

# TypeScript TS「面试题及答案」不断更新

周姐日常事 LV.3

2021年08月24日 21:20 · 阅读 16721

关注



@稀土掘金技术社区

本文只是个人对[# 钉钉前端面试题](#)中 TypeScript面试题，根据网络中的答案进行补充和整理。

## 1. 什么是TypeScript?

Typescript 是一个强类型的 JavaScript 超集，支持ES6语法，支持面向对象编程的概念，如类、接口、继承、泛型等。Typescript并不直接在浏览器上运行，需要编译器编译成纯Javascript来运行。

## 2. 为什么要使用 TypeScript ? TypeScript 相对于 JavaScript 的优势是什么?

增加了静态类型，可以在开发人员编写脚本时检测错误，使得代码质量更好，更健壮。

优势:

1. 杜绝由于变量名写错



196



12



收藏

### 3. TypeScript 中 const 和 readonly 的区别？枚举和常量枚举的区别？接口和类型别名的区别？

**const 和 readonly** : const可以防止变量的值被修改，readonly可以防止变量的属性被修改。

**枚举和常量枚举** : 常量枚举只能使用常量枚举表达式，并且不同于常规的枚举，它们在编译阶段会被删除。常量枚举成员在使用的地方会被内联进来。之所以可以这么做是因为，常量枚举不允许包含计算成员。

**接口和类型别名** : 两者都可以用来描述对象或函数的类型。与接口不同，类型别名还可以用于其他类型，如基本类型（原始值）、联合类型、元组。

### 4. TypeScript 中 any 类型的作用是什么？

为编程阶段还不清楚类型的变量指定一个类型。这些值可能来自于动态的内容，比如来自用户输入或第三方代码库。这种情况下，我们不希望类型检查器对这些值进行检查而是直接让它们通过编译阶段的检查。

### 5. TypeScript 中 any、never、unknown、null & undefined 和 void 有什么区别？

**any** : 动态的变量类型（失去了类型检查的作用）。

**never** : 永不存在的值的类型。例如：never 类型是那些总是会抛出异常或根本就不会有返回值的函数表达式或箭头函数表达式的返回值类型。

**unknown** : 任何类型的值都可以赋给 unknown 类型，但是 unknown 类型的值只能赋给 unknown 本身和 any 类型。

**null & undefined** : 默认情况下 null 和 undefined 是所有类型的子类型。就是说你可以把 null 和 undefined 赋值给 number 类型的变量。当你指定了 --strictNullChecks 标记，null 和 undefined 只能赋值给 void 和它们各自。

**void** : 没有任何类型。例如：一个函数如果没有返回值，那么返回值可以定义为void。

### 6. TypeScript 中 interface 可以给 Function / Array / Class (Indexable) 做声明吗？

```
// 函数声明
interface Say {
  (name: string): void;
}
let say: Say = (name: string):void => {}
// Array 声明
interface NumberArray {
  [index: number]: number;
}
let fibonacci: NumberArray = [1, 1, 2, 3, 5];
// Class 声明
interface PersonalIntl {
  name: string
  sayHi (name: string): string
}
```

## 7. TypeScript 中可以使用 String、Number、Boolean、Symbol、Object 等给类型做声明吗？

```
/* 可以 */
let name: string = "bob";
let decliteral: number = 6;
let isDone: boolean = false;
let sym: symbol = Symbol();
interface Person {
  name: string;
  age: number;
}
```

[javascript 复制代码](#)

## 8. TypeScript 中的 this 和 JavaScript 中的 this 有什么差异？

1. TypeScript: noImplicitThis: true 的情况下，必须去声明 this 的类型，才能在函数或者对象中使用this。
2. Typescript 中箭头函数的 this 和 ES6 中箭头函数中的 this 是一致的。

## 9. TypeScript 中使用 Union Types 时有哪些注意事项？

们不能切问此软口安全的所有安全主共有的岗位或力云。

js 复制代码

```
function getLength(something: string | number): number {
    return something.length;
}

// index.ts(2,22): error TS2339: Property 'length' does not exist on type '>'string | number'
//   Property 'length' does not exist on type 'number'.
```

```
function getString(something: string | number): string {
    return something.toString();
}

// 公共方法和属性可以访问
```

