Gleam で React を書こう

React Tokyo LT 9/19

こまもか

自己紹介

省略

Gleam とは

静的型付けな関数型言語

Erlang と JavaScript(ES6)にコンパイルできる。

JS ターゲットにおいて Node.js, Deno, Bun の 3 つの JavaScript Runtime をサポートしているのが特徴的。

構文

Gleam で書いた FizzBuz

```
list.range(1, 30)
|> list.map(fn(num) {
  case num % 15 {
    3 | 6 | 9 | 12 -> "Fizz"
    5 | 10 -> "Buzz"
    0 -> "FizzBuzz"
    _ -> int.to_string(num)
  }
})
|> string.join("\n")
```

Gleam とフロントエンド

Gleam は JavaScript(しかも ES6)に対応している。

JavaScript が生成できるという事はフロントエンドの開発に使える。

Gleam をフロントエンドの開発に使うのは当然の摂理。

Gleam で React を書こう

Gleam で React を書く利点(主観)

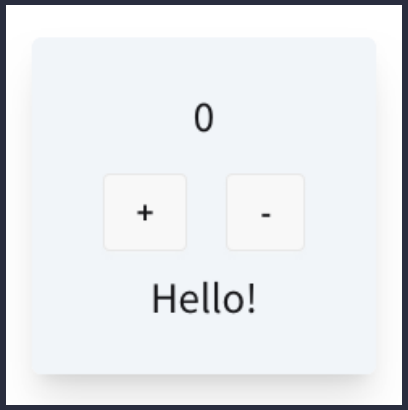
- 関数の外に変数を定義できないため、コンポーネントの純粋性がある程度保たれる
- JS/TS にない構文(パターンマッチ, パイプライン, ブロック)が使える

rewraw について

- Gleam の React wrapper
- Gleam の関数で React コンポーネントを定義できる
- Hooks も使える
- (FFI を書けば)既存の JSX コンポーネントも呼び出せる

やってみた

サンプルは `Comamoca/sandbox-gleam` の `ex_gleam_redraw` ディレクトリにある Hooks を使ったカウンターの例



コードはこんな感じ

```
// コンポーネント定義
use _ <- react.element("Counter")

// useState 定義
let #(count, set_count) = react.use_state(0)
```

```
// コンポーネント定義
use _ <- react.element("Counter")

let #(count, set_count) = react.use_state(0)
```

コードはこんな感じ

```
// useStateの値を表示
html.p([a.class("mx-auto text-2xl pb-5")], [
  // Stateは数値で持っているので文字列型に変換している
  html.text(int.to_string(count)),
]),
```

```
// もちろんコールバックも登録できる
html.button(
  [events.on_click(fn(_) { set_count(count - 1) })],
  [html.text("-")]
)
```

まとめ

- Lustre が主流とはいえ Gleam でも React が使える
- React の膨大な資産 x JS にない構文の組合せは魅力的に見える
- Jotai, shadcn/ui 等既存の資産を使うとどうなるのか気になる
- JavaScript バックエンドでフルスタック Gleam したい