**Typescript**

<https://www.tslang.cn/index.html>

TypeScript 是一种由微软开发的自由和开源的编程语言。它是 JavaScript 的一个超集，TypeScript 在 JavaScript 的基础上添加了可选的静态类型和基于类的面向对象编程。

# 一．开发环境配置

1.安装nodejs

**node –v 查看版本号**

2.下载typescript

**npm install typescript –g（安装ts）**

**tsc –version（查版本）**

**npm init –y（创建package.json）**

**tsc –init(创建tsconfig.json)**

**npm install @types/node --dev-save（安装@types/node，主要是解决模块的声明文件问题，同时安装node\_modules）**

3.新建文件helloWorld.ts

var a:string = "HelloWorld"

console.log(a)

**终端-->运行生成任务--> tsc:构建 – tsconfig.json**

**点击构建后，tsconfig.json配置，生成一个helloWorld.js文件（严格模式下）**

# 二．分类代码编写

4.变量类型的那些事

**TypeScript最大的一个特点就是变量是强类型的，也就是说，在声明变量的时候，我们必须给他一个类型。比如：字符串、数字、布尔，枚举........**

**创建demo03.ts书写创建模型**

//声明数值类型的变量age，但不予赋值

//number类型

var age:number = 18

var stature:number = 178.5

console.log(age)

console.log(stature)

console.log('----------------------')

//string类型

var jspangg:string = "技术胖 jspang.com"

console.log(jspangg)

//bool类型 true/false

var b:boolean = true

var c:boolean = false

console.log('----------------------')

//enum类型（枚举类型）

//世界上人的类型：男人、女人、中性

//一年的季节：春、夏、秋、冬 ，有四个结果。

// enum REN{ nan , nv ,yao}

// console.log(REN.yao)  //返回了2，这是索引index，跟数组很想。

enum REN{

    nan = '男',

    nv = '女',

    yao= '妖'

}

console.log(REN.yao)  //返回了妖 这个字

console.log('----------------------')

//any类型

var t:any =10

t = "jspang"

t = true

console.log(t)

console.log('----------------------')

//null类型

**通过node demo03.js运行**

5.typescript的函数

我们可以把功能相近的需求封装成一个独立的代码块，每次传入不同的变量或参数，就可以实现不同的结果。

**创建demo04.ts书写创建模型**

// function searchXiaoJieJie(age:number):string{

//     return '找到了'+age+'岁的小姐姐'

// }

// var age:number = 28

// var result:string = searchXiaoJieJie(age)

// console.log(result)

//有可选参数的函数

// function searchXiaoJieJie2(age:number,stature?:string):string{

//     let yy:string = ''

//     yy = '找到了'+age+'岁'

//     if(stature !=undefined){

//         yy = yy + stature

//     }

//     return yy+'的小姐姐'

// }

// var result:string  =  searchXiaoJieJie2(22,'大长腿')

// console.log(result)

//有默认参数的函数

// function searchXiaoJieJie2(age:number=18,stature:string='大胸'):string{

//     let yy:string = ''

//     yy = '找到了'+age+'岁'

//     if(stature !=undefined){

//         yy = yy + stature

//     }

//     return yy+'的小姐姐'

// }

// var result:string  =  searchXiaoJieJie2()

// console.log(result)

//有剩余参数的函数

function searchXiaoJieJie3(...xuqiu:string[]):string{

    let  yy:string = '找到了'

    for (let i =0;i<xuqiu.length;i++){

        yy = yy + xuqiu[i]

        if(i<xuqiu.length-1){

            yy=yy+'、'

        }

    }

    yy=yy+'的小姐姐'

    return yy

}

var result:string  =  searchXiaoJieJie3('22岁','大长腿','瓜子脸','水蛇腰')

console.log(result)

6.三种函数的定义方式

**创建demo05.ts书写创建模型**

//函数声明法

// function add(n1:number,n2:number):number{

//     return n1+n2

// }

// console.log(add(1,2))

//函数表达式法

// var add = function(n1:number,n2:number):number{

//     return n1+n2

// }

// console.log(add(1,4))

