第 11 章 Function 类型

学习要点：

1.函数的声明方式
2.作为值的函数
3.函数的内部属性
4.函数属性和方法

主讲教师：李炎恢
合作网站：http://www.ibeifeng.com
讲师博客：http://hi.baidu.com/李炎恢

在 ECMAScript 中，Function(函数)类型实际上是对象。每个函数都是 Function 类型的实例，而且都与其他引用类型一样具有属性和方法。由于函数是对象，因此函数名实际上也是一个指向函数对象的指针。

一．函数的声明方式

1.普通的函数声明

```
function box(num1, num2) {
    return num1+ num2;
}
```

2.使用变量初始化函数

```
var box= function(num1, num2) {
    return num1 + num2;
};
```

3.使用 Function 构造函数

```
var box= new Function('num1', 'num2' ,'return num1 + num2');
```

PS：第三种方式我们不推荐，因为这种语法会导致解析两次代码（第一次解析常规 ECMAScript 代码，第二次是解析传入构造函数中的字符串），从而影响性能。但我们可以通过这种语法来理解"函数是对象，函数名是指针"的概念。

二．作为值的函数

ECMAScript 中的函数名本身就是变量，所以函数也可以作为值来使用。也就是说，不仅可以像传递参数一样把一个函数传递给另一个函数，而且可以将一个函数作为另一个函数的结果返回。

```
function box(sumFunction, num) {
    return sumFunction(num);                    //someFunction
}

function sum(num) {
    return num + 10;
```

```
}

var result = box(sum, 10);                    //传递函数到另一个函数里
```

三．函数内部属性

在函数内部，有两个特殊的对象：arguments 和 this。arguments 是一个类数组对象，包含着传入函数中的所有参数，主要用途是保存函数参数。但这个对象还有一个名叫 callee 的属性，该属性是一个指针，指向拥有这个 arguments 对象的函数。

```
function box(num) {
    if (num <= 1) {
        return 1;
    } else {
        return num * box(num-1);              //一个简单的的递归
    }
}
```

对于阶乘函数一般要用到递归算法，所以函数内部一定会调用自身；如果函数名不改变是没有问题的，但一旦改变函数名，内部的自身调用需要逐一修改。为了解决这个问题，我们可以使用 arguments.callee 来代替。

```
function box(num) {
    if (num <= 1) {
        return 1;
    } else {
        return num * arguments.callee(num-1);//使用 callee 来执行自身
    }
}
```

函数内部另一个特殊对象是 this，其行为与 Java 和 C#中的 this 大致相似。换句话说，this 引用的是函数据以执行操作的对象，或者说函数调用语句所处的那个作用域。PS：当在全局作用域中调用函数时，this 对象引用的就是 window。

```
//便于理解的改写例子
window.color = '红色的';                      //全局的，或者 var color = '红色的';也行
alert(this.color);                            //打印全局的 color

var box = {
    color : '蓝色的',                         //局部的 color
    sayColor : function () {
        alert(this.color);                    //此时的 this 只能 box 里的 color
    }
};

box.sayColor();                               //打印局部的 color
alert(this.color);                            //还是全局的
```

```
//引用教材的原版例子
window.color = '红色的';                    //或者 var color = '红色的';也行

var box = {
    color : '蓝色的'
};

function sayColor() {
    alert(this.color);                      //这里第一次在外面，第二次在 box 里面
}

getColor();

box.sayColor = sayColor;                    //把函数复制到 box 对象里，成为了方法
box.sayColor();
```

四．**函数属性和方法**

ECMAScript 中的函数是对象，因此函数也有属性和方法。每个函数都包含两个属性：length 和 prototype。其中，length 属性表示函数希望接收的命名参数的个数。

```
function box(name, age) {
    alert(name + age);
}
alert(box.length);                          //2
```

PS：对于 prototype 属性，它是保存所有实例方法的真正所在，也就是原型。这个属性，我们将在面向对象一章详细介绍。而 prototype 下有两个方法：apply()和 call()，每个函数都包含这两个非继承而来的方法。这两个方法的用途都在特定的作用域中调用函数，实际上等于设置函数体内 this 对象的值。

```
function box(num1, num2) {
    return num1 + num2;                     //原函数
}

function sayBox(num1, num2) {
    return box.apply(this, [num1, num2]);   //this 表示作用域，这里是 window
}                                           //[]表示 box 所需要的参数

function sayBox2(num1, num2) {
    return box.apply(this, arguments);      //arguments 对象表示 box 所需要的参数
}

alert(sayBox(10,10));                       //20
alert(sayBox2(10,10));                      //20
```

call()方法于 apply()方法相同，他们的区别仅仅在于接收参数的方式不同。对于 call()方法而言，第一个参数是作用域，没有变化，变化只是其余的参数都是直接传递给函数的。

```
function box(num1, num2) {
    return num1 + num2;
}

function callBox(num1, num2) {
    return box.call(this, num1, num2);          //和 apply 区别在于后面的传参
}

alert(callBox(10,10));
```

事实上，传递参数并不是 apply()和 call()方法真正的用武之地；它们经常使用的地方是能够扩展函数赖以运行的作用域。

```
var color = '红色的';                          //或者 window.color = '红色的';也行

var box = {
    color : '蓝色的'
};

function sayColor() {
    alert(this.color);
}

sayColor();                                    //作用域在 window

sayColor.call(this);                           //作用域在 window
sayColor.call(window);                         //作用域在 window
sayColor.call(box);                            //作用域在 box，对象冒充
```

这个例子是之前作用域理解的例子修改而成，我们可以发现当我们使用 call(box)方法的时候，sayColor()方法的运行环境已经变成了 box 对象里了。

使用 call()或者 apply()来扩充作用域的最大好处，就是对象不需要与方法发生任何耦合关系(耦合，就是互相关联的意思，扩展和维护会发生连锁反应)。也就是说，box 对象和 sayColor()方法之间不会有多余的关联操作，比如 box.sayColor = sayColor;

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(yc60.com)联合提供：

本次主讲老师：李炎恢

我的博客：hi.baidu.com/李炎恢/

我的邮件：yc60.com@gmail.com