# GitHubのエンジニアが徹底する社内コミュニケーション8つの原則

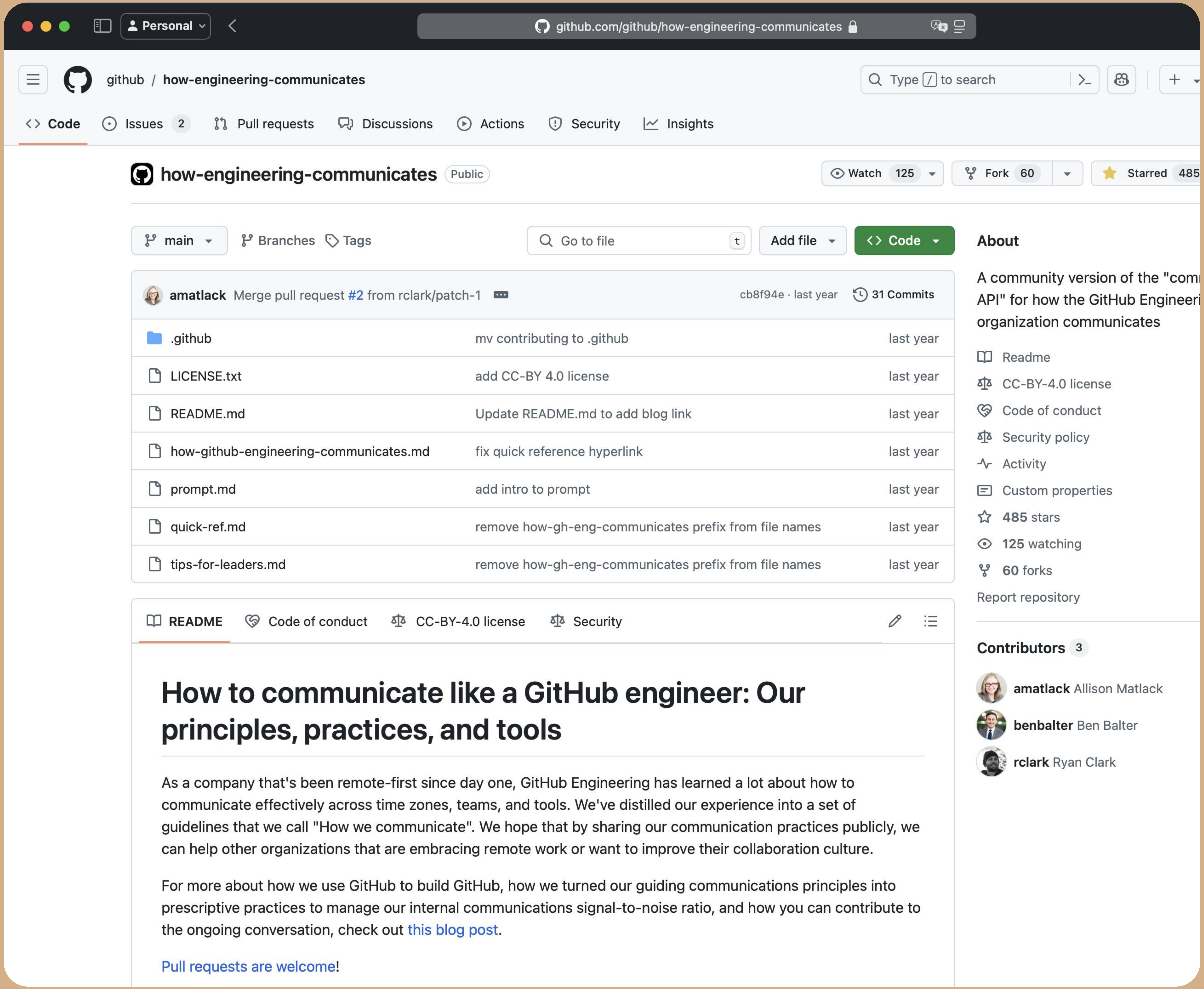
How GitHub Engineering communicates

Sho Mizutani, GitHub

## Customer Success Engineering

GitHub is **Remote First**

# How **GitHub Engineering** communicates



github.com/github/how-engineering-communicates

github / how-engineering-communicates

how-engineering-communicates Public

amatlack Merge pull request #2 from rclark/patch-1 cb8f94e · last year 31 Commits

| | | |
|---|---|---|
| .github | mv contributing to .github | last year |
| LICENSE.txt | add CC-BY 4.0 license | last year |
| README.md | Update README.md to add blog link | last year |
| how-github-engineering-communicates.md | fix quick reference hyperlink | last year |
| prompt.md | add intro to prompt | last year |
| quick-ref.md | remove how-gh-eng-communicates prefix from file names | last year |
| tips-for-leaders.md | remove how-gh-eng-communicates prefix from file names | last year |

## How to communicate like a GitHub engineer: Our principles, practices, and tools

As a company that's been remote-first since day one, GitHub Engineering has learned a lot about how to communicate effectively across time zones, teams, and tools. We've distilled our experience into a set of guidelines that we call "How we communicate". We hope that by sharing our communication practices publicly, we can help other organizations that are embracing remote work or want to improve their collaboration culture.

