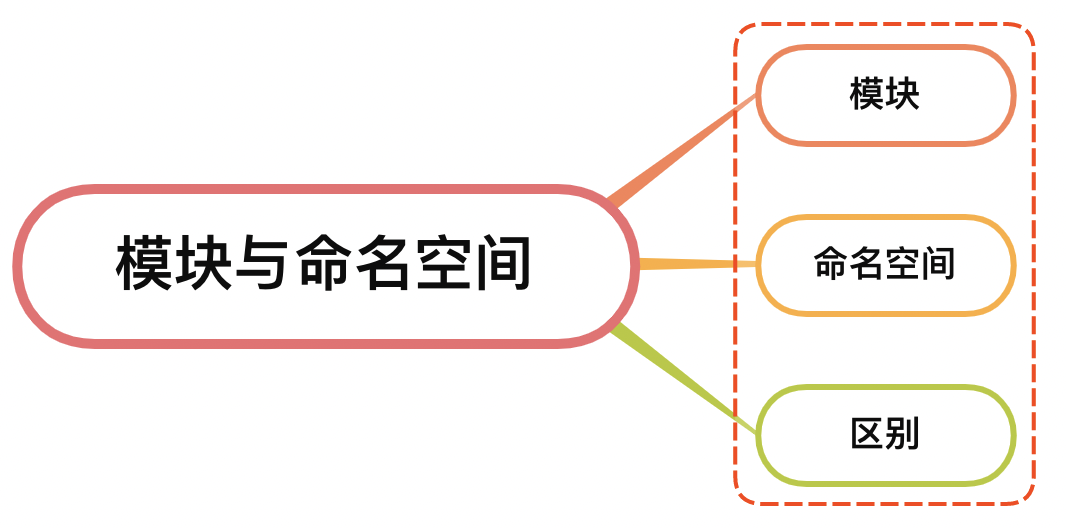
# 面试官：说说对 TypeScript 中命名空间与模块的理解？区别？



## 一、模块

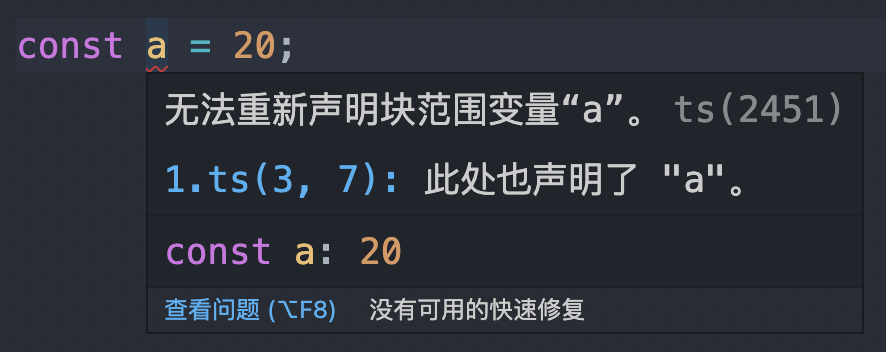
TypeScript 与ECMAScript 2015 一样，任何包含顶级 import 或者 export 的文件都被当成一个模块

相反地，如果一个文件不带有顶级的import或者export声明，那么它的内容被视为全局可见的

例如我们在在一个 TypeScript 工程下建立一个文件 1.ts，声明一个变量a，如下：

const a = 1

然后在另一个文件同样声明一个变量a，这时候会出现错误信息



提示重复声明a变量，但是所处的空间是全局的

如果需要解决这个问题，则通过import或者export引入模块系统即可，如下：

const a = 10;  
  
export default a

在typescript中，export关键字可以导出变量或者类型，用法与es6模块一致，如下：

export const a = 1  
export type Person = {  
 name: String  
}

通过import 引入模块，如下：

import { a, Person } from './export';

## 二、命名空间

命名空间一个最明确的目的就是解决重名问题

命名空间定义了标识符的可见范围，一个标识符可在多个名字空间中定义，它在不同名字空间中的含义是互不相干的

这样，在一个新的名字空间中可定义任何标识符，它们不会与任何已有的标识符发生冲突，因为已有的定义都处于其他名字空间中

TypeScript 中命名空间使用 namespace 来定义，语法格式如下：

namespace SomeNameSpaceName {  
 export interface ISomeInterfaceName { }  
 export class SomeClassName { }  
}

以上定义了一个命名空间 SomeNameSpaceName，如果我们需要在外部可以调用 SomeNameSpaceName 中的类和接口，则需要在类和接口添加 export 关键字

使用方式如下：

SomeNameSpaceName.SomeClassName

命名空间本质上是一个对象，作用是将一系列相关的全局变量组织到一个对象的属性，如下：

namespace Letter {  
 export let a = 1;  
 export let b = 2;  
 export let c = 3;  
 // ...  
 export let z = 26;  
}

编译成js如下：

var Letter;  
(function (Letter) {  
 Letter.a = 1;  
 Letter.b = 2;  
 Letter.c = 3;  
 // ...  
 Letter.z = 26;  
})(Letter || (Letter = {}));

## 三、区别

* 命名空间是位于全局命名空间下的一个普通的带有名字的 JavaScript 对象，使用起来十分容易。但就像其它的全局命名空间污染一样，它很难去识别组件之间的依赖关系，尤其是在大型的应用中
* 像命名空间一样，模块可以包含代码和声明。 不同的是模块可以声明它的依赖
* 在正常的TS项目开发过程中并不建议用命名空间，但通常在通过 d.ts 文件标记 js 库类型的时候使用命名空间，主要作用是给编译器编写代码的时候参考使用

## 参考文献

* https://www.tslang.cn/docs/handbook/modules.html
* https://www.tslang.cn/docs/handbook/namespaces.html
* https://www.tslang.cn/docs/handbook/namespaces-and-modules.html