正课:

1. OOP:

1. OOP: 面向对象编程

什么是对象: 程序中描述现实中一个具体事物的属性和功能的程序结构

什么是面向对象编程: 程序都是先将数据和功能定义在对象中，再按需使用对象的数据和功能

为什么: 便于大量数据和功能的维护和使用

如何: 三大特点:封装，继承，多态

封装:

什么是: 创建对象集中保存一个事物的属性和功能

为什么: 便于大量数据和功能的管理和使用

何时: 要面向对象编程，都要先封装对象，再按需使用对象的属性和功能

如何: 3种:

1. 用{}直接量:

var 对象名={

属性名:属性值,

... : ... ,

方法名:function(){

... this.成员 ...

}

}

问题: 对象的方法要访问对象自己的属性或方法，无法直接使用

为什么: 所有对象的属性和方法默认都不在作用域链当中。任何程序无权自动进入对象中获取成员。

不好的解决: 在方法中: 对象名.成员名

为什么: 紧耦合，如果对象名发生变化，方法的内容必须同时修改才行。

好的解决办法: this

什么是this: 自动指向正在调用当前函数的.前的对象

何时: 只要对象自己的方法，要使用自己的成员，都要加this.

强调: 鄙视时，this与定义时无关，只和调用时有关

访问对象的成员: (属性和方法)

对象.属性名

对象.方法名()

2. 用new创建: 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.新成员=值/function(){...}

js中对象的特点:

1. 对象创建完成后，依然可继续添加新成员

2. 访问对象中不存在的属性，不报错！而是返回undefined

3. 访问对象的成员，既可用.，又可用["成员名"]

选择: 如果成员名是变化的，就必须用[变量]

如果成员名可以写死，就可用.简化

揭示: 其实js底层，所有对象都是关联数组！

所以，对象也可用for in遍历其中的属性

前两种方式的问题: 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，代码重复很多

解决:

3. 用构造函数反复创建统一类型的多个对象:

构造函数: 描述一类对象统一结构的函数

为什么: 重用创建对象的代码！

何时: 要反复创建多个相同结构的对象时

如何: 2步:

1. 定义构造函数:

function 类型名(){

//this->new->{}

this.属性名=值; //将来的对象中一定要有指定的"属性"

this.xxx = 值;

this.方法名=function(){//将来的对象中必须有指定的"方法"

... ...

}

}

2. 调用构造函数反复创建多个对象:

var obj=new Student(属性值,...)

new:

1. 创建一个新的空对象:{}

2. ?

3. 调用构造函数:

自动将构造函数中的this->new->{}

构造函数中的每一句话，都通过强行赋值的方式，为新对象{}添加规定的新属性

4. 返回新对象地址

问题: 放在构造函数中的方法定义，会在每个子对象中重复创建函数对象，浪费内存！

解决:

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: 只要多个子对象需要相同的成员时

如何:

原型对象: 专门保存所有子对象共有成员的父对象

不用创建，定义构造函数时，附赠的

构造函数.prototype -> 当前构造函数的原型对象

如何继承: new的第2步，自动设置本次创建的子对象，继承构造函数的原型对象。

每个子对象都有\_ \_proto\_ \_属性，指向自己的父级对象

向原型对象中添加共有成员:

构造函数.prototype.成员=值/function(){...}

自有属性和共有属性:

获取属性值, 没有任何差别

差别在: 修改时: 自有属性，可用子对象修改

共有属性，不能用某一个子对象修改

只能用原型对象修改

内置对象的构造函数和原型对象:

内置对象: ES标准中规定的，浏览器已经定义好的可直接使用的对象

包括: 11个

String Number Boolean

Array RegExp Date Math

Error

Function Object

global

除了Math和global外，其余都是一种类型:

都包含2部分组成:

1. 构造函数: 创建该类型的子对象

2. 原型对象: 保存该类型的所有子对象共有的成员

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

2大作用:

1. 保存了当前对象可用的所有属性:

2. 控制着属性的使用顺序

多态:

什么是: 一个函数在不同情况下表现出不同的状态

包括: 2种:

1. 重载:

2. 重写:override

什么是: 在子对象中定义和父对象同名的成员

何时: 只要从父对象继承来的成员不好用

如何: