TypeScript 类定义方式如下：

class class\_name {

    // 类作用域

}

**类的数据成员**

class Car {

    // 字段

    engine: string;

    // 构造函数

    constructor(engine: string) {

        this.engine = engine

}

    // 方法

    disp(): void {

        console.log("发动机为 :   " + this.engine)

    }

}

**创建实例化对象**

var object\_name = new class\_name([ arguments ])

**类的继承**

语法格式如下：

class child\_class\_name extends parent\_class\_name

实例

类的继承：实例中创建了 Shape 类，Circle 类继承了 Shape 类，Circle 类可以直接使用 Area 属性：

class Shape {

    Area: number

    constructor(a: number) {

        this.Area = a

    }

}

class Circle extends Shape {

    disp(): void {

        console.log("圆的面积:  " + this.Area)

    }

}

var obj = new Circle(223);

obj.disp()

**static 关键字**

class StaticMem {

    static num: number;

    static disp(): void {

        console.log("num 值为 " + StaticMem.num)

    }

}

StaticMem.num = 12     // 初始化静态变量

StaticMem.disp()       // 调用静态方法

**instanceof 运算符**

instanceof 运算符用于判断对象是否是指定的类型，如果是返回 true，否则返回 false。

class Person { }

var obj = new Person()

var isPerson = obj instanceof Person;

console.log("obj 对象是 Person 类实例化来的吗？ " + isPerson);

**访问控制修饰符**

TypeScript 支持 3 种不同的访问权限。

public（默认） : 公有，可以在任何地方被访问。

protected : 受保护，可以被其自身以及其子类和父类访问。

private : 私有，只能被其定义所在的类访问。

class Encapsulate {

    str1: string = "hello"

    private str2: string = "world"

}

var obj = new Encapsulate()

console.log(obj.str1)     // 可访问

console.log(obj.str2)   // 编译错误， str2 是私有的

**类和接口**

类可以实现接口，使用关键字 implements

interface ILoan {

    interest: number

}

class AgriLoan implements ILoan {

    interest: number

rebate: number

    constructor(interest: number, rebate: number) {

        this.interest = interest

        this.rebate = rebate

    }

}