

Indie microblogging and the fediverse

Manton Reece

@ EFF Austin

Hi. Going to talk for a bit about my background, blogging, the fediverse, and the approach with my platform Micro.blog.



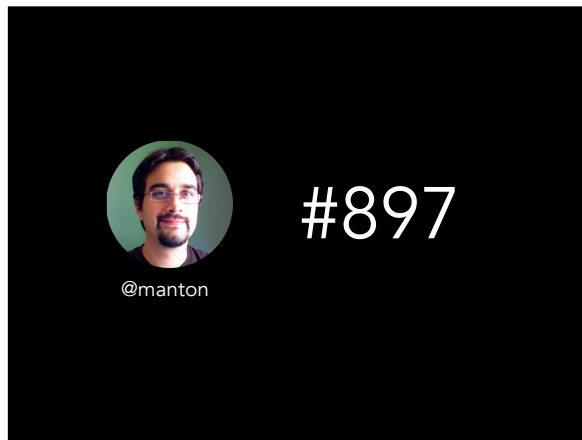
A little about me: I'm a long-time Mac developer, iOS developer, web developer. This is me waiting in line for the first iPhone. I got started programming the Mac in the 90s and still love the Mac, even if I'm less thrilled with Apple as a huge company.

SXSW Interactive started today

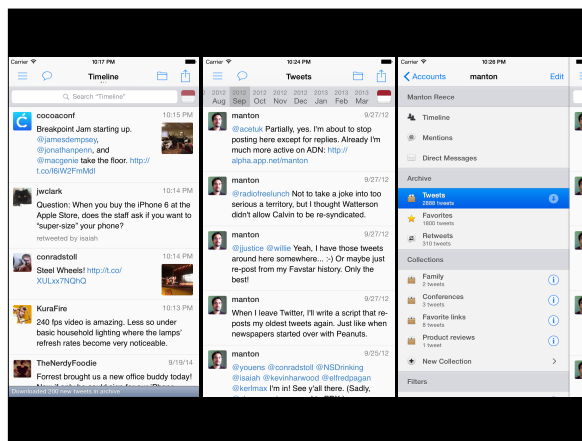
MAR 9, 2002

SXSW Interactive started today. Seems an appropriate time to start a weblog, as if there weren't enough in the world already. Welcome, and enjoy.

I've been blogging for a long time. Over 20 years. This is my first blog post. It's actually a short microblog post. There are a couple interesting things about this post that I'll get back to later.



But first let's talk about social networks. I joined Twitter in 2006. It was silly but I was kind of proud of this 3-digit user ID. I say this just to set the context for where I'm coming from and how excited I was for Twitter in the early days.



I built tools for Twitter. I saw so much potential in the Twitter API. This is an iOS app I built, a third-party Twitter client called Tweet Library.



I created the Tweet Marker API. Used by apps like Tweetbot and Twittrific. Used by dozens of apps across iOS and Android.

"There are several factors that make Twitter a nearly **ideal playground for UI design**. The obvious ones are the growing popularity of the service itself and the relatively small scope of a Twitter client. [...] you will see some very different UI designs, not only in terms of visual style but in terms of layout, structure, and flow."

— John Gruber, April 2009

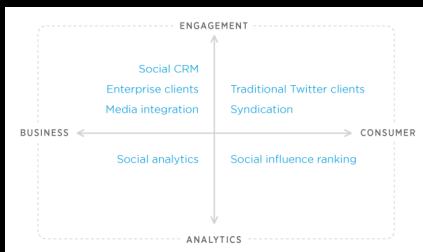
John Gruber quote. Twitter apps were a "playground for UI design". We're now seeing this with the fediverse. So many new apps that work with Mastodon and Bluesky.

Twitterrific

- First use of "tweet" to describe an update.
- First use of a bird icon.
- First character counter as you type.



The word "tweet" itself came not from Twitter but from a third-party app: Twitterrific from the Iconfactory was the first app to use the word tweet. I think there's a lot to be inspired from in those early Twitter apps when third-party developers were excited to build something new.

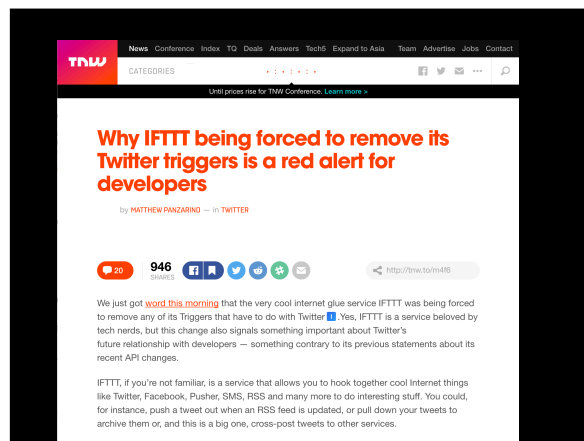


But I grew disillusioned with a series of steps Twitter took to close their API. They took what was a vibrant developer ecosystem and ruined it, before Elon Musk bought the company. First they started discouraging certain types of apps. [read quadrants] Removed RSS feeds. Required registration to use the API.

"Additionally, if you are building a Twitter client application that is accessing the home timeline, account settings or direct messages API endpoints (typically used by traditional client applications) or are using our User Streams product, **you will need our permission if your application will require more than 100,000 individual user tokens.**"

— blog.twitter.com, August 2012

They built that discouragement into the platform itself, so there was actually a limit on the number of auth tokens an app could use. Ceiling for how popular an app could be.



Started shutting down integrations. Article about IFTTT losing access to the Twitter API. This was frustrating because Twitter was turning their back on developers. Developers who had contributed so much to the platform.

"Twitter in 2014 feels like it has settled into a certain default mode of hostility and rage."

— Buzz Andersen, October 2014

And so I left. I stopped posting in 2012. And it kept getting worse. Twitter was bringing out the worst in some people. This feels like the whole web now. Ad-based companies aren't aligned well with users and developers.

"I talk back and I am 'feeding the trolls'. I say nothing and the harassment escalates. I report threats and I am a 'censor'. [...] I have to conclude, after half a decade of troubleshooting, that it may simply be **impossible to make this platform usable for anyone but trolls, robots and dictators.**"

— Lindy West, January 2017

Lindy West writing for the Guardian about the harassment she's had to deal with on Twitter. [read] She deleted her account. There was something fundamentally broken, Twitter didn't plan for harassment from the beginning.



Elon Musk accelerated everything. Twitter had been conflicted before, internally some people wanted an open API, Twitter would alternate between these years of being developer-friendly or developer-hostile, but the pieces for a closed platform were all in motion before Elon. Elon finished dismantling Twitter, but Twitter was always going to be a closed silo, always going to be a dead-end.



I set out to work on some of these problems for blogging with a new service called Micro.blog. Launched in 2017 with a Kickstarter campaign, and has been slowly growing ever since, and adapting to new protocols.

Blog hosting + safe community

- Short microblog posts
- Custom domain names
- Full-length blog posts
- Email newsletters
- Photo and video hosting
- Hugo themes
- Podcasting
- POSSE

Micro.blog has a bunch of features. At it's heart is blog hosting. Photo hosting. Podcast hosting. Also features like blogging about books you're reading. Email newsletters. Open APIs like ActivityPub. Domain name registration.