For more about how we use GitHub to build GitHub, how we turned our guiding communications principles into prescriptive practices to manage our internal communications signal-to-noise ratio, and how you can contribute to the ongoing conversation, check out this blog post.

Pull requests are welcome!

About

A community version of the "com
API" for how the GitHub Engineeri
organization communicates

Contributors 3

amatlack Allison Matlack
benbalter Ben Balter
rclark Ryan Clark

## Be Asynchronous First

非同期ファースト

| 同期 | **非同期** |
|---|---|
| リアルタイム<br>ミーティング | **非リアルタイム**<br>**Issues, PRs**<br>**チャット** |

“チャットは非同期です。同期的なやり取りを要求しないでください。例えば「お疲れさまです」だけ送るのはやめて、最初から内容を送りましょう。”

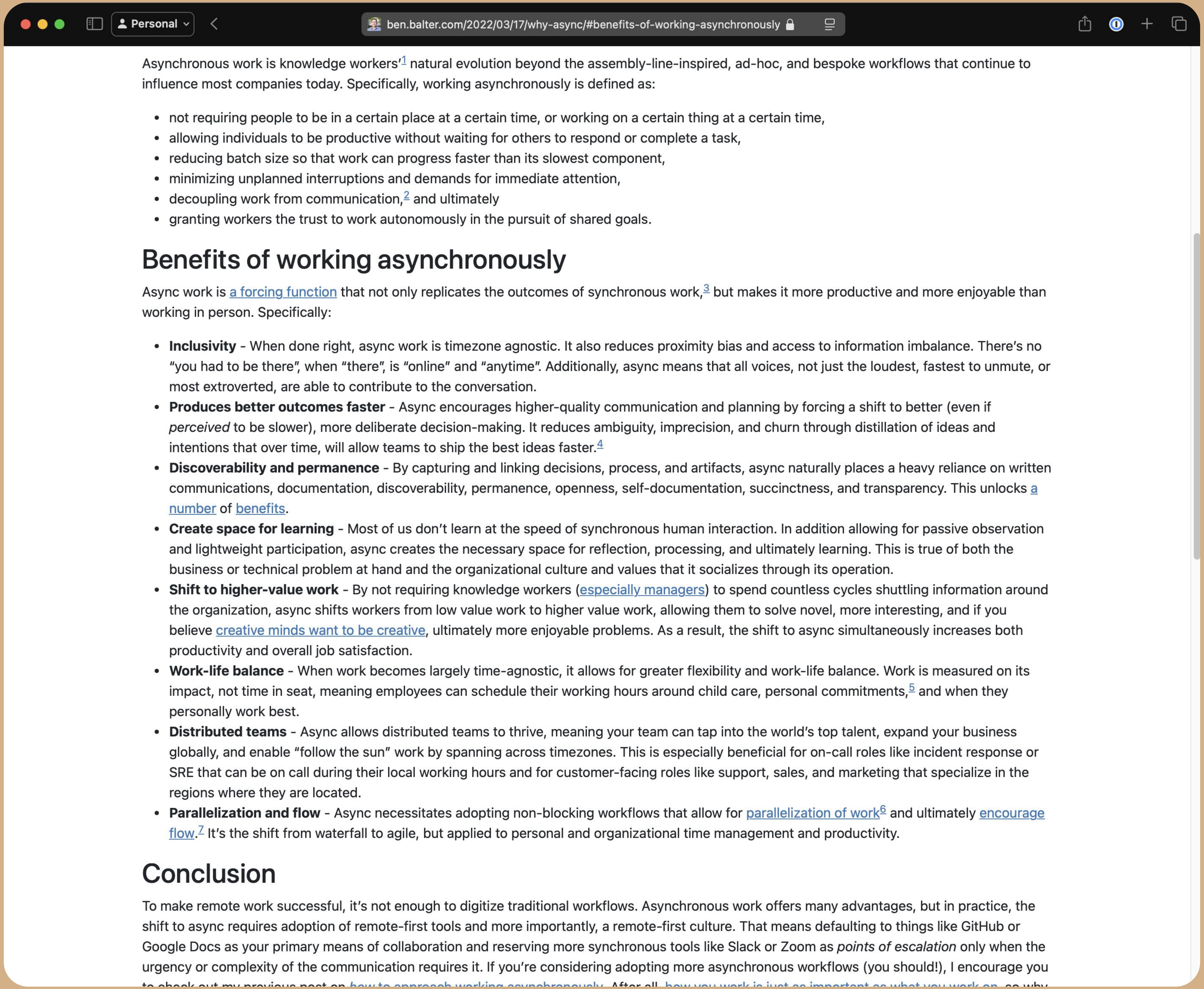
“Chat is asynchronous. Don't force synchronous interactions (e.g., don't just send a "hello"; go ahead and share context/links or ask your question in your initial message)”

## Why?

公平な意思決定
柔軟な働き方

## gh.io/why-async

Why you should work asynchronously


Personal
ben.balter.com/2022/03/17/why-async/#benefits-of-working-asynchronously

Asynchronous work is knowledge workers'[1] natural evolution beyond the assembly-line-inspired, ad-hoc, and bespoke workflows that continue to influence most companies today. Specifically, working asynchronously is defined as:

- not requiring people to be in a certain place at a certain time, or working on a certain thing at a certain time,
- allowing individuals to be productive without waiting for others to respond or complete a task,
- reducing batch size so that work can progress faster than its slowest component,
- minimizing unplanned interruptions and demands for immediate attention,
- decoupling work from communication,[2] and ultimately
- granting workers the trust to work autonomously in the pursuit of shared goals.

### Benefits of working asynchronously

Async work is a forcing function that not only replicates the outcomes of synchronous work,[3] but makes it more productive and more enjoyable than working in person. Specifically:

- **Inclusivity** - When done right, async work is timezone agnostic. It also reduces proximity bias and access to information imbalance. There's no "you had to be there", when "there", is "online" and "anytime". Additionally, async means that all voices, not just the loudest, fastest to unmute, or most extroverted, are able to contribute to the conversation.
- **Produces better outcomes faster** - Async encourages higher-quality communication and planning by forcing a shift to better (even if *perceived* to be slower), more deliberate decision-making. It reduces ambiguity, imprecision, and churn through distillation of ideas and intentions that over time, will allow teams to ship the best ideas faster.[4]
- **Discoverability and permanence** - By capturing and linking decisions, process, and artifacts, async naturally places a heavy reliance on written communications, documentation, discoverability, permanence, openness, self-documentation, succinctness, and transparency. This unlocks a number of benefits.
- **Create space for learning** - Most of us don't learn at the speed of synchronous human interaction. In addition allowing for passive observation and lightweight participation, async creates the necessary space for reflection, processing, and ultimately learning. This is true of both the business or technical problem at hand and the organizational culture and values that it socializes through its operation.
- **Shift to higher-value work** - By not requiring knowledge workers (especially managers) to spend countless cycles shuttling information around the organization, async shifts workers from low value work to higher value work, allowing them to solve novel, more interesting, and if you believe creative minds want to be creative, ultimately more enjoyable problems. As a result, the shift to async simultaneously increases both productivity and overall job satisfaction.
- **Work-life balance** - When work becomes largely time-agnostic, it allows for greater flexibility and work-life balance. Work is measured on its impact, not time in seat, meaning employees can schedule their working hours around child care, personal commitments,[5] and when they personally work best.
- **Distributed teams** - Async allows distributed teams to thrive, meaning your team can tap into the world's top talent, expand your business globally, and enable "follow the sun" work by spanning across timezones. This is especially beneficial for on-call roles like incident response or SRE that can be on call during their local working hours and for customer-facing roles like support, sales, and marketing that specialize in the regions where they are located.
- **Parallelization and flow** - Async necessitates adopting non-blocking workflows that allow for parallelization of work[6] and ultimately encourage flow.[7] It's the shift from waterfall to agile, but applied to personal and organizational time management and productivity.

### Conclusion

To make remote work successful, it's not enough to digitize traditional workflows. Asynchronous work offers many advantages, but in practice, the shift to async requires adoption of remote-first tools and more importantly, a remote-first culture. That means defaulting to things like GitHub or Google Docs as your primary means of collaboration and reserving more synchronous tools like Slack or Zoom as *points of escalation* only when the urgency or complexity of the communication requires it. If you're considering adopting more asynchronous workflows (you should!), I encourage you to check out my previous post on how to approach working asynchronously. After all, how you work is just as important as what you work on, so why

