**学习目标：**

1、apply

2、call

3、bind

函数.apply()方法 和 函数.call()方法

说到底，它们两个只是调用函数的一种方式。实现的功能是：即调用了函数，又为函数指定了this。（自己理解：用在函数自执行时，而不是事件调用，事件调用时，还得再包一层函数）

apply 申请

call 呼叫

如：下面这个json是一个对象，也就是一个环境，同理，window也是一个对象，也是一个环境。

varjson = {

name : "彭作洪",

age : 29

}

function sum(){

console.log(this.name); //弹出调用它的对象的name属性。

}

sum.apply(json); //函数的this为json，它有name这个属性，所以弹出彭作洪

sum.call(json); //函数的this为json，它有name这个属性，所以弹出彭作洪

sum(); //函数里的this是window。它没有name这个属性。所以弹出空字符串

说明： apply和call方法都是函数下面的方法。作用是调用函数，然后在调用函数时，修改函数内部的this指向，指向这个方法传入的第一个参数。

apply()方法接收两个参数：一个是在其中运行函数的作用域，另一个参数是数组。参数数组可以是Array的实例。也可以是arguments对象。

call()方法与apply()方法的作用相同，它们的区别仅在于接收参数的方式不同。对于call()方法而言，第一个参数是this值没有变化，变化的是其余参数都直接传递给函数，换句话说，在使用call()方法时，传递给函数的参数必须逐个列举出来。

至于是使用apply()还是call()，完全取决于你采取哪种函数传递参数的方式。

但如果apply和call的参数为空，哪么函数的this默认就指向了window，这点要注意，如果就是不想改函数的this，可以写上null。或者就不用这两个方法。

解释 call 与 apply 的区别

call 和 apply 都是函数下的一个静态方法，可以通过函数.call() 或函数.apply() 的方式来间接调用该函数，通过 call 或 apply 执行时候的第一个参数改变该函数执行过程中的上下文对象（this），如果第一个参数不存在或者设置成 null/undefined，那么该函数执行过程中的上下文对象指向全局上下文对象，在 JavaScript 中指向了 window 对象。

不同之处在于后续参数上，call 第一个参数以后的参数值将被一一对应的赋值给源函数的形参，而 apply 则是传入一个数组，间接传给函数的 arguments 对象。

函数.bind(第一个参数，第n个参数)。这个方法会创建一个当前函数的实例，这个实例的this值会指向传给bind()函数的第一个参数。

如：

functionppp(){

alert(this);

}

var c = ppp.bind("body");

c(); //body

从此，要调用ppp时，就调用c即可。

它即可以用在函数自执行，也可以用在事件调用上。

又如：bind方法创建函数的一个新的实例，这个新实例的this值指向传给bind()方法的第一个参数。

varjson = {

name : "彭作洪",

age : 29

}

function sum(){

console.log(this.name);

}

var c = sum.bind(json);

c(); //彭作洪

bind和apply\call的区别：

1、apply和call是在调用时，才改函数内部的this指向。而bind是创建一个函数的实例，改变这个实例的this指向。

2、bind是ES5中的方法，在IE9及以上才被支持。

3、bind更强大，它可以自执行，也可以用在事件调用上。而apply和call只能用在自执行。如果要用事件调用，还得再加一层函数。