Micro.blog



Epilogue



Strata



Sunlit

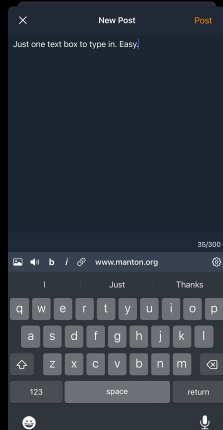


Wavelength

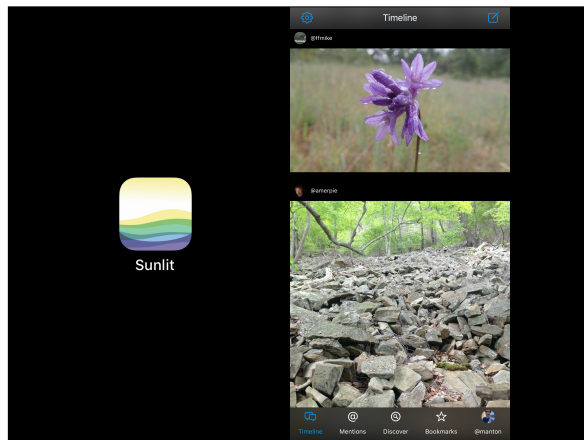
One of the first things to focus on was having open APIs. Unlike Twitter and X making it impossible to have third-party apps, with Micro.blog wanted to encourage lots of apps. That started even with the official Micro.blog apps. We have a suite of mobile apps. [read]



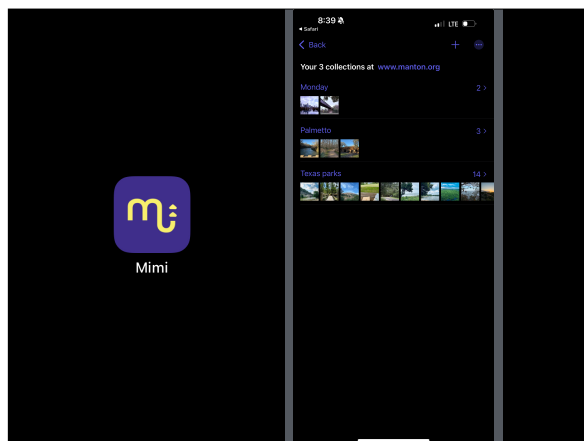
Micro.blog



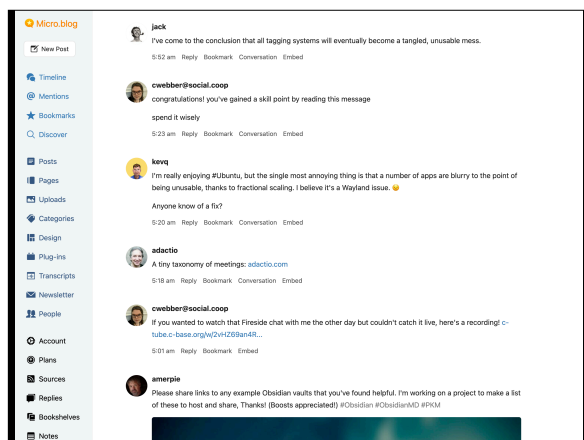
This is the main iPhone app. Simple. I tap Post and it goes to my blog. The blog can be hosted on Micro.blog or the app can also post to WordPress and other platforms that use open APIs.



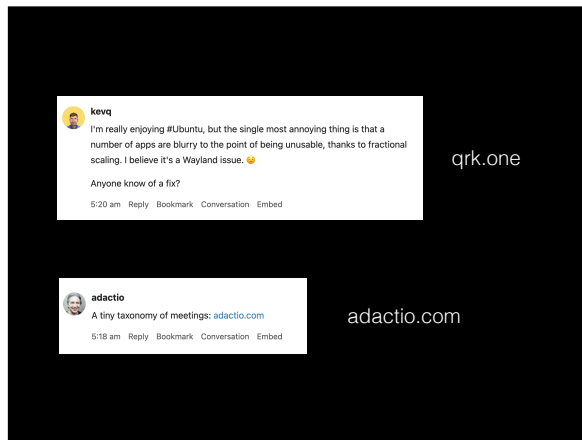
Photoblogging is an important variation of microblogging. So we wanted that to be really easy. Feels sort of like using Instagram. Sunlit is a dedicated app just for photos.



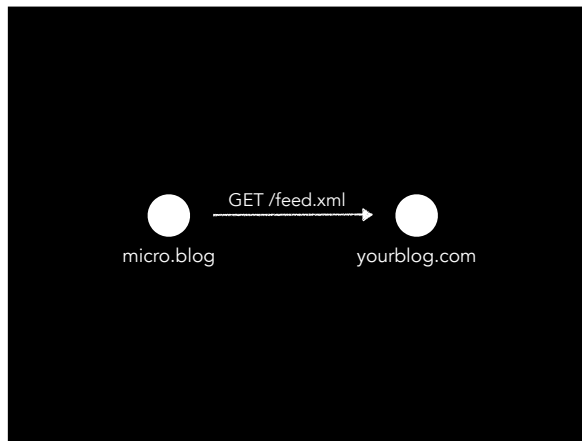
And instead of discouraging third-party apps like Twitter did, we should encourage them. This is a Micro.blog app by developer Sam Grover called Mimi Uploader. It's great. Focused on batch uploading photos and managing collections of photos.



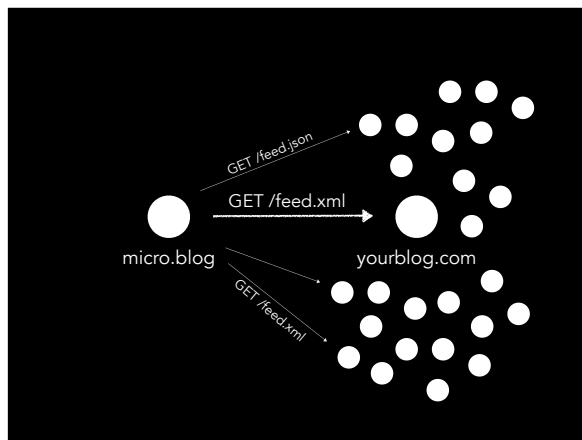
Micro.blog timeline brings all the posts together. Follow users, their posts show up in the timeline. Reverse-chronological list of posts. Replies, viewing conversations.



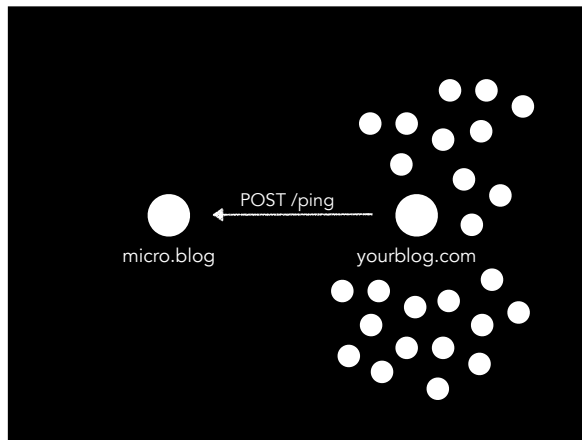
The posts come from RSS feeds. People are posting to their own domain name first, and then Micro.blog reads their feeds and aggregates everything into a unified social timeline.