## 10. TypeScript 如何设计 Class 的声明?

ts 复制代码

```
class Greeter {
    greeting: string;
    constructor(message: string) {
        this.greeting = message;
    }
    greet(): string{
        return "Hello, " + this.greeting;
    }
}

let greeter = new Greeter("world");

// 在声明类的时候，一般类中都会包含，构造函数、对构造函数中的属性进行类型声明、类中的方法。
```

## 11. TypeScript 中如何联合枚举类型的 Key?

ts 复制代码

```
enum str {
    A,
    B,
    C
}

type strUnion = keyof typeof str; // 'A' | 'B' | 'C'
```

1. 都可以描述 对象 或有 函数

2. 都允许拓展(extends)

不同点:

1. type 可以声明基本类型, 联合类型, 元组
2. type 可以使用 typeof 获取实例的类型进行赋值
3. 多个相同的 interface 声明可以自动合并

使用 interface 描述'数据结构', 使用 type 描述'类型关系'

### 13. TypeScript 中 ?.、??、!、!..、\_、\*\* 等符号的含义?

**?. 可选链** 遇到 null 和 undefined 可以立即停止表达式的运行。

**?? 空值合并运算符** 当左侧操作数为 null 或 undefined 时, 其返回右侧的操作数, 否则返回左侧的操作数。

**! 非空断言运算符** x! 将从 x 值域中排除 null 和 undefined

**!.** 在变量名后添加, 可以断言排除undefined和null类型

**\_ 数字分割符** 分隔符不会改变数值字面量的值, 使人更容易读懂数字 .e.g 1\_101\_324。

**\*\* 求幂**

### 15. 简单介绍一下 TypeScript 模块的加载机制?

假设有一个导入语句 `import { a } from "moduleA";`

1. 首先, 编译器会尝试定位需要导入的模块文件, 通过绝对或者相对的路径查找方式;
2. 如果上面的解析失败了, 没有查找到对应的模块, 编译器会尝试定位一个 **外部模块声明** (.d.ts) ;
3. 最后, 如果编译器还是不能解析这个模块, 则会抛出一个错误 `error TS2307: Cannot find module 'moduleA'.`

### 16. 简单聊聊你对 TypeScript 类型兼容性的理解?

**ts 类型兼容:** 当一个类型 Y 可以赋值给另一个类型 X 时, 我们就可以说类型 X 兼容类型 Y。也就是说两者在结构上是一致的, 而不一定非得通过 extends 的方式继承而来

**接口的兼容性:  $X = Y$**  只要目标类型 X 中声明的属性变量在源类型 Y 中都存在就是兼容的 (Y 中的类型可以比 X 中的多, 但是不能少)

## 17. 协变、逆变、双变和抗变的理解？

**协变：**  $X = Y$  Y 类型可以赋值给 X 类型的情况就叫做协变，也可以说是 X 类型兼容 Y 类型

```
interface X { name: string; age: number; }
interface Y { name: string; age: number; hobbies: string[] }
let x: X = { name: 'xiaoming', age: 16 }
let y: Y = { name: 'xiaohong', age: 18, hobbies: ['eat'] }
x = y
```

[typescript](#) [复制代码](#)

**逆变：**  $\text{printY} = \text{printX}$  函数X 类型可以赋值给函数Y 类型，因为函数Y 在调用的时候参数是按照Y类型进行约束的，但是用到的是函数X的X的属性和方法，ts检查结果是类型安全的。这种特性就叫做逆变，函数的参数有逆变的性质。

```
let printY: (y: Y) => void
printY = (y) => { console.log(y.hobbies) }
let printX: (x: X) => void
printX = (x) => { console.log(x.name) }
printY = printX
```

[typescript](#) [复制代码](#)

**双变（双向协变）：**  $X = Y; Y = X$  父类型可以赋值给子类型，子类型可以赋值给父类型，既逆变又协变，叫做“双向协变”（ts2.x 之前支持这种赋值，之后 ts 加了一个编译选项 `strictFunctionTypes`，设置为 `true` 就只支持函数参数的逆变，设置为 `false` 则支持双向协变）

