最近又遇到了JacvaScript中的call()方法和apply()方法，而在某些时候这两个方法还确实是十分重要的，那么就让我总结这两个方法的使用和区别吧。

**1. 每个函数都包含两个非继承而来的方法：call()方法和apply()方法。**

**2. 相同点：这两个方法的作用是一样的。**

**都是在特定的作用域中调用函数，等于设置函数体内this对象的值，以扩充函数赖以运行的作用域。**

一般来说，this总是指向调用某个方法的对象，但是使用call()和apply()方法时，就会改变this的指向。

**call()方法使用示例：**

//例1

<script>

window.color = 'red';

document.color = 'yellow';

var s1 = {color: 'blue' };

function changeColor(){

console.log(this.color);

}

changeColor.call(); //red (默认传递参数)

changeColor.call(window); //red

changeColor.call(document); //yellow

changeColor.call(this); //red

changeColor.call(s1); //blue

</script>

//例2

var Pet = {

words : '...',

speak : function (say) {

console.log(say + ''+ this.words)

}

}

Pet.speak('Speak'); // 结果：Speak...

var Dog = {

words:'Wang'

}

//将this的指向改变成了Dog

Pet.speak.call(Dog, 'Speak'); //结果： SpeakWang

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35

**apply()方法使用示例：**

//例1

<script>

window.number = 'one';

document.number = 'two';

var s1 = {number: 'three' };

function changeColor(){

console.log(this.number);

}

changeColor.apply(); //one (默认传参)

changeColor.apply(window); //one

changeColor.apply(document); //two

changeColor.apply(this); //one

changeColor.apply(s1); //three

</script>

//例2

function Pet(words){

this.words = words;

this.speak = function () {

console.log( this.words)

}

}

function Dog(words){

//Pet.call(this, words); //结果： Wang

Pet.apply(this, arguments); //结果： Wang

}

var dog = new Dog('Wang');

dog.speak();

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31

**3. 不同点：接收参数的方式不同。**

* **apply()方法** 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。

**语法：**apply([thisObj [,argArray] ]);，调用一个对象的一个方法，2另一个对象替换当前对象。

**说明：**如果argArray不是一个有效数组或不是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对象将用作thisObj。

* **call()方法** 第一个参数和apply()方法的一样，但是传递给函数的参数必须列举出来。

**语法：**call([thisObject[,arg1 [,arg2 [,...,argn]]]]);，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：** call方法可以用来代替另一个对象调用一个方法，call方法可以将一个函数的对象上下文从初始的上下文改变为thisObj指定的新对象，如果没有提供thisObj参数，那么Global对象被用于thisObj。

使用示例1：

function add(c,d){

return this.a + this.b + c + d;

}

var s = {a:1, b:2};

console.log(add.call(s,3,4)); // 1+2+3+4 = 10

console.log(add.apply(s,[5,6])); // 1+2+5+6 = 14

* 1
* 2
* 3
* 4
* 5
* 6
* 7

使用示例2：

<script>

window.firstName = "Cynthia";

window.lastName = "\_xie";

var myObject = {firstName:'my', lastName:'Object'};

function getName(){

console.log(this.firstName + this.lastName);

}

function getMessage(sex,age){

console.log(this.firstName + this.lastName + " 性别: " + sex + " age: " + age );

}

getName.call(window); // Cynthia\_xie

getName.call(myObject); // myObject

getName.apply(window); // Cynthia\_xie

getName.apply(myObject);// myObject

getMessage.call(window,"女",21); //Cynthia\_xie 性别: 女 age: 21

getMessage.apply(window,["女",21]); // Cynthia\_xie 性别: 女 age: 21

getMessage.call(myObject,"未知",22); //myObject 性别: 未知 age: 22

getMessage.apply(myObject,["未知",22]); // myObject 性别: 未知 age: 22

</script>

# 另一个例子

<script type="text/javascript">

2 /\*定义一个人类\*/

3 function Person(name,age)

4 {

5 this.name=name;

6 this.age=age;

7 }

8 /\*定义一个学生类\*/

9 functionStudent(name,age,grade)

10 {

11 Person.apply(this,arguments);

12 this.grade=grade;

13 }

14 //创建一个学生类

15 var student=new Student("qian",21,"一年级");

16 //测试

17 alert("name:"+student.name+"\n"+"age:"+student.age+"\n"+"grade:"+student.grade);

18 //大家可以看到测试结果name:qian age:21 grade:一年级

19 //学生类里面我没有给name和age属性赋值啊,为什么又存在这两个属性的值呢,这个就是apply的神奇之处.

20 </script>