So what does this look like. Micro.blog just asks for the RSS feed every once in a while.



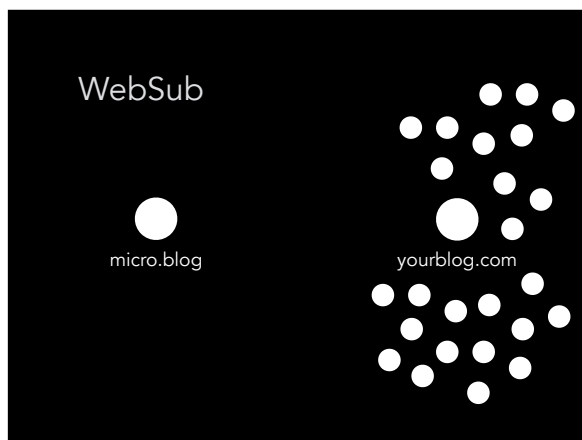
Except that there are a lot of blogs out there. Very inefficient. Micro.blog is constantly working in the background to download feeds over and over to see if anything is new. Polling RSS feeds is slow.



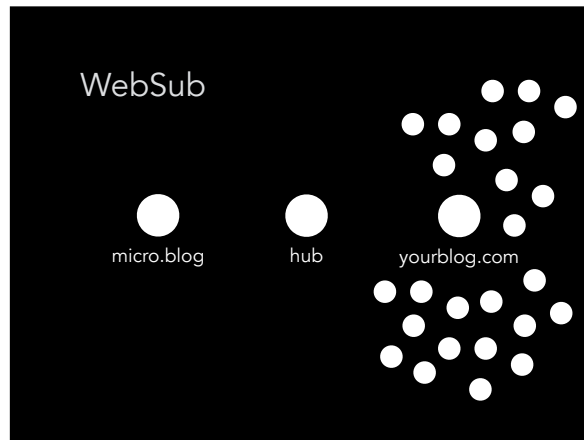
There's another way. The blog can notify the RSS aggregator, in this case Micro.blog, whenever there's a new post. "Ping! I wrote a new post."



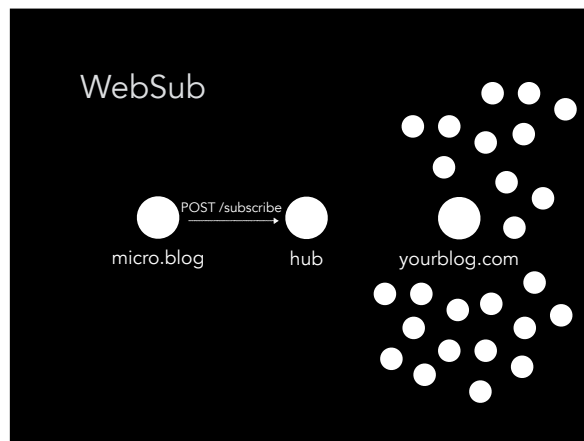
We can do better than that. This is where standards can help make these platforms more efficient. In 2018, there was a W3C working group called the social web working group. They published several important standards as W3C recommendations. Micro.blog supports pretty much all the recommended specs that came out of the working group. [read]



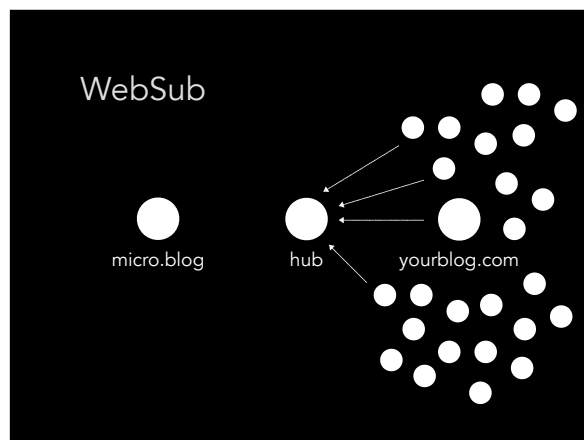
WebSub applies directly to this issue of polling RSS feeds for changes.



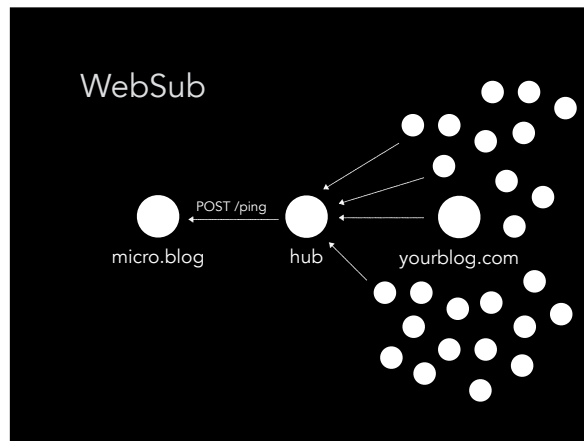
WebSub adds a hub. In your feed, you link to a hub that you will notify about new posts.



Micro.blog can subscribe to your blog in the hub.



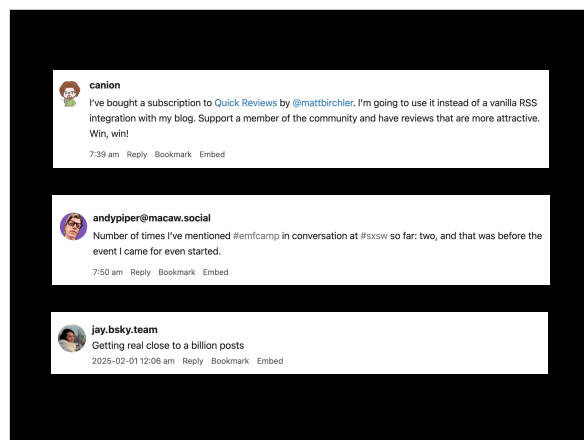
Now all the blogs just tell the hub when there's a new post.



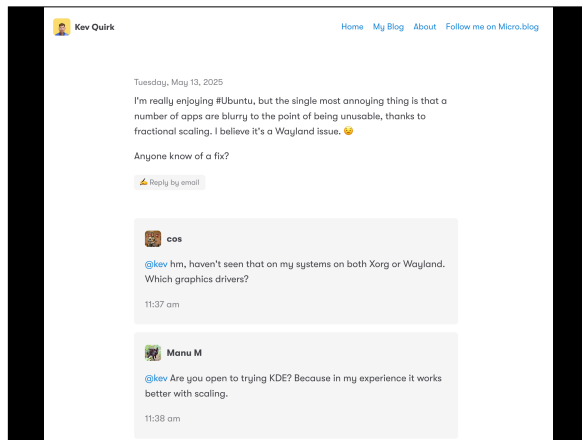
Then the hub notifies Micro.blog whenever one of the blogs changes. Much faster notifications rather than waiting several minutes or longer for a post to show up. And it's more distributed because there can be many hubs.

The social web can weave together many protocols.

