1.安装：npm install -g typescript

2.编译：tsc helloworld.ts

3.Typescript开发工具 VScode自动编译.ts文件：

第一步： tsc --init 生成tsconfig.json 改 “outDir” ： “./js”

第二步： 任务 --运行任务 点击 tsc:监视-tsconfig.json 然后就可以自动生成代码

4.任意类型 any

5.void类型:表示没有类型

```
function run():void{
    console.log('run');
}
run();
```

6.never类型：其他类型，（包括null和undefined）的子类型，代表从不会出现的值。这意味着声明never的变量只能被never类型所赋值。

二：函数定义(函数名定义法)

```
1.//函数声明
function run(){
    return 'run';
}
//匿名函数
var run2 = function(){
    return 'run2';
}
```

2.方法可选参数

//es5里面方法的实参和形参可以不一样，但是ts中必须一样，如果不一样就需要配置可选参数。

```
function getInfo(name:string,age?:number):string{
    if(age){
        return `${name} --- ${age}`;
    }else{
        return `${name} --- 年龄保密`;
    }
}
alert(getInfo('zhangsan'));
```

加问号就是可选可不选的意思

```
function getInfo(name:string,age:number=20):string{
    if(age){
        return `${name} --- ${age}`;
    }else{
        return `${name} --- 年龄保密`;
    }
}

alert(getInfo('张三',30));
```

设置默认参数

//三点运算符　接受新参传过来的值（剩余参数）

```
function sum(...result:number[]):number{
    var sum=0;
    for(var i=0; i<result.length; i++){
    sum+result[i];
    }
    return sum;
}
alert(sum(1,2,3,4));
```

写法一

```
function sum2(a:number,b:number,...result:number[]):number{
    var sum2 = a + b;
    for(var i=0; i<result.length; i++){
        sum2+=result[i];
    }
    return sum2;
}
alert(sum2(1,2,3,4,5,6));
```

写法二

3.ts函数的重载

//java中方法的重载：重载指的是两个或者两个以上同名函数，但它们的参数不一样，这时会出现函数重载的情况

//typescript中的重载，通过为同一个函数提供多个函数类型定义来试下多种功能的目的。

//ts为了兼容es5 以及es6重载的写法和java中有区别

```
//es5中出现同名方法，下面的会替换上面的方法
function css(config){
}
function css(config,value){
}
```

```
//ts中的重载
function getInfo(name:string):string;
function getInfo(age:number):number;
function getInfo(str:any):any{
    if(typeof str==='string'){
        return '我叫: ' + str;
    }else{
        return '我的年龄是' + str;
    }
}
alert(getInfo('张三'));
```

4.箭头函数（this指向问题　箭头函数里面的this指向上下文）

```
//箭头函数
setTimeout(function(){
    alert('run');
},1000);

setTimeout(()=>{
    alert('run');
},1000);
```

三.es5中的类

```
function Person(){                    1.简单的类的定义
    this.name='张三';
    this.age=20;
}
var p = new Person();
alert(p.name);
```

*实例方法一定要new一下才能被调用。静态方法不用new就可以调用。

```
//构造函数和原型链里面增加方法
//以下为构造函数
function Person(){
    this.name='张三';                    //属性
    this.age=20;

    this.run=function(){                 //方法
        alert(this.name + '在运动');
    }
}
//以下为原型链
//原型链上面的属性会被多个实例共享，但构造函数不会
Person.prototype.sex='男';
Person.prototype.work=function(){
    alert(this.name + '在工作');
}
var p = new Person();
p.work();
//静态方法
Person.getInfo=function(){
    alert('我是静态方法');
}
```

*原型链实现继承

原型链实现继承：可以继承构造函数里面的属性和方法，也可以继承原型链上面的属性和方法

```
//es5里的继承
//以下为构造函数
function Person(){
    this.name='张三';                    //属性
    this.age=20;

    this.run=function(){                 //方法
        alert(this.name + '在运动');
    }
}

Person.prototype.sex='男';
Person.prototype.work=function(){
    alert(this.name + '在工作');
}

//web类继承Person类    原型链+对象冒充的组合继承模式
function Web(){
    Person.call(this);  //对象冒充实现继承
}
var w=new Web();
w.run();                //对象冒充可以继承构造函数里面的属性和方法，但是没法继承原型链的属性和方法
w.work();
```

*原型链+构造函数的组合继承模式