**抗变（不变）：** 非父子类型之间不会发生型变，只要类型不一样就会报错

## 18. TypeScript 中对象展开会有什么副作用吗？

1. 展开对象后面的属性会覆盖前面的属性；
2. 仅包含对象自身的可枚举属性，不可枚举的属性将会丢失。

## 19. 类型的全局声明和局部声明

反之，若是在「文件范围」 `import`、`export`，那么该「文件范围」的变量声明则会定向局部声明，不会影响到全局声明。

## 20. TypeScript 中同名的 interface 或者同名的 interface 和 class 可以合并吗？

同名的interface会自动合并，同名的interface和class会自动聚合。

## 21. 如何使 TypeScript 项目引入并识别编译为 JavaScript 的 npm 库包？

1. 选择安装 ts 版本， `npm install @types/包名 --save` ；
2. 对于没有类型的 js 库，需要编写同名的.d.ts文件

## 22. TypeScript 的 tsconfig.json 中有哪些配置项信息？

```
{
  "files": [],
  "include": [],
  "exclude": [],
  "compileOnSave": false,
  "extends": "",
  "compilerOptions": { ... }
}
```

[json](#) 复制代码

`files` 是一个数组列表，里面包含指定文件的相对或绝对路径，用来指定待编译文件，编译器在编译的时候只会编译包含在files中列出的文件。

`include & exclude` 指定编译某些文件，或者指定排除某些文件。

`compileOnSave: true` 让IDE在保存文件的时候根据tsconfig.json重新生成文件。

`extends` 可以通过指定一个其他的tsconfig.json文件路径，来继承这个配置文件里的配置。

`compilerOptions` 编译配置项，如何对具体的ts文件进行编译

## 23. TypeScript 中如何设置模块导入的路径别名？

通过 `tsconfig.json` 中的 `paths` 项来配置。

```
{
  "baseUrl": ".",
  "paths": {
    "@helper/*": ["src/helper/*"],
    "@utils/*": ["src/utils/*"],
    ...
  }
}
```

## 24. declare, declare global是什么?

`declare` 是用来定义全局变量、全局函数、全局命名空间、js modules、class等  
`declare global` 为全局对象 `window` 增加新的属性

[ts 复制代码](#)

```
declare global {
  interface Window {
    csrf: string;
  }
}
```

## 25. 对 TypeScript 类中成员的 public、private、protected、readonly 修饰符的理解?

`public`: 成员都默认为 `public`，被此限定符修饰的成员是可以被外部访问；  
`private`: 被此限定符修饰的成员是只可以被类的内部访问；  
`protected`: 被此限定符修饰的成员是只可以被类的内部以及类的子类访问；  
`readonly`: 关键字将属性设置为只读的。只读属性必须在声明时或构造函数里被初始化。

## 26. keyof 和 typeof 关键字的作用?

`keyof` 索引类型查询操作符 获取索引类型的属性名，构成联合类型。  
`typeof` 获取一个变量或对象的类型。



`Merge<O1, O2>` 是将两个对象的属性合并。

`Compute<A & B>` 是将交叉类型合并

`Intersection<T, U>` 的作用是取 `T` 的属性,此属性同样也存在与 `U` 。

`Overwrite<T, U>` 是用 `U` 的属性覆盖 `T` 的相同属性。

## 28. 数组定义两种方式

[ts](#) [复制代码](#)

```
type Foo= Array<string>;
interface Bar {
    baz: Array<{ name: string, age: number}>
}

type Foo = string[];
interface Bar {
    baz : { name: string, age: number }[]
}
```

分类: 前端      标签: [TypeScript](#)    [面试](#)

## 评论

输入评论 (Enter换行, Ctrl + Enter发送)

## 热门评论 🔥



罗罗卜卜卜卜 

8月前

加油, 继续放个屁股在这儿蹲蹲

👍 1    💬 1




洪川numb

7日前

👍 196

💬 12

★ 收藏

zzusongjinlei  

9月前

第一个阮一峰关于ts的书讲到“类型系统按照「是否允许隐式类型转换」来分类，可以分为强类型和弱类型。”而ts在运行状态下不会修改js的特性，所以认为ts弱类型，不知道作者怎么看。

 2  回复

## 全部评论 12

 最新 最热忘川之水  前端开发

14天前

type 可以使用 typeof 获取实例的类型进行赋值  
let div = document.createElement('div');  
type B = typeof div

 点赞  回复EricYangXD66  前端开发

20天前

 nice job 点赞  回复小黑豆豆   FE @ 996劳工剥削有限...

