TypeScript

基础类型：

let 变量名:类型=值

值可以是模板字符串，模板字符串被反引号**`**包围。可以定义多行文本和内嵌表达式，内嵌表达式被${}包围。

类型为数组时可用：基本类型[]或Array<基本类型>

元组Tuple：[类型1,类型2,……]

枚举：enum Color {Red = 6 , Green , Blue};

let c:Color = Color.Green;

都不赋值默认从0开始，赋值后依次加1，可手动赋不一样的值

任意值：让不清楚类型的变量通过编译阶段的检查，用any（Object类型的变量只允许赋任意值，不允许调用任意方法）

空值：void，当变量用void时只能赋予undefined和null

undefined和null，当指定--strictNullChecks时，undefined和nul只能赋给void和它们各自，当未指定时，还可以赋给别的任何变量

never：返回never的函数存在无法达到的终点

类型断言：把any转化为已知<类型>值或值 as 类型（当在TypeScript中使用jsx时，只允许这种）

变量声明

var的环境保存在function内，let的环境保存在最近的{}内

var可不声明，可先用再声明，可多次声明，let必须先声明再用，必须单次声明

const声明的变量赋值后不能再改变，const声明的变量若为对象，则对象内的属性可以改变

解构数组：

[数组第一个值的别名，数组第二个值的别名， ，数组第四个值的别名，…数组剩下值的别名]

解构对象：

接口：通过结构性子类型检查是否满足其描述，类型检查器不检查属性的顺序，只检查属性是否存在及属性类型是否正确

1 可选属性：在属性名和 : 之间加 ?

2 只读属性：在属性名前加readonly。ReadonlyArray<T>和Array<T>相似，前者把所有可变方法去掉

let a : number[] = [1 , 2 , 3]

let b : ReadonlyArray<number> = a

a = b ; 错

a = b as number[] ; 对

变量用const，属性用readonly

3 额外的属性检查：当对象字面量中属性存在而接口中该属性不存在，且直接赋值会报错。解决（1）用变量中转（2）将对象字面量类型断言（3）接口使用索引签名

4 函数类型：(参数名实际可与它不一样:参数类型):返回类型

构造函数类型：new (参数名实际可与它不一样:参数类型):返回类型

5 索引类型：[参数名无实际意义:索引类型]:返回类型

（1）若接口中既有number索引类型又有string索引类型，则number的返回值应为string返回值的子类型。因为number会转成string去索引

（2）obj.property和obj[”property”]一致

6 类类型：类包含静态部分（中有构造函数）和实例部分，当类implements类类型的接口后只对其实例部分进行类型检查

7 扩展接口：通过extends继承一个或以 , 分割的多个接口

8 接口继承类：继承类的成员但不包括其实现。当接口继承了类中包含private或protect时，这个接口只能被这个类或其子类所实现

类

（1）构造函数：constructor(){}

调用父类构造函数：super();

1. 默认为public
2. private：只能在声明它的类内访问
3. protect：在声明它的类及派生类内访问
4. 当类中含有private或protect时，与它能兼容的类必须也含有这个private或protect，且双方来自同一个private或protect
5. 存取器：set get
6. 抽象类：抽象类和抽象方法前有abstract，抽象方法和接口方法都不包含具体实现，而抽象方法必须包含abstract关键字且可以包含访问修饰符
7. :typeof 类名(是类类型，拥有静态部分)=类名;

: 类名(是实例类型，拥有实例部分)=new 类名();

函数

（1）当没有可选参数时，传入函数的参数个数必须与函数期望的参数个数一致

（2）当有可选参数时，可选参数必须跟在必须参数后面

（3）所有必须参数后的默认参数是可选的，这部分的函数签名用可选参数形式。所有必须参数前的默认参数必须传入undefined来获得默认值

（4）剩余参数：…参数名:参数类型[]

（5）this：

()=>{} 会在函数创建的时候指定this值，而不是在函数调用的时候。

若this在对象字面量里，则类型为any，可用this:对象字面量名为

参数传入对象字面量方法中指出this为对象字面量名的类型