**Javascript中的apply与call详解**

　　JavaScript中有一个call和apply方法，其作用基本相同，但也有略微的区别。

**一、方法定义**

**1、call 方法**

　　语法：call([thisObj[,arg1[, arg2[, [,.argN]]]]])

　　参数 thisObj 可选项。将被用作当前对象的对象。 arg1, arg2, , argN 可选项。将被传递方法参数序列。

　　说明   
　　call 方法可以用来代替另一个对象调用一个方法。call 方法可将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。   
　　如果没有提供 thisObj 参数，那么 Global 对象被用作thisObj。说明白一点其实就是更改对象的内部指针，即改变对象的this指向的内容。这在面向对象的js编程过程中有时是很有用的。

**2、apply方法**

　　语法：apply([thisObj[,argArray]])   
　　定义：应用某一对象的一个方法，用另一个对象替换当前对象。   
　　说明：   
　　如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。   
　　如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj， 并且无法被传递任何参数。

**二、常用实例**

**实例1：call应用实例**

　　引用网上一个代码段，运行后自然就明白其道理。

按 Ctrl+C 复制代码



按 Ctrl+C 复制代码

　　call函数和apply方法的第一个参数都是要传入给当前对象的对象，及函数内部的this。后面的参数都是传递给当前对象的参数。

　　运行如下代码：



<script>

  var func=new function(){this.a="func"}

  var myfunc=function(x){

       var a="myfunc";

       alert(this.a);

       alert(x);

   }

   myfunc.call(func,"var");

</script>



　　可见分别弹出了func和var。先调用func函数，用this.a=”func”替换myfunc中的this.a ; 然后将”var”传递给方法myfunc的参数x可见分别弹出了func和var。

　　对于apply和call两者在作用上是相同的，但两者在参数上有区别的。

　　第一个参数意义都一样。第二个参数：apply传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而call则作为call的参数传入（从第二个参数开始）。  
　　如 func.call(func1,var1,var2,var3)  对应的apply写法为：func.apply(func1,[var1,var2,var3])，同时使用apply的好处是可以直接将当前函数的arguments对象作为apply的第二个参数传入。

**实例2：继承的演示**



// 继承的演示

function base() {

    this.member = " dnnsun\_Member";

    this.method = function() {

        window.alert(this.member);

    }

}

function extend() {

    base.call(this);

    window.alert(member);

    window.alert(this.method);

}



　　上面的例子可以看出，通过call之后，extend可以继承到base的方法和属性。

　　顺便提一下，在javascript框架prototype里就使用apply来创建一个定义类的模式，其实现代码如下：



var Class = {

create: function () {

return function () {

this.initialize.apply(this, arguments);

}

}

}



　　解析：从代码看,该对象仅包含一个方法：Create，其返回一个函数，即类。但这也同时是类的构造函数，其中调用initialize，而这个方法是在类创建时定义的初始化函数。通过如此途径，就可以实现prototype中的类创建模式

　　示例：



var vehicle=Class.create();

vehicle.prototype={

    initialize:function(type){

        this.type=type;

    },

    showSelf:function(){

        alert("this vehicle is "+ this.type);

    }

}

var moto=new vehicle("Moto");

moto.showSelf();



　　运行结果为：this vehicle is Moto

**实例3：**



function add(a, b)

{

alert(a + b);

}

function sub(a, b)

{

alert(a - b);

}

add.call(sub, 3, 1);



　　输出结果为：4

　　这个例子中的意思就是用 add 来替换 sub，add.call(sub,3,1) == add(3,1) ，所以运行结果为：alert(4);  注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

**实例4：**



function Animal() {

this.name = 'Animal';

this.showName = function () {

alert(this.name);

}

}

function Cat() {

this.name = 'Cat';

}

var animal = new Animal();

var cat = new Cat();

//通过call或apply方法，将原本属于Animal对象的showName()方法交给对象cat来使用了。

//输入结果为"Cat"

animal.showName.call(cat, ',');

//animal.showName.apply(cat,[]);



　　输出结果为：cat

　　call 的意思是把 animal 的方法放到cat上执行，原来cat是没有showName() 方法，现在是把animal 的showName()方法放到 cat上来执行，所以this.name 应该是 Cat。

**实例5：实现继承**



function Animal(name) {

this.name = name;

this.showName = function () {

alert(this.name);

}

}

function Cat(name) {

Animal.call(this, name);

}

var cat = new Cat('Black Cat');

cat.showName();



　　输出结果为：Black Cat

　　Animal.call(this) 的意思就是使用 Animal对象代替this对象，那么 Cat中不就有Animal的所有属性和方法了吗，Cat对象就能够直接调用Animal的方法以及属性了。

**实例6：实现多重继承**



var s1 = function(name){

this.name = name;

}

var s2 = function(sex){

this.sex = sex;

}

var s3 = function(age){

this.age = age;

}

var Student = function(name,sex,age,score){

s1.call(this,name);

s2.call(this,sex);

s3.call(this,age);

this.score = score;

}

Student.prototype.construction = Student;

var s = new Student('jack','male','12','100');

console.log(s.name); //输出:jack

console.log(s.sex); //输出:male

console.log(s.age); //输出:12

console.log(s.score);//输出:100



　　这样我们就可以根据各个不同的功能模块分不同的程序员独立开发，最后合并起来，实现了多重继承。