**TypeScript 声明文件**

假如我们想使用第三方库，比如 jQuery，我们通常这样获取一个 id 是 foo 的元素：

$('#foo');

// 或

jQuery('#foo');

但是在 TypeScript 中，我们并不知道 $ 或 jQuery 是什么东西：

这时，我们需要使用 declare 关键字来定义它的类型，帮助 TypeScript 判断我们传入的参数类型对不对：

declare var jQuery: (selector: string) => any;

jQuery('#foo');

declare 定义的类型只会用于编译时的检查，编译结果中会被删除。

上例的编译结果是：

jQuery('#foo');

**声明文件**

声明文件以 **.d.ts** 为后缀，例如：

runoob.d.ts

声明文件或模块的语法格式如下：

declare module Module\_Name {

}

TypeScript 引入声明文件语法格式：

/// <reference path = " runoob.d.ts" />

当然，很多流行的第三方库的声明文件不需要我们定义了，比如 jQuery 已经有人帮我们定义好了：[jQuery in DefinitelyTyped](https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/jquery/index.d.ts" \t "_blank)。

**实例**

以下定义一个第三方库来演示：

CalcThirdPartyJsLib.js 文件代码：

var Runoob;

(function(Runoob) {

    var Calc = (function () {

        function Calc() {

        }

    })

    Calc.prototype.doSum = function (limit) {

        var sum = 0;

        for (var i = 0; i <= limit; i++) {

            sum = sum + i;

        }

        return sum;

    }

    Runoob.Calc = Calc;

    return Calc;

})(Runoob || (Runoob = {}));

var test = new Runoob.Calc();

如果我们想在 TypeScript 中引用上面的代码，则需要设置声明文件 Calc.d.ts，代码如下：

Calc.d.ts 文件代码：

declare module Runoob {

    export class Calc {

        doSum(limit: number): number;

    }

}

声明文件不包含实现，它只是类型声明，把声明文件加入到 TypeScript 中：

CalcTest.ts 文件代码：

/// <reference path = "Calc.d.ts" />

var obj = new Runoob.Calc();

// obj.doSum("Hello"); // 编译错误

console.log(obj.doSum(10));

使用 tsc 命令来编译以上代码文件：

tsc CalcTest.ts