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

Jun 26, 2016 – Jan 15, 2024

Contributions to main, excluding merge commits
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

Jun 26, 2016 – Jan 15, 2024

Contributions to main, excluding merge commits
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")
    }
}
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
Gleam Packages

I found 6 packages matching your search.

**glentities**
HTML entity encoder/decoder for Gleam

Documentation  Repository

**nakai**
HTML generation for Gleam, on the server or anywhere else

Documentation  Repository

**htmgrrrl**
Gleam bindings to htmerl, the fast and memory efficient Erlang HTML SAX parser.

Documentation  Repository

**htmb**

1 month ago

2 months ago

3 months ago
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.
```groovy
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 }
```

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:

```groovy
exciting_list(["Lua", "Gleam"])
// -> True
```
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.

```rust
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?
@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

birdie git:(main) × gleam run -m birdie help
Compiled in 0.01s
Running birdie.main
birdie v0.1

USAGE:
gleam run -m birdie [ <SUBCOMMAND> ]

SUBCOMMANDS:
  review Review all new snapshots one by one
  accept-all Accept all new snapshots
  reject-all Reject all new snapshots
  help Show this help text

birdie git:(main) ×
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?
actor.start("init", fn(msg, state) {
  case msg {
    Set(new_state) -> {
      actor.continue(new_state)
    }
    Get(caller) -> {
      process.send(caller, state)
      actor.continue(state)
    }
  }
})
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

- PostgreSQL
  - [GitHub](https://github.com/lpil/pgo)

- SQLite
  - [GitHub](https://github.com/lpil/sqlight)

- MongoDB
  - [GitHub](https://github.com/massivefermion/mungo)

- Redis
  - [GitHub](https://github.com/massivefermion/radish)
type Msg {
    Incr
    Decr
}

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

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

    div([], [
        button([on_click(Decr)], [text("-")]),
        p([], [text(count)]),
        button([on_click(Incr)], [text("+")])
    ])
}
Gleam LiveView?
let assert Ok(app) =
lustre.component(
    app.init,
    app.update,
    app.view,
    app.on_attribute_change(),
)
lustre.start_actor(initial_state)
**Gleam HTTP ecosystem**

<table>
<thead>
<tr>
<th>HTTP servers:</th>
<th>HTTP clients:</th>
<th>API clients:</th>
<th>Middleware:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mist</td>
<td>gleam_httpc</td>
<td>Plunk</td>
<td>Wisp</td>
</tr>
<tr>
<td>gleam_elli</td>
<td>gleam_hackney</td>
<td>ZeptoMail</td>
<td>Gliew</td>
</tr>
<tr>
<td>gleam_cowboy</td>
<td>gleam_fetch</td>
<td>aws4_request</td>
<td>Bliss</td>
</tr>
<tr>
<td>gleam_plug</td>
<td>finch_gleam</td>
<td>Pushover</td>
<td>gleam(cors)</td>
</tr>
<tr>
<td>CGI</td>
<td>Dove</td>
<td>Gatus</td>
<td>glow_auth</td>
</tr>
<tr>
<td>Stego (Deno)</td>
<td>Conversation (JS)</td>
<td>gleam_sentry</td>
<td></td>
</tr>
<tr>
<td>Conversation (JS)</td>
<td>Nerf</td>
<td>gleam_sendgrid</td>
<td></td>
</tr>
</tbody>
</table>
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
Can you help?

github.com/sponsors/lpil

louis@gleam.run
When is Gleam v1?