TypeScript

Introduction

- JS弱类型＋解释性(运行时报错)
- 大型复杂项目，开发＋调试＋维护
- TS可选静态类型系统(编译时类型检查)
- 配置 tsconfig.json

基本类型

- :number :string :boolean
- 数组(每项类型) :Array<number> :number[]
- 对象 :object 配合其他类型定义内容
- :null :undefined 任何类型的子类(赋值)(配置)
- :void 函数无返回值

简单类型

- 联合类型 :string | number 任选其一
- 字面量 值约束 :'a' | 'b' :{ name: string }
- 元组 定长数组＋每项类型 :[string, number]
- :any 可绕过类型检查

自定义类型

- 类型别名 给已知类型定义别名
  - type List = string[][];
  - type RoundType = 'round' | 'ceil' | 'floor'
- 枚举 取值范围 区分逻辑含义和真实值 编译结果中作为对象保留
  - enum EpisodeStatus = { end = 1, failed = 2 }

Interface

- 约束对象、类、函数
- interface User { id: number, name: string }
- 接口继承 可多继承 组合多个约束标准

  interface I extends A, B {}

Class

- es6类语法基础上加类型语法
- 访问修饰符 public protected private
- 属性列表＋构造函数 简写例子
- 继承 class C extends A {}
- 接口约束类 class C implements I {}

Abstract Class

- 抽象概念 不能实例化 abstract class C{}
- 提取子类共有成员 解决重复代码
- 抽象成员 强约束子类必须实现
  - 抽象属性 abstract widthConfig: { [key:string]: number };
  - 抽象方法 abstract refreshStudentList();

泛型

- 类型变量 定义时无法预知具体类型 调用时确定
- 附属于函数 / 类 / 接口 / 类型别名 <T>
- 解除功能和类型之间的耦合
  - function f<T>(arr:T[]):T[]{ return arr.sort() }
    f<number>([2,1,3]); f<string>('c', 'a', 'b');
- 泛型约束 限制取值范围 <T extends I>

Q & A