Hello, Joe!

Gleam:
Past, Present, Future!

louis@gleam.run
twitter.com/louispilfold
github.com/sponsors/lpil

What's Gleam?

Easy to learn & read

Small & consistent

Type safe

Great tooling

Familiar syntax

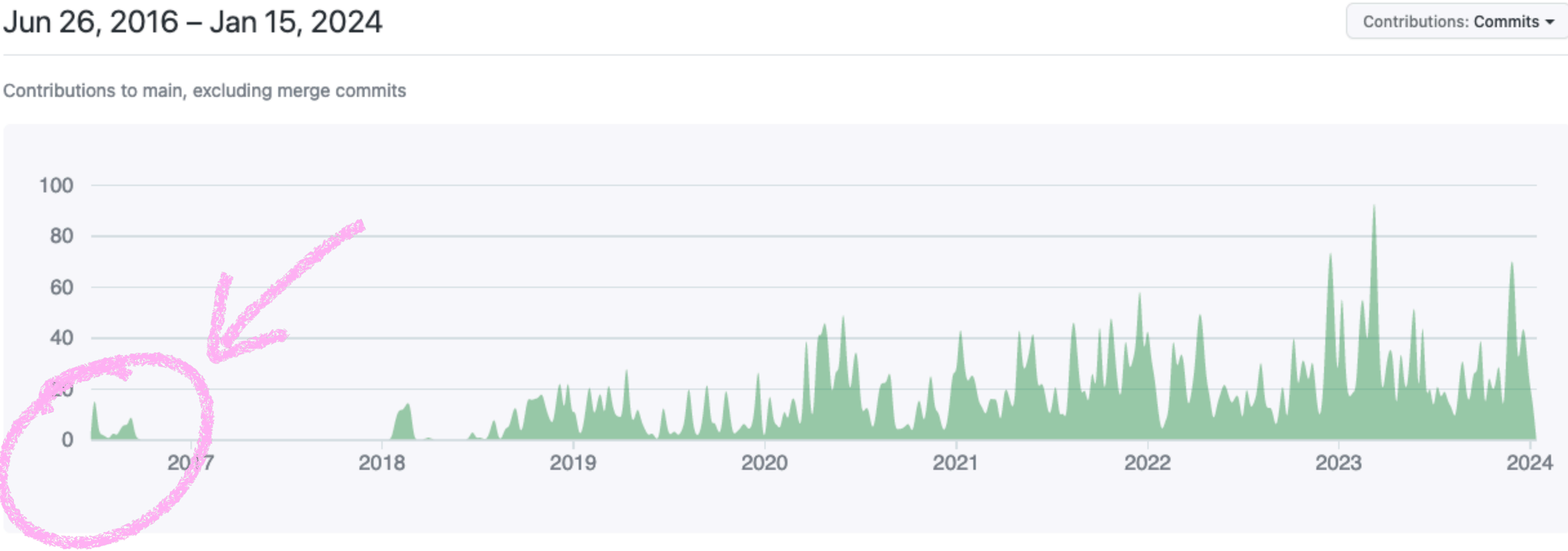
BEAM or JS

```
import gleam/io

pub fn main() {
  io.println("hello, friend!")
}
```

Gleam: Past

How'd we get here?

A history of Gleam



Mid 2017

The very first Gleam

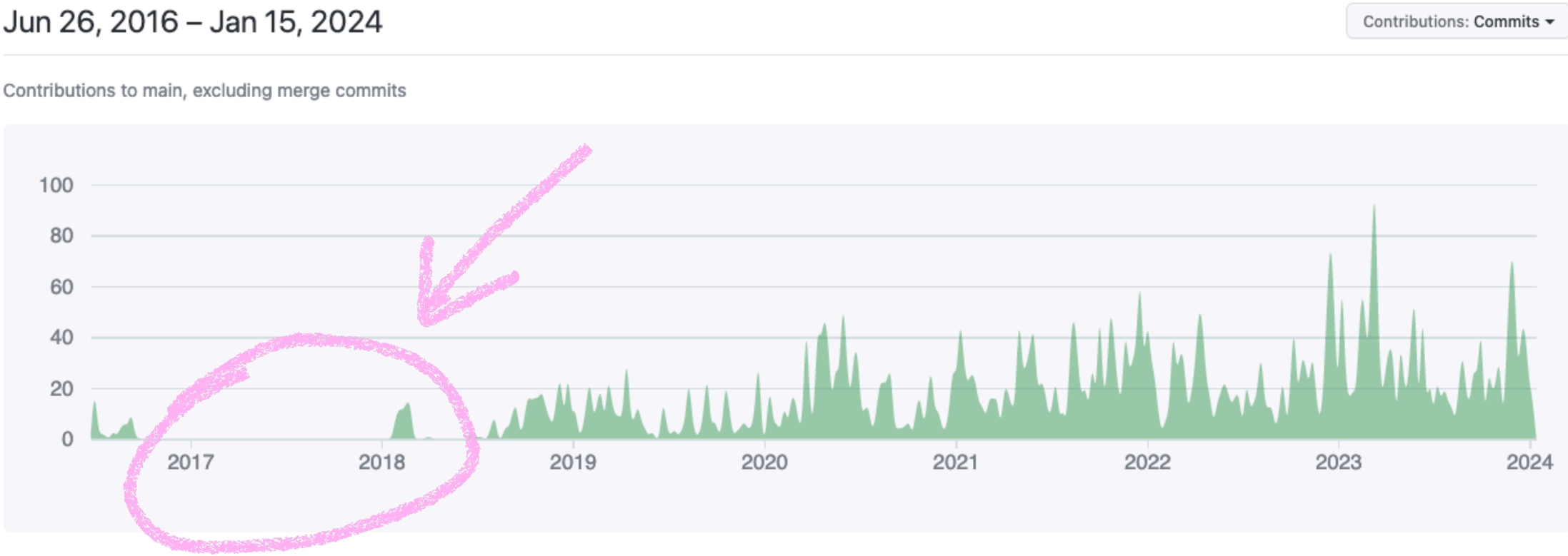
```
module clauses

public count {
  def ([])        { 0 }
  def ([_ | xs]) { count(xs) + 1 }

  test([]       ) { 0 }
  test([1, 2]   ) { 2 }
  test([1, 1, 1]) { 3 }
}
```

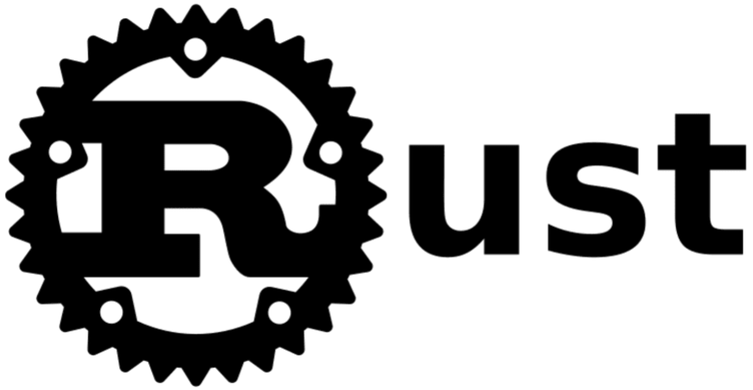
Other fun features

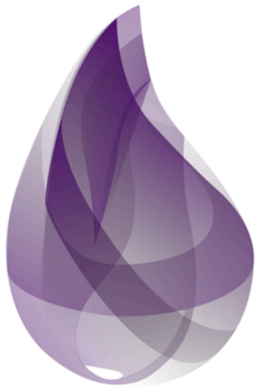
- No type system!
- No real goals!
- Didn't really work!
- A bad Erlang clone in a trenchcoat!

