apply、call与bind

2019年12月10日

11:48

apply与call作用 : 使用时能改变对象的上下文(暂时) ,不同在于 : call 需要把参数按顺序传递进去，而 apply 则是把参数放在数组里。传入多少个参数是不确定时,使用apply是最好的

**例子:**

func.call(this, arg1, arg2);

func.apply(this, [arg1, arg2])

**apply和call的妙用 :**

1.添加数组数据到另一个数组

|  |  |
| --- | --- |
|  | var array1 = [12 , "foo" , {name "Joe"} , -2458];  var array2 = ["Doe" , 555 , 100];  Array.prototype.push.apply(array1, array2);  /\* array1 值为 [12 , "foo" , {name "Joe"} , -2458 , "Doe" , 555 , 100] \*/ |

原因: apply 的 参数是数组的形式

2. 打印多个参数

function log(){

console.log.apply(console, arguments);

};

log(1); //1

log(1,2); //1 2

3.给每一个log消息添加一个前缀

function log(){

var args = Array.prototype.slice.call(arguments);

args.unshift('(app)');

console.log.apply(console, args);

};

**bind()**

bind能改变函数的上下文环境，当调用这个绑定函数时，绑定函数会以创建它时传入 bind()方法的第一个参数作为 this，传入 bind() 方法的第二个以及以后的参数加上绑定函数运行时本身的参数按照顺序作为原函数的参数来调用原函数。

例子:

var bar = function(){

console.log(this.x);

}

var foo = {

x:3

}

bar(); // undefined

var func = bar.bind(foo);

func(); // 3

注意: 在Javascript中，多次 bind() 是无效的,只有第一次的bind()起效.

例子:

var bar = function(){

console.log(this.x);

}

var foo = {

x:3

}

var sed = {

x:4

}

var func = bar.bind(foo).bind(sed);

func(); //?

var fiv = {

x:5

}

var func = bar.bind(foo).bind(sed).bind(fiv);

func(); //3

//3

总结:

apply 、 call 、bind 三者都是用来改变函数的this对象的指向的；

apply 、 call 、bind 三者第一个参数都是this要指向的对象，也就是想指定的上下文；

apply 、 call 、bind 三者都可以利用后续参数传参；

bind 是返回对应函数，便于稍后调用；apply 、call 则是立即调用 。