jsが型安全になったっていい

Natsuki

自己紹介

Natsuki

普段はLaravel + Vueで開発

自己紹介

Natsuki

普段はLaravel + Vueで開発

Vue Fes Japan 2024

JSDocで型注釈

JSDocとは

JavaDocやPHPDocのようなコメント内の注釈から
ドキュメントを生成するツール

または、そのコメント自体

https://jsdoc.app/

JSDocで型注釈

JSDocとは

JavaDocやPHPDocのようなコメント内の注釈から
ドキュメントを生成するツール

または、そのコメント自体

https://jsdoc.app/

```
/** @type {number} */
const i = 0;
```

JSDocで型注釈

JSDocとは

JavaDocやPHPDocのようなコメント内の注釈から
ドキュメントを生成するツール

または、そのコメント自体

https://jsdoc.app/

```
/** @type {number} */
const i = 0;
```

`@type` などのタグで型情報を書くことができ、tscもこの型情報を参照して型チェックを行ってくれる

JSDocで型注釈

JSDocとは

JavaDocやPHPDocのようなコメント内の注釈から
ドキュメントを生成するツール

または、そのコメント自体

https://jsdoc.app/

```
/** @type {number} */
const i = 0;
```

`@type` などのタグで型情報を書くことができ、tscもこの型情報を参照して型チェックを行ってくれる

これとtsconfig.jsonとtscで型チェックを行うことができる

なぜtsファイルじゃないのか

なぜtsファイルじゃないのか

リプレース案件にアサインされた

なぜtsファイルじゃないのか

リプレース案件にアサインされた

「TypeScript使って堅牢なシステムにしたい！！」

なぜtsファイルじゃないのか

リプレース案件にアサインされた

「TypeScript使って堅牢なシステムにしたい！！」

上の偉い人

なぜtsファイルじゃないのか

リプレース案件にアサインされた

「TypeScript使って堅牢なシステムにしたい！！」

上の偉い人「TypeScript？そんなことより開発を急いでくれ」

なぜtsファイルじゃないのか

リプレース案件にアサインされた

「TypeScript使って堅牢なシステムにしたい！！」

上の偉い人「TypeScript？そんなことより開発を急いでくれ」

なぜtsファイルじゃないのか

どうにかして、
こっそりTypeScriptを使えないか

なぜtsファイルじゃないのか

どうにかして、
こっそりTypeScriptを使えないか

JSDocなら、コメントに型が書けるらしい
コメントならバレない...

なぜtsファイルじゃないのか

どうにかして、
こっそりTypeScriptを使えないか

JSDocなら、コメントに型が書けるらしい
コメントならバレない...

※誇張が含まれています

なぜtsファイルじゃないのか

どうにかして、
こっそりTypeScriptを使えないか

JSDocなら、コメントに型が書けるらしい
コメントならバレない...

※誇張が含まれています

ほかにも

既存プロダクトをtsに移行したいけど、テストコードも無いしエンバグが怖い

トランスパイルの時間を無くしたい

とかとか

JSDoc(+tsc)でできること

1. 型注釈の記述

```
// index.ts
function hello(name: string) {
  // ...
}

const LIMIT = {
  min: 0,
  max: 1000,
} as const
```

JSDoc(+tsc)でできること

1. 型注釈の記述

```ts
// index.ts
function hello(name: string) {
  // ...
}

const LIMIT = {
  min: 0,
  max: 1000,
} as const
```

```js
// index.js
/**
 * @param {string} name
 */
function hello(name) {
  // ...
}

const LIMIT = /** @type {const} */ {
  min: 0,
  max: 100,
}
```

JSDoc(+tsc)でできること

2. 型の定義

```ts
// index.ts
interface User<T> {
  name: string
  age: number
  customData: T
}
```

JSDoc(+tsc)でできること

2. 型の定義

```ts
// index.ts
interface User<T> {
  name: string
  age: number
  customData: T
}
```

```js
// index.js
/**
 * @extends T
 * @typedef {{
 *  name: string
 *  age: number
 *  customData: T
 * }} User
 */
```

JSDoc(+tsc)でできること

3. 型のインポート・エクスポート

```
// index.js
/**
 * @import { User } './index.ts'
 */
```

JSDocでできないこと

1. 条件型(Conditional Types)

JSDocでできないこと

1. 条件型(Conditional Types)

```
type IsNumber<T> = T extends number ? true : false;

type T1 = IsNumber<10>;
```

JSDocでできないこと

1. 条件型(Conditional Types)

```
type IsNumber<T> = T extends number ? true : false;

type T1 = IsNumber<10>;
```

型定義だけ、`.d.ts`に書けば解決

JSDocでできないこと

2. TypeScript独自の実装

Enumとかdeclareとか

JSDocでできないこと

2. TypeScript独自の実装

Enumとかdeclareとか

```
/** @enum {number} */
const Answer = {
  YES: 0,
  NO: 1,
}
Answer[0]
```

JSDocでできないこと

2. TypeScript独自の実装

Enumとかdeclareとか

```
/** @enum {number} */
const Answer = {
  YES: 0,
  NO: 1,
}
Answer[0]
```

一応`@enum`はあるが、ただの連想配列なので、TSのように値からアクセスできない

JSDocでできないこと

3. tsファイルでの型補完

JSDocの型はtsファイル内では無視される

JSDocでできないこと

3. tsファイルでの型補完

JSDocの型はtsファイル内では無視される

```
/**
 * @param {string} name
 */
function hello(name) {
  Parameter 'name' implicitly has an 'any' type.
  // ...
}
```

JSDocを使う上での注意

JSDocを使う上での注意

- 「tsファイルよりJSDocの方がオススメ！」というわけではない

  単純に書く文字数が増えるので、通常のWebアプリケーションの開発にはオススメしない

JSDocを使う上での注意

- 「tsファイルよりJSDocの方がオススメ！」というわけではない

  単純に書く文字数が増えるので、通常のWebアプリケーションの開発にはオススメしない

- tsファイルへの移行は楽じゃない

  JSDocはtsファイルに書いても型注釈として解釈されないので、書き直す必要がある

JSDocを使う上での注意

- 「tsファイルよりJSDocの方がオススメ！」というわけではない

  単純に書く文字数が増えるので、通常のWebアプリケーションの開発にはオススメしない
- tsファイルへの移行は楽じゃない

  JSDocはtsファイルに書いても型注釈として解釈されないので、書き直す必要がある
- その他使いたいエコシステムが対応してないことも

おわり