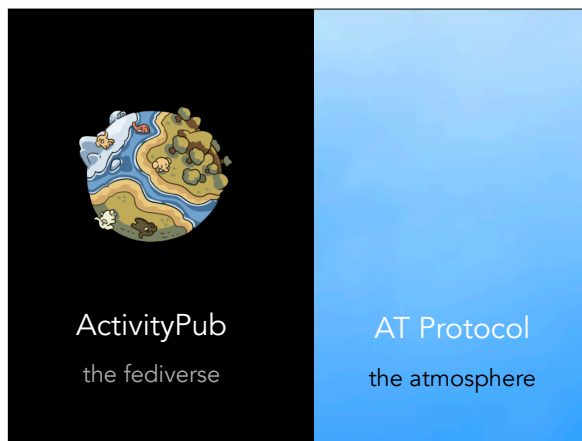
I use this example with WebSub to underscore that the social web can weave together many different protocols. It is open and always evolving. And sometimes there are little gems of earlier standards that we can build on.



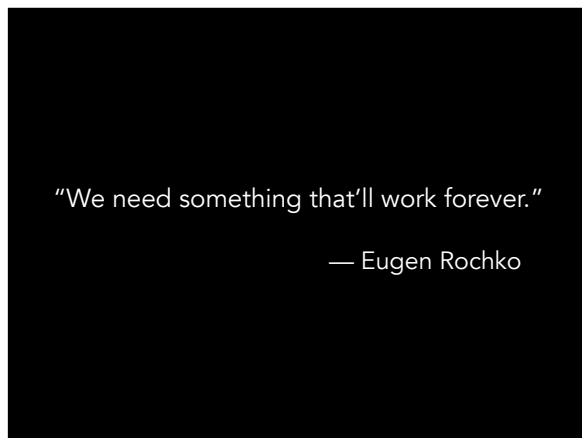
The sources for posts in Micro.blog don't only have to be from RSS feeds, they can be from different platforms that were designed around realtime notifications of new posts. Micro.blog users, fediverse users, Bluesky users. Micro.blog natively supports these different protocols.



And replies from those platforms are also brought back into the blog and social experience in Micro.blog. So if someone replies to this post in Micro.blog or Mastodon, the replies are collected directly on the blog post like comments. You can post to your own blog but still have a presence on other platforms.



Supporting all these different platforms and protocols is possible because of new standards. There is an emerging social web that doesn't depend on siloed, closed platforms with proprietary APIs. Here are two of the open APIs powering the social web. Mastodon with ActivityPub and the fediverse. And Bluesky with AT Protocol and the atmosphere.



Start with ActivityPub. This is from the creator of Mastodon. “We need something that’ll work forever.” He was looking for how he could build a Twitter-like social network with existing open standards.

- ActivityPub
- ActivityStreams
- WebFinger
- HTTP Signatures

What he found was there wasn't one single specification. ActivityPub is really a suite of protocols, if you want to be compatible with Mastodon and other fediverse implementations. First version of Mastodon actually shipped before ActivityPub. It used another standard called OStatus, and then Mastodon migrated everyone to ActivityPub. [read]

GET /well-known/webfinger?resource=acct:manton@indieweb.social

```
{
  "subject": "acct:manton@indieweb.social",
  "aliases": [
    "https://indieweb.social/@manton",
    "https://indieweb.social/users/manton"
  ],
  "links": [
    {
      "rel": "self",
      "type": "application/activity+json",
      "href": "https://indieweb.social/users/manton"
    }
  ]
}
```

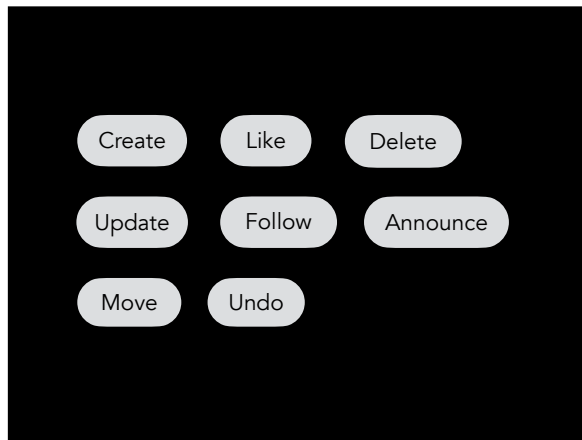
Let's talk about a few of these. WebFinger. Technically the first thing we encounter as users. WebFinger translates email address-style usernames to URLs. Let's say my handle is manton@indieweb.social. Make a web request to this endpoint, get a link to their ActivityPub profile.

GET https://indieweb.social/users/manton

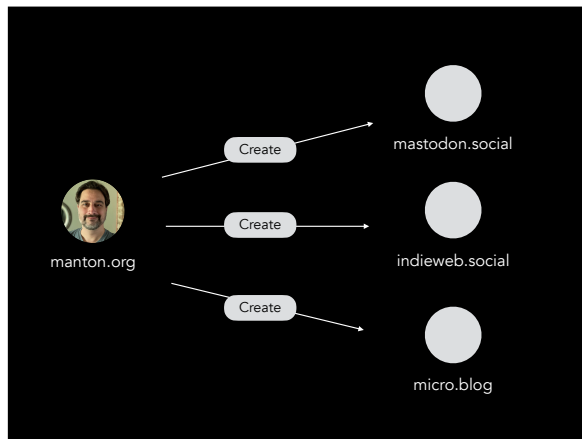
Accept: application/json+activity

```
{
  "@context": [
    "https://www.w3.org/ns/activitystreams",
    "https://w3id.org/security/v1"
  ],
  "id": "https://indieweb.social/users/manton",
  "type": "Person",
  "following": "https://indieweb.social/users/manton/following",
  "followers": "https://indieweb.social/users/manton/followers",
  "inbox": "https://indieweb.social/users/manton/inbox",
  "name": "Manton Reece"
}
```

So next step is to get the profile JSON. ActivityPub calls these profiles "actors". The @context is part of JSON-LD (linked data). Often need the "accept" header, HTTP content negotiation. I simplified the JSON, there's way more in here. Highlight: inbox URL, where we'll send activities to this user.



ActivityPub gets its name because there are all these activities that are sent between servers. [read] When you make a post, it sends a “create” activity to all your followers. When you follow someone, it sends them a “follow” activity.



These activities are sent to other servers on the fediverse. Whether it's creating a new post or following another user.

```
POST /users/manton/inbox

{
  "@context": "...",
  "type": "Create",
  "object": {
    "type": "Note",
    "url": "https://www.manton.org/my-post/"
    "published": "2025-05-13T14:30:00+00:00",
    "content": "<p>Hello world!</p>",
    "tag": []
  }
}
```

Create a new note, sending an HTTP post to the follower's inbox. This is how communication happens between fediverse servers.



Let's talk about Bluesky and the AT Protocol.

"In principle, I don't believe anyone should own or run Twitter. It wants to be a public good at a protocol level, not a company."

— Jack Dorsey

[read] While Jack Dorsey was still CEO of Twitter, he spun up a project called Bluesky. The goal was to develop an open protocol that Twitter could eventually use itself.

"If fate doesn't exist, then we must create it."

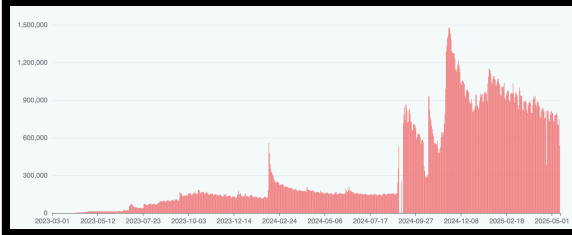
— Jay Graber

There were a series of fascinating decisions that all seemed to happen at once... Elon Musk was buying up Twitter shares. Jack Dorsey was stepping down as Twitter CEO. And Jay Graber, who was on the early group coming up what what Bluesky would look like, was asked to lead Bluesky. She said she would lead it on one condition: if they could spin out Bluesky into its own company independent of Twitter.