29天前

 点赞  回复忘川之水  前端开发

1月前

常量枚举其实就是是在 enum关键字前使用 const 修饰符，常量枚举会在编译阶段被移除

作用：当我们不需要一个对象，而需要对象的值，就可以使用常量枚举，这样就可以避免在编译时生成多余的代码和间接引用

 1  回复Steven果果呀   前端开发工程师

2月前

mark~

 点赞  回复 196 12 收藏

这几个类型怎么ts中找不到

属性合并。  
i并  
T 的属性,此属性  
覆盖 T 的相同属

👍 点赞    💬 回复



坚持前端求学道... LV.2 JY.3

4月前

有Merge这个工具类型吗?

👍 点赞    💬 回复



jskai LV.3 JY.2 前端工程师

5月前

占坑

👍 点赞    💬 回复



努力学习中的max LV.3 JY.3 前端开发

6月前

好文, 关注了~

👍 点赞    💬 回复



罗罗卜卜卜 JY.2

8月前

加油, 继续放个屁股在这儿蹲蹲

👍 1    💬 1



涉川|numb

7月前

起来了 一会腿麻了

👍 1    💬 回复



zzusongjinlei LV.2 JY.2

9月前

第一个阮一峰关于ts的书中讲到“类型系统按照「是否允许隐式类型转换」来分类,可以分为强类型和弱类型。”而ts在运行状态下不会修改js的特性,所以认为ts弱类型,不知道作者怎么看。

👍 2    💬 回复

呛再首 1月前 前端 TypeScript

## 2022年了，我才开始学 typescript，晚吗？（7.5k字总结）

3.7w 629 100

我是leon 2月前 TypeScript 前端 面试

## 你以为学会TypeScript了？先看看这16道题能做对多少先！

9479 189 22

lhvt 2年前 前端 TypeScript

## 再次研究一道网红typescript面试题

4847 10 评论

zxc\_神说要有光 5月前 前端 JavaScript TypeScript

## 为什么说 TypeScript 的火爆是必然？

6.7w 482 298

已禁用 4年前 CSS

## 50道 CSS 基础面试题（附答案）

2.7w 1314 18

前端阿林 6月前 前端 TypeScript

## 「1.9W字总结」一份通俗易懂的 TS 教程，入门 + 实战！

2.1w 620 67

Big shark@LX 9月前 前端 JavaScript TypeScript

## 最全的TypeScript学习指南

6.1w 1436 88

winty 2年前 面试 前端

## 面试题：说说事件循环机制(满分答案来了)

6.3w 1417 123

Jimmy\_fx 10月前 TypeScript 前端

## 2021 typescript史上最强学习入门文章(2w字)

8.6w 2031 112

郭东东 3年前 面试

## 中高级前端大厂面试秘籍，为你保驾护航金三银四，直通大厂(上)

61.7w 6742 484

夜尽天明丶 6月前 TypeScript JavaScript

## 接近天花板的TS类型体操，看懂你就能玩转TS了

2.3w 530 98

vortesnail 6月前 前端 面试

## 做了一份前端面试复习计划，保熟~

31.3w 6341 345

字节前端 1年前 TypeScript 前端

## TypeScript 高级用法

4.9w 1406 51

HearLing 1年前 面试

## 🐼化身面试官出30+Vue面试题，超级干货（附答案） | 牛气冲天新年征文

8.7w 1840 139

前端良文 1月前 TypeScript 面试

## 一起来刻意练习TypeScript类型体操吧(一)

1107 8 评论

苏苏同学 2月前 TypeScript 面试

## TypeScript学习之类型推断、类型断言、双重断言、非空断言、确定赋值断言、类型守卫、...

1323 21 1

yck 4年前 Node.js JavaScript 面试

## 几道高级前端面试题解析

5.2w

1952

80

伟大的兔神 11月前 TypeScript 前端

## 重学TypeScript

2.2w

605

45