Typescript Runtypes

3/8/2024, 1:22:55 AM

2/8 Simulation vs. Reality

- i am working on a product that helps people to use heavy machinery (big trucks, extra large transports) more efficiently
  - looks a litte bit like an 80s computer game on google maps, in the browser
- that means, bugs in that software may have really bad consequences
  - like damaging an expensive transformer
  - blocking an important road for many days (in south africa)
  - delaying projects costing money

3/8 We use Typescript

- we use typescript to implement most of that, in the front and backend
  - which is just an invisible layer of checks that are gone once the program is running
  - isn't that unsafe as hell? it's all just javascript?
  - `` `const vehicle: Truck = unknownJsonData` ``
  - just becomes `` `const vehicle = unknownJsonData` `` <-- no check that its a truck
- even with all types inside being good, we regularly talk JSON to other parties
  - to our frontend
  - to databases
  - to the operating system via env vars, cli arguments and config files

---

4/8 Runtypes Idea

- fortunately, albeight not built into TS, we still have feature hat let us built that ourselves with minimal overhead
  - we can build cheap dynamic and typesafe deserializers ourselves
  - thats what we call *runtypes* in TS
- the basic idea is
  - `` `function truck(x: unknown) ⇒ Truck` ``
  - a function that takes in any json data
  - and makes sure that it conforms to our desired type
  - we only write it once, and TS inferences the type for us
  - so instead of `` `const vehicle: Truck = unknownJsonData` ``
  - we write `` `const vehicle = truck(unknownJsonData)` ``

5/8 Runtypes Schema

- when writing complex types, functional programming comes to our help:
  - combining many functions to build a more complex type
  - const truck = record({ id: string(), weight: integer(), model: string(), })
  - thats it, our simple schema language is ready
  - lightweight functional programming and cheap function calls in JS make this a good approach

---

6/8 Build or Buy

- building your own, in 2024, its still javascript, no way, lets try ~ChatGPT~ npm first
  - there are tons of libraries with different architectures and feature sets
    - function combinators, somethimes with a rich api and tons of integrations, with or without exceptions, full fp or lightweight js style: most popular: zod
    - decoration based: class-validator (from nestjs)
    - simple compilers using eval at runtime (for speed)
    - fully flegded AOT compilers: typia
- seems like people enjoy to explore this space a lot

7/8 Benchmark

- shoutout to Roman / https://github.com/moltar for mainting a list of runtypes and a basic benchmark:
  - https://github.com/moltar/typescript-runtime-type-benchmarks
  - shows some basic performance numbers
  - the benchmarks test cases are readable and allow comparing the single libraries
  - having a list of all those options is already valuable enough
  - :heart: opensource

8/8 Recap

- recap
  - in TS, always check json with runtypes: before writing it to your db, after reading a request payload, after parsing query parameters,
  - github.com/moltar/typescript-runtime-type-benchmarks for a list and benchmark
  - happy coding y'all