最近触ったAltJSをいくつか紹介

まろん。(@devuloper)

写真OK(SNS投稿もOK)

About Me

まろん。

export default
{
Hobbies:
[開発 , ドメイン]
Links:
{
Portfolio: https://montblank.fun
Twitter: @rin_montblank
GitHub: @otoneko1102
}
Langs:
[JavaScript , TypeScript , CoffeeScript , Astro , Svelte]
}

最近したこと

札幌でLT会に参加した (https://blog.montblank.fun/b/3d-winter_2026-02-07)

これは正規表現で実装した冷笑チェッカー

最近の趣味

AltJSに触れること

AltJS とは

JavaScriptに変換（トランスパイル）して動作する代替言語の総称
JavaScriptの仕様が未発達だったES6 (2015) 以前に作られたものが多い印象

AltJS とは

JavaScriptに変換（トランスパイル）して動作する代替言語の総称
JavaScriptの仕様が未発達だったES6 (2015) 以前に作られたものが多い印象
→ ほぼ廃れている

1. TypeScript

Microsoftが開発した静的型付けを追加したAltJS
唯一成功したAltJSと言っても過言ではない

- 型システムによる開発体験の向上
- 現代では標準(なんならJavaScriptよりこっちのほうが使われている気がする)
- トランスパイルすると型情報が消える(←当たり前だけど残念)

1. TypeScript

```
function greet(name: string): string {
  return `Hello, ${name}!`;
}

greet("World");  // OK (string)
greet(123);      // 型エラー (number)
```

2. CoffeeScript

Ruby / (Python)ライクな構文を持つシンプルなAltJS
2009年リリース、ES6以前は人気があった

- インデントベースの構文（`{}`不要）
- シンプルで読みやすい記法
- `->`でアロー関数（ES6より前に実装）
- 現在はほぼ使われていない（使われてほしい）

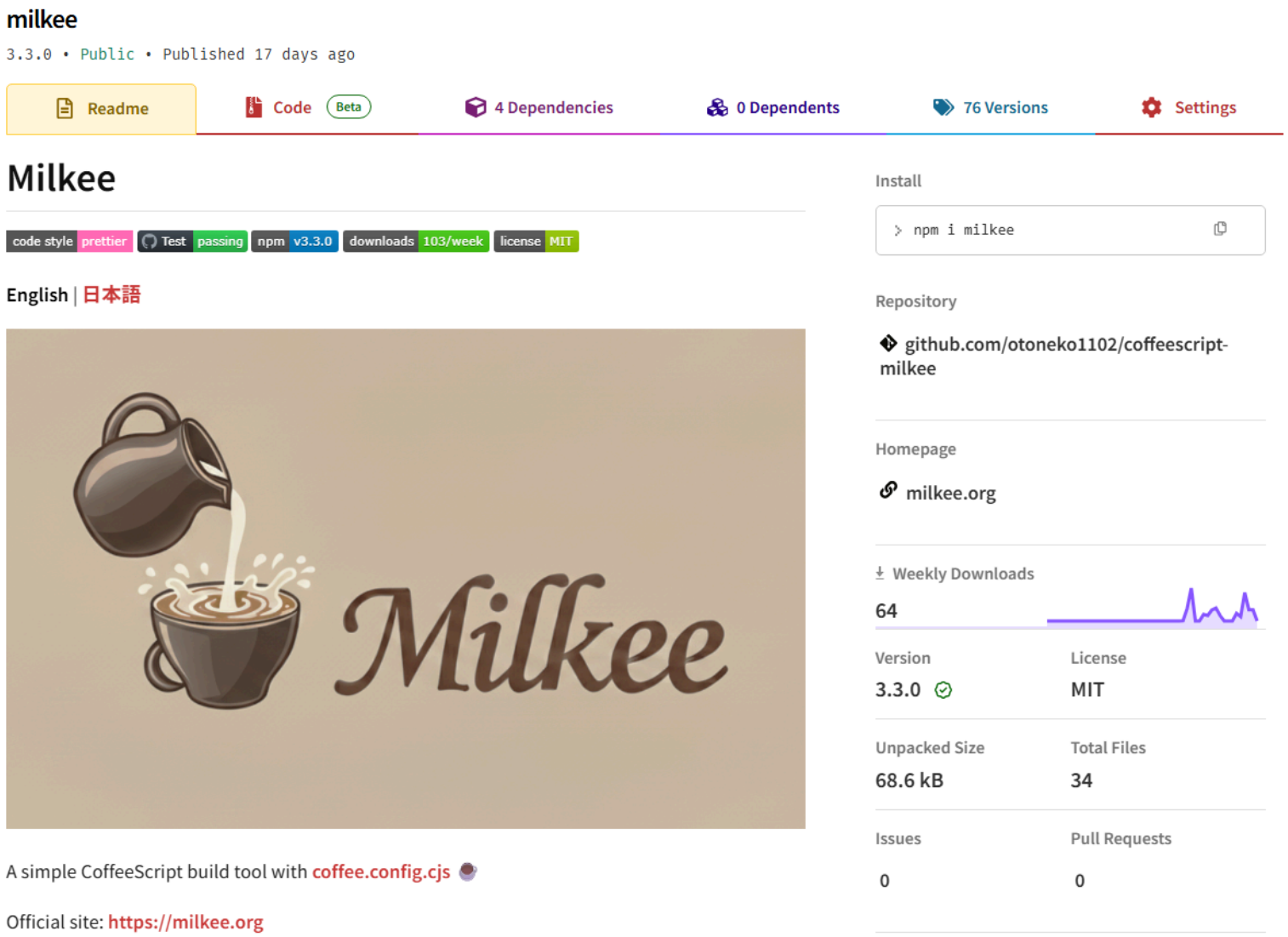
2. CoffeeScript

```
# CoffeeScript - {}不要
greet = (name) ->
  "Hello, #{name}!"

console.log greet "World"

# 範囲指定
numbers = [1..5]  # [1, 2, 3, 4, 5]

# 内包表記
evens = (n for n in numbers when n % 2 is 0)  # [2, 4]
```

