## Function 类型

说起来 ECMAScript 中什么最有意思，我想那莫过于函数了——而有意思的根源，则在于**函数实际上是对象**。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**，不会与某个函数绑定。函数通常是使用函数声明语法定义的，如下面的例子所示。

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

这与下面使用函数表达式定义函数的方式几乎相差无几。

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

最后一种定义函数的方式是使用 Function 构造函数。Function 构造函数可以接收任意数量的参数，但最后一个参数始终都被看成是函数体，而前面的参数则枚举出了新函数的参数。来看下面的例子：

```
var sum = new Function("num1", "num2", "return num1 + num2"); // 不推荐
```

由于函数名仅仅是指向函数的指针，因此函数名与包含对象指针的其他变量没有什么不同。换句话说，一个函数可能会有多个名字，如下面的例子所示。

```
function sum(num1, num2){
    return num1 + num2;
}
alert(sum(10,10));        //20

var anotherSum = sum;
alert(anotherSum(10,10)); //20

sum = null;
alert(anotherSum(10,10)); //20
```

以上代码首先定义了一个名为 sum() 的函数，用于求两个值的和。然后，又声明了变量anotherSum ，并将其设置为与 sum 相等（将 sum 的值赋给 anotherSum ）。注意，**使用不带圆括号的函数名是访问函数指针**，而非调用函数。此时， anotherSum 和 sum 就都指向了同一个函数，因此 anotherSum() 也可以被调用并返回结果。即使将 sum 设置为 null ，让它与函数“断绝关系”，但仍然可以正常调用anotherSum()