正课:

1. 闭包:

2. 面向对象:

1. 闭包:

鄙视时: 画简图: 找两样东西

1. 受保护的变量是谁？

2. 外层函数共生了几个内层函数孩子

规律: 外层函数是妈妈，外层函数返回到外部的内层函数时生的孩子。

1. 每调用一次妈妈，就包一个红包给孩子

2. 调用一次妈妈生的多个孩子，共用同一个红包。

2. 面向对象:

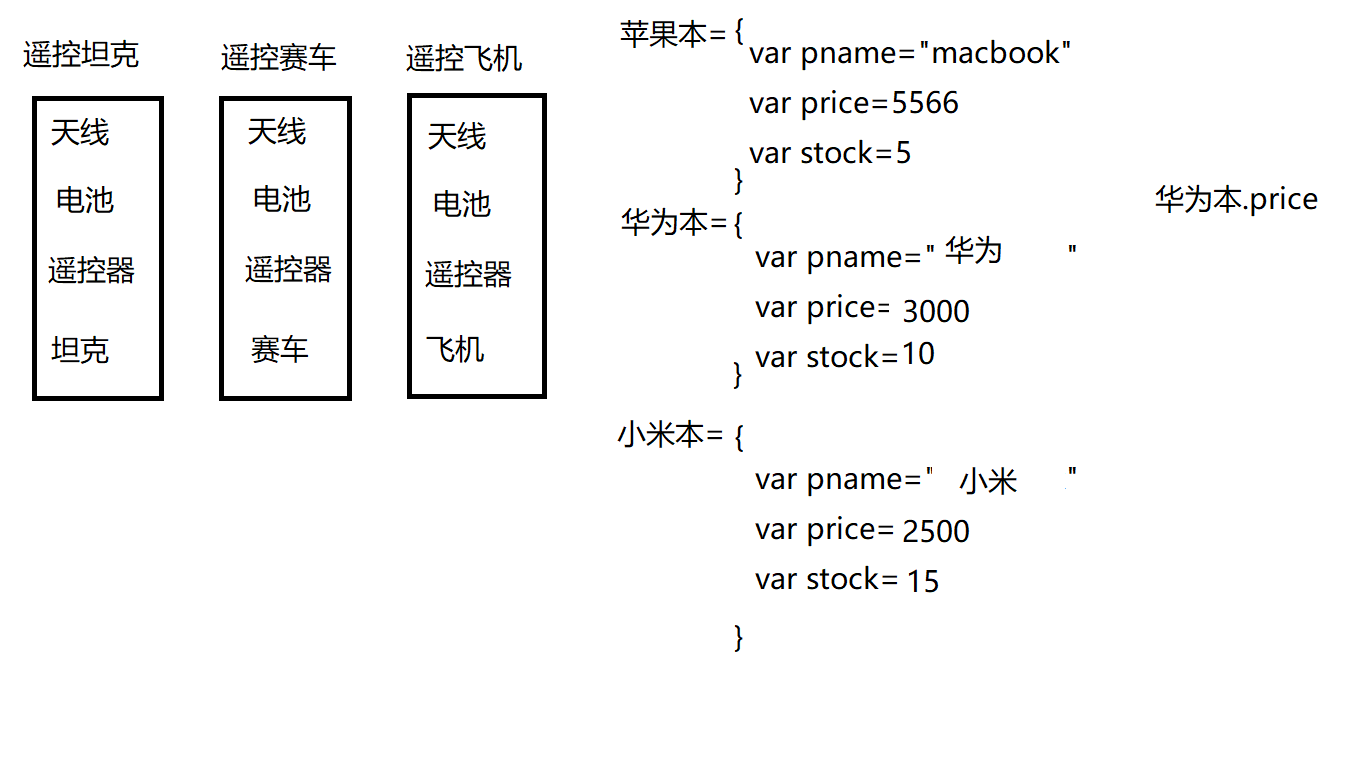
什么是对象:

用途: 对象是描述现实中一个具体事物的属性和功能的程序结构。

本质: 程序中集中存储一个事物的属性和功能的一块存储空间，再起一个名字

什么是面向对象编程: 程序都是先将数据封装在对象中，然后再按需使用对象中的成员。这样的编程方式，就称为面向对象编程。

为什么: 便于大量数据的维护和使用



何时: 今后所有程序都是用面向对象方式实现的

如何: 面向对象三步/三大特点: 封装，继承，多态

1. 封装:

什么是: 创建一个对象，集中保存一个事物的属性和功能。

为什么: 便于大量数据的维护和使用

何时: 只要使用面向对象方式编程，都要先将数据和功能封装在对象中，然后再按需使用。

如何: 3种:

1. 用{}创建一个对象:

var 对象名={

属性名: 值,

... : ... ,

方法: function(){

}

}

如何访问对象的成员:

访问对象的属性: 对象名.属性名

其实属性就是保存在对象中的变量而已。只不过，要找到属性，得先找到对象，再用.操作符，进入对象中，访问属性。

访问对象的方法: 对象名.方法()

其实方法就是保存在对象中的函数而已。只不过，要先找到对象，然后用.操作符进入对象中，才能找到内部的方法加()调用。

问题: 在对象自己的方法，直接使用属性名却无法访问到自己的属性值。报错: 属性名未定义

原因: 所有不带.的变量，默认只能在作用域链中(临时函数作用域和window)查找。但是对象的{}又不是作用域，所以对象是不包含在作用域链中的。所以，直接使用属性名，无法找到藏在对象内的属性的。

不好的解决办法: 在属性名前加"对象."，勾引着引擎进入对象中查找属性。

问题: 紧耦合: 如果外部修改，内部被迫跟着修改。

好的解决: 松耦合: 外部改变，内部代码不用变，也能自动适应！——this.属性名

什么是this: 在调用函数时，临时指向正在调用函数的.前的对象的关键词

总结: 只要对象自己的方法，想访问自己的属性，都必须加"this."

错误: ~~函数定义在哪个对象中，this就指哪个对象~~

正确: this与函数定义在哪儿无关！只与函数调用这一瞬间，.前的对象有关！

2. 用new创建: 2步:

1. 先创建空对象: var obj=new Object()

2. 强行向空对象中添加新属性:

obj.属性=值

obj.方法=function(){ ... }

揭示: 所有js对象底层，其实都是关联数组。

对象 vs 关联数组:

相同: 4个

1. 都可用["成员名"]或.成员名两种方式，访问自己的成员。.成员名 其实就是 ["成员名"] 的简写。也就说 .成员名 到底层会被自动翻译为 ["成员名"]

坑: 如果成员名来自于一个变量！则不能用".变量"方式访问成员。因为".变量"，会被翻译为["变量"]，出错了！因为变量不能放在""里！

解决: 今后只要成员名不是写死的，而是来自于一个变量，只能用[变量] 方式访问，不能带引号。

2. 都可被for in循环遍历每个成员。

3. 随时可以给数组或对象在任何位置添加新成员，而不会报错！而是自动创建该成员。

所以，如果想给一个已经创建完的对象，添加新属性，只有一个野蛮的办法：强行赋值！

4. 访问数组或对象中不存在的位置，不会报错！而是返回undefined！

不同: 类型:

关联数组: 是数组家孩子

对象: 是Object家孩子

问题: 只能创建一个对象。如果反复创建多个相同结构的对象时。代码很冗余，不便于维护。

解决:

3. 用构造函数:

什么是: 专门描述一类对象统一结构的函数。

为什么: 重用对象的结构定义。

何时: 只要程序中，需要反复创建同一类型的多个对象时。

如何: 2步:

1. 定义构造函数来描述一类对象的统一结构

function 类型名(形参1,...){

this.属性名=形参1;

