SPAs Without Javascript

Simplify Your Stack

Brian Meeker

Sr. Software Engineer at GetThru

Create a pet

https://pet-mash.fly.dev

Brian Meeker

Website

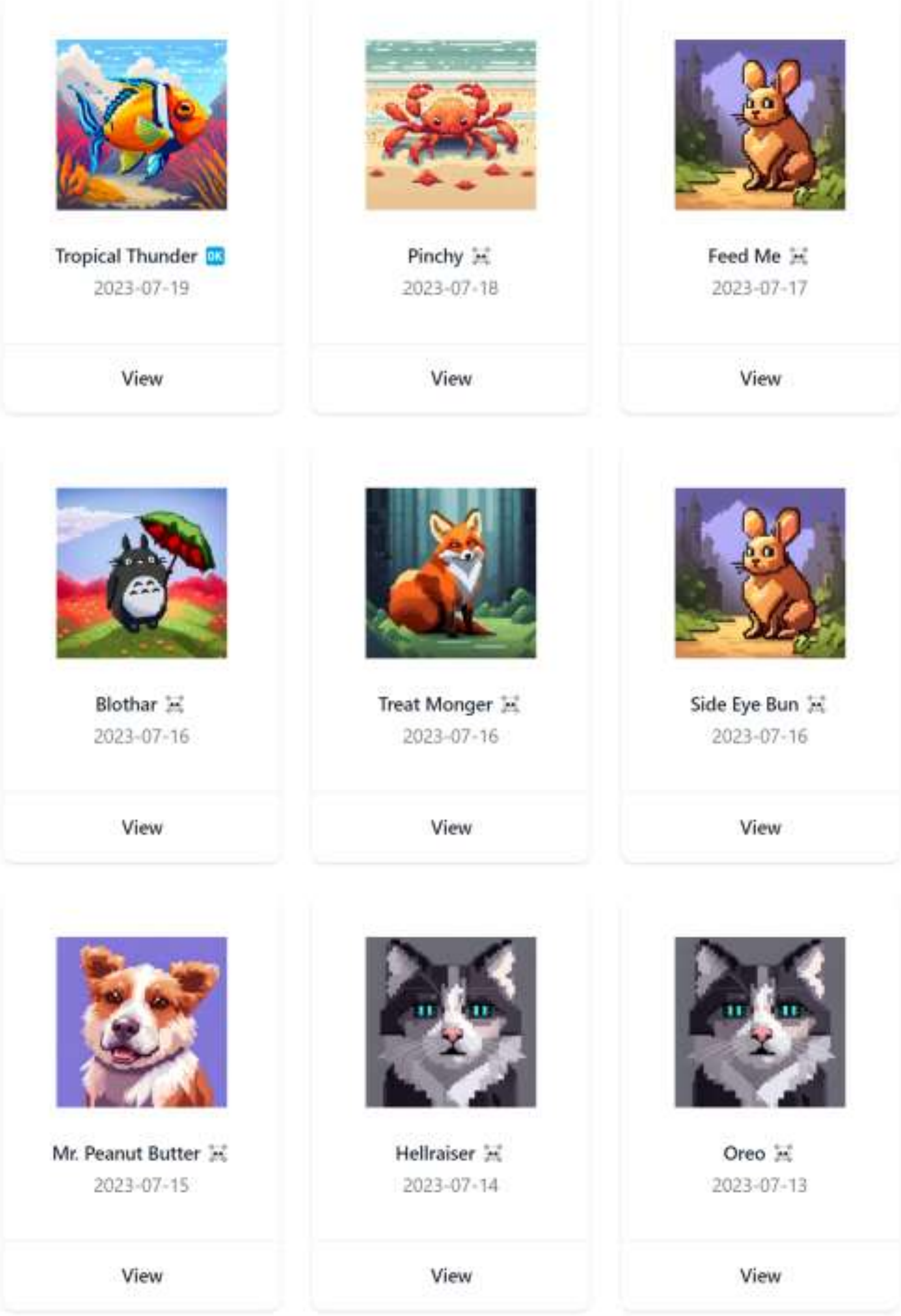
- https://brianmeeker.me

GitHub

- @CuriousCurmudgeon

Bluesky

- @brianmeeker.bsky.social



Tropical Thunder 🆗
2023-07-19

Pinchy ☠
2023-07-18

Feed Me ☠
2023-07-17

Blothar ☠
2023-07-16

Treat Monger ☠
2023-07-16

Side Eye Bun ☠
2023-07-16

Mr. Peanut Butter ☠
2023-07-15

Hellraiser ☠
2023-07-14

Oreo ☠
2023-07-13

Agenda

- □ Hotwire
- □ HTMX
- □ Phoenix LiveView Deep Dive

Goal

You Build Something

Every Speaker Is Biased

Your Tech Stack Is Secondary

What Is a SPA?

