Advanced Typescript

Benefits

- Type as a Documentation
- Reduce some common bugs in javascript
- More confident in refactoring

Warning

- Take more time from the beginning
- Typescript couldn't help you much if the codebase has a bad architecture

Basic

1. Structural type vs Nominal type
2. Problems with any and as
3. Narrowing type
4. Function overload
5. Unknown type

Advanced

6. Generic
7. Conditional Type
8. Mapped Type
9. Template Literal Type
10. Decorator
11. This in class
12. Challenges

Structural type vs Nominal type

Structural type vs Nominal type

Nominal type

```
class Foo {
  method(input: string): number { ... }
}
class Bar {
  method(input: string): number { ... }
}
let foo: Foo = new Bar(); // ERROR!!
```

Structural type

```

interface ProductA {
  name: string;
}

interface ProductB {
  name: string;
  description: string;
}

function getProductName(product: ProductA) {
  return product.name;
```

References

- https://www.typescriptlang.org/docs/handbook/2/basic-types.html
- https://www.typescriptlang.org/cheatsheets
- https://mariusschulz.com/blog/series/typescript-evolution
- https://learntypescript.dev/
- https://github.com/total-typescript/type-transformations-workshop

Thank you for listening!