以表达式定义的函数，函数的名称是可选的。

一个函数定义表达式包含名称，函数的局部作用域将会包含一个绑定到函数对象的名称。

嵌套函数变量作用域规则：可以访问嵌套它们（或多重嵌套）的函数的参数和变量。

调用JS函数的4种方法：作为函数、作为方法、作为构造函数、通过它们的call()和apply()方法间接调用。

1.作为普通函数调用：1>函数返回值：如果该函数返回是因为解释器到达结尾，返回值是undefined。return后面有语句，返回语句值，return后面没有语句，返回undefined；ECS3非严格模式ECM5，函数调用，调用上下文为全局对象。严格模式下，调用上下文是undefined。

以函数形式调用的函数通常不使用this关键字。

2.作为方法调用：调用上下文为对象，函数体可以通过使用关键字this引用该对象。

方法链：当方法的返回值是一个对象，这个对象还可以再调用它的方法。这种方法调用序列中的每次调用结果都是另外一个表达式的组成部分。有一些函数用作内部函数或私有函数，这种函数通常以一条下划线为前缀。

以表达式定义个函数在定义之前无法调用。

return语句会导致函数停止执行，并返回它的表达式的值给调用者。如果return语句没有一个与之相关的表达式，则它返回undefined值。

如果一个函数不包含return语句，那它只执行函数体中的每条语句，并返回undefined值给调用者。

以语句声明形式定义个函数不能出现在循环、条件判断或try/cache/finally/with语句中。

调用函数的4种方式：作为函数、作为方法、作为构造函数、通过call()、apply()方法间接调用。

对于普通的函数调用，函数的返回值成为调用表达式的值。

当方法的返回值是一个对象，这个对象还可以再调用它的方法。这种方法调用序列中，每次的调用结果都是另外一个表达式的组成部分。

错误观点：调用嵌套函数时this会指向外层函数的上下文。

构造函数没有形参，JS构造函数调用的语法是允许省略实参列表和圆括号的。

JS中函数是对象。

arguments指向实参对象的引用，实参对象是一个类数组对象，实质是Object类型。arguments包含一个length属性

“不定实参函数”：可以接收人一个数的实参，不定实参函数实参个数不能为0。

实参对象具有：callee、caller属性。在严格模式下，对这两个属性的读写操作会产生一个类型错误。

在非严格模式下，callee属性指代当前正在执行的函数。

caller是非标准的，指代调用当前正在执行的函数的函数，大多数浏览器实现了这个属性。通过caller属性可以访问调用栈。

callee属性在匿名函数中来递归调用自身。例：var factorial=function(x){ if(x<=1) return 1;return x\*arguments.callee(x-1);}

在JS中是无法声明只在一个代码块内可见的变量的，基于这个原因，我们常简单地定义一个函数用作临时地命名空间，在这个命名空间内定义地变量不会污染到全局命名空间。

变量作用域是在函数定义时决定的。函数对象可以通过作用域链相互关联起来，函数体内部地变量可以保存在函数作用域内，这种特性称为“闭包”。

关联到闭包地作用域链都是‘活动的’。

函数嵌套不会将作用域内的私有成员复制一份，也不会对所绑定地变量生成静态快照。

函数的length属性是只读属性，代表函数参数的数量，这里的参数指的是‘形参’。

prototype是属性指向原型对象的引用。当将函数用作构造函数时，新创建的对象会从原型对象上继承属性。

call()和apply（）方法的第一个实参是要调用函数的母对象，它是调用上下文，在函数体内可以通过this来获得对它的引用。

对于call()来说，第一个调用上下文实参之后的所有实参要传入待调用函数的值。例：f.call(p,1,2):以对象p的方法的形式调用函数f()，并传入两个实参。

apply()所有实参都放在一个数组中：f.apply(p,{1,2});

bind()：主要作用是将函数绑定至某个对象。bind()除第一个实参之外，传入bind()的实参会绑定至this。

bind()返回一个函数对象，这个函数对象的length属性是绑定函数的形参个数减去绑定实参的个数（length的值不能小于0）。

Function()构造函数可以传入任意数量的字符串实参，最后一个实参所表示的文本就是函数体。它可以包含任意的JS语句，每两条语句之间用分号分隔。

Function（）构造函数允许JS在运行时动态地创建并编译函数。Function（）所创建地函数并不使用词法作用域。函数体代码地编译总是会在顶层函数执行。