1、函数。

JS中函数分为系统函数和自定义函数。

函数是一个功能体，提供若干个数据，返回处理的结果。用于将重复使用的代码封装。

定义一个函数：

function 函数名(参数1,参数2){

函数体; //重复执行的代码。

}

调用函数：函数名();会执行函数体中的代码。

练习：创建函数getSum(),封装计算1-100之间整数和，调用多次。

创建带有参数的函数。

function 参数名(参数列表){//参数列表可以是多项的。用于接收数据，

}

function getSum(n1, n2) { //n1,n2形式参数，用于接收数据，参数未赋值，返回undefined

return n1 + n2;

}

var sum = getSum(10, 12); //10,12实参，实际要传递的参数。函数调用。

console.log(sum)

参数：创建函数时的参数称为形参，用于接收数据，调用函数时的参数称为实参，实际要传递的数据，实参会赋值给形参，多个参数之间用逗号 ，隔开，实参的数量可以和形参的数量不等，如果形参未赋值为undefined。

调用函数，得到的是函数体中return后的值。

return：返回值，函数调用后得到的结果，return执行后，结束函数调用，后面代码不再执行。函数中没有return或者return后没有值，会得到undefined，

单独使用undefined是为了结束函数

function get(){

return ; //没有返回值，为了结束函数。

}

var res = get();

console.log(res) =========>undefined

function get1(){

}

var res\_ = get();

console.log(res\_) ========>undefined

对比return和break。

break用于结束switch或者循环，

return 用于函数中结束函数的调用

作用域：

①变量的作用域：变量的可访问范围。

全局作用域：在函数外使用var声明的变量，可以在任何的作用域下访问。===>全局变量

函数作用域：在函数内部使用var声明的变量，只能在当前的函数下访问。===>局部变量

function fn(){

n1 = 2;

}

fn();

console.log(n1); //函数内声明变量不使用var，声明的是一个全局变量。不推荐使用这种方式，后期严格模式下会报错

function fn(){

var m1=m2=m3=4;

}

fn()

console.log(m3); //4

console.log(m2); //4

console.log(m1); //n is not defined

var m1=m2=m3=4;

======>等价于 m3=4;//不加var，全局

m2=m3;//不加var，全局

var m1=m2;//局部。

变量声明提前：变量提升，JS程序执行前，会将var声明的变量提升到所在作用域的最前边，只是提升声明，赋值不提升。

//var a;

console.log(a); //undefined

var a = 1;

function fn(n){

//参数属于局部变量。

console.log(n) //2

}

fn(2)

console.log(n) //n is not defined，形参属于局部变量，

var m = 4; //全局变量

function fn(m){

m = m + 3; //m=7,形参属于局部变量

return m; //访问到当前作用域的m，不会访问全局变量

}

fn(7);

//全局作用域下，访问全局的m

console.log(m) //4

函数作用域：

全局函数：在全局作用域下创建的函数，可以在任意作用域。

局部函数：在函数之中创建的函数，只能在当前函数作用域下调用。

函数提升：JS程序执行前，会将function关键字创建的函数提升到所在作用域的最前边，可以先调用在创建，

递归：

在函数内调用自身这个函数，本身是一个死循环。

使用递归：要有结束条件，结合return。