So farewell then, TypeScript

... we hardly knew ye

HI I'M DAVE

- Writing Server Side (Java|Type)Script for most of my career.
- No that's not a joke.
- Flow, my tears.

ShriramKrishnamurthi @ShriramKMurthi · Apr 9

One of my colleagues is having students do projects in @typescript, and claims the coding styles are a mess despite the types. Can anyone suggest a style guide for TypeScript code that they like, that a course might adopt and require of its students? Thanks!

19 4 27

https://twitter.com/ShriramKMurthi/status/1512883907053727749

TypeScript is *not* a language (imho)

It is a (pretty good) type system for JavaScript

If studying *SICP* using Common Lisp is like "putting ketchup on caviar"...

... then TypeScript is like putting ketchup on JavaScript

Take a look at the "Goals" and "Non-goals"

https://github.com/Microsoft/TypeScript/wiki/TypeScript-Design-Goals

TypeScript is welded to JavaScript

So what's the recommended coding style for JavaScript?

LOL ARE YOU FUCKING KIDDING ME?

If anything, a TypeScript makes this worse

Bare-Metal Arrogance + FP Smugness = Rust

-- Me

JavaScript Chaos + FP Smugness = TypeScript

-- Also Me

"But Dave," you say, "surely you have some opinions on how to use TypeScript?"

No kidding

1. (Type|Java)Script is a poor tool for domain modelling with types, don't use it on the server.

You can't even override equality operations FFS

Good luck with DDD in most forms suckas

2. TypeScript is *THE BEST* client-side language.

TS ❤️ JS, this is a good thing!

Context switching is *also a good thing* when it's actually a *different context*.

3. Choose one of these two *incredible* styles...

a) Object Oriented Programming

TypeScript looks like C#, so just write it like C#

Beat up the language though

e.g. `Sett` by the Pelicans provides a conventional interface for equality in sets.

b) "Functional Programming"

NO, NOT THIS

amount of time I've spent in codebases of PURE FUNCTION SOUP

OR THIS

https://gcanti.github.io/fp-ts/

"Functional Cosplay"

-- Nat Pryce

Try...

`export` a type/interface from a module (file)

create a `namespace` to shadow the type

`export` related functions within the `namespace`

```ts
export interface Dog {
  name: string;
}

export namespace Dog {
  export function newDog(name: string): Dog {
    return { name };
  }

  export function sayName(dog: Dog): string {
    return dog.name;
  }
}
```

```
import { Dog } from './Bob';

const dog: Dog = Dog.newDog('bob');
```

Thoughts for the future

JS is influenced by Smalltalk / Self. What do they do?

Object Oriented != Class Oriented; just use objects everywhere

Opportunity for mixing dynamic patterns with static patterns

Thank you