# 一 环境

本地环境：npm install -g typescript

在线环境：<http://www.typescriptlang.org/play/index.html>

# 二 HelloWorld

编写Hello.ts文件：

export class Hello {

}

在命令行使用命令：tsc Hello.ts 就可以将该ts文件编译为js文件：

"use strict";

exports.\_\_esModule = true;

var Hello = /\*\* @class \*/ (function () {

function Hello() {

}

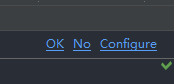
return Hello;

}());

exports.Hello = Hello;

Webstorme的集成：

新建ts文件时，会出现提示



点击OK即可。

# 三 字符串特性

*//字符串模板作为参数传递--不需要方法的小括号*function test(templateStr,name,age){  
 console.log(templateStr);  
 console.log(name);  
 console.log(age);  
}  
let myname = 'lisi';  
let getAge = function () {  
 return 18;  
};  
test`name is ${myname},age is ${getAge()}`;

# 四 类型检查

## 1 常见类型

let myname:string = 'lisi';  
*//有了类型的变量在作为参数时，会被检查类型*myname = 13; *//这里会有错误提醒，但是实际编译完毕后，仍然会执行，因为JS不报错*

常用的类型是：number boolean string void any

*//类型定位any，就可以赋任何值，不会报错*let alias:any = "test";  
alias = 13;

*//void表示函数没有返回值*function test():void {  
 return 3; *//返回了值会报错*}

## 2 自定义类型

定义一个Person类型：

class Person {  
 name:string;  
 age:number;  
}  
let zs:Person = new Person();  
zs.name = 'lisi';  
zs.age = 18;

## 3 参数默认值

function test(a:string,b:string = 'hi'){  
 console.log(a);  
 console.log(b);  
}  
test('xx','yy');  
test('xx');

注意：默认值必须在参数列表最后面。

## 4 可选参数

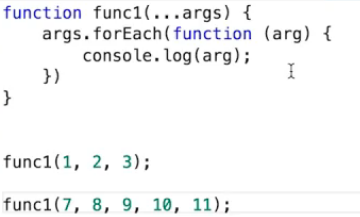
function test(a:string,b?:string,c:string = 'hi'){  
 console.log(a);  
 console.log(b);  
 console.log(c);  
}  
test('xx'); *//一个参数也可以执行，因为b可选  
//此时输出b的值为undefined*

声明了可选参数，需要在函数体内部说明没有传入可选参数，可选参数应该如何处理；

可选参数只能写在必选参数后面，不能写在默认参数后面。

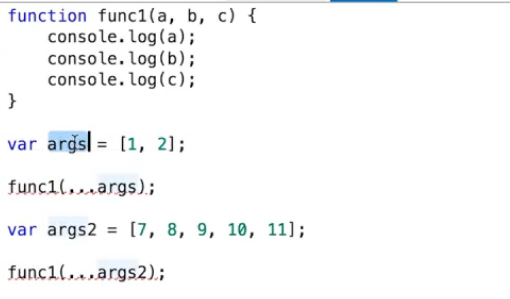
## 5 Rest 和 Spread 操作符

### 5.1 用来声明任意数量的方法参数



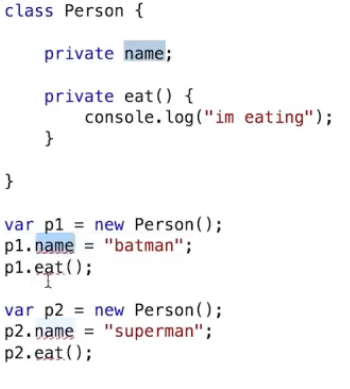
### 5.2 把任意长度数组转化为固定数量参数

typescript未支持，但是编译后的js能够执行



# 五 面向对象

## 1 类的私有成员 private



如上所示，私有成员访问会报错。

注意：如果没有修饰符，则默认都是public，但是在构造函数中，有所区别。

如果构造函数的参数没有修饰符，那么在构造函数外部是无法使用这个参数的；

如果构造函数的参数使用了public修饰符，那么在构造函数外部可以使用该参数

## 2 类的保护成员 protect

protect修饰的成员也不能被外部访问，但是可以被子类继承。

## 3 泛型

export class Person {  
 constructor(public name:string){  
 console.log('constructor');  
 }  
 work(){  
 console.log(this.name + 'can work');  
 }  
}  
*//定义一个空数组，这个数组内的元素必须都是Person*let workers: Array<Person> = [];  
workers[0] = new Person('lisi');  
console.log(workers);