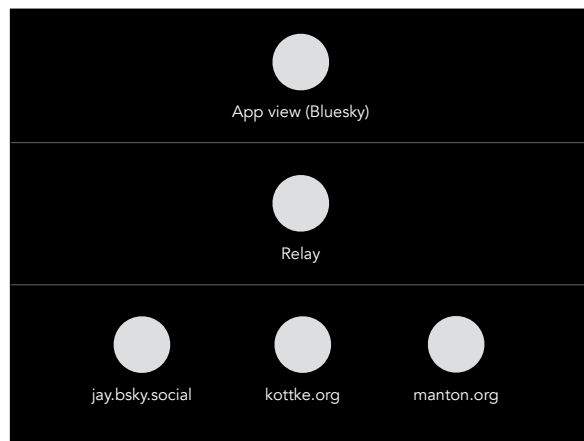
Mastodon: **8 million users** (12.5 million fediverse)

Bluesky: **35 million users**

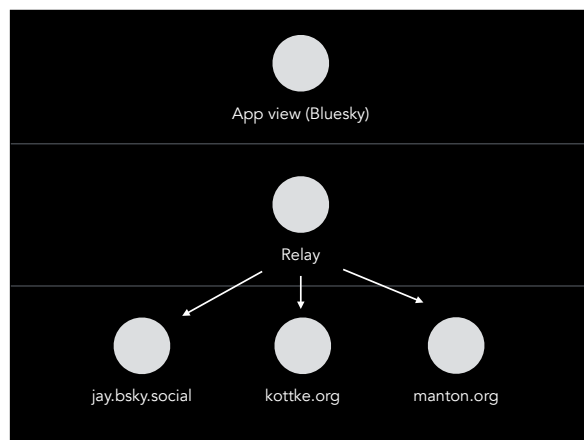
Bluesky daily posters:



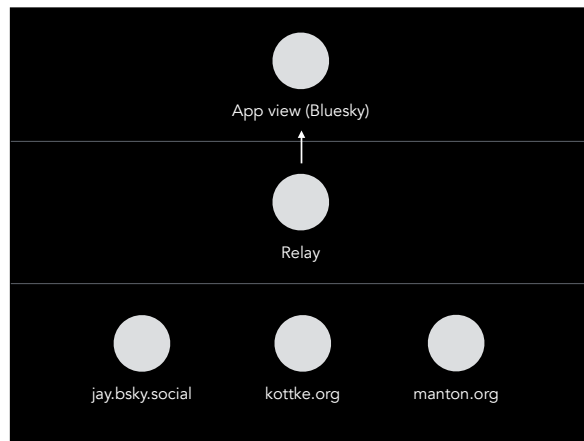
Bluesky has grown a lot. Mastodon and Bluesky. Graph of new posts per day. Huge spikes as people left X.



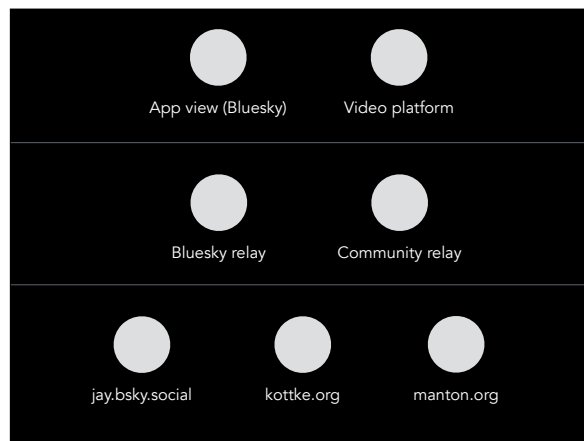
What is the architecture of Bluesky and the AT Protocol. It's not federated like Mastodon. But it's not strictly centralized either. At the bottom you have personal data servers. Servers that hold your posts, follows. Domain names for identity.



In the middle you have one or more relays. Relays aggregate all the posts from the personal data servers and combine them into a firehouse of all data on the network.



And then that is sent back up to the app view, like the Bluesky app where people are interacting with the platform.



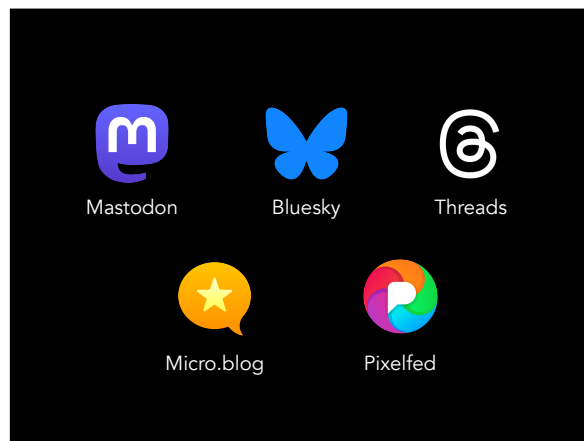
Where the resiliency comes is you could add new relays. You could add new apps views, for example one focused only on video content. And you could clone and replace Bluesky if something happened to the company, or was taken over by new leadership. So the architecture is modular.

```
{
  "id": "app.bsky.feed.post",
  "defs": {
    "main": {
      "type": "record",
      "description": "Record containing a Bluesky post.",
      "record": {
        "required": ["text", "createdAt"],
        "properties": {
          "text": {
            "type": "string",
            "maxGraphemes": 300,
            "description": "The primary post content. May be an empty string, if there are embeds."
          },
          "facets": {
            "type": "array",
            "description": "Annotations of text (mentions, URLs, hashtags, etc)",
            "items": { "type": "ref", "ref": "app.bsky.richtext.facet" }
          }
        }
      }
    }
  }
}
```

AT Protocol has a format called lexicons that keeps all of this data coordinated. A lexicon describes what the data in certain formats looks like. So Bluesky has defined a lexicon for short-form microblogging. 300 characters. [read] Facets for character ranges of styled text. You could have other lexicons and all of the data records would co-exist in someone's personal data server.

