17.9.12 函数调用模式

在 JavaScript 中，共有 4 种函数调用模式：方法调用模式、函数调用模式、构造器调用模式和 apply 调用模式。这些模式在如何初始化 this 上存在差异。

【提示】

调用运算符是小括号，小括号内可以包含零个或多个用逗号隔开的表达式。每个表达式产生一个参数值。每个参数值被赋予函数声明时定义的形参。当实际参数（arguments）的个数与形式参数（parameters）的个数不匹配时不会导致运行时错误。如果实际参数值过多，超出的参数值将被忽略。如果实际参数值过少，缺失的值将会被替换为 undefined。不会对参数值进行类型检查，任何类型的值都可以被传递给参数。

【示例 1】方法调用模式。

当一个函数被保存为对象的一个属性值时，将称之为一个方法。当一个方法被调用时，this 被绑定到当前调用对象。

```
var obj = {
    value : 0,
    increment : function(inc) {
        this.value += typeof inc === 'number' ? inc : 1;
    }
}
obj.increment();
document.writeln(obj.value);     // 1
obj.increment(2);
document.writeln(obj.value);    // 3
```

在上面代码中创建了 obj 对象，它有一个 value 属性和一个 increment 方法。increment 方法接受一个可选的参数，如果该参数不是数字，那么默认使用数字 1。

increment 方法可以使用 this 去访问对象，所以它能从对象中取值或修改该对象。this 到对象的绑定发生在调用的时候。这个延迟绑定使函数可以对 this 高度复用。通过 this 可取得 increment 方法所属对象的上下文的方法称为公共方法。

【示例 2】函数调用模式。

当一个函数不是一个对象的属性时，它将被当作一个函数来调用：

```
var sum = add(3, 4);     //7
```

当函数以此模式调用时，this 被绑定到全局对象。这是语言设计上的一个缺陷，如果语言设计正确，当内部函数被调用时，this 应该仍绑定到外部函数的 this 变量。这个设计错误的后果是方法不能利用内部函数来帮助它工作，因为内部函数的 this 被绑定了错误的值，所以不能共享该方法对对象的访问权。

解决方案：如果该方法定义一个变量并将它赋值为 this，那么内部函数就可以通过这个变量访问 this。按照约定，将这个变量命名为 that。

```
var obj = {
    value : 1,
    doub : function() {
        var that = this;
        var helper = function() {
            that.value = that.value * 2;
        };
        helper();
    }
}
obj.doub();
document.writeln(obj.value);     // 2
```

【示例 3】构造器调用模式。

JavaScript 是基于原型继承的语言，对象可以直接从其他对象继承属性。当今大多数语言都是基于类的语言，虽然原型继承有着强大的表现力，但它偏离了主流用法，并不被广泛理解。JavaScript 为了能够兼容基于类语言的编写风格，提供了一套基于类似类语言的对象构建语法。

如果在一个函数前面加上 new 运算符来进行调用，那么将创建一个隐藏连接到该函数的 prototype 原型对象的新实例对象，同时 this 将会被绑定到这个新实例对象上。注意，new 运算符也会改变 return 语句的行为。

```
var F = function(string) {
    this.status = string;
};
F.prototype.get = function() {
    return this.status;
};
var f = new F("new object");
document.writeln(f.get()); //"new object"
```

上面代码创建一个名为 F 的构速函数，此函数构建了一个带有 status 属性的对象。然后，为 F 所有实例提供一个名为 get 的公共方法。最后，创建一个实例对象，并调用 get 方法，以读取 status 属性的值。

结合 new 前缀调用的函数被称为构造函数。按照约定，构造函数应该保存在以大写格式命名的变量中。如果调用构造函数时没有在前面加上 new，可能会发生非常糟糕的事情，既没有编译时警告，也没有运行时警告，所以大写约定非常重要。

【示例 4】apply 调用模式。

JavaScript 是函数式的面向对象编程语言，函数可以拥有方法。apply 就是函数的一个基本方法，使用这个方法可以调用函数，并修改函数体内的 this 值。apply 方法包括两个参数：第一个参数设置绑定给 this 的值;第二个参数是包含函数参数的数组。

```
var array = [5, 4];
var add = function() {
    var i, sum = 0;
    for( i = 0; i < arguments.length; i += 1) {
        sum += arguments[i];
    }
    return sum;
};
var sum = add.apply({}, array);      // 9
```

上面代码构建一个包含两个数字的数组，然后使用 apply 方法调用 add()函数，将数组 array 中的元素值相加。

```
var F = function(string) {
    this.status = string;
};
F.prototype.get = function() {
    return this.status;
};
var obj = {
    status: 'obj'
};
var status = F.prototype.get.apply(obj);     //"obj"
```

上面代码构建了一个构造函数 F，为该函数定义了一个原型方法 get，该方法能够读取当前对象的 status 属性的值。然后定义一个 obj 对象，该对象包含一个 status 属性，使用 apply 方法在 obj 对象上调用构造函数 F 的 get 方法，将会返回 obj 对象的 status 属性值。