What was the point?

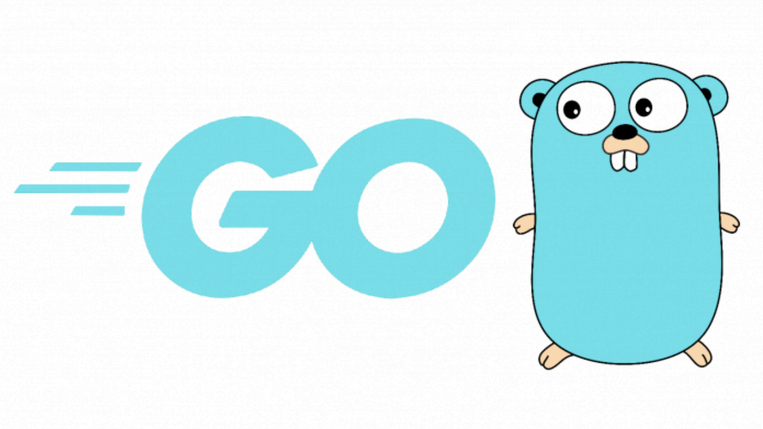
A history of Gleam







elixir



February 2018

```
module Main
export User, greet/1
from IO import println

type User =
  | LoggedIn(String)
  | Guest

spec |String| ⇒ Nil
fn check(user) =
  case user
  | LoggedIn("Al") ⇒ println("Hi Al!")
  | LoggedIn(name) ⇒ println("Welcome back!")
  | Guest ⇒ println("Hello! Please log in")
```

```
import gleam/io

pub enum User =
  | LoggedIn(String)
  | Guest


pub fn check(user: User) {
  case user {
  | LoggedIn("Al") → io:println("Hi Al!")
  | LoggedIn(name) → io:println("Welcome back!")
  | Guest → io:println("Hello! Please log in")
  }
}
```

v0.6 - December 2019

```
import gleam/io

pub type User {
  LoggedIn(name: String)
  Guest
}

pub fn check(user: User) {
  case user {
    LoggedIn("Al") → io.println("Hi Al!")
    LoggedIn(name) → io.println("Welcome back!")
    Guest → io.println("Hello! Please log in")
  }
}
```

Language changes

First class modules

Row typed records

Dedicated enum syntax

Labelled arguments

"use" expression sugar

String concatenation operator

JavaScript target

The build tool

Batteries included

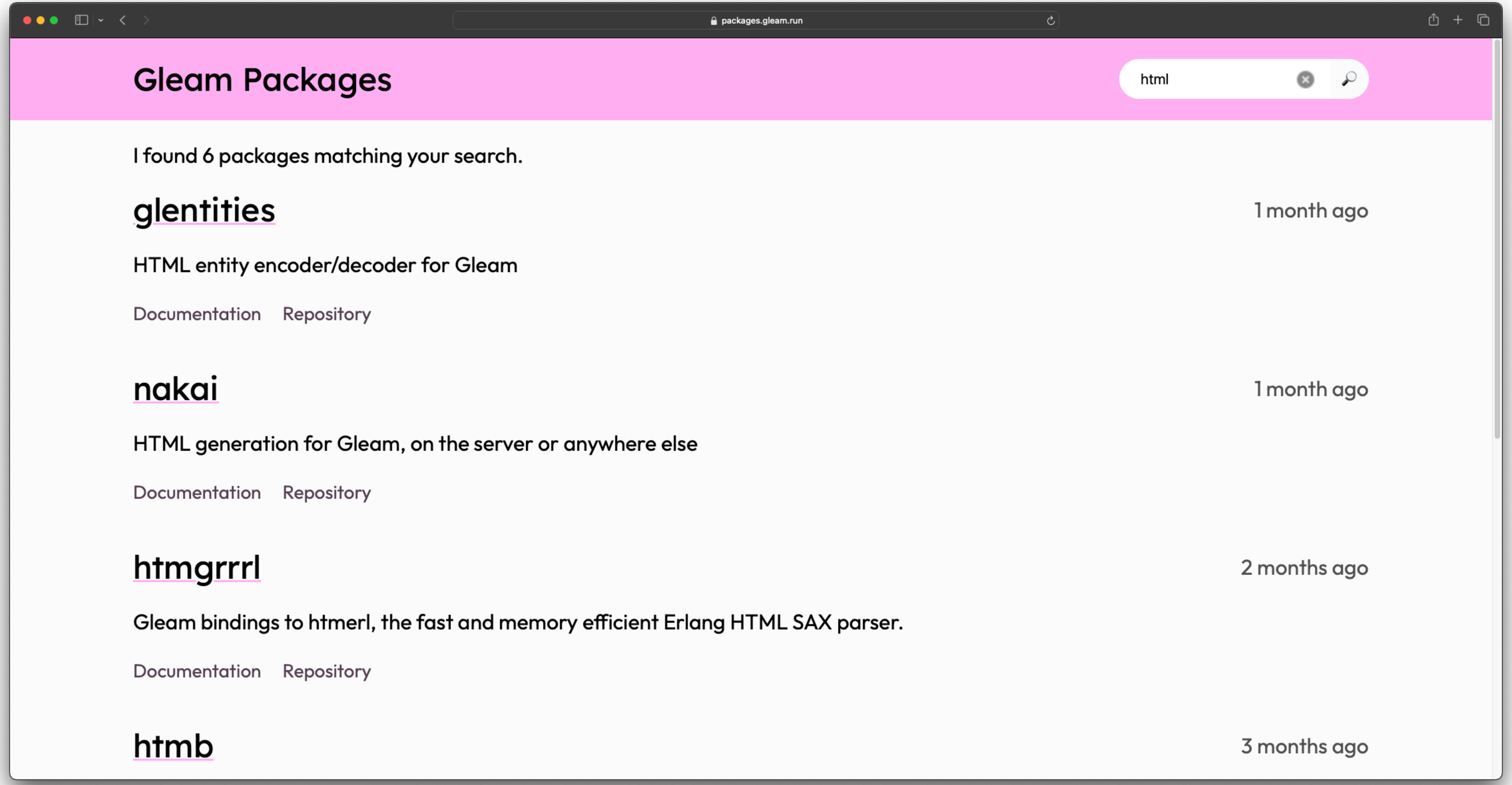
Easy to use

Hex package management

Code formatter

Language server

```
$ gleam test
  Resolving versions
Downloading packages
 Downloaded 2 packages in 0.02s
  Compiling gleam_stdlib
  Compiling gleeunit
  Compiling justin
   Compiled in 1.20s
    Running justin_test.main
.....
Finished in 0.018 seconds
5 tests, 0 failures
```


packages.gleam.run
Gleam Packages
html
I found 6 packages matching your search.
glentities
1 month ago
HTML entity encoder/decoder for Gleam
Documentation Repository
nakai
1 month ago
HTML generation for Gleam, on the server or anywhere else
Documentation Repository
htmgrrrl
2 months ago
Gleam bindings to htmerl, the fast and memory efficient Erlang HTML SAX parser.
Documentation Repository
htmb
3 months ago

simov.github.io

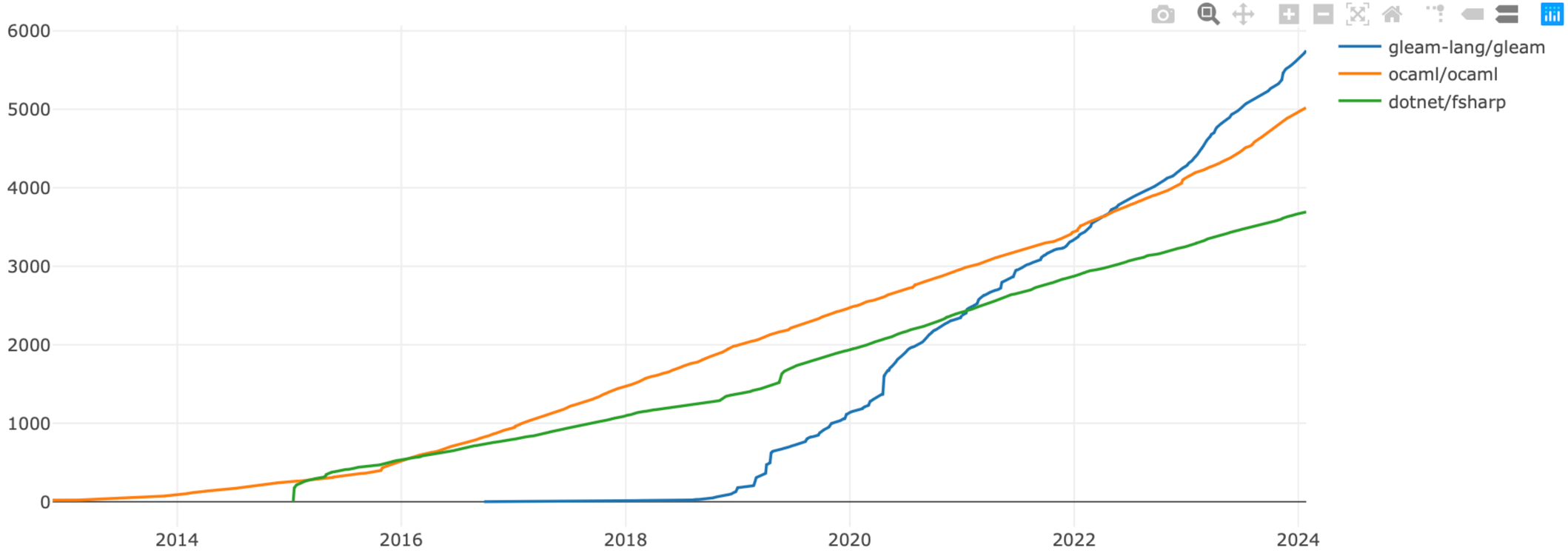
★ GitHub Stars

4697

user/repo

add

export



gleam-lang/gleam | ocaml/ocaml | dotnet/fsharp

exercism.org

Learn Discover Contribute More

Sign up Log in

Gleam

982 students

30 contributors
30 mentors

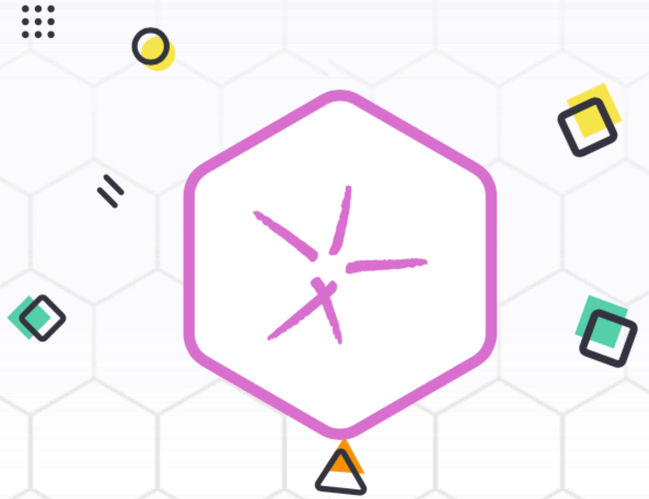
About Gleam Learn Practice

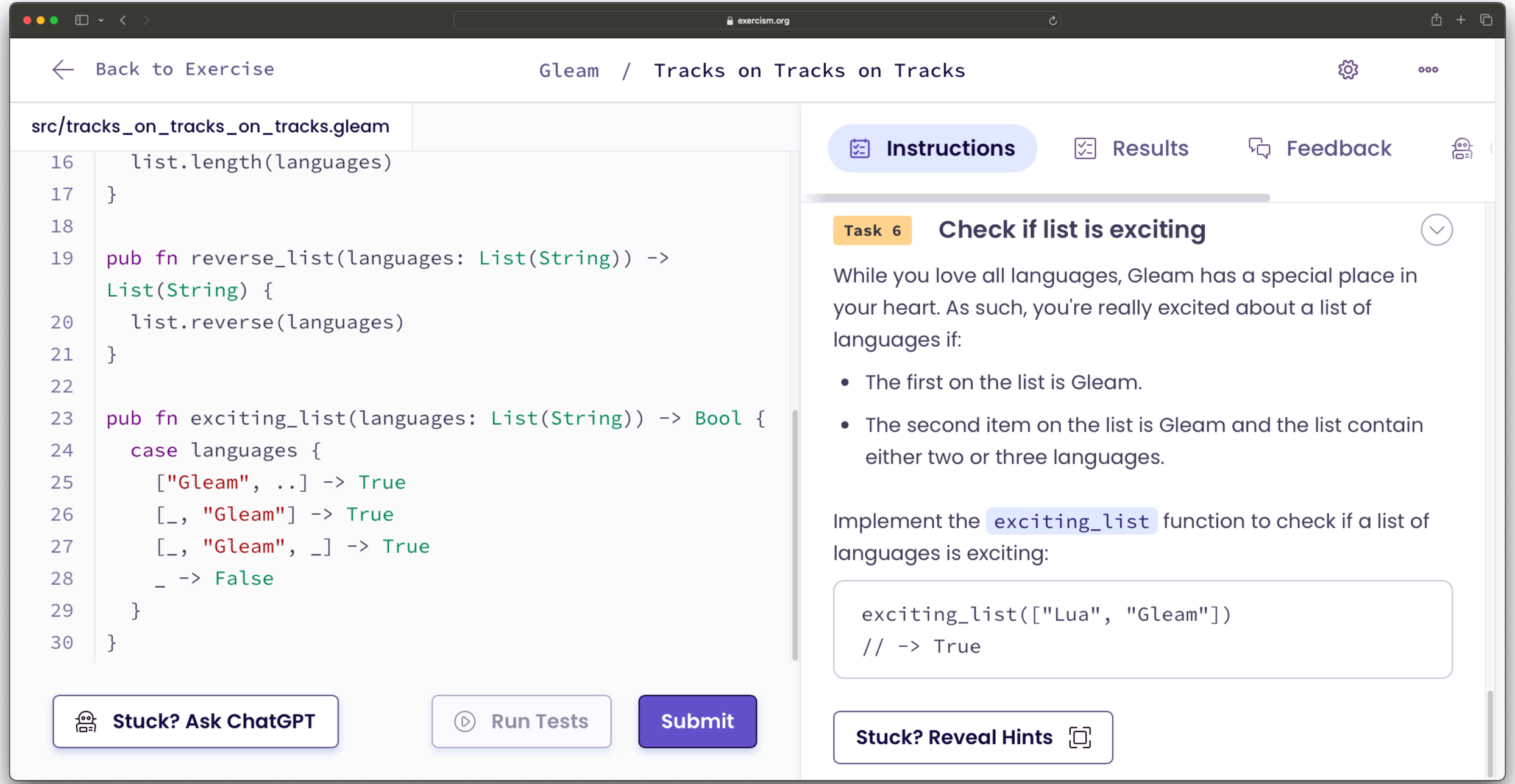
Want to learn and master Gleam?

Join Exercism's Gleam Track for access to **124 exercises** grouped into 36 Gleam Concepts, with automatic analysis of your code and **personal mentoring**, all **100% free.**

Join the Gleam Track

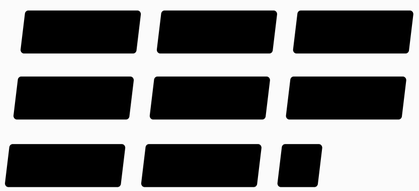
Explore concepts



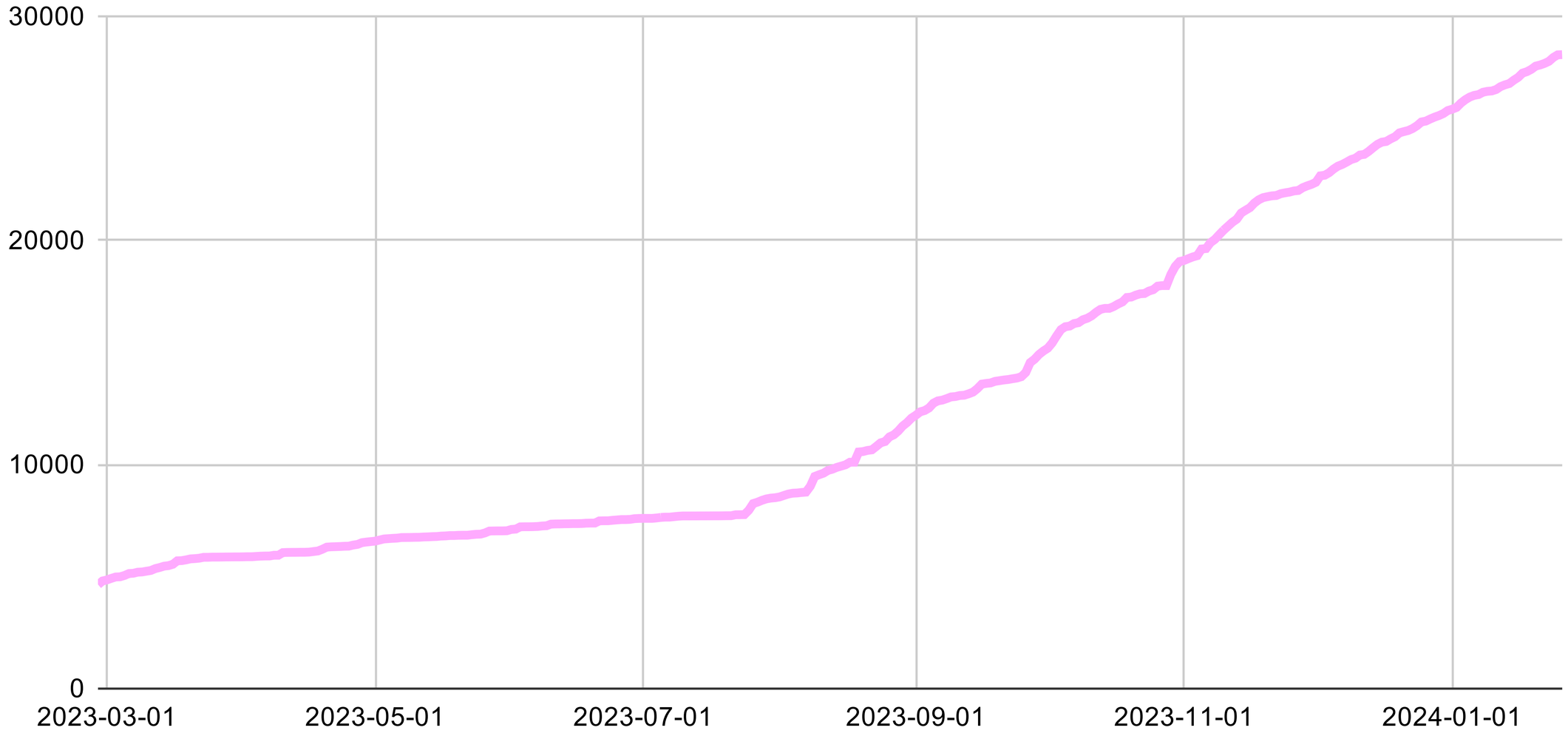

exercism.org
Back to Exercise
Gleam / Tracks on Tracks on Tracks
src/tracks_on_tracks_on_tracks.gleam
16 list.length(languages)
17 }
18
19 pub fn reverse_list(languages: List(String)) ->
List(String) {
20 list.reverse(languages)
21 }
22
23 pub fn exciting_list(languages: List(String)) -> Bool {
24 case languages {
25 ["Gleam", ..] -> True
26 [_, "Gleam"] -> True
27 [_, "Gleam", _] -> True
28 _ -> False
29 }
30 }
Stuck? Ask ChatGPT
Run Tests
Submit
Instructions
Results
Feedback
Task 6
Check if list is exciting
While you love all languages, Gleam has a special place in your heart. As such, you're really excited about a list of languages if:
The first on the list is Gleam.
The second item on the list is Gleam and the list contain either two or three languages.
Implement the exciting_list function to check if a list of languages is exciting:
exciting_list(["Lua", "Gleam"])
// -> True
Stuck? Reveal Hints

Your journey through Gleam

Learn and master concepts to achieve fluency in Gleam.



Exercism submissions



Higher order functions

In Gleam functions are values. They can be assigned to variables, passed to other functions, and anything else you can do with values.

Here the function `add_one` is being passed as an argument to the `twice` function.

Notice the `fn` keyword is also used to describe the type of the function that `twice` takes as its second argument.

Back — Contents — Next

```
import gleam/io

pub fn main() {
  // Call a function with another function
  io.debug(twice(1, add_one))

  // Functions can be assigned to variables
  let function = add_one
  io.debug(function(100))
}

fn twice(argument: Int, function: fn(Int) -> Int) -> Int {
  function(function(argument))
}
```

```
3
101
```

Gleam: Present

Where are we now?

