https://www.zhihu.com/question/20289071

obj.call(thisObj, arg1, arg2, ...);
obj.apply(thisObj, [arg1, arg2, ...]);
两者作用一致，都是把obj(即this)绑定到thisObj，这时候thisObj具备了obj的属性和方法。或者说thisObj『继承』了obj的属性和方法。

LZ要先明白存在call和apply的原因，才能记得牢一点：
在javascript OOP中，我们经常会这样定义：

```
function cat(){
}
cat.prototype={
food:"fish",
say: function(){
alert("I love "+this.food);
}
}

var blackCat = new cat;
blackCat.say();
```

但是如果我们有一个对象whiteDog = {food:"bone"},我们不想对它重新定义say方法，那么我们可以通过call或apply用blackCat的say方法：blackCat.say.call(whiteDog);
所以，可以看出call和apply是为了动态改变this而出现的，当一个object没有某个方法，但是其他的有，我们可以借助call或apply用其它对象的方法来操作。
用的比较多的，通过document.getElementsByTagName选择的dom 节点是一种类似array的array。它不能应用Array下的push,pop等方法。我们可以通过：
var domNodes = Array.prototype.slice.call(document.getElementsByTagName("*"));
这样domNodes就可以应用Array下的所有方法了。
其他的就不提了，讲多了反而迷惑。

obj.call(thisObj, arg1, arg2, ...);
obj.apply(thisObj, [arg1, arg2, ...]);

两者作用一致，都是把obj(即this)绑定到thisObj，这时候thisObj具备了obj的属性和方法。或者说thisObj『继承』了obj的属性和方法。

唯一区别是apply接受的是数组参数，call接受的是连续参数。

```
function add(j, k){
    return j+k;
}

function sub(j, k){
    return j-k;
}
```

我们在控制台运行：

```
add(5,3); //8
add.call(sub, 5, 3); //8
add.apply(sub, [5, 3]); //8

sub(5, 3); //2
sub.call(add, 5, 3); //2
sub.apply(add, [5, 3]); //2
```

通过call和apply，我们可以实现对象继承。示例：

```
var Parent = function(){
    this.name = "yjc";
    this.age = 22;
}

var child = {};

console.log(child);//Object {} ,空对象

Parent.call(child);

console.log(child); //Object {name: "yjc", age: 22}
```

以上实现了对象的继承。

call和apply可以用来重新定义函数的执行环境，也就是this的指向。通过一个操作DOM的例子来理解。

```
function changeStyle(attr, value){
    this.style[attr] = value;
}
var box = document.getElementById('box');
window.changeStyle.call(box, "height", "200px");
```

call中的第一个参数用于指定将要调用此函数的对象，在这里，changeStyle函数将被box对象调用，this指向了box对象，如果不用call的话，程序报错，因为window对象中没有style属性。

apply的用法:

```
window.changeStyle.apply(box, ['height', '200px']);
```

=========更新：2016年10月08日========

如果call或apply的第一参数是null的话， this指向window

函数调用的三种方式:

```
obj.myFunc();
myFunc.call(obj,arg);
myFunc.apply(obj,[arg1,arg2..]);
```

```
function cat(){};
        /* cat.prototype.eat=function(){
                console.log("I like "+this.food);
        };
        cat.prototype.food="fish"; */
        cat.prototype={
                        food:"fish",
                        eat:function(){
                                console.log(" I like "+this.food);
                        }
        }
```

```
var blackCat = new cat();
blackCat.eat();

var dog = {food:"bone"};
blackCat.eat.call(dog);//call 调用另一个对象的方法


function add(a,b){
        console.log(a+b);
}
function sub(c,d){
        console.log(c-d);
}
add(3,4);
sub(1,5);
add.call(sub,3,4);//7
add.apply(sub,[3,4]);//7
```