Typescript `strictNullChecks`:

What, why, how

Who am I?

- Professional dev since 2007
- EPAMer since 2017
- Member of EPAM Advanced Software Engineering
- Recognized as Chief Software Engineer since Dec/2022

Let’s begin!

Part 1 - What

```
TypeError: Cannot read properties of undefined (reading 'filter')
    at computeRootPlayersTotalScore (index.js:3:40)
    at index.js:7:3
    at ModuleJob.run (node:internal/modules/esm/module_job:193:25)
    at async Promise.all (index 0)
    at async ESMLoader.import (node:internal/modules/esm/loader:530:24)
    at async loadESM (node:internal/process/esm_loader:91:5)
    at async handleMainPromise (node:internal/modules/run_main:65:12)
```

Scenario

- The project is deployed to production
- Clients begin consuming it
- Application behaves weirdly
- Your SLO error budget is consumed quickly
- You scratch your head asking yourself: *“What’s the root cause?”*

What `strictNullChecks` is?

- It’s a Typescript compiler option
- Defaults to `false`

From TS Docs

"When `strictNullChecks` is `false`, `null` and `undefined` are effectively ignored by the language.

This can lead to unexpected errors at runtime."

```ts
interface Entity {
  name: string,
  type: "PERSON" | "ORG"
  age?: number
}

const aPerson = {
  name: "John Doe",
  type: "PERSON",
  age: 39
}

const aCompany {
  name: "XPTO Inc.",
  type: "ORG",
}

const logAge = (entity: Entity) => console.log(entity.name + " is " + age.toString() + " years old");

logAge(aPerson); // John Doe is 39 years old
logAge(aCompany); // Runtime error
```

What are the problems when it is `false`?

- It won't warn you when you're accessing a possibly `null` or `undefined` value
- It will compile the code into JavaScript, letting a potential bugs slip into production
- When provoked, runtime errors will be thrown, letting your application unstable

Let’s check more!

Part 2 - Why

Type-safety

- Enables speed
- Problems are detected early, while typing the code
- Better code (quality/reliability/readability)
- Reduced cost of maintenance

- It’s a best practice

Others have done it too

- Projects I've joined with in the past went through the same issue (2017, 2019)
- Back in 2018, the entire VS Code repo didn't have `strictNullChecks` enable! (Issue #60565)
- Currently, actively driving the same initiative
- Next, you!

Let’s do it!

Part 3 - How

How to get started?

```
{
  "extends": "./tsconfig.json",
  "compilerOptions": {
    "noEmit": true,
    "strictNullChecks": true,
    "strictPropertyInitialization": true
  },
  "files": [
    // Files strict null checked goes here
  ],
  "exclude": [
    // Files that are not the primary focus, like tests
    // Files with big number of null check erros might require special attention
  ]
}
```

Simple process

- Add a file to custom `tsconfig.json#files` property
- Run a build with null checks
- Find and fix errors
- Rinse & repeat

Let’s check some examples

Let’s check some tooling

Thank you!