函数

学习目标：

函数

变量的作用域

函数的作用域

递归(掌握)

JS函数 自定义函数

分为两种

一种是js提供的，例如typeof parseInt,parseFloat,alert...；这些事可以直接使用的。

另一种是自定义函数(function)，也称为方法(method)；

自定义函数是一段已经预定义好的代码块，并且可以反复执行。

1. 定义函数和使用函数
2. 普通函数

声明:function 函数名(){

函数体;

}

调用:函数名();

1. 带参数的函数

声明:function 函数名(参数列表){ //形参

函数体;

}

调用:函数名(参数列表); //实参

例:function say(name,age){

console.log(‘我叫’+name+’,今年’+age+’岁’);

}

say(‘jerry’,18);

1. 带返回值的函数

声明:function 函数名(参数列表){

函数体;

return值;

}

调用:函数名(参数列表);

例: function calc(a,b){

return a+b; //函数里return后面的语句不能执行

console.log(‘该命令不会执行’);

}

var res1=calc(16,18);

var res2=calc(13,6);

console.log(res1+res2); //53

说明:函数值没有添加return,返回结果是undefined;

函数里有return,如果return后面的值为空,返回结果是undefined;

Return后的所有语句不能被执行,return就是函数结束

例1：计算前n个数的乘积

function cj(n){

var s=1;

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

s\*=i;

}

return s;

}

var r=cj(5);

console.log(r);

例2：计算是否是闰年

function isLeap(y){

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

return true;

}

return false;

}

var s=isLeap(2000);

if(s){

console.log(366);

}else{

console.log(365);

}

2.变量的作用域

变量的作用域指变量的可访问范围。

分为全局作用域和局部(函数)作用域

全局作用域下的变量可以在任何位置访问到

局部作用域下的变量只能在局部(函数内部)访问到

**注意：**函数内部声明的变量没有使用var关键字，声明的变量就成为全局变量。

**变量提升**:JS程序在执行前，声明的变量都会加载到程序的最前面，但只提升声明，赋值还是在原来的位置。函数内部也存在变量的提升

例2：console.log(a); //undefined

var a=1;

console.log(a); // 1

例3：var n=10;

function f(n){

n=n+10;

}

f(n);

console.log(n) //10

3.函数的作用域

全局函数、局部函数

局部作用域下声明的函数，只能在函数内部访问；

全局作用域下声明的函数，可以在任何位置访问。

**函数提升**：在程序执行前，声明的函数会提升到程序的最前面

例：f();

function f(n){

console.log(123); //123

}

**4.递归**

在函数的内部调用自身

例1：使用递归计算阶乘

function jc(n){

if(n==1){

return;

}

return n\*jc(n-1);

}

jc(5);

例2：使用递归计算斐波那契数列的第n项值

function f(n){

if(n==1 || n==2){

return 1;

}

return f(n-1)+f(n-2);

}

console.log(f(12));

**函数——0813**

arguments.callee 函数内部访问函数名

如果形参的数量大于实参的数量，未赋值的形参结果未undefined;

如果实参的数量大于实参的数量，结果没有任何变化

**匿名函数**

1.声明与调用

例：var 变量名称=function (形参列表){ //把一个匿名变量发到变量里

函数体;

}

变量名称(实参列表); //调用函数

使用匿名函数的方式不存在函数提升，因为函数名称是使用变量表示的，只存在变量提升。

2.回调函数

匿名函数以实参形式传递

例1：匿名函数的自调用 ——创建一个局部作用域，防止全局污染

(function(){

console.log('hello');

})();

例2：

function f(i,j){

return i()+j();

}

var s=f(

function(){return 1;}, //匿名函数以实参形式传递

function(){return 2;}

)

console.log(s);

**全局函数**

1.对一个URI进行编码(中文)—encodeURI()\decodeURI()

例：var uri='http://www.codeboy.com/?kw=戴尔&addr=中国'；

var res1=encodeURI(uri); //编码

var res2=decodeURI(res1); //解码

2.检测是否为NaN—isNaN() 返回true 或 false

3.检查是否为有限值—isFinite() 返回true 或 false

1/0 -> infinity(无限值)

4.转换为整型—parseInt()

5.转换为浮点型—parseFloat()

6.执行**字符串表达式**的值—eval()

例1：var a=1,b=3,c=eval(‘a+b’); //c=4

例2：var p=eval(prompt('')); //2+3\*5/6\*8-13

alert(p); //9