Do You Even Need a SPA?

Answer Me These Questions Three

1. Is your core functionality real-time?
2. Are rich UI interactions core to your product?
3. Is there lots of shared state between screens?

Frameworks

- Hotwire (Ruby)
- HTMX (Bring your own back end)
- Phoenix LiveView (Elixir)
- ~~Blazor (C#)~~
- ~~LiveWire (PHP)~~

JavaScript is fine

Context switching is not

Write Only Necessary JS

HOTWIRE

HTML Over The Wire

RAILS

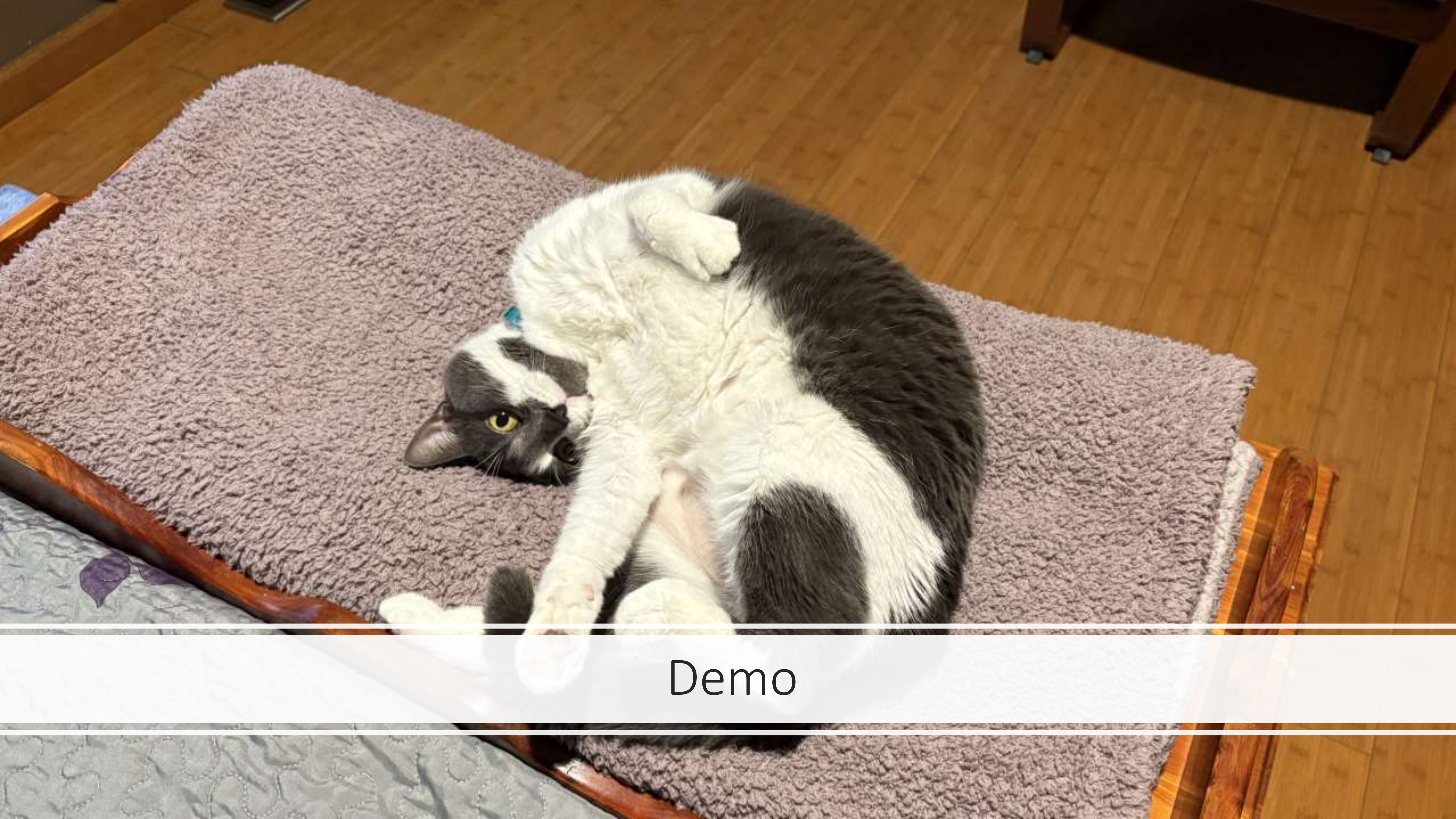
Wasn’t Turbo Already a Thing?

