1. 函数

isNaN(数据) / parseInt / parseFloat / Number / prompt 等

函数分为：系统函数 和 自定义函数 （自己创建）

**function：功能体** 函数（方法），可以接受若干数据，返回处理的结果。 用于封装反复执行的代码。

1. 创建普通函数

function 函数名称( ){

函数体 //要封装的代码}

调用（）

1. 创建带有参数的函数

|  |
| --- |
| function 函数名称（参数列表//用于接收传递的数据）{  函数体 //  }  调用（） |

例：function haha(num1,num2){ //形参

console.log(num1+num2);

}

haha(3,4); //实参

haha(7,45);

1. 调用

函数名称：实际传递的数据

参数：创建函数时的参数成为形参，调用时参数成为实参。

实参会赋值给形参，多个参数间逗号隔开。参数可有0或多个，如果形参未被赋值则值为undifined 。

例：封装计算1~任何数字间所有整数和，调用多次

function dam (a){

for (var i=1,sum=0;i<=a ;i++ )

{ sum += i;

}

console.log(sum);

}

dam(8);

dam(15);

…

例2：创建函数，封装计算任意两年之间闰年个数，调用多次

function rn(a,b){

for (var i=a,sum =0;i<=b;i++ )

{ if ((i%4===0 && i%100!==0)||i%400===0)

{ sum += 1;

console.log(i,sum);

}

}

}

rn(2000,2100);

1. 创建带有返回值的函数

|  |
| --- |
| function 函数名称（参考数列）{  函数体；  ruturn值； //返回值，函数执行返回结果  } |

return在循环中会停止后续代码执行。如果函数中没有return或者return后没有加任何值，返回undefined。

例：

function add(n1,n2){

console.log(n1+n2);

return n1+n2; //要用return保存数据，a才能获得值，否则a==undefined

}

var a=add(87,93);

※ return只是返回函数执行得到的值，内部声明的函数仍旧无法在外面使用！

例：三个数比大小

function com(n1,n2,n3){

/\* if (n1>=n2)

{ return n1;

}else {return n2;}（可考虑if-else）\*/

return (n1>n2? n1:n2)>n3?(n1>n2? n1:n2):n3;

}

var A=com(87,58,70);

console.log(A);

对比break和return用法

break用于结束switch语句，还可以用于提前结束循环。

return用于函数中返回值，可以结束函数后续代码执行。

例：

function judge(y){

if ((y%4===0 && y%100!==0)||y%400===0)

{return true; //后面省略了else，因为return阻止了后面代码执行

}

return false;

}

// console.log(judge(2020));

function ri(a){

if (judge(a))

{ return 366;

}return 365;

}

console.log(ri(2019));

例：1~任意数字的阶乘的和

function jc(n){

for(var i=1,num=1;i<=n;i++){

num\*=i;

}

return num;

}

/\*jc(5); //1~5 的乘积

console.log(jc(5)); //因为是内部的函数这里只能打印jc（5）而不是num\*/

function qh(n){ //创建1~任何数的和，再将jc函数带入任何数

for(var i=1,sum=0;i<=n;i++)

{ sum += jc(i);}

return sum;

}

qh(10); //输出数10

console.log(qh(10));

1. 作用域

变量或者函数的可访问范围

全局作用域：在全局使用var声明的变量，可以在任意范围内访问到

局域作用域：在函数使用var声明的变量，只能在函数内部访问到

（函数内不用var声明的变量默认全局变量，但不推荐，后期会报错）

var a=1;// 全局作用域下的变量——全局变量

function fn(){

var b=2; //函数作用域下使用var的变量——局部变量

console.log(a);}

fn(); //调用函数

console.log(a);

1. 变量提升

JS程序在执行前，会将使用var关键字声明的变量提升到所在作用域的最前，但赋值还是在原来的位置。

1. 函数提升

同变量提升，JS执行前会将function关键字提升到所在作用域最前，在调用时才会执行其中代码。

1. 函数作用域

在全局作用域下创建的函数可在任意位置访问；在函数作用域下创建的函数只能在其函数内部访问。