The Gleam Programmin

1 Event
Browse Channels
Members

rules
moderator-only
announcements

MODERATION
gleam-team

GLEAM
general
sharing 1
teashop
codegen

lpil
Online

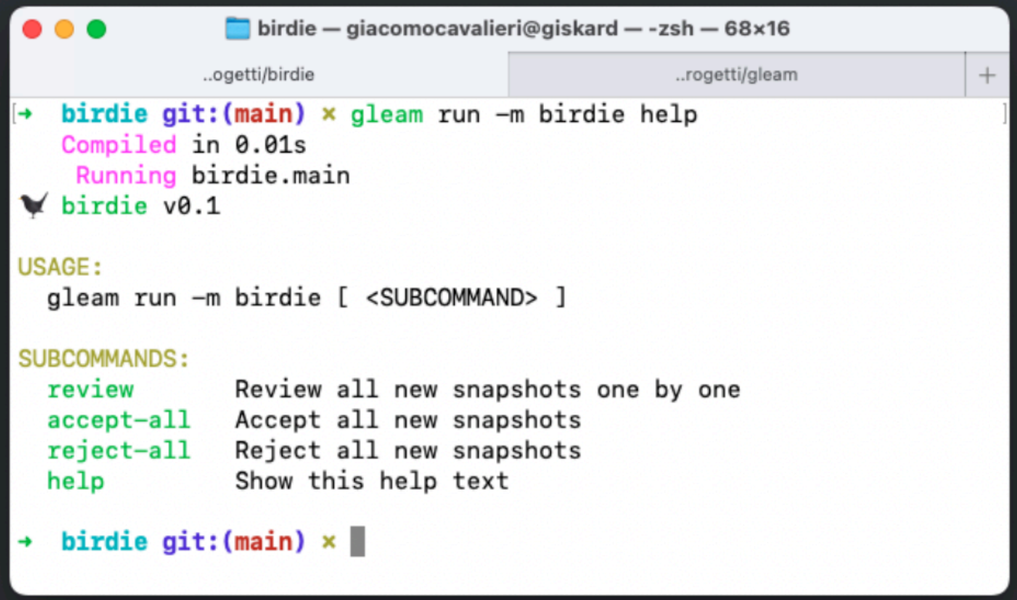
general | A place to talk about Gleam

Search

@lpil that's totally what I wanted when I share an SVG

inoas Today at 15:11
they are afraid if svgs eh 😉

Jak Today at 15:13
It's looking good! I'm just missing the interactive review bit and I'll publish v1



Message #general

I 💖 the Gleam community!

