**call() apply()方法**

1、每个函数都包含两个非继承而来的方法：apply()和call()。   
2、他们的用途相同，都是在特定的作用域中调用函数。   
3、接收参数方面不同，apply()接收两个参数，一个是函数运行的作用域(this)，另一个是参数数组。  
call()方法第一个参数与apply()方法相同，但传递给函数的参数必须列举出来。   
　　例1：

代码如下:

window.firstName = "diz";   
window.lastName = "song";   
var myObject = { firstName: "my", lastName: "Object" };   
function HelloName() {   
　　console.log("Hello " + this.firstName + " " + this.lastName, " glad to meet you!");   
}   
HelloName.call(window); //huo .call(this);   
HelloName.call(myObject);

运行结果为：   
Hello diz song glad to meet you!   
Hello my Object glad to meet you!

例2：

代码如下:

function sum(num1, num2) {   
return num1 + num2;   
}   
console.log(sum.call(window, 10, 10)); //20   
console.log(sum.apply(window,[10,20])); //30

　　分析：在例1中，我们发现apply()和call()的真正用武之地是能够扩充函数赖以运行的作用域，如果我们想用传统的方法实现，请见下面的代码：

代码如下:

window.firstName = "diz";   
window.lastName = "song";   
var myObject = { firstName: "my", lastName: "Object" };   
function HelloName() {   
console.log("Hello " + this.firstName + " " + this.lastName, " glad to meet you!");   
}   
HelloName(); //Hello diz song glad to meet you!   
myObject.HelloName = HelloName;   
myObject.HelloName(); //Hello my Object glad to meet you!

　　见加红的代码，我们发现，要想让HelloName()函数的作用域在对象myObject上，我们需要动态创建myObject的HelloName属性，此属性作为指针指向HelloName()函数，这样，当我们调用myObject.HelloName()时，函数内部的this变量就指向myObjecct，也就可以调用该对象的内部其他公共属性了。   
　　通过分析例2，我们可以看到call()和apply()函数的真正运用之处，在实际项目中，还需要根据实际灵活加以处理！   
　　一个小问题：再看一看函数中定义函数时，this变量的情况

代码如下:

function temp1() {   
console.log(this); //Object {}   
function temp2() {   
console.log(this); //Window   
}   
temp2();   
}   
var Obj = {};   
temp1.call(Obj); //运行结果见上面绿色的注释！！！！

执行结果与下面的相同：

代码如下:

function temp1() {   
console.log(this);   
temp2();   
}   
function temp2() {   
console.log(this);   
}   
var Obj = {};   
temp1.call(Obj);

**4、bind()方法**　　支持此方法的浏览器有IE9+、Firefox4+、Safari5.1+、Opera12+、Chrome。它是属于ECMAScript5的方法。直接看例子：

代码如下:

window.color = "red";   
var o = { color: "blue" };   
function sayColor(){   
alert(this.color);   
}   
var OSayColor = sayColor.bind(o);   
OSayColor(); //blue

　　这里，sayColor()调用bind()方法，并传入o对象，返回了OSayColor()函数，在OSayColor()中，this的值就为o对象。

**1、方法定义**

**call方法:**   
语法：call([thisObj[,arg1[, arg2[,   [,.argN]]]]])   
定义：调用一个对象的一个方法，以另一个对象替换当前对象。   
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。   
  
**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。

**2、常用实例**

a、**Java代码**

1. function add(a,b)
2. {
3. alert(a+b);
4. }
5. function sub(a,b)
6. {
7. alert(a-b);
8. }
10. add.call(sub,3,1);

 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。

b、**Java代码**

1. function Animal(){
2. **this**.name = "Animal";
3. **this**.showName = function(){
4. alert(**this**.name);
5. }
6. }
8. function Cat(){
9. **this**.name = "Cat";
10. }
12. var animal = **new** Animal();
13. var cat = **new** Cat();
15. //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。
16. //输入结果为"Cat"
17. animal.showName.call(cat,",");
18. //animal.showName.apply(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

c、实现继承

**Java代码**

1. function Animal(name){
2. **this**.name = name;
3. **this**.showName = function(){
4. alert(**this**.name);
5. }
6. }
8. function Cat(name){
9. Animal.call(**this**, name);
10. }
12. var cat = **new** Cat("Black Cat");
13. cat.showName();

 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.

d、多重继承

**Java代码**

1. function Class10()
2. {
3. **this**.showSub = function(a,b)
4. {
5. alert(a-b);
6. }
7. }
9. function Class11()
10. {
11. **this**.showAdd = function(a,b)
12. {
13. alert(a+b);
14. }
15. }
17. function Class2()
18. {
19. Class10.call(**this**);
20. Class11.call(**this**);
21. }

 很简单，使用两个 call 就实现多重继承了  
当然，js的继承还有其他方法，例如使用原型链，这个不属于本文的范畴，只是在此说明call 的用法。说了call ，当然还有 apply，这两个方法基本上是一个意思，区别在于 call 的第二个参数可以是任意类型，而apply的第二个参数必须是数组，也可以是arguments  
还有 callee，caller..