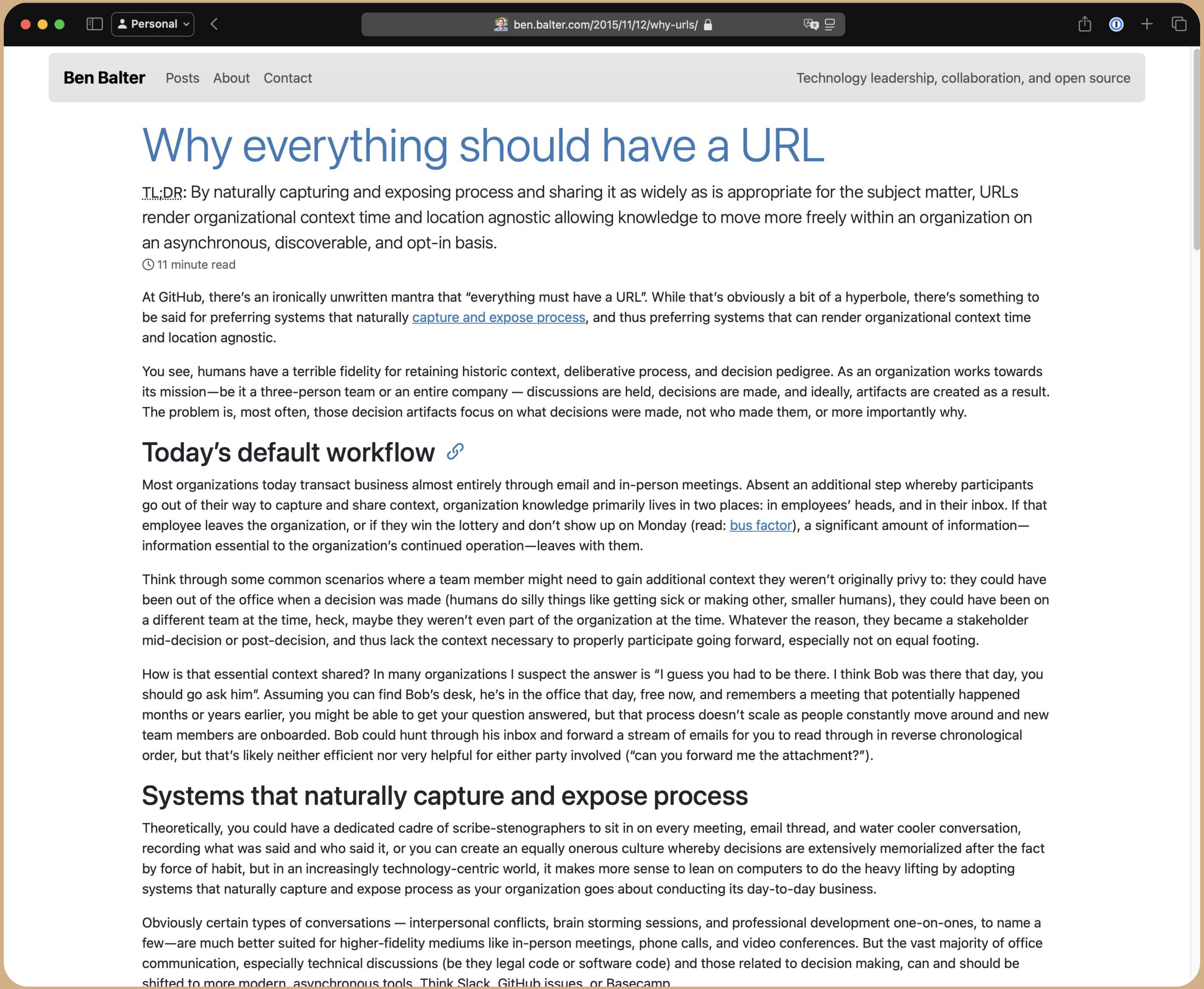
## Write Things Down

記録を残す

“If it doesn't have a URL, it didn't happen”

## gh.io/why-urls

## Why everything should have a URL



ben.balter.com/2015/11/12/why-urls/

Ben Balter Posts About Contact Technology leadership, collaboration, and open source

# Why everything should have a URL

TL;DR: By naturally capturing and exposing process and sharing it as widely as is appropriate for the subject matter, URLs render organizational context time and location agnostic allowing knowledge to move more freely within an organization on an asynchronous, discoverable, and opt-in basis.

11 minute read

At GitHub, there's an ironically unwritten mantra that "everything must have a URL". While that's obviously a bit of a hyperbole, there's something to be said for preferring systems that naturally capture and expose process, and thus preferring systems that can render organizational context time and location agnostic.

You see, humans have a terrible fidelity for retaining historic context, deliberative process, and decision pedigree. As an organization works towards its mission—be it a three-person team or an entire company — discussions are held, decisions are made, and ideally, artifacts are created as a result. The problem is, most often, those decision artifacts focus on what decisions were made, not who made them, or more importantly why.

## Today's default workflow

Most organizations today transact business almost entirely through email and in-person meetings. Absent an additional step whereby participants go out of their way to capture and share context, organization knowledge primarily lives in two places: in employees' heads, and in their inbox. If that employee leaves the organization, or if they win the lottery and don't show up on Monday (read: bus factor), a significant amount of information—information essential to the organization's continued operation—leaves with them.

Think through some common scenarios where a team member might need to gain additional context they weren't originally privy to: they could have been out of the office when a decision was made (humans do silly things like getting sick or making other, smaller humans), they could have been on a different team at the time, heck, maybe they weren't even part of the organization at the time. Whatever the reason, they became a stakeholder mid-decision or post-decision, and thus lack the context necessary to properly participate going forward, especially not on equal footing.

How is that essential context shared? In many organizations I suspect the answer is "I guess you had to be there. I think Bob was there that day, you should go ask him". Assuming you can find Bob's desk, he's in the office that day, free now, and remembers a meeting that potentially happened months or years earlier, you might be able to get your question answered, but that process doesn't scale as people constantly move around and new team members are onboarded. Bob could hunt through his inbox and forward a stream of emails for you to read through in reverse chronological order, but that's likely neither efficient nor very helpful for either party involved ("can you forward me the attachment?").

## Systems that naturally capture and expose process

Theoretically, you could have a dedicated cadre of scribe-stenographers to sit in on every meeting, email thread, and water cooler conversation, recording what was said and who said it, or you can create an equally onerous culture whereby decisions are extensively memorialized after the fact by force of habit, but in an increasingly technology-centric world, it makes more sense to lean on computers to do the heavy lifting by adopting systems that naturally capture and expose process as your organization goes about conducting its day-to-day business.

Obviously certain types of conversations — interpersonal conflicts, brain storming sessions, and professional development one-on-ones, to name a few—are much better suited for higher-fidelity mediums like in-person meetings, phone calls, and video conferences. But the vast majority of office communication, especially technical discussions (be they legal code or software code) and those related to decision making, can and should be shifted to more modern, asynchronous tools. Think Slack, GitHub issues, or Basecamp