They're super smart and twice as nice

github.com/rawhat/mist

Pure Gleam

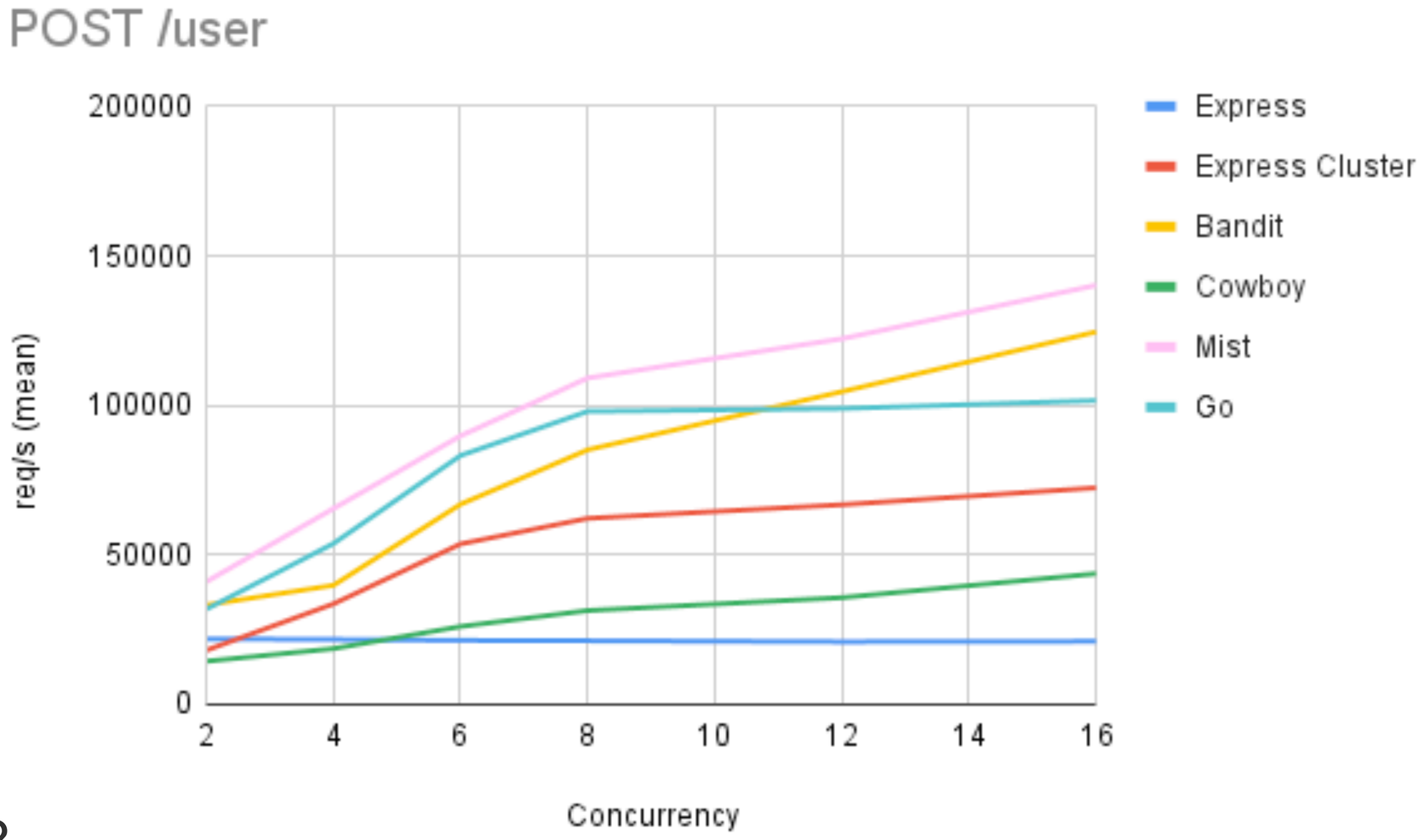
Type safe

HTTP1.1

WebSockets

HTTP or HTTPS

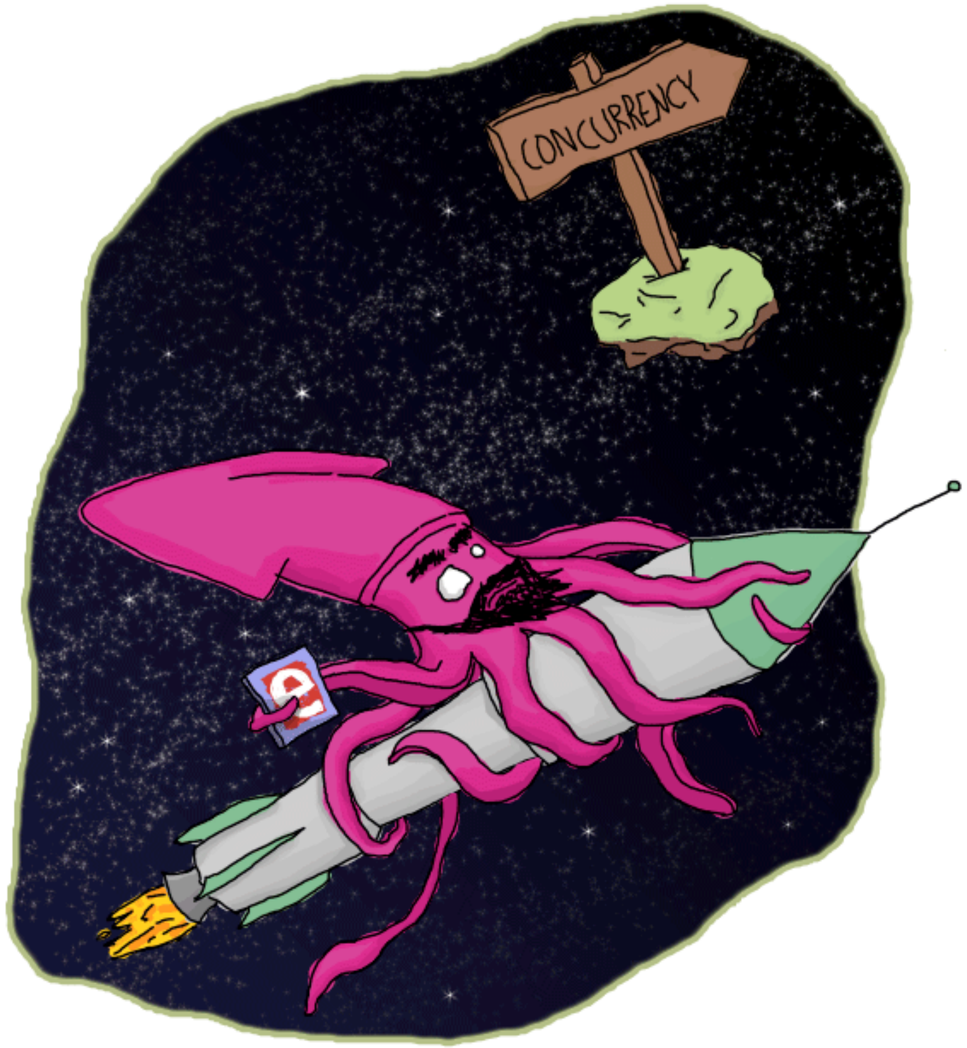
Super fast



Maybe HTTP2 in future?

github.com/gleam-lang/otp

```
actor.start("init", fn(msg, state) {
  case msg {
    Set(new_state) → {
      actor.continue(new_state)
    }
    Get(caller) → {
      process.send(caller, state)
      actor.continue(state)
    }
  }
})
```



github.com/gleam-wisp/wisp

```gleam
pub fn handle_request(req: Request) -> Response {
  use form <- wisp.require_form(req)

  let result = {
    use name <- try(list.key_find(form.values, "name"))
    Ok("<h1>Hi, " <> wisp.escape_html(name) <> "!</h1>")
  }

  case result {
    Ok(content) -> wisp.html_response(from_string(content), 200)
    Error(_) -> wisp.bad_request()
  }
}
```

Databases

github.com/lpil/pgo

github.com/lpil/sqlight

github.com/massivefermion/mungo

github.com/massivefermion/radish

github.com/lustre-labs/lustre

```
fn view(model) {
  let count = int.to_string(model)

  div([], [
    button([on_click(Decr)], [text("-")]),
    p([], [text(count)]),
    button([on_click(Incr)], [text("+")])
  ])
}
```

```
type Msg {
  Incr
  Decr
}

fn update(model, msg) {
  case msg {
    Incr → model + 1
    Decr → model - 1
  }
}
```

Gleam LiveView?

github.com/lustre-labs/lustre

Lustre 🤝 Sprocket

```
let assert Ok(app) =
  lustre.component(
    app.init,
    app.update,
    app.view,
    app.on_attribute_change(),
  )
  |> lustre.start_actor(initial_state)
```

Gleam HTTP ecosystem

HTTP servers:

- Mist
- gleam_elli
- gleam_cowboy
- gleam_plug
- CGI
- Stego (Deno)
- Conversation (JS)

HTTP clients:

- gleam_httpc
- gleam_hackney
- gleam_fetch
- finch_gleam
- Dove
- Conversation (JS)
- Nerf

API clients:

- Plunk
- ZeptoMail
- aws4_request
- Pushover
- Gatus
- gleam_sentry
- gleam_sendgrid

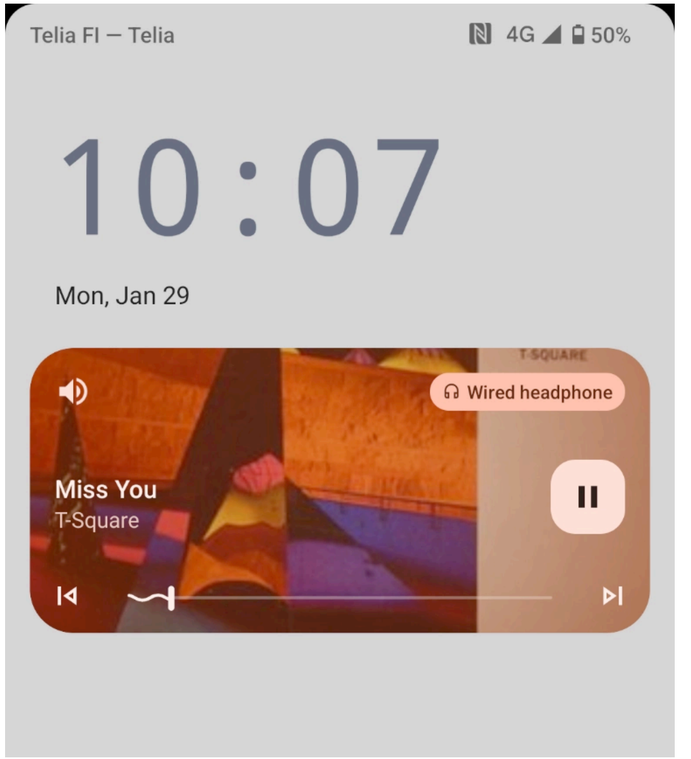
Middleware:

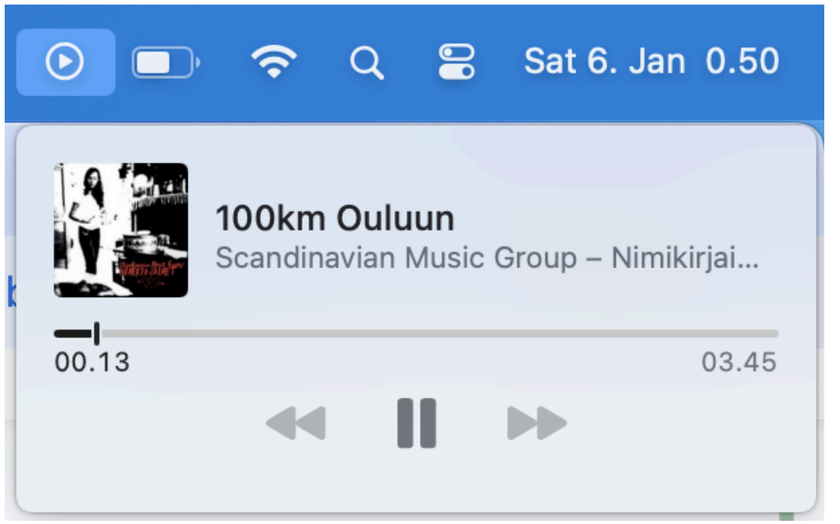
- Wisp
- Gliew
- Bliss
- gleam_cors
- glow_auth