ActivityPub	AT Protocol
<ul style="list-style-type: none"> • Federated across many servers • Identity based on community servers • Followers and posts can be moved while servers are online • ActivityPub is a W3C open standard 	<ul style="list-style-type: none"> • Single large app view server, distributed data stores • Identity with domain names • Followers and posts can be moved anytime without changing identity • AT Protocol not yet put to standards group

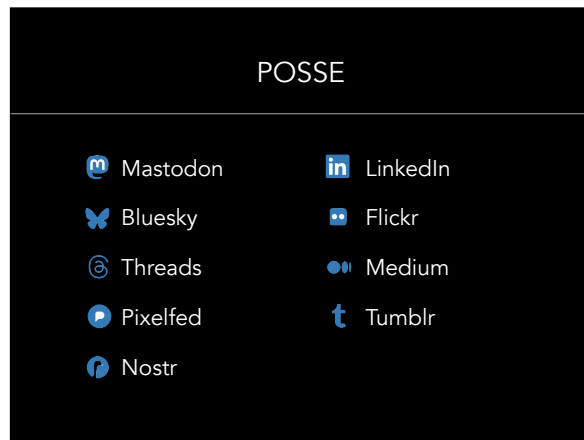
Comparing these two platforms. [read]



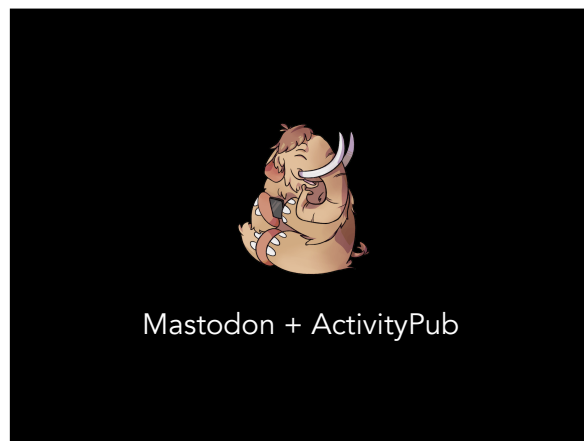
We now have many platforms. Mastodon. Bluesky. Threads. Micro.blog. Pixelfed, a photo sharing fediverse platform, and many others. We talked about getting posts into a platform like Micro.blog, but what about sending posts out, even if a platform has a proprietary API? How do we tie all of this together with blogs?



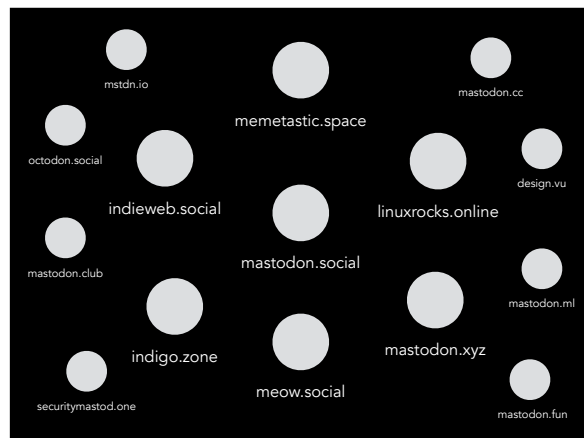
One way is called POSSE: post on your own site, syndicate elsewhere. This is a term coined by the IndieWeb community. Copying posts between platforms, but starting on your blog where you own your content.



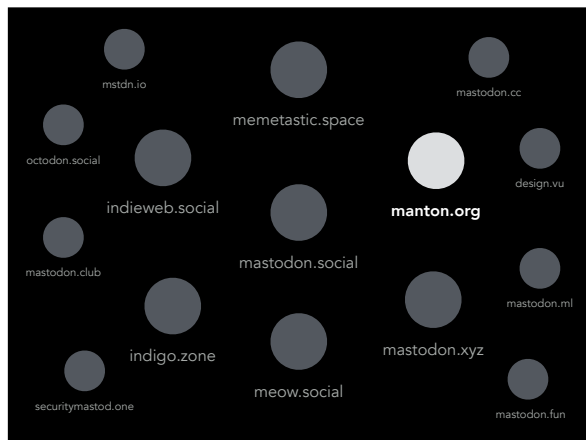
Micro.blog has cross-posting to all these services. [read] Post to your blog first, the post is copied and reformatted to other services automatically. Truncation.



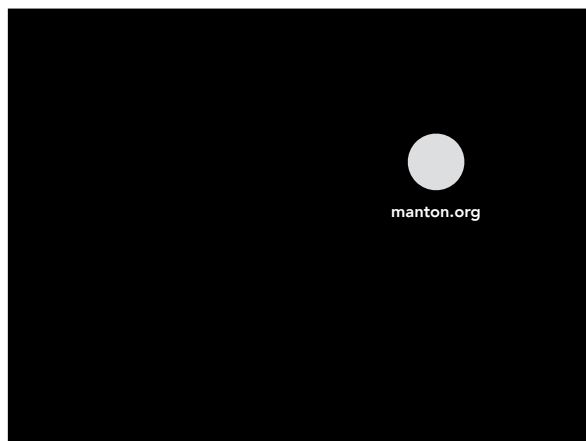
Mastodon again deserves special attention. So in addition to cross-posting, Micro.blog natively supports ActivityPub. ActivityPub is more integrated into the platform, not just a copy of posts with separate accounts on a different platform.



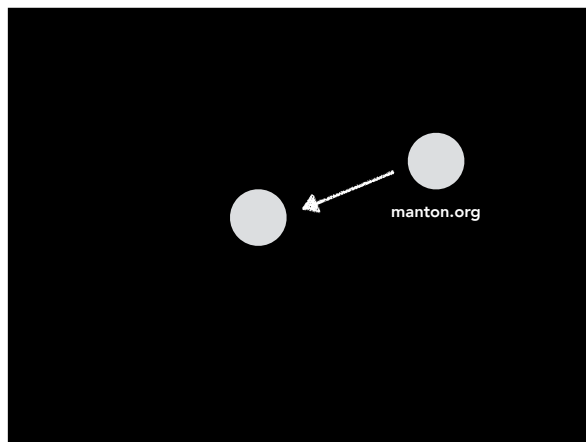
The fediverse is made up a bunch of servers. A dozen people or thousands on each server. Mastodon is very much based around communities.



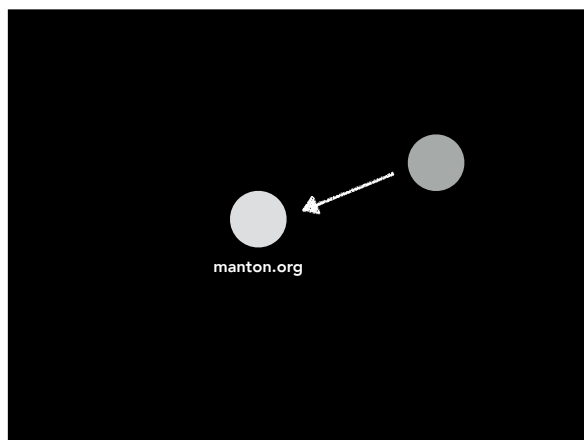
But what I really want is just me. What I really want is a return to how web publishing used to work with domain names for identity.



If my web host goes out of business, or they change their policies and I'm not happy with them anymore, I can move my site and nothing will break.



Find a new place to put my site. A different hosting company.



Point the domain name at the new site. Everything just works. I have done this several times over the 23 years of blogging on my own site.

Personal domain names.

The most important thing to controlling your own content.

Control is not really about running your own server. It's about domain names. Even if you have a Mastodon server and community that you love, you should have a blog at your own domain name too.

@manton@micro.blog

By default everyone on Micro.blog gets a fediverse handle with username at micro.blog, but...

@manton@manton.org

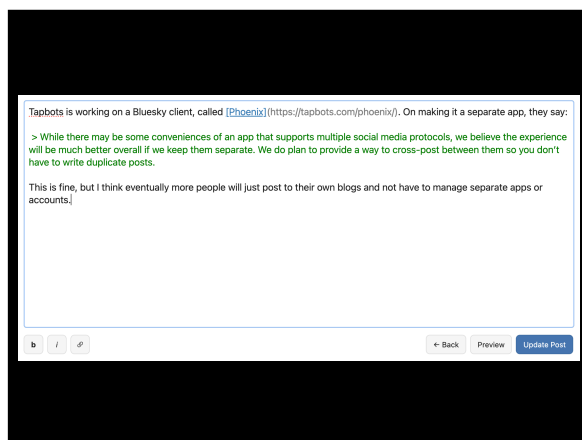
When Micro.blog hosts your blog, your ActivityPub handle can use your own domain name. So Micro.blog is one platform on the fediverse but we can host thousands of domain names.

