\*\*\*\*OOP

封装

\*\*\*\*\*继承

多态

什么是OOP: 程序中都是用对象来描述现实中一个具体事物

什么是对象: 封装一个事物的属性和功能的程序结构

为什么OOP: 便于大程序的维护

何时OOP: 今后写程序都用OOP

\*\*\*面向对象三大特点: 封装，继承，多态

封装:

什么是: 将一个事物的属性和功能集中定义在一个对象中

为什么: 便于大程序的维护

何时: 只要使用OOP，都要先将事物的属性和功能封装在一个对象中，再反复使用对象的功能。

如何: 事物的属性，会成为对象的属性

事物的功能，会成为对象的方法

属性和方法统称为对象的成员

3种:

1. 直接量:

var obj={

属性名: 属性值,

... : ... ,

方法名 (){

... this.属性名 ...

},

...

}

何时: 在创建对象时，就知道所有成员

\*\*\*\*\*什么是this: 引用正在调用函数的对象的关键词

简单说: 就是点前的对象

为什么: 方法中不加this的变量，默认仅在作用域链中找，不会去对象中查找。

何时: 只要对象的方法向使用自己的属性时，必须加this

2. 用new创建:

var obj=new Object();//{}

obj.属性名=值;

obj.方法名=function(){

... this.属性 ...

}

何时: 在创建对象时，暂时不知道对象的成员

强调: js中的对象即使创建完成，仍然可以添加新成员

\*\*\*js中一切对象底层都是关联数组！

obj.属性 => obj["属性"]

console.dir(obj) 存储结构和关联数组是完全一样的

简写: new Object() => new Object 或 Object()

问题: 以上两种封装方式仅适合封装一个单独的对象

解决: 构造函数

3. 构造函数:

什么是: 描述一类对象统一结构的函数——

何时: 只要反复创建多个相同结构的对象时，都要先定义统一的构造函数，再使用构造函数反复创建对象

如何: 2步:

1. 定义构造函数描述统一结构:

function 类型名(属性参数){

this.属性名=属性参数;

this.方法名=function(){

... this.属性 ...

}

}

2. 用new调用构造函数，传入具体对象的属性值

var obj=new 类型名(属性值);

如何访问对象的属性和方法:

obj.属性 -> 用法和普通变量完全一样

obj.方法() -> 用法和普通函数完全一样

方法也是属性，只不过保存的值是一个函数而已