**Call和Apply的作用与区别**

JavaScript中的每一个Function对象都有一个apply()方法和一个call()方法，它们的语法分别为：

/\*apply()方法\*/function.apply(thisObj[, argArray])

/\*call()方法\*/function.call(thisObj[, arg1[, arg2[, [,...argN]]]]);

**它们各自的定义：**

　　　   apply：应用某一对象的一个方法，用另一个对象替换当前对象。例如：B.apply(A, arguments);即A对象应用B对象的方法。

　　　　call：调用一个对象的一个方法，以另一个对象替换当前对象。例如：B.call(A, args1,args2);即A对象调用B对象的方法。

**它们的共同之处：**

　　　　都“可以用来代替另一个对象调用一个方法，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”。

**它们的不同之处：**

　　　　apply：最多只能有两个参数——新this对象和一个数组argArray。如果给该方法传递多个参数，则把参数都写进这个数组里面，当然，即使只有一个参数，也要写进数组里。如果argArray不是一个有效的数组或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。

　　　　call：它可以接受多个参数，第一个参数与apply一样，后面则是一串参数列表。这个方法主要用在js对象各方法相互调用的时候，使当前this实例指针保持一致，或者在特殊情况下需要改变this指针。如果没有提供thisObj参数，那么 Global 对象被用作thisObj。

　　　　实际上，apply和call的功能是一样的，只是传入的参数列表形式不同。

**示例代码：**

　　（1）基本用法

[IMG_256](http://www.cnblogs.com/lengyuehuahun/p/javascript:void(0);" \o "复制代码)

function add(a,b){

return a+b;

}function sub(a,b){

return a-b;

}var a1 = add.apply(sub,[4,2]);　　//sub调用add的方法var a2 = sub.apply(add,[4,2]);

alert(a1); //6

alert(a2); //2  
  
/\*call的用法\*/  
var a1 = add.call(sub,4,2);

[IMG_257](http://www.cnblogs.com/lengyuehuahun/p/javascript:void(0);" \o "复制代码)

1. 改变this指向

如下例：

|  |  |  |
| --- | --- | --- |
| 代码如下 | 复制代码 | |
| function b() { alert(this) } b(); //window b.call(); //window b.call(“a”,2,3); //a | |

再看一个复杂的例子：

|  |  |  |
| --- | --- | --- |
| 代码如下 | 复制代码 | |
| function Animal() { this.name=”animal”; this.showName=function() { alert(this.name) } } function Cat() { this.name=”cat”; } var animal = new Animal(); var cat = new Cat(); animal.showName(); //结果为animal animal.showName.call(cat); //原本cat没有showName方法，但是通过call方法将animal的showName方法应用到cat上,因此结果为cat | |

（3）实现继承

[IMG_258](http://www.cnblogs.com/lengyuehuahun/p/javascript:void(0);" \o "复制代码)

function Animal(name){

this.name = name;

this.showName = function(){

alert(this.name);

}

}

function Cat(name){

Animal.apply(this,[name]);

}

var cat = new Cat("咕咕");

cat.showName();

/\*call的用法\*/

Animal.call(this,name);

[IMG_259](http://www.cnblogs.com/lengyuehuahun/p/javascript:void(0);" \o "复制代码)

　　（4）多重继承

[IMG_260](http://www.cnblogs.com/lengyuehuahun/p/javascript:void(0);" \o "复制代码)

function Class10(){

this.showSub = function(a,b){

alert(a - b);

}

}

function Class11(){

this.showAdd = function(a,b){

alert(a + b);

}

}

function Class12(){

Class10.apply(this);

Class11.apply(this);

// Class10.call(this);

//Class11.call(this); }

var c2 = new Class12();

c2.showSub(3,1); //2

c2.showAdd(3,1); //4

[IMG_261](http://www.cnblogs.com/lengyuehuahun/p/javascript:void(0);" \o "复制代码)

**apply的一些其他巧妙用法**

（1）Math.max 可以实现得到数组中最大的一项：

　　　　因为Math.max不支持Math.max([param1,param2])也就是数组，但是它支持Math.max(param1,param2...)，所以可以根据apply的特点来解决 var max=Math.max.apply(null,array)，这样就轻易的可以得到一个数组中的最大项（apply会将一个数组转换为一个参数接一个参数的方式传递给方法）

          这块在调用的时候第一个参数给了null，这是因为没有对象去调用这个方法，我只需要用这个方法帮我运算，得到返回的结果就行，所以直接传递了一个null过去。

　　　 用这种方法也可以实现得到数组中的最小项：Math.min.apply(null,array)

　　（2）Array.prototype.push可以实现两个数组的合并

　　　　同样push方法没有提供push一个数组，但是它提供了push(param1,param2...paramN)，同样也可以用apply来转换一下这个数组，即：

 　　　 1.var arr1=new Array("1","2","3");

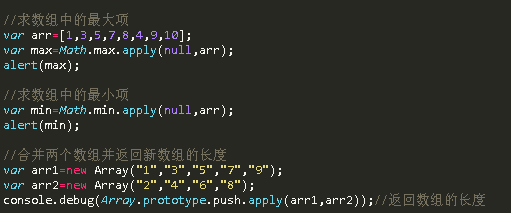
          2.var arr2=new Array("4","5","6");

          3.Array.prototype.push.apply(arr1,arr2);    //得到合并后数组的长度，因为push就是返回一个数组的长度

          也可以这样理解，arr1调用了push方法，参数是通过apply将数组转换为参数列表的集合

**通常在什么情况下，可以使用apply类似Math.max等之类的特殊用法：**

一般在目标函数只需要n个参数列表，而不接收一个数组的形式，可以通过apply的方式巧妙地解决这个问题。



1. Array.prototype.slice.call(arguments)

将具有length属性的对象转成数组 。