javaScript中的面向对象（oop）

Js面向对象概述

面向对象

对代码的一种抽象，对外统一提供调用接口的编程思想

基于类的面向对象和基于原型的面向对象

基于原型的面向对象方式中，对象（object）则是依靠构造器（constructor）利用 原型（prototype）构造出来的。

js面向对象的名词解释

属性：事物的特性

方法：事物的功能

对象：事物的一个实例

原型：Js函数中由prototype属性引用了一个对象，即原型对象（原型）

js中的闭包

闭包：闭包是一个拥有许多变量和绑定了这些变量的环境的表达式（通常是一个函数）

js字面式对象声明对象

```
var obj = {
            属性名称:属性值,
            属性名称:属性值,
            属性名称:属性值,
            ...
            方法名称:function() {},
            方法名称:function() {},
            ...
          }
```

new操作符后跟Object构造函数

```
var obj = new Object();
obj.属性 = 属性值;
obj.属性 = 属性值;
obj.方法 = function (str){
    方法代码
};
```

js中构造方法声明对象

```
function test([参数列表]) {
 this.属性 = 属性值;
 this.方法 = function () {
    方法中代码
 }
}
var obj = new test(参数列表);
```

js中工厂方式声明对象

```
function createObject(name, age){
    var obj = new Object();
    obj.name = name;
    obj.age = age;
    obj.run = function () {
        return this.name + this.age + '运行中...';
    };
    return obj;
}
var box1 = createObject('zhangsan', 100);
var box2 = createObject('lisi', 200);
```

js中原型模式声明对象

```
function test(){
}
test.prototype.属性 = 属性值;
test.prototype.属性 = 属性值;
test.prototype.属性 = function(){
    执行代码
}
var obj = new test();
```

js中混合模式声明对象

```
function test(v1, v2, v3) {
        this.v1 = v1;
        this.v2 = v2;
        this.v3 = v3;
}
test.prototype.方法名称 = function () {
      执行代码
}
 var obj = new Blog(v1, v2, v3);
```

js遍历对象的属性和方法

```
var ren = {};
ren.name = "zhangsan";
ren.age = 18;
ren.demo = function (){
    document.write("aaaaa");
}
for(var i in ren){
    //alert(i); //弹出属性和方法名称
    alert(ren[i]); //弹出属性和方法的内容
}
```

封装的概念

封装(Encapsulation)：把对象内部数据和操作细节进行隐藏

原型和原型链

原型：是利用prototype添加属性和方法

原型链：JS在创建对象（不论是普通对象还是函数对象）的时候，都有一个叫做__proto__的内置属性，用于指向创建它的函数对象的原型对象prototype

原型继承

利用原型让一个引用类型继承另一个引用类型的属性和方法

构造函数继承

在子类内部构造父类的对象实现继承

call和apply的用法

call：调用一个对象的一个方法，以另一个对象替换当前对象

apply：应用某一对象的一个方法，用另一个对象替换当前对象

js面向对象的关键词

instanceof、delete、call、 apply、arguments、callee、this

对象冒充

将父类的属性和方法一起传给子类作为特权属性和特权方法