StimulusReflex

Turbo Drive

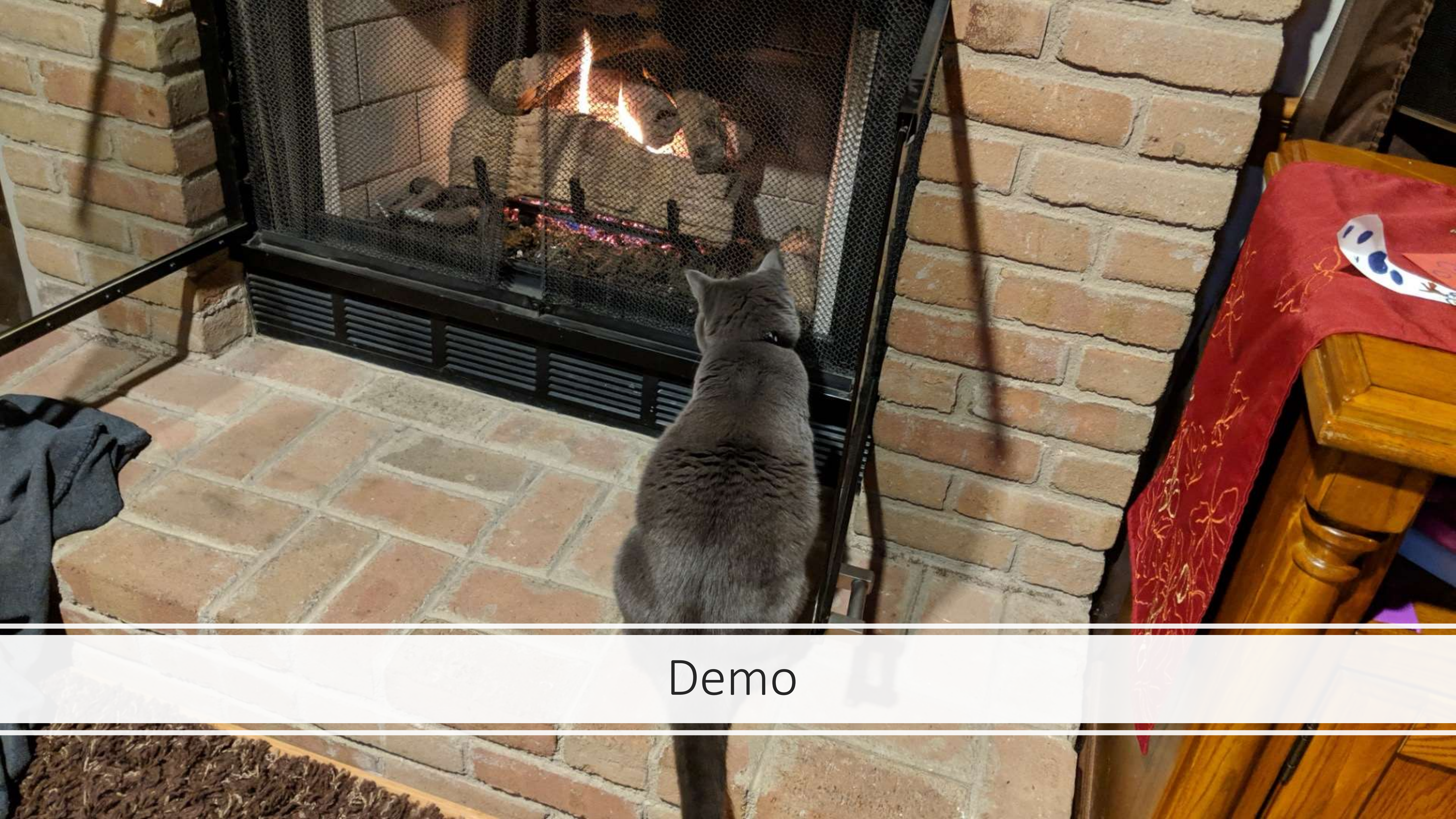
Turbo Frames

Turbo Streams



Demo

</> htmx



Demo

Phoenix Framework

The Elixir programming language wraps functional programming with immutable state and an actor-based approach to concurrency in a tidy, modern syntax. And it runs on the industrial strength, high performance, distributed Erlang VM

- Dave Thomas - Programming Elixir

LET'S SEE WHO YOU REALLY ARE!
IT WAS OLD MAN ERLANG ALL ALONG!
imgflip.com

The Actor Model

In the Erlang VM, all code runs in tiny concurrent processes, each with its own state. Processes talk to each other via messages. And since all communication happens by message-passing, exchanging messages between different machines on the same network is handled transparently by the VM, making it a perfect environment for building distributed software.