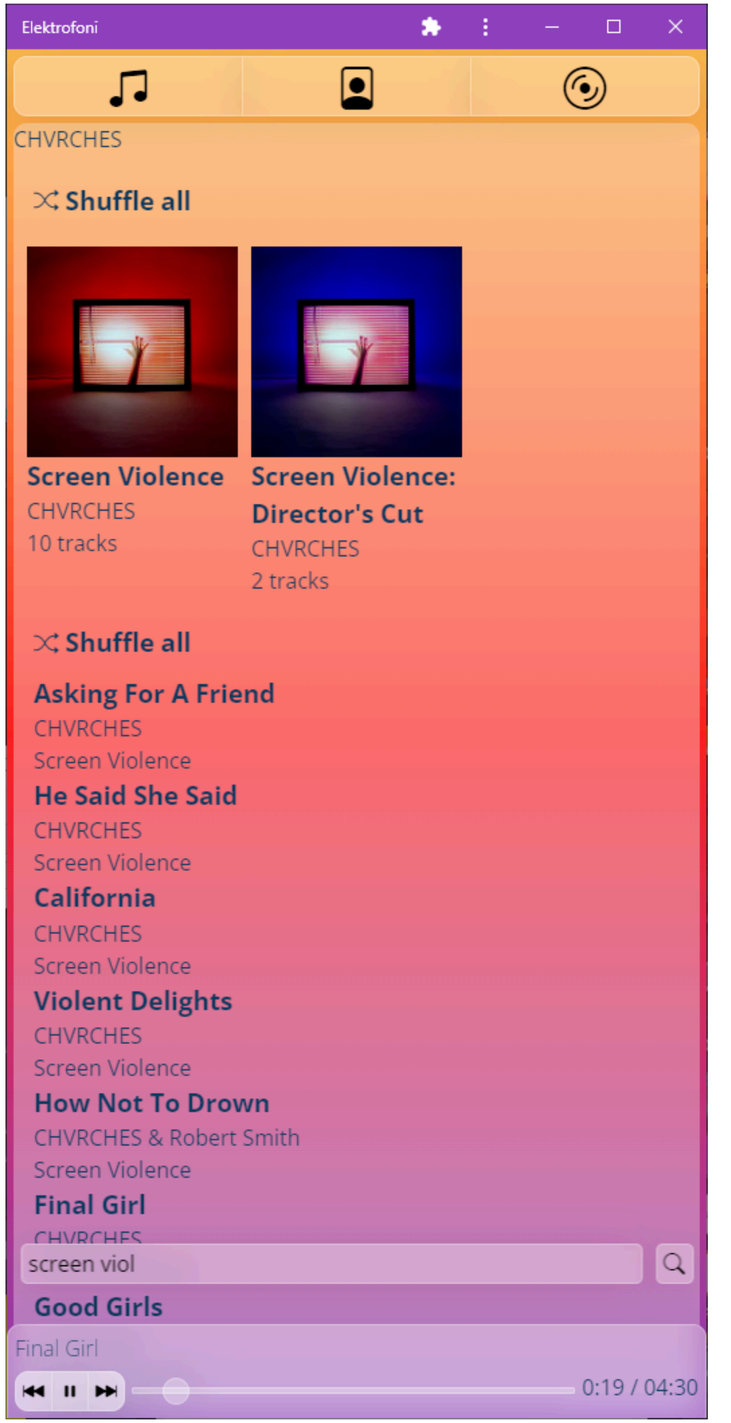
github.com/erikareads/teashop

gitlab.com/Nicd/elektrofoni

- Music streaming
- Library management
- Library search
- Scrobbling to last.fm
- Lock screen support
- Media key support