## 4 接口

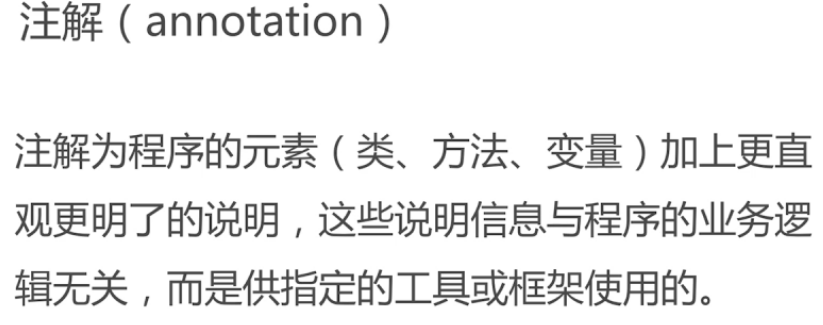
### 4.1 实现接口属性

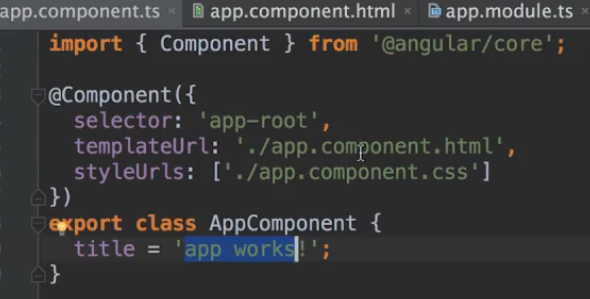
interface Person {  
 name:string;  
 age:number;  
}  
  
class Student {  
 *//构造函数支持接口* constructor(public config:Person){  
  
 }  
}  
  
*//在创建对象时，必须满足接口定义的属性*let s1 = new Student({  
 name:'lisi',  
 age:18  
});

### 4.2 实现接口

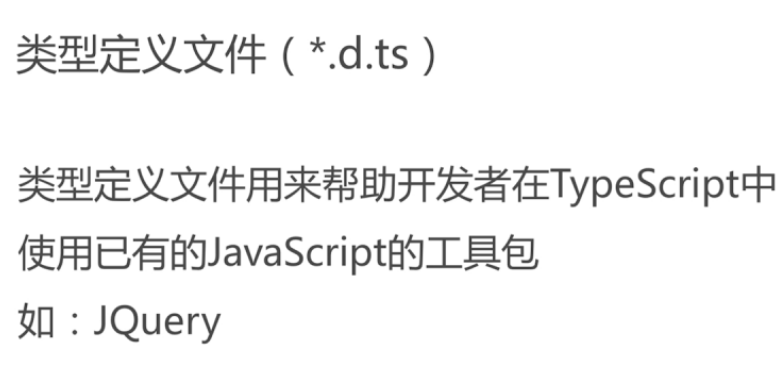
interface Person {  
 name:string;  
 age:number;  
 run();  
}  
  
class Student implements Person{  
 name:string;  
 age:number;  
 constructor(){  
  
 }  
 *//实现接口方法* run(){  
 console.log(this.name + ' run....');  
 }  
}  
  
let s1 = new Student();  
s1.name = 'lisi';  
s1.age = 15;  
s1.run();

## 5 注解





## 6 类型定义文件



这个文件其实是一个typescript文件模块，提供给开发者直接使用，比如引入jQuery的类型定义文件后，就可以使用$与jQuery的方法：

