2018-01-01JavaScript设计模式

this 、call 和 apply

1 . this 的指向

1 . 作为对象的方法调用

当函数作为作为对象的方法被调用时，this 指向改对象

```
var obj = {
    a : 1,
    b : function(){
        return this.a;
    }
}
console.log(obj.b)       // 1
```

2 . 作为普通函数调用

当函数不作为对象调用时，也就是我们常说的普通函数调用，此时的 this 总是指向全局对象 window 。

```
window.name = 'Gloable Name';
var getName = function(){
    return this.name
}
console.log(getName())       // Gloable Name    此时的函数内部的 this 指向的是全局 window。
```

或者：

```
window.name = 'Gloable Name';
var obj = {
    name: 'seven',
    getName:fucntion(){
        return this.name;
    }
}
console.log(obj.getName());      // seven    对象调用时 this 指向的是当前的对象
var getName = obj.getName;
console.log(getName());          // Gloable Name    对象的函数被当做普通函数调用
```

3 . 构造器调用

构造器里面的 this 指向 new 关键字返回的这个对象

```
var MyClass = function(){
    this.name = 'seven';
}
var obj = new MyClass();
console.log(obj.name);           // seven
```

但是，使用 new 调用构造函数的时候，还需要注意一个问题，如果构造函数显示的返回了一个 object 类型的对象，那么此次运算结果最终会返回一个对象，而不是我们之前的 this;

```
var MyClass = function(){
    this.name = 'seven';
```

```
        return {
            name : 'anne';
        }
    }
    var obj = new MyClass();
    console.log(obj)                // anne
```

如果构造器不显示的返回任何数据，或者是返回一个非对象类型数据就不会出现上面的问题

4 . Function.prototype.call 或者 Function.prototype.apply 调用

跟普通的函数调用对比，用 Function.prototype.call 或 Function.prototype.apply 可以动态的改变传入函数的 this

```
    var obj1 = {
        name: 'seven',
        getName:function(){
            return this.name;
        }
    }
    var obj2 = {
        name: 'anne'
    }
    console.log(obj1.getName())     // seven
    console.log(obj1.getName.call(obj2))        // anne
```

2 . call 和 apply

1 . call 和 apply 的区别

Function.prototype.call 和 Function.prototype.apply 都是非常常用的方法。它们的作用一模一样，区别仅在于传入参数形式的不同

apply 接收两个参数，第一个参数指定了函数体内的 this 的指向，第二个参数为一个带下标的集合，这个集合可以是数组，也可以是类数组。

```
    var func = function(a,b,c){
        alert([a,b,c]);             // [1,2,3]
    }
    func.apply(null,[1,2,3]);
```

call 传入的参数量不固定，跟 apply 相同的是，第一个参数也代表函数体内的 this 指向，第二个参数开始往后，每一个参数依次被传入函数

```
    var func = function(a,b,c){
        alert([a,b,c]);             // [1,2,3]
    }
    func.call(null,1,2,3);
```

3 . call 和 apply 的用途

1 . 改变 this 的指向

call 和 apply 最常见的用途就是改变函数内部的 this 的指向。