call 和 apply 是 JS 中比较重要的两个方法， 一般在框架和组件设计中用的较多，比如 jQuery Code。 那么这两个方法是做什么的呢，下面我们通过代码来了解：

```
1 function f(){
2       console.log(this.name);
3  }
4
5  var obj = {};
6  obj.name = "obj";
7
8  f.call(obj); //obj
```

f()中输出了 this.name, 这里的 this 指的是函数在运行时的调用者（关于 JS 中的 this,请参考：JS 作用域和 this 关键字），因此 f()输出的结果取决于调用它的对象。

obj 是 javascript 中的一个普通对象，obj.name = "obj"; 是在 obj 上添加了一个属性 name,并赋值"obj"。 一般情况下,只有 obj 能访问自己的属性 name,f 为何能访问到 obj.name 呢？

call 方法的作用就体现在这里：

f.call(obj) 翻译成白话文就是：在 obj 对象的执行环境调用 f()，相当于 obj.f() ，此时 f 中的 this 指的就是 obj, this.name 相当于 obj.name,所以输出"obj"。

这是对 call 方法最简单的解释。

call 还有个兄弟 apply, apply 和 call 唯一的区别是传递参数形式不同，call(obj,args..) 可以传递一个参数列表， apply(obj,args[]) 只能传递一个 Array 参数。

当然，这兄弟俩还有其他作用，那就是 JS 中的继承。

JS 中没有诸如 Java、C#等高级语言中的 extend 关键字，因此 JS 中没有继承的概念，如果一定要继承的话，call 和 apply 可以实现这个功能：

```
1 function Animal(name,weight){
2       this.name = name;
3       this.weight = weight;
4  }
5
6  function Cat(){
7       Animal.call(this,'cat','50');
```

```
 8         //Animal.apply(this,['cat','50']);
 9
10         this.say = function(){
11             console.log("I am " + this.name+",my weight is " +
this.weight);
12         }
13     }
14
15     var cat = new Cat();
16     cat.say();//I am cat,my weight is 50
```

从输出结果看，Cat 确实继承了 Animal 中的属性，继承的关键在于 Animal.call(this,'cat','50') 这句话！

翻译成白话文就是：在执行 Cat()时，先在其调用对象 this 的执行环境调用 Animal()，相当于 this.Animal()。 this.Animal()执行完以后，this 上就产生了 name 和 weight 属性，而 say 方法的调用者也是 this，因此 say()中访问的 this.name 就是 Animal 中的 name 属性。所以输出的是 Animal 的属性值。

```
var cat = new Cat();
```

这句话执行之后，this 就是 cat， cat.say() 就相当于 this.say() 。

至于 this 为什么是 cat， 请参考：剖析 JS 中 new 的运行机制

update at 2015-12-11 12:04:16

update at 2015-12-11 19:48:45

update at 2015-12-15 13:00:23