//箭头函数

var add = (n1:number,n2:number):number=>{

    return n1+n2

}

console.log(add(2,2))

7.函数中变量的作用域

**创建demo06.ts书写创建模型**

//变量的作用域，函数划分    void空值

// var yangzi = '刘德华'

// function zhengXing():void{

//     console.log('技术胖整形成了'+yangzi+'的样子')

// }

// zhengXing()

// console.log(yangzi)

//全局变量，局部变量

// var yangzi = '刘德华'

// function zhengXing():void{

//     var yangzi:string = '马德华'

//     console.log('技术胖整形成了'+yangzi+'的样子')

// }

// zhengXing()

// console.log(yangzi)

// var yangzi:string = '刘德华'

// function zhengXing():void{

//     console.log('技术胖整形成了'+yangzi+'的样子')     //此处yangzi变量提升undefined

//     var yangzi:string = '马德华'

//     console.log('技术胖整形成了'+yangzi+'的样子')

// }

// zhengXing()

// console.log(yangzi)

//let

function zhengXing(): void {

    var yangzia: string = '刘德华'

    {

        let yangzib: string = '小沈阳'

          //因为ts编译成js，他自动给我们加了ES5的处理，ES5里是没有let关键字的,js中自动转为var可正常使用

        console.log('技术胖整形成了' + yangzib + '的样子')

    }

    console.log('技术胖整形成了' + yangzia + '的样子')

    console.log('技术胖整形成了' + yangzib + '的样子')

}

zhengXing()

**attention：**

在函数里用var定义一个yangzi的变量,我们再函数的外部读取这个变量，你会发现是读取不到的

当局部变量与全局变量重名的时候，在函数体内是局部变量起作用；如果重名，就有变量提升undefined，这是一个坑

当有let时，因为ts编译成js，他自动给我们加了ES5的处理，ES5里是没有let关键字的,js中自动转为var可正常使用

8.引用类型：数组

TypeScript中的数据分为值类型和引用类型。值类型我们前几节课讲的都是，但是引用类型我们在这套视频中还没有讲解

js中的值类型和引用类型的区别：<https://www.cnblogs.com/leiting/p/8081413.html>

**创建demo07.ts书写创建模型**

// let jspang = {

//     name:'技术胖',

//     website:'jspang.com',

//     age:18,

//     saySometing:function(){

//         console.log('为了前端技术')

//     }

// }

// console.log(jspang.name)

// jspang.saySometing()

//数组--- Array String Date RegExp

// let arr1:number[ ]     //声明一个数值类型的数组

// let arr2:Array<string>  //声明一个字符串类型的数组

//字面量赋值法

//构造函数赋值法

//字面量赋值法

//定义一个空数组，数组容量为0

// let arr1:number[] = []

// //定义一个数组时，直接给数组赋值

// let arr2:number[] = [1,2,3,4,5]

// //定义数组 的同事给数组赋值

// let arr3:Array<string> = ['jspang','技术胖','金三胖']

// let arr4:Array<boolean> = [ true,false,false]

//构造函数赋值法

let arr1:number[] = new Array()

let ara2:number[] = new Array(1,2,3,4,5)

let arr3:Array<string> = new Array('jspang','技术胖','金三胖')

let arr4:Array<boolean> = new Array(true,false,false)

//声明一个元祖类型

let x : [string,number]

//正确的初始化

x = ['hello',10]

//错误的初始化方法

x = [10,'hello']

元祖是一种特殊的数组，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

9.引用类型：字符串

* 基本类型字符串：由单引号或者双引号括起来的一串字符串。
* 引用类型字符串：用new 实例化的 String类型。

**创建demo08.ts书写创建模型**

// 基本类型字符串：由单引号或者双引号括起来的一串字符串

// 引用类型字符串：用new 实例化的 String类型

// var jspang:string = "技术胖"

// let jspanga:String = new String("jspang.com")

// console.log(jspang)

// console.log(jspanga)

// console.log(jspang.length)

// console.log(jspanga.length)

//查找字符串

// let something:string = "清早起来打开窗，心情美美哒，我要出去找小姐姐，心情美美哒。"