## Make work visible and overcommunicate

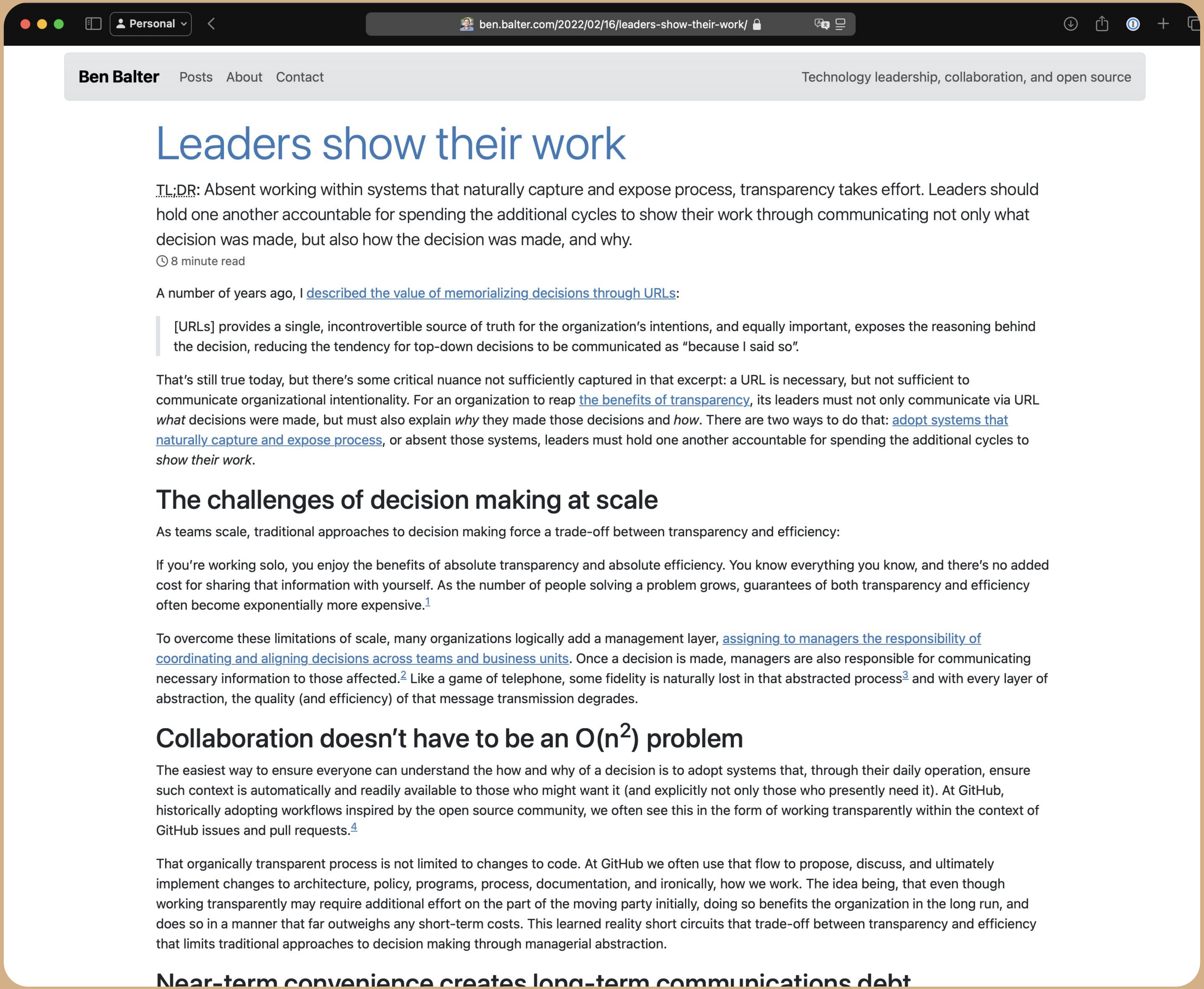
作業・成果を可視化し、積極的に発信する

## Why?

可視化 → 透明性を高める
発信 → 作業の重複を避ける

gh.io/show-their-work

## Leaders show their work



ben.balter.com/2022/02/16/leaders-show-their-work/

**Ben Balter** Posts About Contact

Technology leadership, collaboration, and open source

# Leaders show their work

TL;DR: Absent working within systems that naturally capture and expose process, transparency takes effort. Leaders should hold one another accountable for spending the additional cycles to show their work through communicating not only what decision was made, but also how the decision was made, and why.

8 minute read

A number of years ago, I described the value of memorializing decisions through URLs:

> [URLs] provides a single, incontrovertible source of truth for the organization's intentions, and equally important, exposes the reasoning behind the decision, reducing the tendency for top-down decisions to be communicated as "because I said so".

That's still true today, but there's some critical nuance not sufficiently captured in that excerpt: a URL is necessary, but not sufficient to communicate organizational intentionality. For an organization to reap the benefits of transparency, its leaders must not only communicate via URL *what* decisions were made, but must also explain *why* they made those decisions and *how*. There are two ways to do that: adopt systems that naturally capture and expose process, or absent those systems, leaders must hold one another accountable for spending the additional cycles to *show their work*.

## The challenges of decision making at scale

As teams scale, traditional approaches to decision making force a trade-off between transparency and efficiency:

If you're working solo, you enjoy the benefits of absolute transparency and absolute efficiency. You know everything you know, and there's no added cost for sharing that information with yourself. As the number of people solving a problem grows, guarantees of both transparency and efficiency often become exponentially more expensive.[1]

To overcome these limitations of scale, many organizations logically add a management layer, assigning to managers the responsibility of coordinating and aligning decisions across teams and business units. Once a decision is made, managers are also responsible for communicating necessary information to those affected.[2] Like a game of telephone, some fidelity is naturally lost in that abstracted process[3] and with every layer of abstraction, the quality (and efficiency) of that message transmission degrades.

## Collaboration doesn't have to be an $O(n^2)$ problem

The easiest way to ensure everyone can understand the how and why of a decision is to adopt systems that, through their daily operation, ensure such context is automatically and readily available to those who might want it (and explicitly not only those who presently need it). At GitHub, historically adopting workflows inspired by the open source community, we often see this in the form of working transparently within the context of GitHub issues and pull requests.[4]

That organically transparent process is not limited to changes to code. At GitHub we often use that flow to propose, discuss, and ultimately implement changes to architecture, policy, programs, process, documentation, and ironically, how we work. The idea being, that even though working transparently may require additional effort on the part of the moving party initially, doing so benefits the organization in the long run, and does so in a manner that far outweighs any short-term costs. This learned reality short circuits that trade-off between transparency and efficiency that limits traditional approaches to decision making through managerial abstraction.

## Near-term convenience creates long-term communications debt

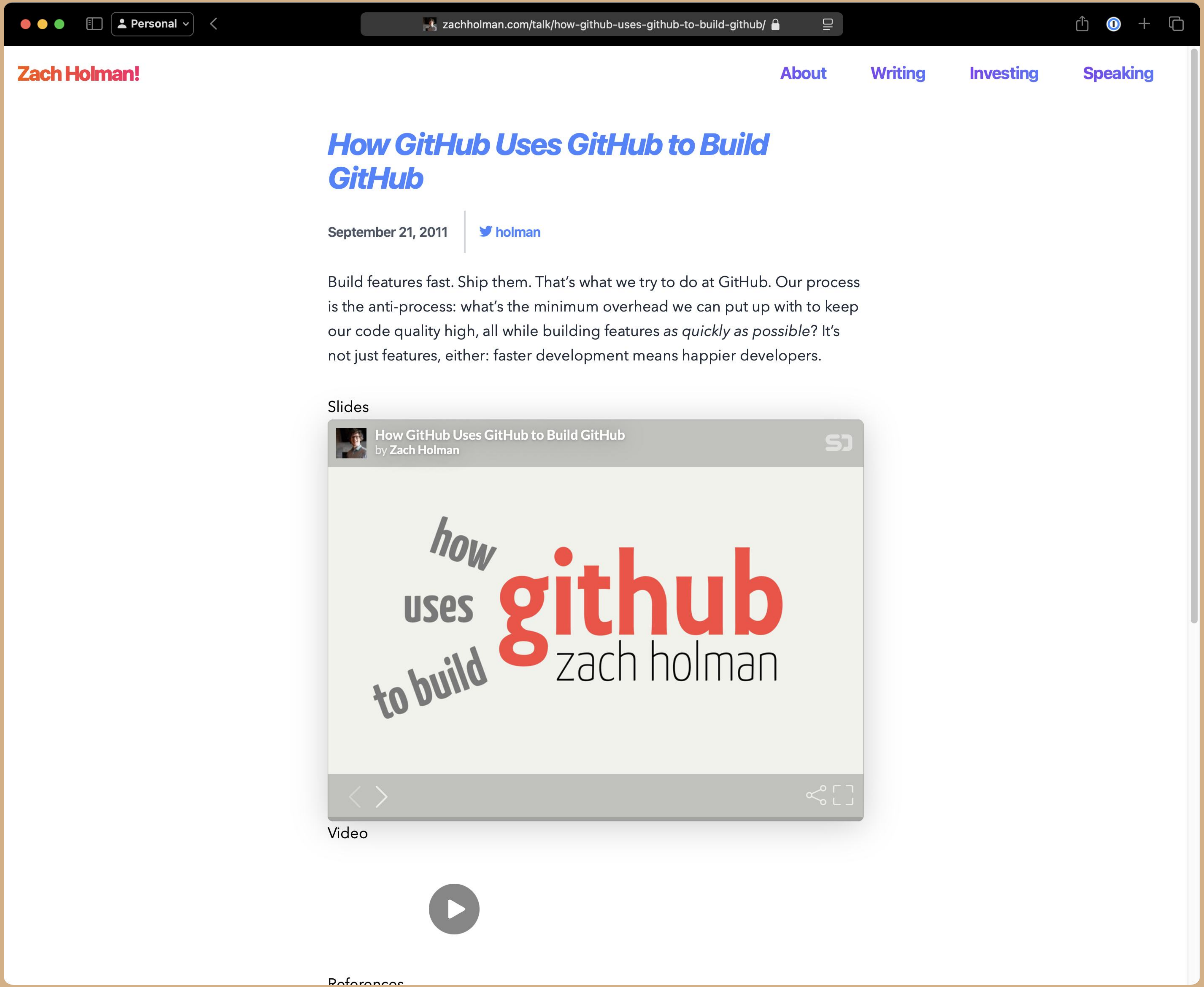
# Prefer GitHub tools and workflows

GitHubのツールとワークフローを優先する

## Why?

可視化・発信しやすい

# How GitHub Uses GitHub to Build GitHub



zachholman.com/talk/how-github-uses-github-to-build-github/

Zach Holman!

About Writing Investing Speaking

## How GitHub Uses GitHub to Build GitHub

September 21, 2011 | holman

Build features fast. Ship them. That's what we try to do at GitHub. Our process is the anti-process: what's the minimum overhead we can put up with to keep our code quality high, all while building features *as quickly as possible*? It's not just features, either: faster development means happier developers.

