self-hosting cloud storage is surprisingly easy. Install Syncthing on your devices, and you're essentially self-hosting a cloud storage solution. One downside of both options is that they don't use OS-level features, such as downloading files on demand. Proton Drive does, but I already use Proton for mail, so according to my "one service for one thing" rule, I'm not going to use Proton Drive. Password Manager Use 1Password or self-host Bitwarden using Vaultwarden. Photos Use Ente or self-host Immich. Search Use Kagi or self-host SearXNG. Calendars Self-host using Baikal. Those are the most important ones that allowed me to remove myself from Google's services completely. They're not the only services I self-host. For example, I use Paperless-NGX to manage my documents, Karakeep for bookmarks, and Pi-hole to block the 40% of my Internet traffic that is useless to me. However, they're the primary ones. About Self-Hosting Self-hosting this stuff seems daunting. It sounds like you need a server, handle admin work, and perform a lot of maintenance to keep everything running smoothly. In reality, if you read this blog, you can self-host something like Immich. All you need is an old PC and some way to back it up. You probably already have both of these. Here's what to do: Install Docker. It's free. Install Tailscale on all of your devices, create an account, and log in. Now you can access them from each other even if they're not on the same network. It's also free. Install the software you want to self-host on the computer you installed Docker on. Every app that can be self-hosted has a tutorial on how to do that. For example, here's the one form Immich. It's a three-step process, and you're done. This does mean that you will be responsible for backing up your data. But you're already responsible for that because Google can take away your access to the stuff on its cloud any time it wants. It's just easier now to keep backups, because all of the data is on your local storage to begin with. Anyway, here's a list of other cool stuff you can self-host. And now stop uploading your data to Google. If you require a short url to link to this article, please use https://ignco.de/809But wait, there's more!Want to read more like this? Buy my book's second edition! Designed for Use: Create Usable Interfaces for Applications and the Web is now available DRM-free directly from The Pragmatic Programmers. Or you can get it on Amazon, where it's also available in Chinese and Japanese.

- Daylight Saving Time (2024/04/14 08:25)
  I've never spent a lot of time thinking about daylight saving time. It's a thing that annoys me once a year and makes me happy once a year. I never gave it much more thought than that, other than to kind of vaguely think that it seemed pretty stupid, and having the impression that this was a commonly held belief. A few weeks ago, I read Craig's weeknotes, where he wrote: My social feeds have been full of people grumbling about daylight savings, and that’s set to kick off again next week when UK clocks change. But I love it. (The clocks changing – not the grumbling.) In an instant, we’ll go from sunset around 18:30 to 19:30, meaning I’ll be able to start playing football in the street with my daughter after dinner again." This actually got me wondering what the real impact of daylight saving time was. A quick search led me to SunriseSunset.io, a super cool website that provides a free API that returns sunset and sunrise times for any location on earth. So I wrote some code to visualize the effect of daylight saving time at my location. Here it is: From top to bottom, this shows the daylight for each day. You can clearly see when daylight saving

time becomes active, and when we switch back to standard time. The blue areas on either side are the times when I'm typically asleep. At least in my location for my sleep pattern, daylight saving time makes sense. I gain some light in the evening without losing any in the morning. However, standard time doesn't seem to make a lot of sense. It would be more sensible for daylight saving time to be used for the whole year. I now lose some morning light because I sleep through it. By keeping daylight saving time all year, I would get that lost light back in the evenings. The only cost would be very dark mornings around the winter solstice. If you want to see a graph like this for your location and your sleep pattern, go to ignorethecode.net/dst and see for yourself.[^note] You need to give the page access to your location for it to work. If you get an error, try again in a few minutes. And if you find this useful, buy the guy who made the API a coffee. [^note]: Note that if you're publishing generated pictures, you're probably doxing your own location, since the light pattern narrows down possible locations you could live at. If you require a short url to link to this article, please use https://ignco.de/805But wait, there's more!Want to read more like this? Buy my book's second edition! Designed for Use: Create Usable Interfaces for Applications and the Web is now available DRM-free directly from The Pragmatic Programmers. Or you can get it on Amazon, where it's also available in Chinese and Japanese.