// let xiaoJieJie:string = "美美哒"

// let jspang:string = "js胖"

// console.log(something.indexOf(xiaoJieJie))   //10

// console.log(something.lastIndexOf(xiaoJieJie)) //25

// console.log(something.indexOf(jspang)) //查找不到，返回-1

//截取字符串

// let something:string = "清早起来打开窗，心情美美的，我要出去找小姐姐，心情美美的。"

// let xiaoJieJie:string = "小姐姐"

// console.log(something.substring(8)) //到index8

// console.log(something.substring(8,14)) //从index8到14+1

//替换字符串

let something:string = "清早起来打开窗，心情美美的，我要出去找小姐姐，心情美美的。"

let xiaoJieJie:string = "小姐姐"

console.log(something.replace(xiaoJieJie,'小哥哥'))

**//字符串常用方法：**

**查找/截取/替换**

10.引用类型：日期对象

TypeScript中使用Date这个引用类型来存储日期对象，如果你要声明一个日期变量时，记得也要注明它的类型是Date

**创建demo09.ts书写创建模型**

//不传任何参数

// let d:Date = new Date()

// console.log(d)   //当前时间 2020-04-25T19:24:22.407Z

//传递一个整数

// let d:Date = new Date(1000)

// let da:Date = new Date(2000)

// console.log(d)  //1970-01-01T00:00:01.000Z

// console.log(da) //1970-01-01T00:00:02.000Z

//传递一个字符串

// let d1:Date = new Date('2018/09/06 05:30:00')

// let d2:Date = new Date('2018-09-06 05:30:00')

// let d3:Date = new Date('2018-09-06T05:30:00')

// console.log(d1)

// console.log(d2)

// console.log(d3)

//均为   2018-09-05T21:30:00.000Z  （UTC+8:00  北京时间）

//传递表示年月日分秒的变量

// let d:Date = new Date(year,month,day,hours,minutes,seconds,ms);

// year 表示年份，4位数字。

// month表示月份，数值是0(1月)~11(12月)之间的整数。

// day 表示日期。数值是1~31之间的整数。

// hours 表示小时，数值是0-23之间的整数。

// minutes 表示分钟数，数值是0~59之间的整数。

// seconds 表示秒数，数值是0~59之间的整数。

// ms 表示毫秒数，数值是0~999之间的整数

let d:Date = new Date(2000,9,20,13,21,58,666);     //2000-10-20T05:21:58.666Z

console.log(d)

11.引用类型：正则表达式

正则表达式：<https://www.jianshu.com/p/e0ea541aa773>

**创建demo10.ts书写创建模型**

//构造函数声明法

// let reg1:RegExp = new RegExp("Jspang")  //表示字符串规则里含有jspang

// console.log(reg1)

// let reg2:RegExp = new RegExp("jspang",'gi')

// console.log(reg2)

// //字面量声明（一般用此种方式）

// let reg3:RegExp = /jspang/

// let reg4:RegExp = /jspang/gi

// RegExp对象包含两个方法：test( )和exec( ),功能基本相似，用于测试字符串匹配。

// test(string) ：在字符串中查找是否存在指定的正则表达式并返回布尔值，

// 如果存在则返回 true，不存在则返回 false。

// exec(string) : 用于在字符串中查找指定正则表达式，如果 exec() 方法执行成功，

// 则返回包含该查找字符串的相关信息数组。如果执行失败，则返回 null。

//test

// let reg1:RegExp =  /jspang/i

// let website:string = 'jspang.com'

// let result1:boolean = reg1.test(website)

// console.log(result1)    //true

//exec

let reg1:RegExp =  /jspang/i

let website:string = 'jspang.com'

console.log(reg1.exec(website))

//[ 'jspang', index: 0, input: 'jspang.com', groups: undefined ]

RegExp对象包含两个方法：test( )和exec( ),功能基本相似，用于测试字符串匹配。

* **test(string)** ：在字符串中查找是否存在指定的正则表达式并返回布尔值，如果存在则返回 true，不存在则返回 false。
* **exec(string)** : 用于在字符串中查找指定正则表达式，如果 exec() 方法执行成功，则返回包含该查找字符串的相关信息数组。如果执行失败，则返回 null。

