正课:

1. 面向对象:

1. 面向对象:

什么是: 程序中都是用对象结构来描述现实中的事物

为什么: 便于大量数据的维护

何时: 所有程序都是用面向对象的思想管理数据和功能

如何: 面向对象三大特点: 封装，继承，多态

1. 封装:

什么是: 创建一个对象结构，保存一个事物的属性和功能

为什么: 便于大量数据的维护

何时: 今后所有数据都是先封装在对象中，再按需使用

如何: 3种:

1. 用{}创建一个对象:

var 对象名={

属性: 属性值,

... : ... ,

功能: function(){

... this.属性名 ...

}

}

访问对象的成员: 成员=属性+方法

访问属性: 对象名.属性

调用方法: 对象名.方法()

问题: 对象自己的方法内，想访问自己的属性，也不能直接属性，报错： 属性 未定义

为什么: 对象的属性没有直接保存在一个函数的作用域链中，而引擎不能擅自进入对象中获取数据。

解决一: 对象名.属性/方法

问题: 紧耦合: 对象名发生变化，必须同时改内部的对象名，一旦忘改，就出错！

解决二: 松耦合: 能不能自动获得对象名——this

什么是this: 正在调用当前函数的.前的对象

每个函数与生俱来的关键词，可自动获得.前的对象

鄙视时: this只和调用时.前的对象有关，和保存在哪儿毫无关系！

如果一个函数前没加任何.，默认是window.

总结: 何时: 只要对象自己的方法内，想使用对象自己的属性和其它方法时，都必须用this.

2. 用new: 2步: 同关联数组的创建过程

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

obj.属性名=值;

obj.方法名=function(){ ... }

揭示了: js底层所有对象都是关联数组

vs 关联数组相比:

相同:

1. js中的对象，可随时添加新属性和新方法，而不会报错！

2. 访问对象中不存在的属性，不会报错，而是返回undefined

3. 访问对象中的成员，有两种方式:

简写: 对象名.属性名/方法名()

// ||等效

何时: 如果要访问的属性名是写死的

完整: 对象名["属性名"]/["方法名"] ()

何时: 如果要访问的属性只能在运行时动态获得

4. 对象也可以用for in遍历每个属性

缺点: 一次只能创建一个对象，如果反复创建多个相同结构的对象时，代码会很繁琐。

解决: 构造函数

3. 用构造函数反复创建多个相同结构的对象

什么是构造函数: 专门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

如何: 2步:

1. 定义构造函数：

function 类型名(形参1, 形参2, ... ){

this.属性名=形参1;

... = ...;

this.方法名=function(){ ... this.属性名 ...}

}

2. 用构造函数创建多个对象:

用new调用构造函数，创建出规定格式和功能的对象

var 对象名=new 类型名(属性值,属性值,...)

鄙视题: new的原理: 4件事:

1. 创建一片空的存储空间

2.

3. 用新的空对象作为主语调用构造函数

new Student(属性值,...)

=> new.Student(属性值,...) this->new{}

↓

=> this.属性名=形参

↓

=>new{}.属性名=属性值

在new{}中强行添加该新属性，并将属性值保存进对象中——野蛮的强行赋值方式

结果: new{}中就反复添加了规定的新属性和方法

4. 返回新对象的结果