正课:

\*\*\*OOP

1. 什么是:

2. 封装:

3. 继承:

4. 多态:

1. 什么是:

对象: 集中存储多个数据的存储空间

本质: 其实对象底层就是关联数组

面向对象: 程序中，都是用对象结构，来描述现实中一个具体事物的属性和功能。

为什么: 现实中，任何一项数据，都依附于某个具体事物，才有意义

何时: 只要在程序中，描述一个事物，都必须先将事物的属性和功能集中定义在对象中，再使用。

如何: 三大特点:

封装: 创建一个对象，描述现实中一个具体事物的属性和功能

继承: 偷懒?

多态: 个性?

2. 封装:

什么是: 创建一个对象，描述现实中一个具体事物的属性和功能

为什么: 现实中，任何一项数据，都依附于某个具体事物，才有意义

何时: 只要使用面向对象的方法编程，都要先创建对象

如何: 3种:

1. 用对象直接量:

var obj={//创建一个新对象的意思

属性名:值,

... : ... ,

方法名(){...}, //~~方法名:function(){...}~~

... (){...},

}

现实中，事物的属性，会成为对象的属性,

其实就是保存在对象中的一个普通变量而已

事物的功能，会成为对象的方法

其实就是保存在对象中的一个普通函数而已

何时: 创建对象时，已知对象的成员

问题: 对象的方法中，不能写死属性值, 因为对象的属性都可能发生变化。

解决: 错误的解决1: 在方法中直接使用属性名

因为不用.访问的任何变量，默认只能在作用域链中找

不会自动进入对象中找属性

错误的解决2: 直接使用对象名.属性名

因为对象名只是一个普通的变量名，可能发生变化！

正确的解决: 今后，只要对象的方法，想使用自己的属性值，都要用this.属性名访问

访问对象的成员: 成员=属性+方法

obj.成员名: 属性的用法和普通变量完全一样

方法的用法和普通函数完全一样

如果访问一个不存在成员: 不报错！而是返回undefined

遍历成员: for...in

for(var key in obj){

key //当前属性名

obj[key]//当前属性值

}

2. 用new:

如何: 2步:

1. 先创建空对象: var obj={}; //new Object();

2. 向空对象中添加新属性: obj.新属性名=值;

js中的对象，可随时强行添加新属性和方法

前两种创建对象方式问题: 一次只能创建一个对象。反复创建相同结构的同一类型的对象时，代码会大量重复。

解决: 用构造函数

3. 用构造函数

什么是构造函数: 描述一类对象，统一结构的函数

何时: 只要想反复创建同一类型的多个对象时, 都要先用构造函数定义统一的结构

如何: 2步:

1. 定义构造函数: 定义同一类型对象的统一结构

function 类型名(属性参数,...){

this.属性名=属性参数;

... =属性参数;

this.方法名=function(){

... this.属性名 ...

}

}

2. 调用构造函数，反复创建新对象:

var obj=new 类型名(属性值,...);

new: 4件事:

1. 创建一个新的空对象

2. 让新对象自动继承构造函数的原型对象

3. 调用构造函数, 自动将函数中的this指向新对象

4. 返回新对象的地址保存在变量中

3. 继承:

什么是: 父对象的成员，子对象无须重复创建，就可直接使用

为什么: 放在构造函数中的方法，会为每个对象创建一个相同的副本——浪费内存！

何时: 只要同一类型的多个子对象，需要有共同的方法定义或属性值时，都要在共同的父对象中定义一份即可。所有子对象共用！

作用: 代码重用! 节约内存!

如何:

原型对象: 集中存储同一类型的子对象共有成员的父对象

创建: 不用自己创建，而是在定义构造函数时，买一赠一，自动创建。

继承: 在调用构造函数，创建子对象时，new会自动设置子对象继承构造函数的原型对象

为构造函数添加共有成员:

构造函数.prototype.成员名=值;

调用时: 直接通过"子对象.共有成员名"的方式访问

原型链prototype chain:

由多级父元素，逐级继承，形成的链式结构

保存着: 多有对象可用的成员

控制着: 对象成员的使用顺序:

先用自有成员

如果自己没有，才延原型链向父级查找

自有属性和共有属性:

自有属性: 保存在对象本地的仅自己所有的属性

共有属性: 保存在原型链中的所有子对象共有的属性

访问:

获取: 子对象.属性名

修改: 自有属性: 子对象.属性名=值;

共有属性: 原型对象.属性名=值;

内置对象的原型对象:

其实: 能new的内置对象类型都是构造函数

都有自己对应的原型对象

内置对象的原型对象中，存储着该类型的子对象共有的所有API

问题: 旧浏览器，无法使用新的API

比如: 数组的indexOf方法

解决: 2步:

1. 判断当前浏览器中的原型对象里是否包含指定方法

2. 如果不包含，才自定义方法，加入原型对象中

4. 多态: 同一个函数在不同情况下，表现出不同的状态

重写(override):

什么是: 在子对象中，定义和父对象中同名的成员

为什么: 从父对象继承来的成员，不一定都是好用的！

何时: 只要子对象觉得从父对象继承来的成员不好用，就要在本地重写同名成员，覆盖父对象的

结果: 优先使用自有成员，不再去父对象中找

如何: 在子对象中，定义和父对象中同名的成员