# 三．进阶面向对象

12.面向对象编程：类的声明和使用

我们先用class关键字声明了一个类，并在里边声明了name和age属性。constructor为构造函数。构造函数的主要作用是给类中封装的属性进行赋值

**创建demo11.ts书写创建模型**

//类是对象具体事物的一个抽象，对象是类的具体表现

class XiaoJieJie{

    name:string;

    age:number;

    constructor(name:string,age:number){

        this.name = name

        this.age = age

    }

    say(){

        console.log('小哥哥好')

    }

}

let jiejie:XiaoJieJie = new XiaoJieJie('范冰冰',18)

console.log(jiejie)

jiejie.say()

// XiaoJieJie { name: '范冰冰', age: 18 }

// 小哥哥好

**生成的demo11.js**

"use strict";

//类是对象具体事物的一个抽象，对象是类的具体表现

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

    function XiaoJieJie(name, age) {

        this.name = name;

        this.age = age;

    }

    XiaoJieJie.prototype.say = function () {

        console.log('小哥哥好');

    };

    return XiaoJieJie;

}());

var jiejie = new XiaoJieJie('范冰冰', 18);

console.log(jiejie);

jiejie.say();

// XiaoJieJie { name: '范冰冰', age: 18 }

// 小哥哥好

13.面向对象编程：修饰符

TypeScript语言和Java还有C#很像（因为我只会这两个面向对象的语言），类中属性的访问可以用访问修饰符来进行限制。访问修饰符分为：public、protected、private。

* public:公有修饰符，可以在类内或者类外使用public修饰的属性或者行为，默认修饰符。
* protected:受保护的修饰符，可以本类和子类中使用protected修饰的属性和行为。
* private : 私有修饰符，只可以在类内使用private修饰的属性和行为。

**创建demo12.ts书写创建模型**

//访问修饰符

// public:公有修饰符，可以在类内或者类外使用public修饰的属性或者行为，默认修饰符。

// protected:受保护的修饰符，可以本类和子类中使用protected修饰的属性和行为。

// private : 私有修饰符，只可以在类内使用private修饰的属性和行为。

// class XiaoJieJie2{

//     public sex:string

//     protected name:string

//     private age:number

//     public constructor(sex:string,name:string,age:number){

//         this.sex=sex

//         this.name=name

//         this.age=age

//     }

//     public sayHello(){

//         console.log('小哥哥好')

//     }

//     protected sayLove(){

//         console.log('我爱你')

//     }

// }

// var jiejie2:XiaoJieJie2 = new XiaoJieJie2('女','热巴',22)

// console.log(jiejie2.sex)

// console.log(jiejie2.name)   //报错

// console.log(jiejie2.age)    //报错

// jiejie2.sayHello()

// jiejie2.sayLove()    //报错

//只读修饰符

class Man{

    public readonly sex:string = '男'

}

var man:Man = new Man()

man.sex='女'  //报错

14.面向对象编程：继承和重写

在使用TypeScript这门语言时，一个最重要基本功就是面向对象编程，那对类的扩展就变的格外重要，扩展经常使用的手段就是继承。

继承：允许我们创建一个类（子类），从已有的类（父类）上继承所有的属性和方法，子类可以新建父类中没有的属性和方法。

**extends**关键字就是继承的重点，它相当于我们扑倒女神，小蝌蚪从身体内迸发的一瞬间，所以我们一定要牢记这种感受.....不，是关键词.

但是有一点需要我们注意，TypeScript不支持多重继承。

**类的重写**就是在子类中重写父类的方法。例如，我的儿子“技术帅”发现兴趣爱好是找小姐姐，完成不了“”一天赚一个亿“”的目标，它需要多个兴趣，开平台网站。这时候我们就用到了重写。用super调用

**创建demo13.ts书写创建模型**

//类的继承

class Jspang{

    public name:string

    public age : number

    public skill: string

    constructor(name:string,age:number,skill:string){

        this.name = name

        this.age = age

        this.skill = skill

    }

    public interest(){

        console.log('找小姐姐')