“The result is a loose federation of documents — **many small pieces loosely joined**. But in what has turned out to be simply the first cultural artifact and institution the Web has subtly subverted, the interior structure of documents has changed, not just the way they are connected to one another. The Web has blown documents apart.”

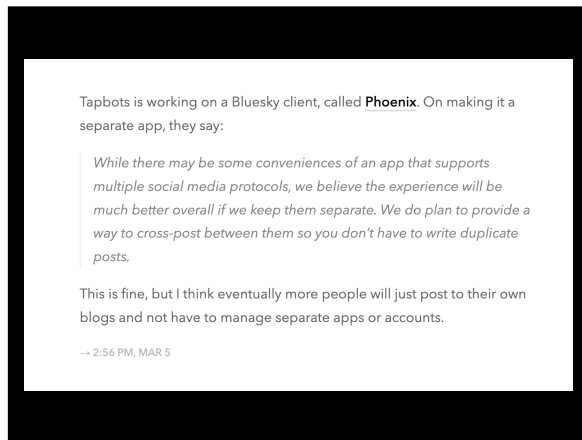
— David Weinberger

Another thing that makes Micro.blog unique is that it’s rooted in the web and HTML.

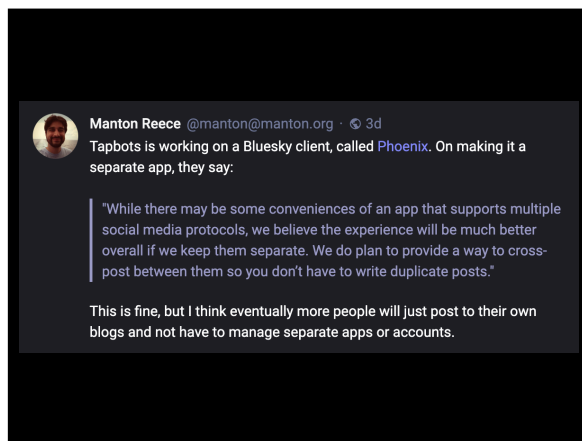
Micro.blog is always about the web first. Hypertext. Links.



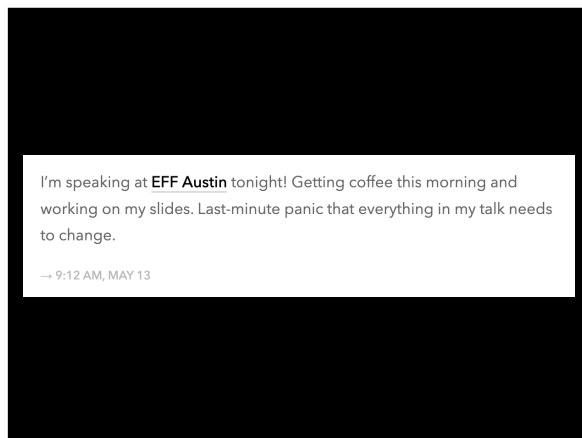
So it looks like this. This is a short post on my blog. Wrote this with Markdown. It has a Markdown link and a block quote. You could also use HTML directly.



This is what that post looks like on my blog. Publishing with Hugo. You could have custom templates. Ends up as static HTML on servers separate from the main Micro.blog platform.



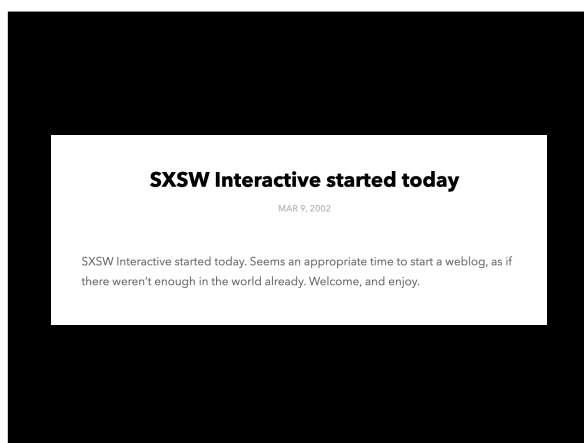
And this is what it looks like via ActivityPub on Mastodon. When there's a choice in how to do something, we do it in a way that is most like the web itself. Links and block quotes and img tags. Not about new data structures. Just web stuff. Photo posts are img tags.



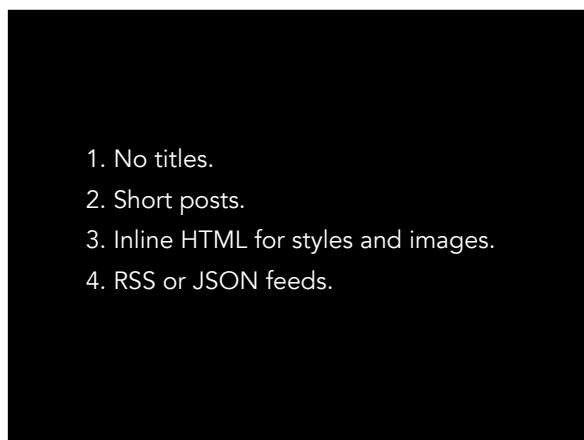
Carry this over to other cross-posted platforms where possible. Here's another short microblog post. This is what it looks like on my blog.



This is a screenshot of the same post on Bluesky. We convert these HTML links to Bluesky facets so the inline links work on Bluesky too.



I want to return to my first blog post. I think in hindsight this old post says a lot. "SXSW started today..." Title wasn't originally there. When migrating posts, something I was using took the first few words and made the title. It's a short post, like something you'd see on X or Threads or Mastodon.

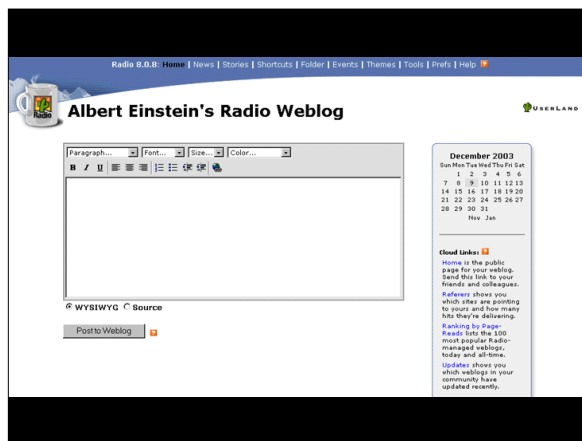


Guidelines. Short posts, 300 characters or less is generally what Micro.blog uses. Don't reinvent existing standards that work perfectly well. The mistake that many Twitter competitors made is inventing some new complicated spec and no one adopts it. Start with blogs because all the basics are built in. No titles. Leave the title blank.

RSS 2.0

“All elements of an item are optional, however at least one of **title** or **description** must be present.”

RSS spec: title is optional. Many modern blog systems assume there will always be a title.