- José Valim - Programming Elixir foreword

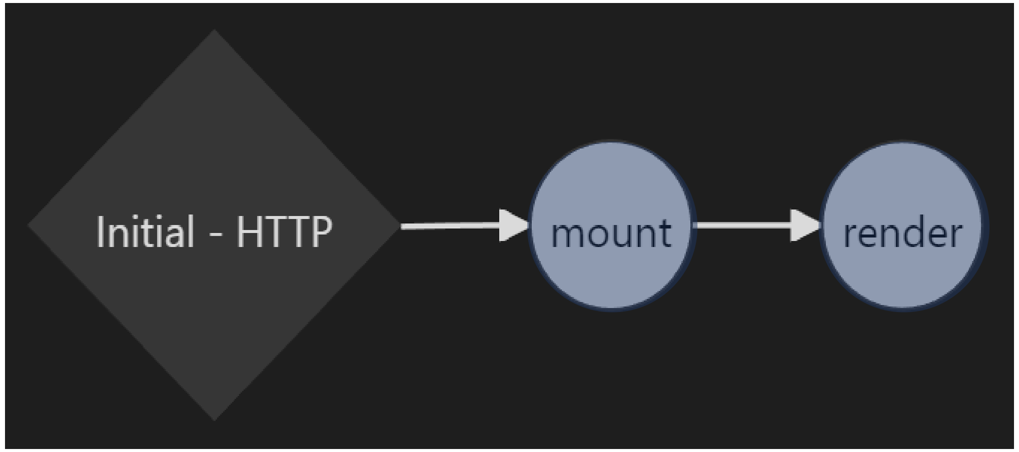
LiveView

“A LiveView is a process that receives events, updates its state, and renders updates to a page as diffs.”

