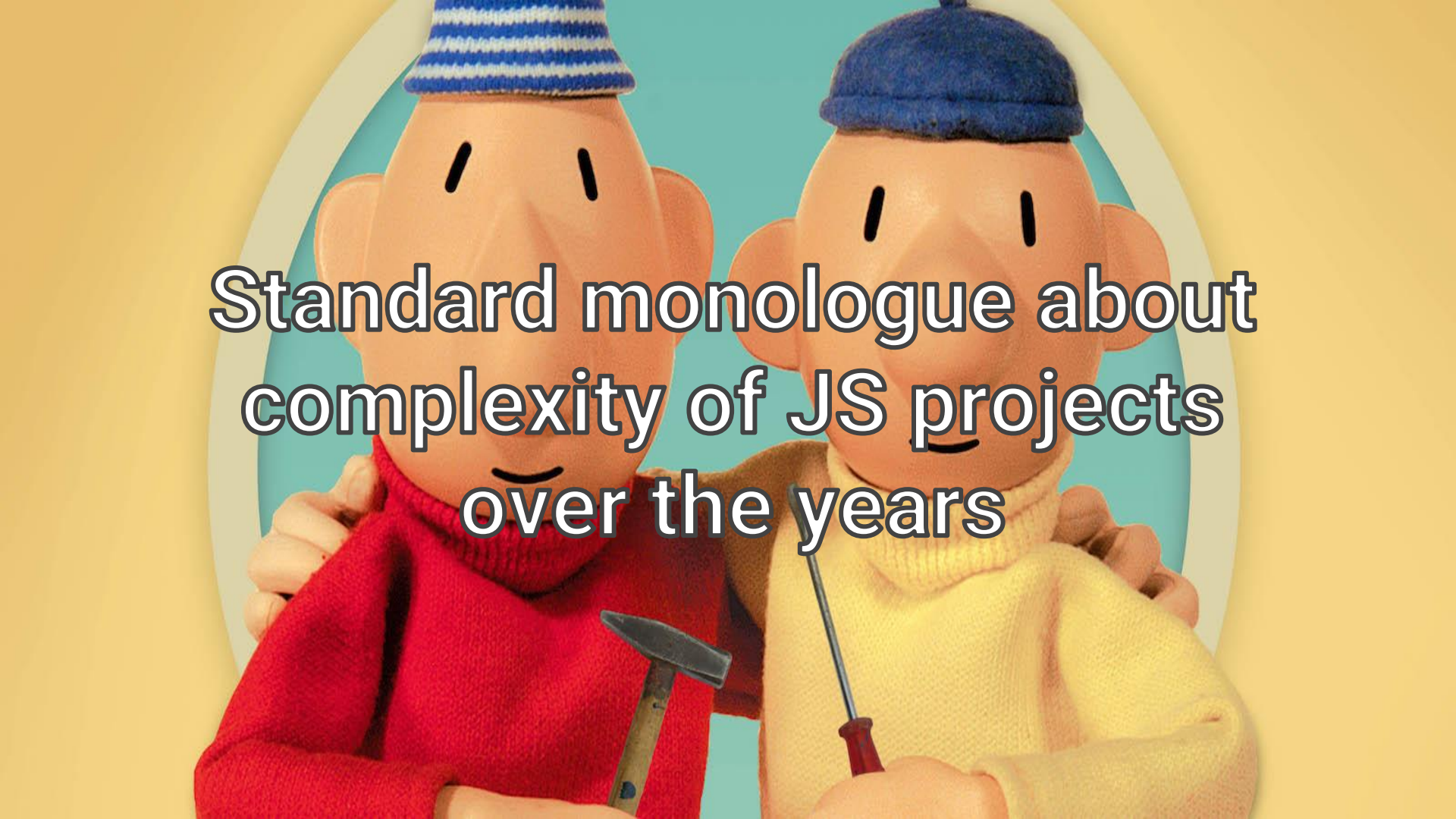
# Typing JavaScript

(TypeScript )

Kresimir Antolic
Frontend dictator at Oradian
@kantolic

I've seen things you people wouldn't believe


Standard monologue about
complexity of JS projects
over the years

# Let’s talk about TypeScript

Any JavaScript is TypeScript.

# TypeScript extends JavaScript.

# TypeScript extends JavaScript

- Types (and type interference)
- Interfaces
- Generics
- Enum
- ES6 flair and more (Classes, decorators, async/await..)
- Great tooling
- EXCELLENT editor support
- React support (tsx)

# DISCLAIMER

This does not turn your JS into Java.

JS +  = TS

# Batteries included

# Types

- boolean
- number
- string
- array
- tuple
- Enum
- any
- Void

```
let name:string = 'unicorn';
```

# Interfaces

Describe shape of values.

```ts
interface Animal {
   name: string,
   power: Power,
   magical?: boolean
 };

 interface Power {
   name: string
 }
```

# Generics

Components that work over many types

```
function move<T extends Being>(t: T): void {
  t.eat();
  if ((t as Person).walk) {
      (t as Person).walk();
  } else if ((t as Bird).fly) {
      (t as Bird).fly();
  }
}
```

# ES2015 + stuff

- Classes
- Fat arrows
- Modules
- Symbols
- Iterators, Generators
- Async/Await

# Decorators

Annotate and modify classes and properties at design time.

```
function touchMe(target) {
  // Adds a property on target
  target.touched = true;
}

@touchMe
class MyClass { }

MyClass.touched;
```

And a lot more complicated stuff.

Just move to it already.

# How to refactor to Typescript?

- Change extension to TS and follow what the compiler tells you
- Rules: noImiplicitAny, strickNullChecks, noImlicitThis
- Type definitions -  description of the shape/interface of the JS lib you’re using
- Modules
- declare var - declare a variable that may not have originated from a TypeScript file

Not sold?
But like the idea of typing?

# Other options

- Facebook Flow
- Purescript
- Elm
- Dart
- Clojure
- Google Closure Compiler (using JS docs)

# TypeScript Goods

- Coding is less stressful
- Refactoring
- Focuses on the things that would likely be errors.
- Readable
- Code completion

# TypeScript Bads

- Doesn't solve your mistakes
- Introduces complexity
- More knowledge of programing needed
- You still need to know JS
- Dosen’t make JS more performant
- Can lead to over-engineering
- Illusion of security

# Illusive type-safety

```html
<input type="number" id="ovo">
<button onclick="doCalc()">Calc</button>
```

```
function doCalc() {
  let x = document.ovo.value;
  console.log(multi(x,x), add(x,x));
};


function multi(a: number, b: number): number {
  return a * b;
}

function add(a: number, b: number): number {
  return a + b;
}
```

(try 2);

Typescripts marks intent - your code is more explicit.

It’s more understandable.

It’s Just Another Test.

Typescripts just makes getting work done easier.

# Resources

- https://www.typescriptlang.org/play/
- https://www.typescriptlang.org/docs/
- https://github.com/DefinitelyTyped/DefinitelyTyped

# Thanks.

Kresimir Antolic

Frontend dictator at Oradian

@kantolic