Gleamがアツい

2026.07.16
個人開発集会 in
Vket2026Summer 技術学術WEEK

About Me

- 創好リナ (Lina Tsukusu)
  - Github: `neverclear86`
- VTuber系フリーランスエンジニア
  - Web系がメイン
- 最近YouTube本格再開

突然ですが

サーバーサイドを書く言語迷っていませんか?

TSは楽でいいけど
パフォーマンスで不安がある

Rustは速いけどサーバーを作るには仰々しい

Goとか良いけど
もっと関数型で書きやすく

今日はそんな皆さんに
ぴったりの言語を紹介します

Gleam

ユーザーフレンドリーで
スケーラブルで
静的型付けの
シンプルな関数型言語

???  ???

ひとつずつ見ていこう

シンプルな関数型言語

どれくらいシンプルなのか

シンプル界隈で有名なGoは
例外処理、クラス、継承が無い

Gleamも文法がかなり少ない

Gleamにない文法

try - catch

try - catch

class

try - catch

class

for

try - catch

class

for

while

try - catch

class

for

while

if

 < パターンマッチングと
関数があればプログラミング
できるでしょ

FizzBuzzを書いてみよう

```gleam
import gleam/io
import gleam/int

pub fn main() {
  loop(1, 100, fn (i) {
    let str = case i % 3, i % 5 {
      0, 0 -> "FizzBuzz"
      0, _ -> "Fizz"
      _, 0 -> "Buzz"
      _, _ -> int.to_string(i)
    }
    io.println(str)
  })
}

fn loop(i: Int, end: Int, call: fn(Int) -> Nil) -> Nil {
  case i > end {
    True -> Nil
    False -> {
      call(i)
      loop(i + 1, end, call)
    }
  }
}
```

確かに書ける...

```gleam
import gleam/io
import gleam/int
import gleam/list

pub fn main() {
  list.range(1, 100)
  |> list.map(fizzbuzz)
  |> list.each(io.println)
}

fn fizzbuzz(n: Int) -> String {
  case n % 3, n % 5 {
    0, 0 -> "FizzBuzz"
    0, _ -> "Fizz"
    _, 0 -> "Buzz"
    _, _ -> int.to_string(n)
  }
}
```

list関数とパイプを使った例

関数型なので
コレクションの
扱いは分かりやすい

シンプル故にミスを防げる

Goと似たような思想

さらに静的型付け

コンパイル時点で
ミスを概ね潰せる

ユーザーフレンドリー

1. エラーが分かりやすい

```
error: Type mismatch
    ┌── /src/main.gleam:16:5
    │
16  │         _, _ -> n
    │         ^^^^^^^^^

This case clause was found to return a different type than the previous
one, but all case clauses must return the same type.

Expected type:

    String

Found type:

    Int
```

提案までしてくれる

Rustっぽい感じだね

2. LSPが高機能

型アノテーション

import文の自動挿入

caseの不足パターン挿入

パイプを使う形に変換

存在しない関数の定義生成 ...etc

VSCode専用じゃないので
Vimとかでも使える!

3. gleamコマンドが
オールインワン

プロジェクト作成

ビルド・実行

フォーマッター

パッケージマネージャ

LSP

ドキュメント生成

DenoやBunみたいに使える

スケーラブル

GleamはJSかBEAMにビルドできる

ErlangのVM[BEAM]がオーパーツ過ぎる

BEAMの特徴

1. 並列処理が軽量で高速

OSプロセスより軽量に
BEAM内でプロセスを大量作成可能

プロセスはメモリを共有せず
メッセージで通信する

2. 壊れても落ちない

プロセスが分離してるので
一つ落ちてもシステムは落ちない

Supervisorが監視していて
自動でプロセスを再起動

3. ホットスワップで
再起動なしにアップデート可能

パッケージをアップデートしたら
次に生成するプロセスから適応

4. ほっといても止まらない

先述のプロセス分離やSupervisor
などの特徴により
年単位で放置しても正常に動作

これがGleam

ここまでの説明を聴いて
なにか思いませんか

私は思いました

🤔 < なんだかAI適性が高そう

このAIの時代にAI適性は
あればあるだけ良いとされてます

静的型付けでシンプルな言語

充実したコマンドやLSP

-> AIが間違えず、
素早く綺麗に書ける

適当にClaudeに頼んだら大体一発で動くやつが出てくる

BEAMは並列処理が得意

-> IoT, マイクロエージェントによる頻繁・多数のアクセスに耐えれる

リアルタイム通信のための
大量WebSocketとかもいけそう

更新をホットスワップできる

-> AIで爆速修正しまくっても
安全にアップデートできる

ほとんどメンテしないでも
ずっと動いてる

-> 私のような
ズボラ個人開発者には嬉しい

さらに!!

Gleamならなんと!!

JSターゲットもできる

フロントも
関数型で書けちゃう!

😟 < とはいえフロントはTSがいい

TSの型定義も生成可能

さらに!!!!!!

マスコットがかわいいね!!

ついでに

文法が分かりやすいので
関数型の勉強の入口としても
ベネ(良し)

接続が多いバックエンドや
並列処理をしたいときは
使ってみてね

JSを関数型で書きたい時も
オススメ

以上