正课:

1. OOP:

1. OOP:

面向对象三大特点：封装， 继承， 多态

1. 封装: 创建对象，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管理

何时: 只要使用面向对象思想，都要先创建对象，再按需访问对象的内容

如何: 3种:

1. 用{}直接量:

var lilei={

sname:"Li Lei",

sage:11,

intr:function(){

... this.sname ... this.sage ...

}

}

问题: 即使函数自己的方法内，也不能直接使用自己的成员变量。

为什么: 成员变量默认没有保存在作用域链上，而是保存在某个对象内部。如果不加"对象."前缀，默认无法访问对象中的属性。

解决:

不好的解决: 写死lilei.sname

紧耦合: 如果对象名变量发生变化: lilei->ll，如果内部不跟着改，就会出错！

好的解决: this.sname

this: 函数运行时，临时创建的，指向正在调用当前函数的.前的对象： lilei.intr() intr中的this->lilei

hmm.intr() intr中的this->hmm

强调: 要判断this一定不要看保存在哪里，只看何时调用，以及调用时.前使用的是谁

访问对象成员:

访问属性: lilei.sage

访问方法: lilei.intr();

2. 用new: 2步:

1. 创建空对象: var lilei=new Object();

lilei: {}

2. 向空对象中，添加新属性:

js中的对象可随时强行添加新属性:

lilei.sname="Li Lei";//强行赋值

//lilei: { sname:"Li Lei" }

lilei.sage=11;

//lilei: { sname:"Li Lei", sage:11 }

揭示: js中一切对象底层，都是关联数组

1. 向对象中一个不存在的属性强行赋值，也不会报错，而会自动添加指定属性

2. 访问对象中一个不存在属性值，也不会报错，而是返回undefined

3. 访问对象中的属性，也有两种方式:

1. 标准方式: 对象["属性名"]

||

2. 简写: 对象.变量

如果要访问的属性名不是固定的，来自于变量或动态拼接生成。只能用[变量或js表达式]

问题: 反复创建多个相同结构的对象时，代码会大量重复，导致不便于维护

解决:

3. 用构造函数(constructor)反复创建同一类型的相同结构的多个对象

为什么: 为了重用对象结构的定义

让对象的结构只定义一次，就可反复使用

何时: 反复创建同一类型的相同结构的多个对象

如何:

2步:

1. 定义构造函数，来描述同一类型的所有对象的统一结构——设计图

function Student(sname, sage){

//新对象:{}

this.sname=sname;

//新对象:{ sname:" Li Lei " }

this.sage=sage;

//新对象:{ sname:" Li Lei ", sage:11 }

this.intr=function(){

... ...

}

//新对象:{ sname:" Li Lei ", sage:11, intr:function(){} }

}

2. 调用构造函数，创建新对象:

var lilei=new Student("Li Lei",11);

new做了4件事:

1. 创建一个新对象，开辟一个新的存储空间

2. ？

3. 用新对象调用Student

new可将this自动指向新对象

依次执行构造函数内的每条语句:

强行给新对象赋值规定的属性，结果新对象中就自动添加了规定的新属性和参数值

4. new将新对象的地址，返回给前面的变量

总结: 只要仅创建一个对象，用{}

如果反复创建多个相同结构的对象，用构造函数

js中最核心的原理: 一切对象底层都是关联数组

问题: 每个对象中都重复创建了相同的方法定义，浪费内存！

解决:

2. 继承

什么是: 父对象的成员，子对象无需重复创建，就可直接使用

为什么: 代码重用，节约内存

何时: 如果多个子对象，都需要一个相同的成员时，就要用继承。

如何:

js中的继承，都是继承原型对象(prototype)

什么是原型对象: 集中保存了所有孩子共有的功能的父对象

何时: 今后，只要多个孩子需要共同的功能时，都要将功能仅放在原型对象中一份即可，所有孩子，自动可以使用！

创建原型对象: 不用你创建！买一增一

每定义一个构造函数，就附赠一个空的原型对象:

构造函数.prototype={ 原型对象 }

如何继承: 不用自己继承！

new的第二步: 让新的子对象继承构造函数的原型对象

lilei.\_\_proto\_\_=Student.prototype;

咱们只负责向原型对象中添加共有成员！

只能靠强行赋值新成员的方式，自动添加:

Student.prototype.intr=function(){}

原型链:prototype chain

什么是: 由多级父对象逐级继承，形成的链式结构

保存着一个对象可用的所有成员

控制着对象成员的使用顺序: 先自有，再向上找共有

补: for 专门遍历数字下标的数组或类数组对象

for in 专门遍历自定义字符串下标的关联数组和对象

in 依次获得当前对象可用的一切非隐藏的成员！

3. 多态

什么是: 一个函数在不同情况下表现出不同的状态

2种:

1. 重载overload:

2. 重写override:

什么是: 在子对象中定义和父对象中同名的成员，实现覆盖父对象成员的目的

为什么: 爹的成员不总是好用的

何时: 只要爹的成员不好用，就可在子对象中定义同名成员