Blogging kind of used to work this way without titles. Radio Userland from 2003, developed at Dave Winer's company. No title field in the UI. Getting back to some of the things that were good about the early web, but updating them for the modern world.

```
{"url": "jsonfeed.org"}
```

JSON Feed Version 1

by Brent Simmons and Manton Reece

The JSON Feed format is a pragmatic syndication format, like [RSS](#) and [Atom](#), but with one big difference: it's JSON instead of XML.

For most developers, JSON is *far* easier to read and write than XML. Developers may groan at picking up an XML parser, but decoding JSON is often just a single line of code.

Our hope is that, because of the lightness of JSON and simplicity of the JSON Feed format, developers will be more attracted to developing for the open web.

Here's a simple example:

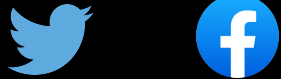
```
{
  "version": "https://jsonfeed.org/version/1",
  "title": "My Example Feed",
  "home_page_url": "https://example.org/",
  "feed_url": "https://example.org/feed.json",
  "items": [
    {
      "id": "2",
```

We can make it even more obvious for microblogging. A few years ago, Brent Simmons (who developed the feed reader NetNewsWire) approached me about whether the time was right for a modern version of RSS based on JSON. The result was JSON Feed.

JSON Feed

"Microblogs, which are often plain text and without titles. So much web writing today is Twitter-like, which is actually plain text."

We wanted to call out microblogs specifically. [\[read\]](#)



On the opposite side of indie microblogging are ad-based platforms. Twitter is this kind of walled garden. Tweets only exist at Twitter.com URLs. If Twitter bans you from their platform, your tweets are gone. The URLs break. You can't take tweets anywhere else like you can with posts at your own domain name. Because of this, Twitter almost never removes anyone's account, even when they should.

Meta envisages social media filled with AI-generated users

"Meta is betting that **characters generated by artificial intelligence will fill its social media platforms** in the next few years as it looks to the fast-developing technology to drive engagement with its 3bn users."

And Meta's vision of the future is quite dismal. Article about Meta in the Financial Times. [\[read\]](#) In the age of AI, content will be abundant. Ad-based platforms feed off abundance, printing money faster as they fill ad inventory. There can never be too much content for algorithmic timelines — more data to rank by engagement, more data to funnel through the outrage machine to see what sticks — so algorithmic timelines will always trend toward AI slop. It's infinite content, some AI generated, using up all of our attention.

"They're blind to the voices silenced by hate and intimidation. The voices that need to be protected. **But anyone who's ever tended a garden knows that for the good stuff to grow, you have to deal with the bad stuff.** You can't let the weeds choke the vegetables. You'll go hungry."

— Mike Monteiro

With everything driven by algorithms, there will be less human oversight. Mike Monteiro is someone who had been really critical of Twitter leadership years ago. He wrote about Twitter that... [read]

Walled gardens.

Walled gardens with closed APIs — like Facebook, Twitter, Instagram, even the App Store — usually become developer-hostile platforms. This is what was frustrating to me about Twitter going back to those first few slides. Twitter and now X is not just closed but it's a walled garden that is full of weeds. [click]

Walled gardens.

Now imagine instead a platform based on blogs, with all the open formats and protocols I've been talking about. Because you post to your own domain name, your writing is independent of any one platform.

Open gardens.

This isn't a walled garden. It's an open garden. The flowers and plants on the inside of the garden are the same flowers that grow outside. The soil is the same, whether your blog is hosted on Micro.blog or it's hosted on WordPress or Ghost. It's made of the same web stuff.

Tweets can only exist natively on Twitter / X. But a microblog post is the same everywhere.

A tweet can only exist natively on x.com. But a microblog post is the same everywhere, inside and outside the garden.

Tweets can only exist natively on Twitter / X. But a microblog post is the same everywhere.

And so that gives us the freedom to pull the weeds in Micro.blog. We can curate the social network. We can tend the garden, knowing people have choices for where to post.

Freedom to pull the weeds.



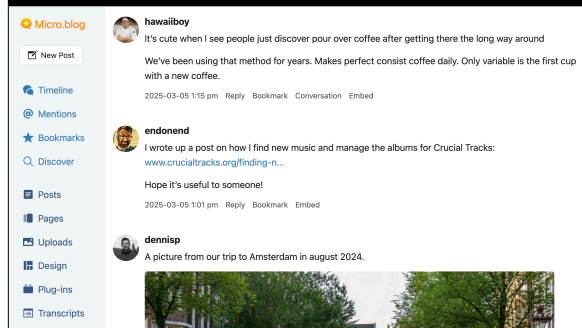
Leaving features out

- No follower counts
- No algorithms
- No likes
- No ads

Micro.blog tries to do a few things a little differently. Not copying all features from Twitter and other big platforms. Not a popularity contest. [read] Ad-based platforms have incentives for engagement. Our business isn't ads so we don't need engagement. Slower, quieter space on the web. With posts written by humans.



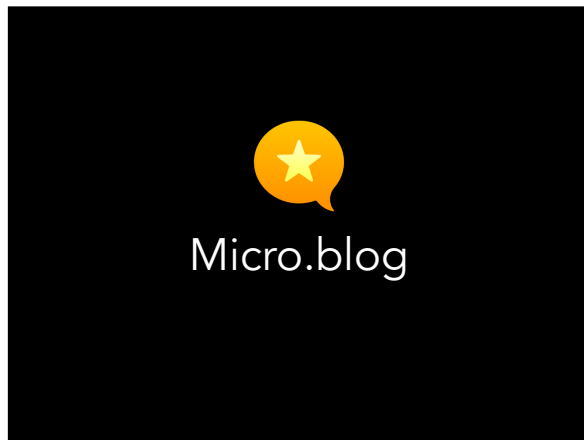
Curated discovery



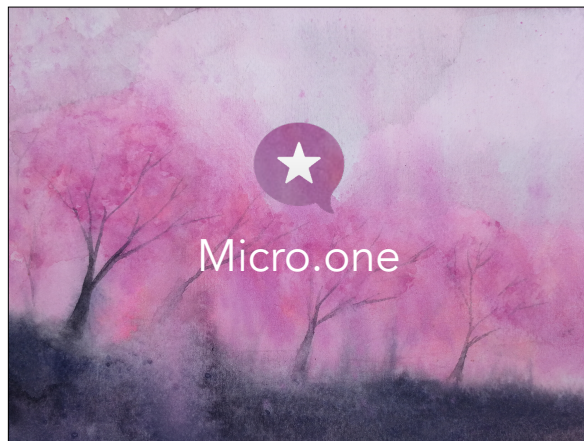
Screenshot of the Discover page in Micro.blog. We manually curate this page, highlighting posts to help people discover new users to follow.



Every platform can make their own decisions around features and curation. There is a growing list of web-friendly, blog-based platforms that support ActivityPub. WordPress and Ghost. The social side pioneered by Mastodon, rethinking social for smaller platforms, and we're meeting in the middle with traditional blogs. Social can emphasize conversations. Blogs can emphasize permanence and your own posts.



So that's what we're trying to do with Micro.blog. Trying to create a quieter, maybe slower corner of the open web, centered around blog hosting and still connecting to other platforms.



Also earlier this year, launched Micro.one, a special subscription that's only \$1/month. Blog hosting, ActivityPub at your own domain name.



That's it, thanks! Questions?