- Phoenix.LiveView Documentation

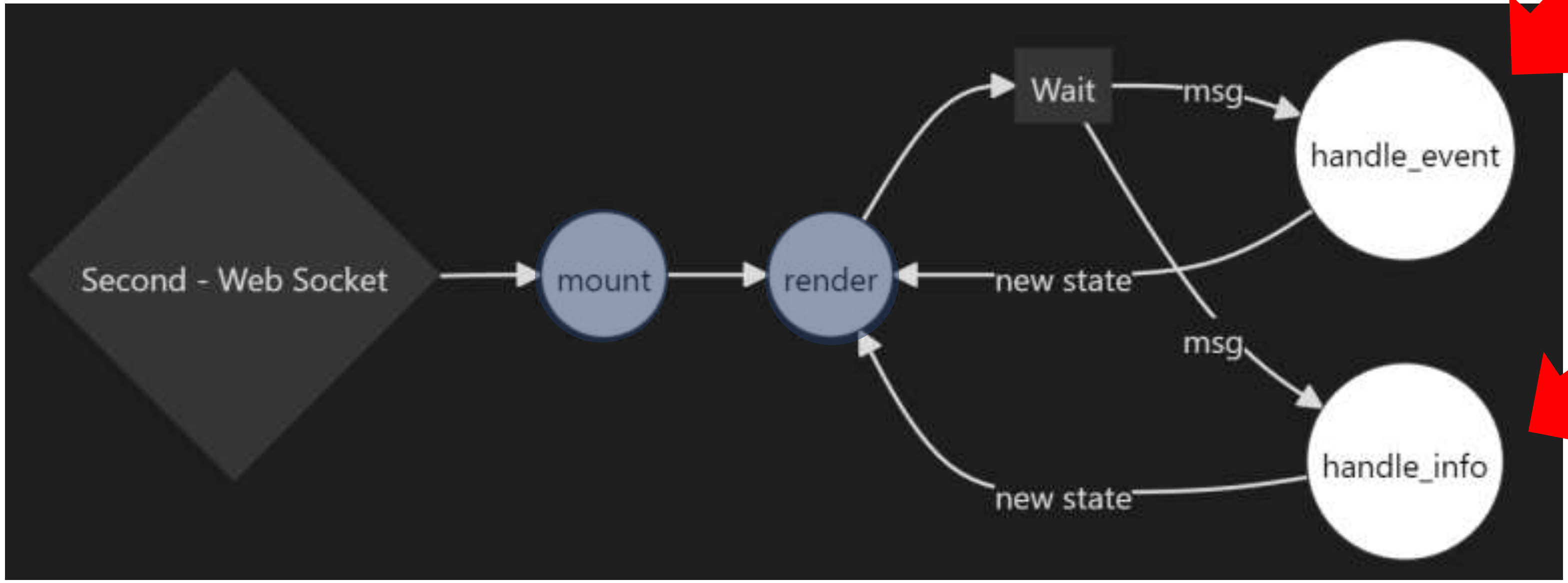
LiveView Lifecycle

Initial GET request



From client over web socket

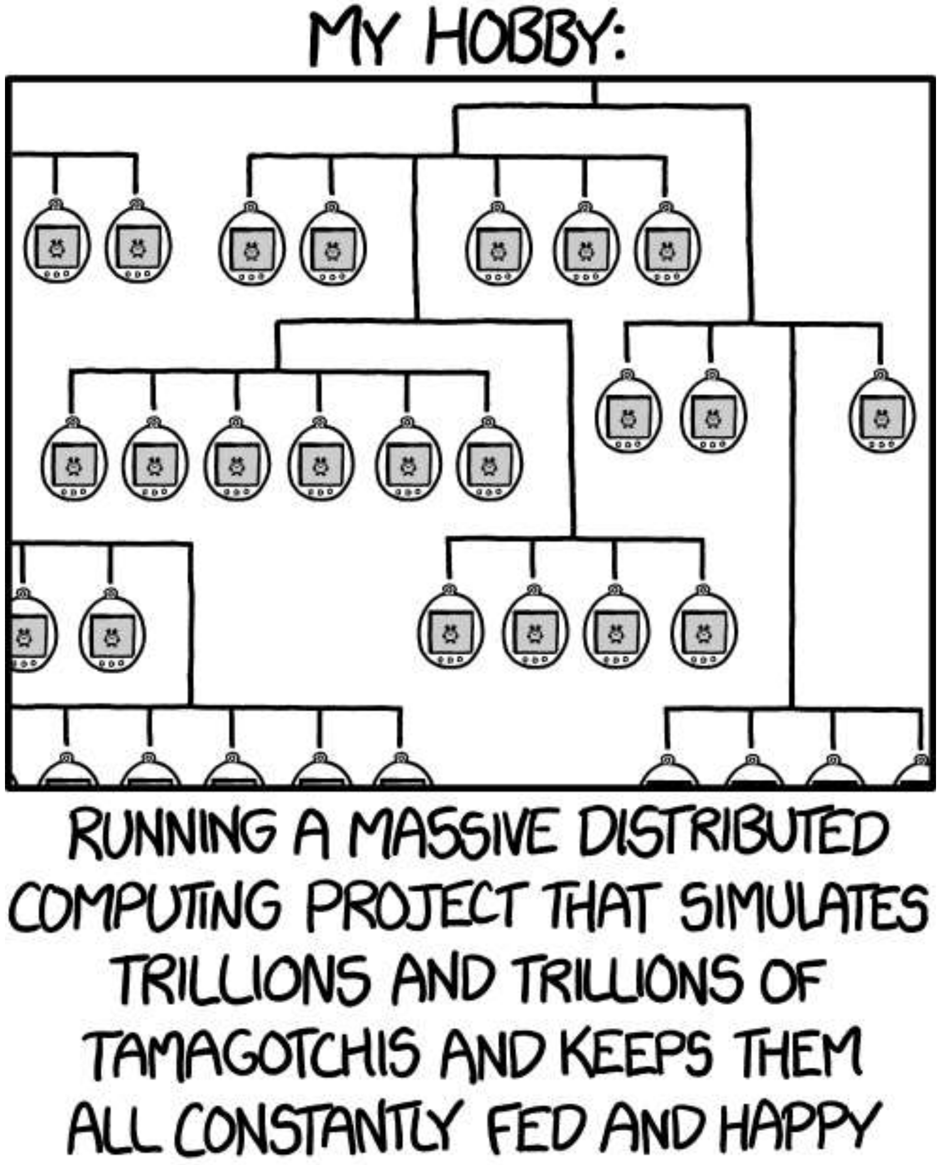
Web Socket



From other process

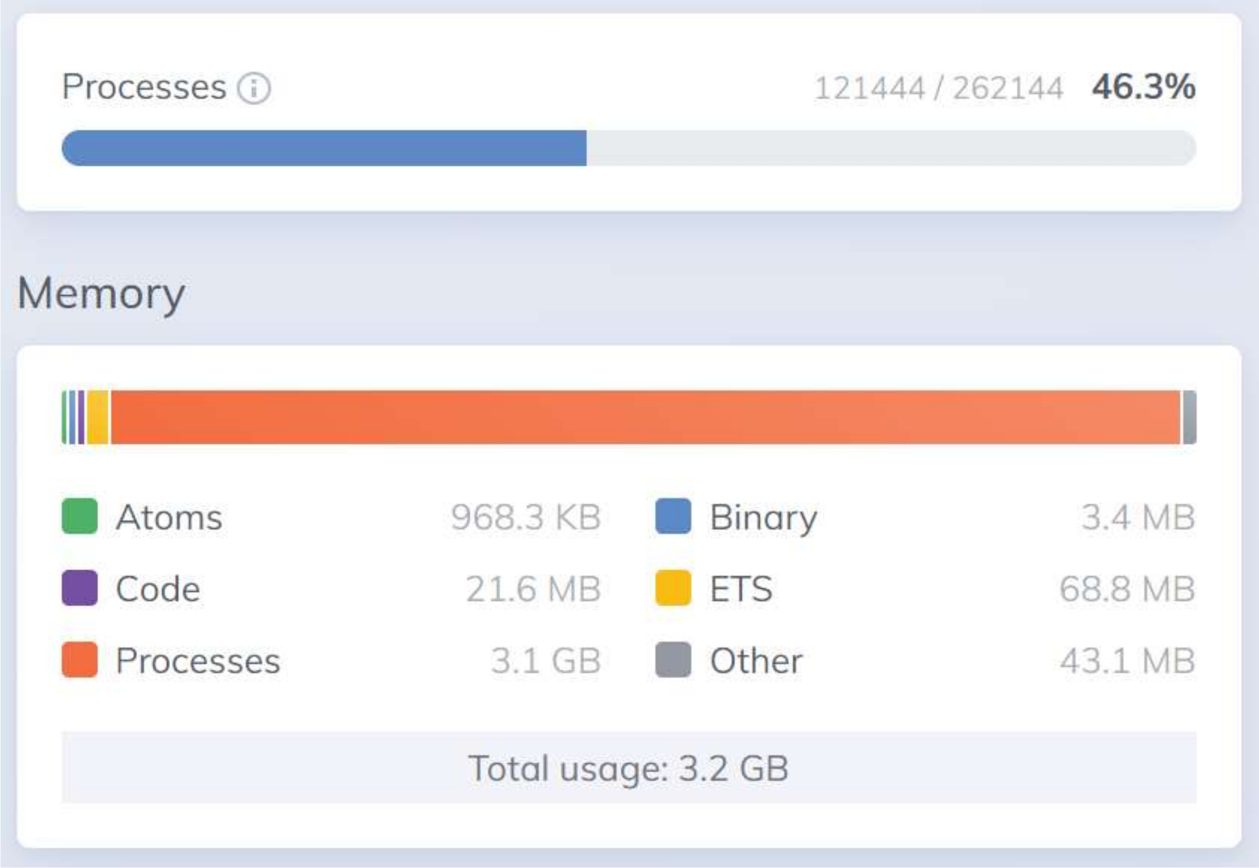
Live Components

Function Components


MY HOBBY:
RUNNING A MASSIVE DISTRIBUTED COMPUTING PROJECT THAT SIMULATES TRILLIONS AND TRILLIONS OF TAMAGOTCHIS AND KEEPS THEM ALL CONSTANTLY FED AND HAPPY

https://xkcd.com/1546/

Scalability



Processes ⓘ 121444 / 262144 **46.3%**

Memory

🟩	Atoms	968.3 KB	🟦	Binary	3.4 MB
🟪	Code	21.6 MB	🟨	ETS	68.8 MB
🟧	Processes	3.1 GB	⬜	Other	43.1 MB

Total usage: 3.2 GB

Demos

LiveView Limitations

- Memory pressure at scale
- Requires an always-on connection
- Latency
- JS hooks can get messy

Website

https://brianmeeker.me

GitHub

https://github.com/CuriousCurmudgeon/live_pet

Bluesky

- @brianmeeker.bsky.social

https://github.com/CuriousCurmudgeon/spas_without_js_talk