Slides

How GitHub Uses GitHub to Build GitHub
by Zach Holman

Video

# Embrace collaboration

コラボレーションを推進する

## Why?

早く頻繁なフィードバック
→より良い意思決定

# Foster a culture that values documentation maintenance

ドキュメントのメンテナンスを重視する文化を作る

## Why?

更新されていないドキュメントは非効率を生む

## Communicate openly, honestly, and authentically

オープンに、正直に、誠実に意思を伝える

I. インクルーシブな言葉を使う

II. ✨絵文字を使う✨

lululemon
Opening this winter

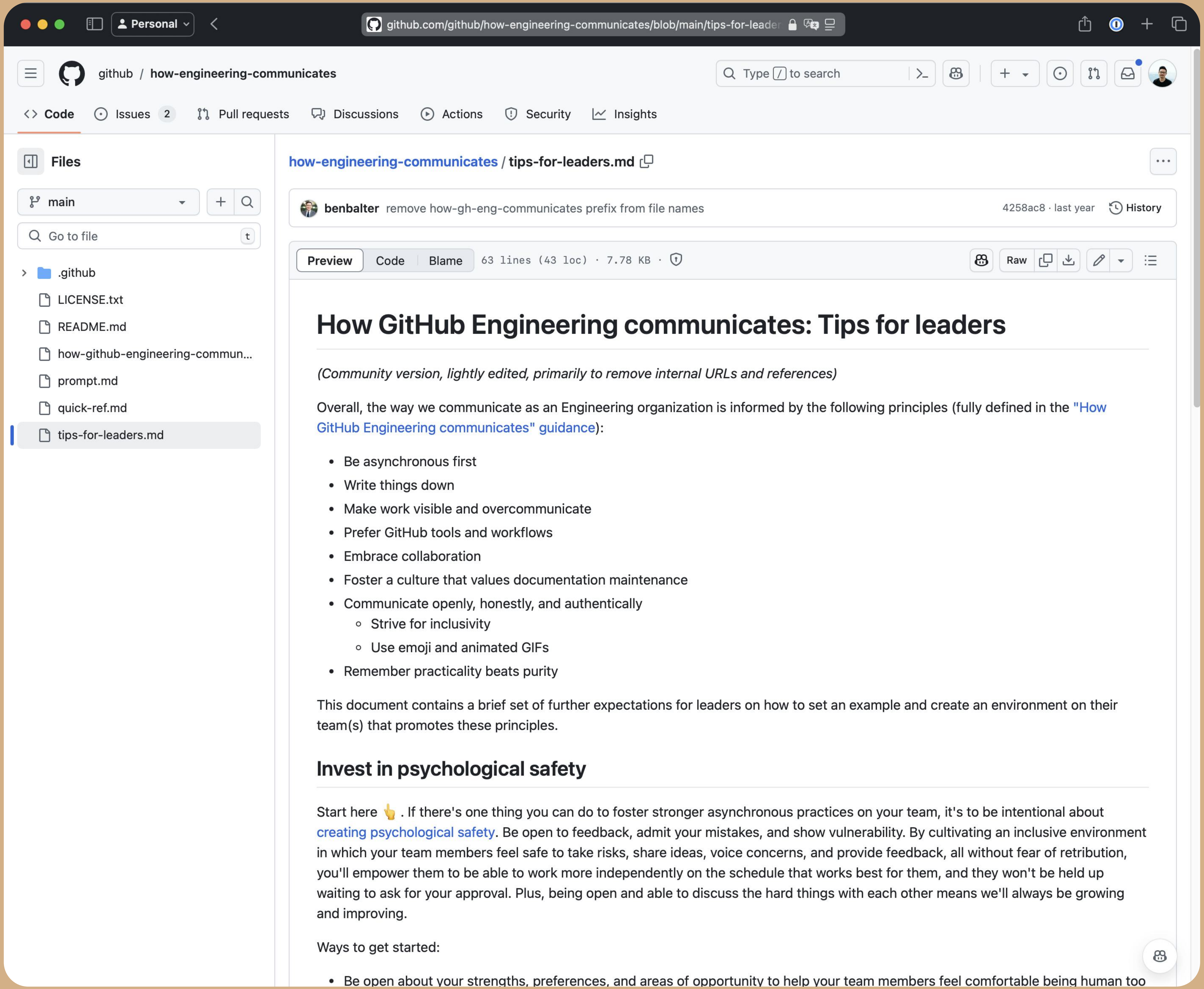
I. インクルーシブな言葉を使う

II. ✨絵文字を使う✨

## Remember practicality beats purity

実用性は完璧さに勝る

# Tips for leaders



github / how-engineering-communicates

how-engineering-communicates / tips-for-leaders.md

benbalter remove how-gh-eng-communicates prefix from file names 4258ac8 · last year

63 lines (43 loc) · 7.78 KB

# How GitHub Engineering communicates: Tips for leaders

*(Community version, lightly edited, primarily to remove internal URLs and references)*

Overall, the way we communicate as an Engineering organization is informed by the following principles (fully defined in the "How GitHub Engineering communicates" guidance):

- Be asynchronous first
- Write things down
- Make work visible and overcommunicate
- Prefer GitHub tools and workflows
- Embrace collaboration
- Foster a culture that values documentation maintenance
- Communicate openly, honestly, and authentically
  - Strive for inclusivity
  - Use emoji and animated GIFs
- Remember practicality beats purity

This document contains a brief set of further expectations for leaders on how to set an example and create an environment on their team(s) that promotes these principles.

## Invest in psychological safety

Start here 👆 . If there's one thing you can do to foster stronger asynchronous practices on your team, it's to be intentional about creating psychological safety. Be open to feedback, admit your mistakes, and show vulnerability. By cultivating an inclusive environment in which your team members feel safe to take risks, share ideas, voice concerns, and provide feedback, all without fear of retribution, you'll empower them to be able to work more independently on the schedule that works best for them, and they won't be held up waiting to ask for your approval. Plus, being open and able to discuss the hard things with each other means we'll always be growing and improving.

Ways to get started:

- Be open about your strengths, preferences, and areas of opportunity to help your team members feel comfortable being human too

01. 非同期ファースト
02. 記録を残す
03. 作業・成果を可視化し、積極的に発信する
04. GitHubのツールとワークフローを優先する
05. コラボレーションを推進する
06. ドキュメントのメンテナンスを重視する文化を作る
07. オープンに、正直に、誠実に意思を伝える
08. 実用性は完璧さに勝る

# Thank You!

## Resources

- How GitHub Engineering communicates
- gh.io/why-async
- gh.io/why-urls
- gh.io/show-their-work
- How GitHub Uses GitHub to Build GitHub
- How to communicate like a GitHub engineer: our principles, practices, and tools