2. CoffeeScript

CoffeeScriptは
触ってみて思ったよりよかったので
開発補助ツールを作成した
(Builder/Plugins/Formatter/Minify/etc.)
CoffeeScript流行れ(無理)

https://milkee.org



3. LiveScript

関数型プログラミングに特化したAltJS
CoffeeScriptから派生（2011年）

- 強力な関数型プログラミング機能
- パイプライン演算子 `|>`
- パターンマッチング
- 独特な演算子（`++`, `--` など）
- **完全に廃れている**
- ~~名前がややこしい~~

3. LiveScript

```
# LiveScript
greet = (name) -> "Hello, #{name}!"

console.log greet "World"

# 範囲指定
numbers = [1 to 5]  # [1, 2, 3, 4, 5]

# 固有の演算子
a = 10
b = a + 2  # 12
c = a ++ b # [10, 12] (配列結合)
```

3. LiveScript

パイプライン演算子 `|>`

データを次々と関数に流し込んで処理する

```
# 従来の書き方: 変数に代入（冗長）
step1 = 10 * 2       # 20
step2 = step1 + 5    # 25
step3 = step2 - 3    # 22
console.log step3

# パイプライン演算子: 簡潔で読みやすい
10
  |> (* 2)          # 2倍する → 20
  |> (+ 5)          # 5を足す → 25
  |> (- 3)          # 3を引く → 22
  |> console.log    # 表示   → 22
```

3. LiveScript

固有の演算子

JavaScriptにはない便利（？）な演算子

```
# 配列結合 ++
[1 2] ++ [3 4]           # [1 2 3 4]
[1 2] ++ [3 4] ++ [5 6] # [1 2 3 4 5 6]

# 暗黙の引数 it
[1 to 5] |> (.map (* 2))   # it が省略されている
# これは (.map (it) -> it * 2) と同じ

# 関数合成 >>
double = (* 2)
add-one = (+ 1)
f = double >> add-one   # 2倍してから1を足す
f 5                     # 11
```

3. LiveScript

その他の便利な特徴

```
# パターンマッチング
{name, age} = {name: "Alice", age: 25}

# switch文が式になる
grade = switch score
  | score >= 90 => "A"
  | score >= 80 => "B"
  | score >= 70 => "C"
  | otherwise   => "D"

# バックコール（ネストが深くなるのを防止できる）
data <-! fetch-data
result <-! process data
console.log result
```

直感的に理解しやすい

まとめ

- TypeScript: 現役で使われている(ほぼ)唯一のAltJS
- CoffeeScript: シンプルで読みやすかったが今は過去の遺産
- LiveScript: 関数型特化も普及せず

でも触ってみると面白い

最近の活動

JavaScript系で100万回Hello, World!する方法を考えるプロジェクト
制約: for/while/foreach/再帰を使わない

https://github.com/oto-home/how-to-do_hello-world_a-million-times.js

これはgzipで二回圧縮した `Hello, World!` x 1000000 を解凍してコンソールに出力するサンプル

```
const hello_gz_gz = `
H4sIAAAAAAACA+3cPQpBAQAA4IfSKz+DsqAo2cnuAk5ge4PBBUwKi8kFWFgcQNlls1okFhcwvFmJ
nELq+w7yVedh8JVMPY6tXFDLBttqMa73883DvtGNJ9HmVeqdR6vx8BK2AQAAAAAAAAAAAAAAAAAAA
gF/Z3ceVwvcICJ/vWzk9WCwBAAAAAAAAAAAA+DPXKNGZZU7T9Tn4APdK+vhBagAA
`

const binary = Uint8Array.from(atob(hello_gz_gz), c => c.charCodeAt(0));

const byteStream = new ReadableStream({
  start(controller) {
    controller.enqueue(binary);
    controller.close();
  }
});

const decompressedStream = byteStream
  .pipeThrough(new DecompressionStream('gzip'))
  .pipeThrough(new DecompressionStream('gzip'));  // 二度漬けgzip

const resultText = await new Response(decompressedStream).text();

console.log(resultText);
```