... = ... ;

this.方法名=function(){

}

}

2. 调用构造函数按照统一结构创建对象

var obj=new 类型名(属性值1，...);

new的原理: 4件事:

1. 新建一个空对象

2. ？

3. 用new调用构造函数。将构造函数中的this，都吸引到new上！

然后通过强行赋值的方式！给新对象new，添加规定好的属性！

凡是构造函数中规定的属性和方法，最终都会添加到孩子身体里！

4. 返回新对象的地址给变量保存起来。

问题: 构造函数虽然实现了代码重用，但是浪费了内存！

2. 继承:

什么是: 爹的成员，孩子无需重复创建，就可直接使用。

为什么: 代码重用，节约内存

何时: 只要多个孩子都需要共用的成员，都要通过继承使用！

如何:

1. 不用自己设置继承关系！其实，js中已经帮我们设置好了继承关系。咱们直接使用即可！

js实现继承，都是通过继承原型对象来实现的。

原型对象: 专门集中保存一类子对象的共有成员的父对象。

原型对象是在定义构造函数时，附赠的。只不过，这个原型对象暂时是空的。

如何找到原型对象: 每个构造函数都有一个属性叫prototype，指向自己附赠的这个原型对象。

其实原型对象中，也有一个constructor属性，指回构造函数。

子对象何时如何继承的父对象: new的第2步: 让子对象的\_ \_proto\_ \_属性指向妈妈的原型对象。凡是从\_ \_proto\_ \_属性指出的关系，都叫继承关系。

结果: 原型对象中的成员，子对象无需创建，可直接使用！

2. 如何向原型对象中添加共有成员:

强行赋值:

妈妈.prototype.成员名=值

比如: 向所有学生的共同的爸爸里强行添加一个公用的方法intrSelf()。

Student.prototype.intrSelf=function(){

... ...

}

结果: 凡是Student生的孩子，都能.intrSelf()来直接调用爹里的函数。

比如： lilei.intrSelf()

hmm.intrSelf()

自有属性和共有属性：

自有属性: 保存在子对象中，归对象个人所有的属性

共有属性: 保存在原型对象中，归多个子对象共有的属性

获取属性值: 二者没有任何差别！

子对象.属性名

修改属性值:

自有属性，可直接用子对象修改

比如: lilei.sage++

共有属性，只能用原型对象修改

如果强行用子对象.共有属性修改

会给这个子对象添加一个同名的自有属性。从此，这个子对象和父对象的属性，无关了。

内置对象的原型对象:

任何一种类型，其实都是由构造函数和原型对象组成。内置类型也是这样。

js包含哪些内置类型：

String Number Boolean

Array Date ~~Math~~ RegExp

Error

Function Object

~~global~~

所有内置类型也包含构造函数和原型对象：

构造函数: 创建该类型的子对象

原型对象: 保存该类型的所有子对象共有的方法和属性。

比如: Array类型就包含2部分:

1. 数组类型构造函数:

function Array(){ ... }

所以，创建数组可以使用new Array()

2. 数组类型也有一个原型对象，包含所有数组对象共用的函数:

为一个类型添加一个共有的自定义函数:

何时: 只要项目中，经常对一个类型的对象做一种操作，而这个类型的原型对象中又不包含这种函数。

如何: 比如: 为所有数组添加求和函数sum()

1. 在数组类型的原型对象中添加sum函数

2. 用数组类型的子对象，调用这个sum函数。

作业:

1. function fun(){//妈妈生了几个孩子？

for(var i=0,arr=[];i<3;i++){

arr[i]=function(){ console.log(i) }

}

return arr;

}

var funs=fun();//因为妈妈只被调用了一次，所以只有一个红包

funs[0]();//?

funs[1]();//?

funs[2]();//?

2. function change(o){

o.name="西西";

o={};

o.name="小红";

}

var o={ name: "小静", age: 11 }

change(o);  
 console.log(o)//?

单词列表：

1. method 方法

2. function 功能/函数

3. constructor 构造函数