- Steal These Surface Duo Ideas (2024/03/16 09:32)
  As has happened with my Fold 3, after about a year of using it, my Fold 4's screen started to delaminate,[^delaminategate] so I had to send it in for repairs.[^why] When looking for a cheap replacement phone to use while my Fold 4 is away, I noticed that the Surface Duo is now available for around 400 bucks, so I picked one up. [^delaminategate]: I guess technically, the built-in screen protector, the one that is supposed to not be removed from the screen, started to remove itself. It's kind of funny how that just happens, and people accept it. If Apple released a phone whose screen would just fall apart after a year, the media couldn't stop screaming about delaminategate - as they should. I'm just not sure how Android phone manufacturers get away with this shit. [^why]: Why do I keep buying foldable phones if they keep falling apart? Because they're so much better than regular phones that the advantages vastly outweigh having to send them in for repair once a year. Then, something interesting happened: I started to absolutely love it. I think this is due to three things. The Duo's Aspect Ratios The Surface Duo's aspect ratios make a ton of sense. When it's folded, it has a 86×116 mm screen, which is just a great aspect ratio. It's wider than most screens, but less tall, which makes the screen more reachable when held in one hand, and means content like websites work much better. When it's unfolded, it essentially has an 176×116 mm screen, which is great for reading books, or watching YouTube videos (although there is a black bar in the middle, so it doesn't work as well for watching actual movies, where you care about the aesthetics). The Fold 4, on the other hand, has a 58×148 mm screen when folded, which is a weird, skinny, tall aspect ratio. Worse, when it's unfolded, at 150×148 mm, it's essentially square, which means that that most things you'd want to use a larger screen for don't fit well. To me, at least, it's painfully obvious that the Duo's aspect ratios make much more sense. Default Unfolded App Behavior The way apps behave on the Duo seems odd at first, but turn out to be exactly what I want. On the Fold 4, the unfolded phone acts as a tablet, and opened apps immediately take up the whole screen. Not so on the Duo: apps take up only half of the screen by default. At first, I found this annoying, but I soon realized that it encouraged me to open apps side-by-side more often, rather than switching between them. Have to log into some account that 1Password can't fill in automatically? Instead of switching between the two apps, just open 1Password next to the app I'm logging into. I'm responding to an email? Open a browser next to my email client so I can see both at the same time. Need to find a time slot for an appointment, and company policy doesn't allow me to open my calendar in my regular calendar app? Open my personal calendar app and my company calendar app side-by-side. All of this would have technically been possible on the Fold, I just never did it, because by default, apps take over the whole screen. It's much more convenient to have them take over

half the screen by default, and then enlarge them when necessary, rather than the other way around. Single-Screen Mode Unlike most other foldable phones, you can fold the Duo's screens so the screens are outside of the phone, on the front and the back. So instead of having a small outside screen, and a large foldable inside screen, like the Fold 4, the Duo only has the inside screen, but it can either be folded closed, so the screen is protected, or it can be folded fully open, so both screens are on the outside. Why is this great? Because at that point, it's just a regular phone with a great screen that has a perfect aspect ratio. And because there are only two screens, it's a phone that is much thinner, but has a much larger screen, than a closed Fold 4. In general, the way the Duo is designed, it's much more versatile than most "real" folding phones. It's quite unfortunate that the Duo (and its much better sequel, the Duo 2) never caught on. This is likely mainly due to the first Duo's utterly abysmal software, which never really got fixed, even after multiple software updates. At this point, it seems that the Duo line of smartphones is dead, but I do hope that some of these ideas make it to other phones. If you require a short url to link to this article, please use https://ignco.de/798But wait, there's more!Want to read more like this? Buy my book's second edition! Designed for Use: Create Usable Interfaces for Applications and the Web is now available DRM-free directly from The Pragmatic Programmers. Or you can get it on Amazon, where it's also available in Chinese and Japanese.

