unknown型を使おう

anyやめませんか

自己紹介

- 三宮在住のシステムエンジニア
- システム何でも屋
  - 業務用クラサバシステムから帳票・バッチ、レガシー・モダンWeb開発まで
  - フロントエンド、バックエンド、DB設計、インフラ

unknown型とは

- TypeScript 3.0で導入された型
- 型安全に不明な型を処理するためのもの
  - anyのようなもの

unknownとanyの比較

- unknown

```
export const hoge = (param: unknown): number => {
  return param + 1;
};
```

```
export const hoge = (param: unknown) => {
  return param.value;
};
```

- any

```
export const piyo = (param: any): number => {
  return param + 1;
};
```

```
export const piyo = (param: any) => {
  return param.value;
};
```

unknown型を処理する例

- 型検査を実装することで適切に扱うことができる

```ts
type Result = {
  text: string;
};

0 個の参照
export const hoge = (param: unknown): Result => {
  if (param === undefined || param === null) {
    throw new Error('nullish');
  } else if (typeof param === 'object' && 'value' in param && typeof param.value === 'string') {
    return {
      text: param.value
    };
  } else {
    throw new Error('unexpected value');
  }
};
```

ユースケース

- anyを使うケースでは基本的にはunknown型を受け取り、型検査をしたうえで処理する
- unknown型のままでいいものは、型検査をしない
  - これによって使わない値として責務を分けることができる
- { [key: string]: any }も可能であればunknownにするとより型安全になる
  - どうしても難しい場合でも{ [key: string]: unknown }

unknown型を使う上での長短

- Pros
  - 型の保証を得やすい
    - 例えばany型変数を引き回すと型の保証がないが、unknownであれば保証しやすい
  - 型を騙すのに使える
    - “hoge” as unknown as numberと書くことで数値型にできる
    - 実装と設計の乖離のようなことが発生するので極力回避したいが現実的には必要なケースもある
- Cons
  - 型検査のコストが重い
    - 型検査の処理が必要であり、パフォーマンスが犠牲になる

最後に

- そもそも不明な型を発生させないことが大切
  - API要求や応答を始めとした外部値は必要がない限り仕様通りの値が来るものとして解釈し、異常な型が来て処理が続行できない場合は言語側のランタイムエラーに任せるのも一つ
  - もちろん、異常値を許さない処理であればバリデーションする必要はある
- 複数の型が来ることを想定する場合はジェネリクスやUnion型、クラスなどを用いるのも一つ
  - <Hoge>, <Hoge extends Piyo>, ‘hoge’ | ‘piyo’ | ‘fuga’
- Anyを許さない設定にする
  - tsconfig.json
    - “noImplicitAny”: true
  - ESLint
    - plugin:@typescript-eslint/recommended
    - “@typescript-eslint/no-explicit-any”: “error”