用 Function 类直接创建函数的语法如下：

var function\_name = new function(arg1, arg2, ..., argN, function\_body)

**注意：**

***1：被函数使用的参数的名称必须是合法命名的。参数名称是一个有效的JavaScript标识符的字符串，或者一个用逗号分隔的有效字符串的列表;例如“×”，“theValue”，或“A，B”。***

***2： 使用Function构造器生成的函数，并不会在创建它们的上下文中创建闭包；它们一般在全局作用域中被创建。当运行这些函数的时候，它们只能访问自己的本地变量和全局变量，不能访问Function构造器被调用生成的上下文的作用域。这和使用带有函数表达式代码的 eval 不同。***

***3 : 以调用函数的方式调用Function的构造函数 (不是用new关键字) 跟以构造函数来调用是一样的.***

使用构造函数等同于直接声明一个函数:

function function\_name(atg1,arg2,…,argN){

Function\_body;

}

函数只不过是一种引用类型，它们的行为与用 Function 类明确创建的函数行为是相同的。

Function 实例从 Function.prototype 继承了一些属性和方法。 同其他构造函数一样， 你可以改变构造函数的原型从而使得所有的Function实例的属性和方法发生改变。

**Function的属性：**

Function.arguments

以数组形式获取传入函数的所有参数。此属性已被arguments替代。

Function. length

用于指定的函数的参数的个数。

Function.caller

获取调用函数的具体对象。

Function.length

获取函数的接收参数个数。

Function.name

获取函数的名称。

Function.displayName

获取函数的display name。

Function.prototype.constructor

声明函数的原型构造方法，详细请参考 Object.constructor 。

**Function的方法：**

Function.prototype.apply()

在一个对象的上下文中应用另一个对象的方法；参数能够以数组形式传入。

Function.prototype.bind()

bind()方法会创建一个新函数,称为绑定函数.当调用这个绑定函数时,绑定函数会以创建它时传入 bind()方法的第一个参数作为 this,传入 bind()方法的第二个以及以后的参数加上绑定函数运行时本身的参数按照顺序作为原函数的参数来调用原函数.

Function.prototype.call()

在一个对象的上下文中应用另一个对象的方法；参数能够以列表形式传入。

Function.prototype.isGenerator()

若函数对象为generator，返回true，反之返回 false。

Function.prototype.toSource()

获取函数的实现源码的字符串。 覆盖了 Object.prototype.toSource 方法。

Function.prototype.toString()

获取函数的实现源码的字符串。覆盖了 Object.prototype.toString 方法。