- Rethinking Window Management (2023/07/31 18:28)
  Over at Tobias Bernard's GNOME Blog, he writes about a new approach to tiling window managers. Window management is probably the single worst aspect of current operating systems, and his ideas for how a modern tiling window manager might work are extremely compelling to me. See also: Bluetile. If you require a short url to link to this article, please use https://ignco.de/804
- Tricking Monty Hall (2023/07/15 09:20)
  Yesterday, a friend sent me a screenshot of an Instagram story from mordlustderpodcast asking for an intuitive explanation of the Monty Hall problem (which, as I found out, is called the "Ziegenproblem", or "goat problem", in German). The basic problem is this: You're in a game show. You have three boxes in front of you. One of them contains a win (let's say the key to an expensive car), and two of them contain nothing (or goats, if you prefer that, although I personally feel that winning a goat would be pretty cool). You pick a box. The show's host then reveals the contents of one of the two remaining boxes, but, importantly, always a "goat box." At this point, the host gives you the option of either sticking with the box you picked originally, or switching to the other, unopened box. Should you switch or stick with your originally selected box? Intuitively, it seems like it shouldn't matter. There are two boxes, so it's a 50-50 chance of winning either way, right? But in reality, you should switch boxes, since that will increase your chance of winning from 33% to 66%.[^onepercent] [^onepercent]: Yes, it's true, one of the goats ate the remaining 1%. Or I just didn't want to type out an infinite number of 3s and 6s. This seems extremely strange to most people, including a lot of mathematicians. However, writing a program running this scenario a bunch of times quickly shows that it is true: it's better to switch. Thinking about this, I came up with what I feel is a pretty intuitive explanation for why you should switch, and since it's an explanation I haven't seen anywhere else, I thought I'd put it up here, in case it helps anyone.[^intuitive] [^intuitive]: Note that this is not the only intuitive way of explaining the problem. A pretty common approach that also works is to imagine that there are a large number of boxes, instead of just three. Imagine that you could pick two boxes out of the three. Obviously, if you can pick two boxes, then your chance of winning is 2 out of 3, or 66%, right? Well, you actually can, if you trick the show host. When the host shows you the three boxes, in your mind, without telling the host, pick two of them: But then, importantly, you lie to the show host, and say that you actually picked the third box, the one you did not pick: Now what happens? You've just tricked the show host into revealing which one of the two boxes you picked contains a goat! The host just did you a huge

favor, and opened one of the two boxes you picked. Now you've learned something: you've learned which of the two boxes you've picked definitely contains a goat. So now all you have to do is open the other box you've secretly picked. From the game's point of view, picking a box and then switching is the same as picking two boxes, falsely claiming you picked the third box, and then opening the remaining of your two boxes. But because you have picked two boxes, which gives you a 2/3 chance of winning, and because you have then tricked to show host into opening your goat box for you, the only way you can lose is if you've picked two goat boxes to begin with. And that's why switching boxes gives you a 66% chance of winning. Addendum: This great discussion on Hacker News provides other intuitive ways of explaining the problem. I particularly like this bit of feedback: This is incorrect, the goats and car are behind doors. They are not inside cardboard boxes. This is correct, and I do apologize. If you require a short url to link to this article, please use https://ignco.de/802But wait, there's more!Want to read more like this? Buy my book's second edition! Designed for Use: Create Usable Interfaces for Applications and the Web is now available DRM-free directly from The Pragmatic Programmers. Or you can get it on Amazon, where it's also available in Chinese and Japanese.

From:
https://wiki.tromjaro.alexio.tf/ - **TROMjaro wiki**

Permanent link:
https://wiki.tromjaro.alexio.tf/doku.php?id=news:ux:ignore_the_code

Last update: **2021/10/30 11:41**