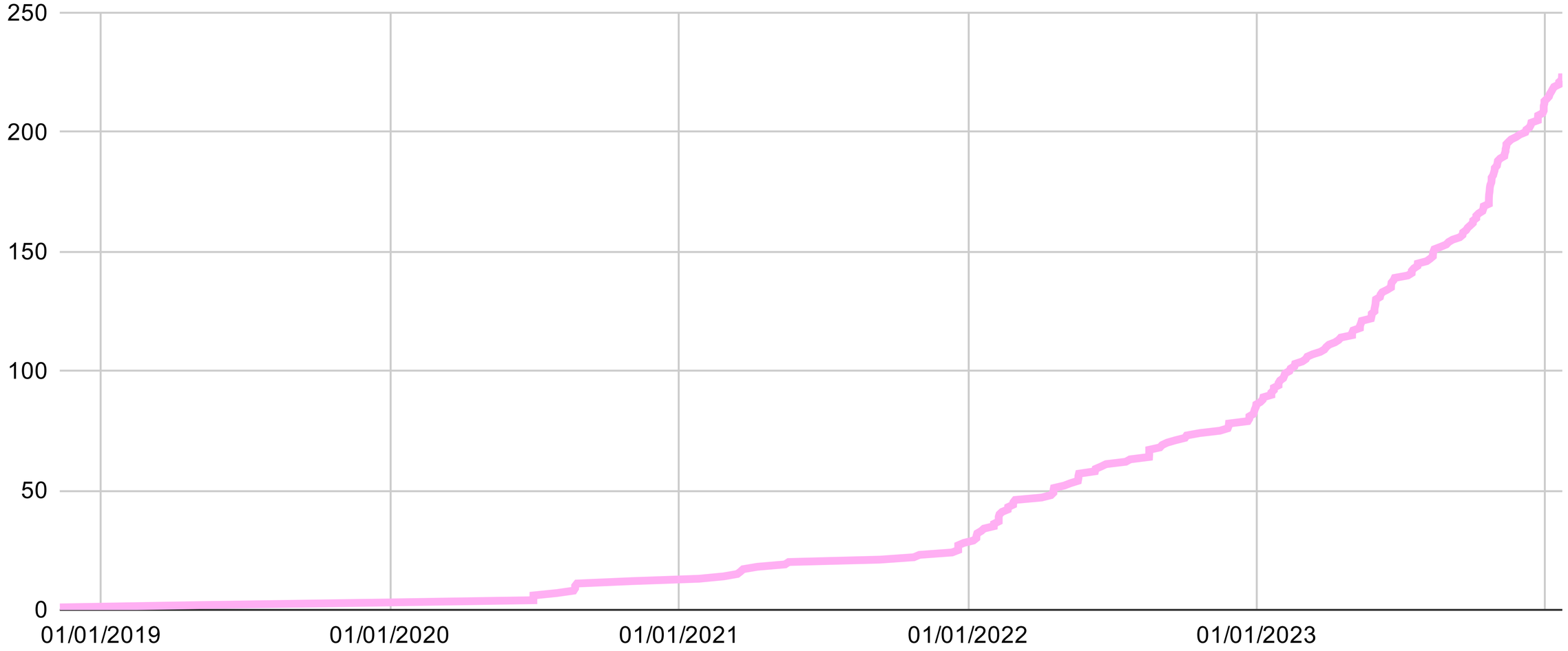






Published Packages

1.2% of Hex



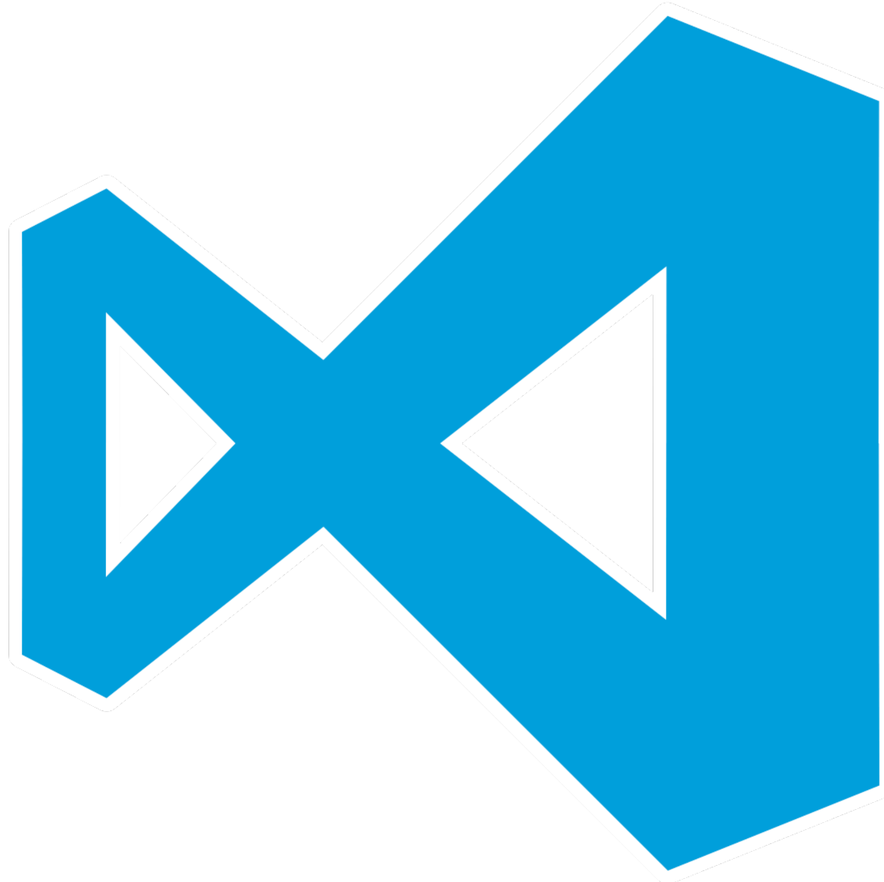
Gleam: Future

Where are we going?

The language server

What's a language server?

Reliability and consistency



Language server additions

Features:

- Find references
- List symbols
- Rename
- Autocomplete imports
- Autocomplete fields
- Import insertion
- Call argument docs

Refactorings:

- Extract variable
- Extract function
- Add annotations
- Add record fields
- Add all case clauses
- Surround with block
- Promote to const

Code generators:

- Dynamic decoder
- JSON encoder
- List variants
- Variant to string
- Variant from string
- Compare

Breaking changes

This slide intentionally left blank

Gleam v1.0.0

Gleam v1: Focus on production usage

1

Productivity for Gleam users

No breaking changes

No language bloat

Keep improving DX

More documentation

2

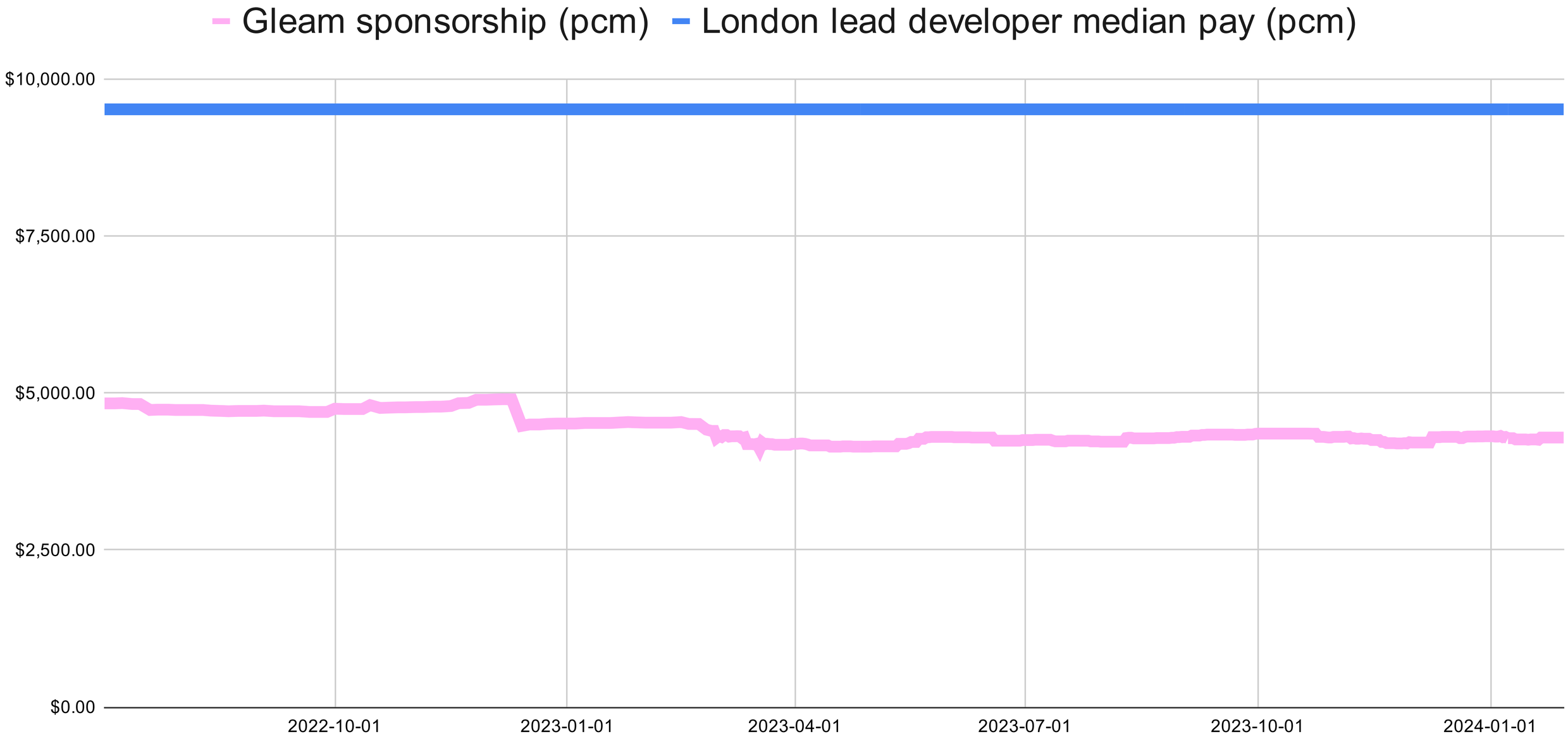
Sustainability for the project

Impactful additions only

Document everything internal

Find more corporate sponsorship

Explore other revenue streams


Gleam sponsorship (pcm)
London lead developer median pay (pcm)
$10,000.00
$7,500.00
$5,000.00
$2,500.00
$0.00
2022-10-01
2023-01-01
2023-04-01
2023-07-01
2023-10-01
2024-01-01

Can you help?

github.com/sponsors/lpil

louis@gleam.run

When is Gleam v1?