**正课:**

**1. \*\*\*\*\*OOP:**

**1. OOP**

**什么是面向对象: 程序都是先用对象结构描述现实中一个具体事物，再按需使用事物的属性和功能。**

**为什么: 为了便于大量数据的管理和维护**

**何时: 今后所有程序都使用面向对象的方式实现**

**如何: 三大特点: 封装，继承，多态**

**封装:**

**什么是: 创建一个对象结构来集中保存一个事物的属性和功能**

**为什么: 便于大量数据的使用和维护**

**何时: 只要使用面向对象方式编程，都要先创建对象**

**如何: 3种:**

**1. 用{}创建:**

**var obj={**

**属性名:值,**

**... : ...**

**方法名~~:function~~(){ //ES6: 方法名(){**

**... ...**

**}**

**... ...**

**}**

**对象是描述现实中一个事物的属性和功能的程序解构**

**事物的属性会成为对象中的属性**

**事物的功能会成为对象中的方法**

**访问对象成员: 成员={ 属性+方法 }**

**先用对象名找到对象，再用.找到对象中的成员:**

**访问对象的属性: 对象.属性**

**对象的属性，其实就是保存在对象中的变量，用法和普通变量完全一样**

**访问对象的方法: 对象.方法()**

**对象的方法，其实就是保存在对象中的函数，用法和普通函数完全一样**

**问题: 对象自己的方法中，要使用自己的属性，不能直接写属性名**

**原理: 默认情况下，不带任何前缀的变量，只能在作用域链范围内查找，无法自动进入某个对象内。**

**解决: 2种:**

**1. 改为对象.属性名**

**问题: .前的变量名与对象名紧耦合**

**2. 改为this.属性名**

**this: 自动指向正在调用当前函数的.前的对象的关键字**

**总结: 对象自己的方法要使用自己的属性必须加this.**

**2. 用new: 2步:**

**1. 先创建空对象: var obj=new Object();**

**简写: 可省略new或()，但至少保留一个**

**2. 再向空对象中强行添加新属性**

**obj.属性名=值;**

**obj.方法名=function(){ ... }**

**揭示: 其实, js对象底层也是关联数组:**

**1. 即可用["属性名"]访问元素，又可简写为.属性名**

**.属性名 等效于 ["属性名"] .key 等效于 ["key"]**

**何时用. : 如果要访问的属性名是固定不变的**

**何时用[]: 如果要访问的属性名是动态获得的**

**2. 访问不存在的属性，不报错，而是返回undefined**

**3. 强行给不存在的属性赋值，不报错，而是强行添加该属性**

**4. 遍历对象每个属性也可以用for in循环**

**问题: 以上两种方式，只适合创建一个单独的对象**

**如果反复创建同一类型的多个对象时，代码会很繁琐**

**3. 用构造函数:**

**什么是: 描述一类对象统一结构的函数**

**为什么: 代码重用！**

**何时: 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，描述同一类型所有对象的同一结构**

**function 类型名( 属性参数,... ){**

**this.属性名=属性参数;**

**this. ... = ...;**

**this.方法名=function(){**

**... this.属性...**

**}**

**}**

**2. 调用构造函数:**

**var obj=new 类型名(属性值,...)**

**new共做了4件事:**

**1. 创建一个新的空对象**

**2. ？**

**3. 用新对象调用构造函数**

**new 可将构造函数中的this自动指向新对象**

**构造函数通过强行赋值新属性的方式，为新对象添加规定的新属性。**

**4. 返回新对象的地址保存到等号左边的变量中**

**问题: 浪费内存**

**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: 如果多个子对象要使用相同的方法时**

**如何: js中继承都是通过原型对象实现的**

**什么是原型对象: 集中存储所有子对象共有成员的父对象**

**如何获得原型对象: 买一赠一**

**每创建一个构造函数，都自动附赠一个原型对象**

**何时，如何继承:**

**new的第2步: 让新子对象继承构造函数的原型对象**

**子对象.\_\_proto\_\_=构造函数.prototype**

**向原型对象中添加共有成员**

**构造函数.prototype.共有成员=值**

**自有属性和共有属性:**

**自有属性: 保存子对象本地，仅当前对象自己所有的属性**

**共有属性: 保存在原型对象中，归所有子对象共有的属性**

**获取属性值: 二者没有差别: 子对象.属性名**

**修改属性值:**

**修改自有属性，必须通过子对象修改**

**修改共有属性，必须通过原型对象修改**

**如果强行用子对象修改共有属性，会给子对象添加一个同名的自有属性。从此子对象与共有属性再无关系**

**内置类型/对象:**

**什么是: ES标准规定的，浏览器已经实现的类型**

**包括: 11个:**

**String Number Boolean**

**Array Date RegExp Math{}**

**Error**

**Function Object**

**global{} 在浏览器中被window代替**

**凡是能new的都是一种类型:**

**包括2部分:**

**1. 构造函数: 专门用于创建该类型的子对象**

**2. 原型对象: 专门保存该类型所有子对象的共有成员**

**为类型添加一种新函数:**

**原型链:**

**什么是: 由多级父对象逐级继承，形成的链式结构**

**保存着所有对象的成员**

**控制着成员的使用顺序: 先自有，再共有**

**鄙视: 如何判断一个对象是不是数组！有几种方式**

**自定义继承:**

**3种:**

**1. 仅修改一个对象的父对象:**

**子对象.\_\_proto\_\_=father**

**Object.setPrototypeOf(子对象, father)**

**2. 修改多个对象的父对象:**

**构造函数.prototype=father**

**时机: 必须在创建子对象之前，就改**

**3. 两种类型间的继承:**

**问题: 两种类型间包含部分相同的属性结构和方法定义**

**解决: 定义抽象父类型: 2步:**

**1. 定义抽象父类型:**

**父类型的构造函数中保存相同的属性结构**

**父类型的原型对象中保存相同的方法定义**

**2. 让子类型继承父类型:**

**让子类型原型对象继承父类型原型对象**

**在子类型构造函数中调用父类型构造函数**