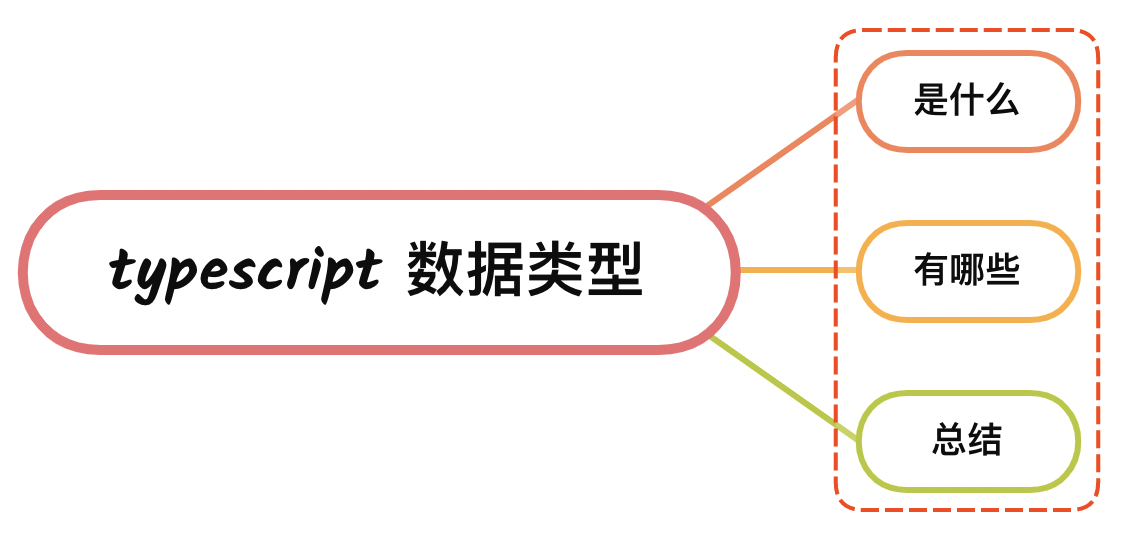
# 面试官：说说 typescript 的数据类型有哪些？



## 一、是什么

typescript 和 javascript几乎一样，拥有相同的数据类型，另外在javascript基础上提供了更加实用的类型供开发使用

在开发阶段，可以为明确的变量定义为某种类型，这样typescript就能在编译阶段进行类型检查，当类型不合符预期结果的时候则会出现错误提示

## 二、有哪些

typescript 的数据类型主要有如下：

* boolean（布尔类型）
* number（数字类型）
* string（字符串类型）
* array（数组类型）
* tuple（元组类型）
* enum（枚举类型）
* any（任意类型）
* null 和 undefined 类型
* void 类型
* never 类型
* object 对象类型

### boolean

布尔类型

let flag:boolean = true;  
// flag = 123; // 错误  
flag = false; //正确

### number

数字类型，和javascript一样，typescript的数值类型都是浮点数，可支持二进制、八进制、十进制和十六进制

let num:number = 123;  
// num = '456'; // 错误  
num = 456; //正确

进制表示：

let decLiteral: number = 6; // 十进制  
let hexLiteral: number = 0xf00d; // 十六进制  
let binaryLiteral: number = 0b1010; // 二进制  
let octalLiteral: number = 0o744; // 八进制

### string

字符串类型，和JavaScript一样，可以使用双引号（"）或单引号（'）表示字符串

let str:string = 'this is ts';  
str = 'test';

作为超集，当然也可以使用模版字符串``进行包裹，通过 ${} 嵌入变量

let name: string = `Gene`;  
let age: number = 37;  
let sentence: string = `Hello, my name is ${ name }

### array

数组类型，跟javascript一致，通过[]进行包裹，有两种写法：

方式一：元素类型后面接上 []

tsx let arr:string[] = ['12', '23']; arr = ['45', '56'];

方式二：使用数组泛型，Array<元素类型>：

tsx let arr:Array<number> = [1, 2]; arr = ['45', '56'];

### tuple

元祖类型，允许表示一个已知元素数量和类型的数组，各元素的类型不必相同

let tupleArr:[number, string, boolean];  
tupleArr = [12, '34', true]; //ok  
typleArr = [12, '34'] // no ok

赋值的类型、位置、个数需要和定义（生明）的类型、位置、个数一致

### enum

enum类型是对JavaScript标准数据类型的一个补充，使用枚举类型可以为一组数值赋予友好的名字

enum Color {Red, Green, Blue}  
let c: Color = Color.Green;

### any

可以指定任何类型的值，在编程阶段还不清楚类型的变量指定一个类型，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，这时候可以使用any类型

使用any类型允许被赋值为任意类型，甚至可以调用其属性、方法

let num:any = 123;  
num = 'str';  
num = true;

定义存储各种类型数据的数组时，示例代码如下：

let arrayList: any[] = [1, false, 'fine'];  
arrayList[1] = 100;

### null 和 和 undefined

在JavaScript 中 null表示 "什么都没有"，是一个只有一个值的特殊类型，表示一个空对象引用，而undefined表示一个没有设置值的变量

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

let num:number | undefined; // 数值类型 或者 undefined  
console.log(num); // 正确  
num = 123;  
console.log(num); // 正确

但是ts配置了--strictNullChecks标记，null和undefined只能赋值给void和它们各自

### void

用于标识方法返回值的类型，表示该方法没有返回值。

function hello(): void {  
 alert("Hello Runoob");  
}

### never

never是其他类型 （包括null和 undefined）的子类型，可以赋值给任何类型，代表从不会出现的值

但是没有类型是 never 的子类型，这意味着声明 never 的变量只能被 never 类型所赋值。

never 类型一般用来指定那些总是会抛出异常、无限循环

let a:never;  
a = 123; // 错误的写法  
  
a = (() => { // 正确的写法  
 throw new Error('错误');  
})()  
  
// 返回never的函数必须存在无法达到的终点  
function error(message: string): never {  
 throw new Error(message);  
}

### object

对象类型，非原始类型，常见的形式通过{}进行包裹

let obj:object;  
obj = {name: 'Wang', age: 25};

## 三、总结

和javascript基本一致，也分成：

* 基本类型
* 引用类型

在基础类型上，typescript增添了void、any、emum等原始类型

## 参考文献

* https://www.tslang.cn/docs/handbook/basic-types.html