class 类

> 通过class关键字来构建类，其实现的功能和构造函数差不多，视为一个语法糖

声明和使用

```
class Phone {
    constructor(brand, price)//名字固定，但是不写也是合法的，只是没有初始化的属性而已
    {
        this.brand = brand
        this.price = price
    }

    call() {
        console.log("打电话啊")
    }
}

let OnePlus = new Phone('one+', 2999)
OnePlus.call()
console.log(OnePlus.price)
```

> 注意constructor是固定名字的，每次实例化就会执行这个函数，相当于python中的init

静态成员 static

实例对象中的属性是和构造对象中的属性不相通的,和原型是相通的

```
function Phone(){}

Phone.name = 'a'
Phone.change = function() {}
```

```
let nokia = new Phone()
console.log (nokia.name)//undefine

Phone.prototype.size = '5.5'
nokia.size//5.5
```

对应的静态属性和方法在es6+中这样写

```
class Phone {
    constructor(brand, price)//名字固定
    {
        this.brand = brand
        this.price = price
    }

  static name = 'nokia'//实例对象不能访问
  static function() {}
}
```

继承

构造函数中的继承

```
function Phone(brand, price) {
    this.brand = brand
    this.price = price
}

Phone.prototype.call = function () {
    console.log('我能打电话')
}

function smartPhone(brand, price, color, size) {
    Phone.call(this, brand, price)//改变this的指向，指向外面这个this
    this.color = color
    this.size = size
}

smartPhone.prototype = new Phonef
smartPhone.prototype.constructor = smartPhone//注意这里要指回去

smartPhone.prototype.playGame = function() {
```

```
    console.log ('我还可以打游戏')
}//给子类指定方法
```

es6+

```
//class实现
class Phone {
    constructor(brand, price) {
        this.brand = brand
        this.price = price
    }

    call() {
        console.log('我可以打电话')
    }
}

class smartPhone extends Phone {//这里就已经可以用父类的方法了
    constructor(brand, price, color, size) {
        super(brand, price)//继承父类的
        this.color = color
        this.size = size
    }

    photo() {
        console.log('可以拍照')
    }

    call() {
        console.log('重写父类的call方法')
    }

}

const sp = new smartPhone('xiaomi', 1999, 'red', '5.5')
sp.call()
```

简洁明了

get 和 set

> get是读取的时候触发的函数，set的时候触发的函数，一般用于判断赋值是否合法

```
class Phone () {
  get price() {
    console.log('属性被修改')//读取这个属性的时候会执行这个函数
    return 'iloveyou'//返回值就是修改值
  }
  set price(newVal)//一定要有一个参数
    {
    console.log('价格属性被修改')
  }
}

let s = new Phone ()
s.price = 'free'//会执行set那一行
console.log(s.price)//会执行get那个函数
```