# 函数对象

函数是被设计为执行特定任务的代码块，通过函数可以将具有相同或者相似逻辑的代码包裹起来，通过函数调用执行这些被包裹的代码逻辑。函数的作用就是有利于精简代码方便复用。

## 函数定义

### 普通函数

function square(number) {  
 return number \* number;  
}

### 函数表达式(匿名函数)

虽然上面的函数声明在语法上是一个语句，但函数也可以由函数表达式创建。这样的函数可以是**匿名**的；它不必有一个名称。例如，函数square也可这样来定义：

const square = function(number) { return number \* number; };   
var x = square(4); // x gets the value 16

即将匿名函数赋值给一个变量，并且通过变量名称进行调用 我们将这个称为函数表达式。

let fn = function(){   
 // 函数体   
}

调用匿名函数

fn()

### 立即执行函数

声明一个函数，并马上调用这个匿名函数就叫做立即执行函数。在定义好一个函数后，直接执行。  
立即执行函数的写法

第一种写法

(function(arg1,arg2){   
 //函数体   
})(x,y)

第二种写法

(function(arg1,arg2){  
 //函数体  
}(x,y))

立即执行函数作用

* 不必为函数命名，避免了污染全局变量。
* 立即执行函数内部形成了一个单独的作用域，可以封装一些外部无法读取的私有变量。
* 封装变量。

# 函数传参

## 传递参数

传递多个参数

function funcname(arg1,arg2){  
 //函数体  
}  
//调用函数  
funcname(1,2)

传递列表参数

function funcname(Arr){  
 //函数体  
}  
//调用函数  
funcname([1,2,3,4])

默认参数

function funcname(arg1=0,arg2=0){  
 //函数体  
}  
//调用函数  
funcname(1,2)

逻辑中断（逻辑短路）

function funcname(arg1=0,arg2=0){  
 arg1 = arg1 || 0  
 arg2 = arg2 || 0  
}  
//调用函数  
funcname(1,2)

# 函数返回值

使用return关键字来返回数据

function getSum(x,y) {   
 return x + y   
}   
console.log(get(x,y))

## 有返回值的函数

在函数体中使用 return 关键字能将内部的执行结果交给函数外部使用。

函数内部只能出现 1 次 return，并且 return 后面代码不会再被执行，所以 return 后面的数据不要换行写

## 无返回值的函数

函数可以没有 return，这种情况函数默认返回值为 undefined

# 函数作用域

通常来说，一段程序代码中所用到的名字并不总是有效和可用的，而限定这个名字的可用性的代码范围就是这 个名字的作用域。作用域的使用提高了程序逻辑的局部性，增强了程序的可靠性，减少了名字冲突。

## 作用域区分

### 全局作用域

作用于所有代码执行的环境

### 局部作用域

作用于函数内的代码环境，就是局部作用域。因为其与函数有关系，所以也称为函数作用域

### 块级作用域

块作用域由{}包括，if语句和for语句中的{}。

## 变量作用域

### 全局变量

函数外部let 的变量，全局变量在任何区域都 可以访问和修改

### 局部变量

函数内部let的变量，局部变量只能在当前函 数内部访问和修改

### 块级变量

{}内部的let变量。let定义的变量,只能在块作用域里访问,不能跨块访问,也不能跨函数访问。