    }

}

let jspangObj:Jspang = new Jspang('技术胖',18,'web')

jspangObj.interest()

//实例化 子继承父类extends

class JsShuai extends Jspang{

    public xingxiang:string = '帅气'

    public interest(){

//重写就是在子类中重写父类的方法，用super调用

        super.interest()

        console.log('建立电商平台')

    }

    public zhuangQian(){

        console.log('一天赚了一个亿')

    }

}

let shuai = new JsShuai("技术帅",5,'演讲')

shuai.interest()

shuai.zhuangQian()

15.面向对象编程：接口

作为一个前端，和后端最多的交流就是向后端要接口，而这些接口是如何定义的，有时候我们好像漠不关心。其实前端也有接口的定义，特别如果你要靠着前端技术作一个全栈，那接口的编写也是必不可少的一项技能。

定义接口的关键字是interface

**?**代表可选择参数接口，可以写也可以不写

**创建demo14.ts书写创建模型**

//定义接口的关键字是interface

interface Husband {

    sex:string

    interest:string

    maiBaoBao?:Boolean   //?代表可选择参数接口，可以写也可以不写

}

let myhusband:Husband ={ sex:'男',interest:'看书、作家务',maiBaoBao:true}

console.log(myhusband)//{ sex: '男', interest: '看书、作家务', maiBaoBao: true }

//规范化类型接口---匹配

interface  SearchMan{

    (source:string,subString:string):boolean

}

let mySearch:SearchMan

mySearch = function(source:string,subString:string):boolean{

    let flag =source.search(subString)

    return (flag != -1) //匹配判断flag=1,如果没有匹配到return false，else true

}

console.log(mySearch('高、富、帅、德','胖')) //false

**生成的demo14.js**

"use strict";

var myhusband = { sex: '男', interest: '看书、作家务', maiBaoBao: true };

console.log(myhusband);

var mySearch;

mySearch = function (source, subString) {

    var flag = source.search(subString);

    return (flag != -1);

};

console.log(mySearch('高、富、帅、德', '胖')); //false

16.面向对象编程：命名空间

在制作大型应用的时候，为了让程序更加有层次感和变量之间不互相干扰，我们可以使用命名空间来构建程序。

举个小例子：比如“德华”这件事，帅哥也有叫德华的，二师兄也有叫德华的。那我们要如何区分那。这对于女孩子选老公来说非常重要啊。

export：输出<https://www.jianshu.com/p/8eba0dc06bdb>

**创建demo15.ts书写创建模型**

//命名空间的使用

namespace shuaiGe{

    //输出export

    export class Dehua{

        public name:string = '刘德华'

        talk(){

            console.log('我是帅哥刘德华')

        }

    }

}

namespace bajie{

    export class Dehua{

        public name:string = '马德华'

        talk(){

            console.log('我是二师兄马德华')

        }

    }

}

let dehua1:shuaiGe.Dehua   = new shuaiGe.Dehua()

let dehua2:shuaiGe.Dehua   = new bajie.Dehua()

dehua1.talk()   //我是帅哥刘德华

**生成的demo15.js**

"use strict";

//命名空间的使用

var shuaiGe;

(function (shuaiGe) {

    //输出export

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

        function Dehua() {

            this.name = '刘德华';

        }

        Dehua.prototype.talk = function () {

            console.log('我是帅哥刘德华');

        };

        return Dehua;

    }());

    shuaiGe.Dehua = Dehua;

})(shuaiGe || (shuaiGe = {}));

var bajie;

(function (bajie) {

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

        function Dehua() {

            this.name = '马德华';

        }

        Dehua.prototype.talk = function () {

            console.log('我是二师兄马德华');

        };

        return Dehua;

    }());

    bajie.Dehua = Dehua;

})(bajie || (bajie = {}));

var dehua1 = new shuaiGe.Dehua();

var dehua2 = new bajie.Dehua();

dehua1.talk(); //我是帅哥刘德华