An 'exceptionally' demo

by Ivan Hofer

What is an error?

- **unrecoverable**

- application can't function if condition is not met
- e.g.
  - missing environment variable on startup
  - DB migrations don't succeed

- application should crash

What is an exception?

- **recoverable**

- affects just a part of your application
  - temporary:
    e.g. external server is down
  - input-dependent
    e.g. passing incorrect data

- application should continue to work

DEMO

Fun Fact

Do you know how much the core of the library adds to your bundle size? Just as much as it takes a PC to save this paragraph on disk.

132 bytes (minimized and gzipped)

- just a few lines of code
- with some TypeScript magic

Implementation / runtime

```
const exceptionally = Symbol.for('exceptionally')

export const isExceptionallyResult = value ⇒ value?.exceptionally === exceptionally

const wrap = (success, data) ⇒ isExceptionallyResult(data)
    ? data
    : Object.assign(() ⇒ data,
        {
            exceptionally,
            isSuccess: success,
            isException: !success,
        },
    )

export const success = data ⇒ wrap(true, data)

export const exception = data ⇒ wrap(false, data)
```

Advantages

- code flow stays linear
- full TypeScript support
  - auto-inferred types
  - don't need to look at the implementation
  - see if exceptions are unhandled
- huge help when refactoring

Disadvantages

- learning curve
- you need to wrap everything
  - can be solved with a proxy
- supports only code you own
  - unless you wrap the functionality

Shift left

- resolve issues as early as possible
  - in your brain
  - while writing the code ← here
  - while manual testing
  - automatic tests (CI) ← and here
  - staging environment
  - production system

- reduces costs

Conclusion

- will your code be bugsafe? **NO**
  but it will make you aware of potential bugs
- can you ignore writing tests? **NO**
  but you'll need to write fewer of them
- does it replace error reporting services? **NO**
  using such a service as fallback is a great idea
- should you know implementation details of a function you use? **NO**
  TypeScript should tell you what all possible outcomes are

Links

- https://github.com/ivanhofer/exceptionally
  - try it
  - share your feedback
  - Code (https://github.com/ivanhofer/exceptionally-demo)

- https://github.com/ivanhofer
  - typesafe-i18n (https://github.com/ivanhofer/typesafe-i18n)
  - Svelte(Kit) TypeScript Showcase (https://github.com/ivanhofer/sveltekit-typescript-showcase)