**四、javascript函数**

**一、认识函数**

**1.函数概念**

函数就是把完成特定功能的一段代码[抽象出来]，使之成为程序中的一个[独立实体]，起个名字（函数名）。可以在同一个程序或其他程序中多次重复使用（通过函数名调用）。

【格式】

function 函数名(){

函数体/完成特定功能的一段代码

}

//【注】函数的声明,给计算机写说明书

function print(){

for(var i = 0; i < 10; i++){

document.write("world Hello!<br />");

}

}

//【注】函数的调用(去执行说明书)，函数中的代码必须调用以后才能执行

print();

**2.分类**

(1)内置函数/系统函数

alert();

document.write();

parseInt();

Math.pow();

(2)用户自定义函数

(3)构造函数

**3.有参函数**

【注】封装一个函数，功能，计算两个数之和。

未知数：两个数。

【注】形参(形式参数)的方式去设置未知数

function add(num1, num2){ //num1 = 3, num2 = 5;

alert(num1 + num2);

}

//【注】调用函数的时候，需要传参

//【注】传参，使用实参(实际参数)给形参赋值

add(3, 5);

【注】标识符的命名规则。

(1)函数名命名规则符合标识符的规范

(2)函数名一定要体现当前函数的功能

(3)必须使用全拼

(4)形参也必须符合数据的类型，和数据的信息

【例】1、编写一个函数，计算两个数字的和、差、积、商

要求：使用传参的形式

function operation(num1, operator, num2){

switch(operator){

case "+":

alert(num1 + num2);

break;

case "-":

alert(num1 - num2);

break;

case "\*":

alert(num1 \* num2);

break;

case "/":

alert(num1 / num2);

break;

default:

alert("error");

break;

}

}

operation(10, "-", 5);

**4.数组arguments**

//【注】在每一个函数内部，都有一个内置的数组，这个数组叫做arguments。

//【作用】所有传入参数，都会按照传参的顺序，存入到数组arguments中

【注】arguments.length 数组，里面所存值的个数

【注】访问其中的每一个元素。是通过下标访问

arguments[下标] 【注】下标是从0开始

【例】求任意数的和。参数个数不确定。

function sum(){

var sum = 0;

for(var i = 0; i < arguments.length; i++){

sum += arguments[i];

}

alert(sum);

}

sum(10, 20, 30, 40, 50);

**5.函数返回值**

(1)如果没有设置返回值，函数的返回值默认是 undefined

【注】如果，我们想要将函数内部的运算结果，在外部拿到。需要设置返回值。通过return设置返回值

(2)表达式：1、功能2、值

函数可以看做是表达式：

1、函数的功能

2、函数的值 == 函数中return后面所跟表达式的值

(3)【注】使用return的时候，需要注意

每一个函数，只能有一个return生效，并且函数执行到return以后，就终止执行。return后面所写的代码是不去执行的。

function show(){

alert("return前");

return 0;

alert("return后");

}

show(); //return后是不执行的

**6.函数作用域**

(1)【注】凡是函数，都有大括号，每一个函数大括号就相当于是一个作用域。

(2)【注】在作用域内访问变量的时候，采取就近原则。

(3)【注】一个函数的作用域，代表这个函数的生命周期，当这个函数执行完成以后，在该函数作用域内，声明的变量和形参，就直接被销毁了。

① var a = 10;

function show(){

var a = 5;

alert(a);

}

alert(a); //10

show(); //5

alert(a); //10

//====================================

② var a = 10;

function show(){

a = 5;

alert(a);

}

alert(a); //10

show(); //5

alert(a); //5

(4)全局作用域：不包含在任何一个函数内，直接声明在全局的变量

局部作用域：声明的函数内部

全局变量

局部变量

【封装函数原则】尽量保证当前函数功能的单一性。

function show(){

var a = 10;

alert(a);

}

show();

alert(a); //ReferenceError: a is not defined

**7.事件驱动函数**

事件驱动函数

document.getElementById(id);

通过id找到符合条件的标签

【例1】

<script>

window.onload = function(){ //这个函数，在页面加载完成以后执行

//1、找到这个节点

var oBtn = document.getElementById("btn");

//2、绑定事件

oBtn.onclick = function(){

alert(1);

}

var oDiv = document.getElementById("div1");

oDiv.onmouseover = function(){

oDiv.style.backgroundColor = "blue";

}

oDiv.onmouseout = function(){

oDiv.style.backgroundColor = "yellow";

}

}

/\*

**onclick 单击**

**onmouseover 鼠标移入**

**onmouseout 鼠标移除**

**文本框获取焦点和失去焦点时(onfocus、onblur)**

\*/

</script>

</head>

<body>

<input id = "btn" type = "button" value = "按钮" />

<div id = "div1"></div>

</body>

【例2】

<script>

function operation(num1, operator, num2){

switch(operator){

case "+":

return num1 + num2;

break;

case "-":

return num1 - num2;

break;

case "\*":

return num1 \* num2;

break;

case "/":

return num1 / num2;

break;

default:

return "error";

break;

}

}

window.onload = function(){

$("equal").onclick = function(){

var num1 = Number($("num1").value);

var num2 = Number($("num2").value);

var operator = $("operator").value;

alert(operation(num1, operator, num2));

}

}

//获取元素节点的函数

function $(id){

return document.getElementById(id);

}

</script>

</head>

<body>

<input id = "num1" type = "text" placeholder = "请输入数字" />

<input id = "operator" type = "text" placeholder = "请输入运算符" />

<input id = "num2" type = "text" placeholder = "请输入数字" />

<input id = "equal" type = "button" value = "=" />

</body>

**8.函数递归**

(1)递归的概念：

函数自己调用自己叫做递归。

(2)递归函数特点：

①函数自己调用自己

②必须有参数

③必须return

(3)递归的作用：

凡是循环做的事情递归都可以做

循环做不了的事情，递归也可以做

递归特点：

①我写的出来

②但是我看不懂

(5)递归黄金法则：

①先去找临界值，无需计算，就得出的结果

②找这一次和上一次的关系

③假设当前函数可以被调用了，写出第n次和第n-1次的关系。

sum(n) = sum(n - 1) + n;

【例】

function sum(n){

if(n == 1){

return 1;

}

return sum(n - 1) + n;

}

alert(sum(100));

/\*

sum(100) = sum(99) + 100;

sum(99) = sum(98) + 99;

...

sum(3) = sum(2) + 3;

sum(2) = sum(1) + 2;

\*/

/\*

计算1到100的和

\*/

/\* var sum = 0;

for(var i = 0; i < 100; i++){

sum += i + 1;

}

alert(sum);\*/

**9.菲波那切数列**

兔子繁殖问题，设有一对新生兔子，从第四个月开始他们每个月月初都生一对兔子

新生的兔子从第四个月月初开始又每个月生一对兔子

按此规律，并假定兔子没有死亡,n(n <= 20)个月月末共有多少对兔子？

/\*

一月兔 二月兔 三月兔 成熟兔

1 1 0 0 0

2 0 1 0 0

3 0 0 1 0 1

4 1 0 0 1 2

5 1 1 0 1 3

6 1 1 1 1 4

7 2 1 1 2 6

8 3 2 1 3

9 4 3 2 4

10 6 4 3 6 19

11 9 6 4 9 28

\*/

/\*

递归

rabbit(4) = rabbit(4 - 1) + rabbit(4 - 3);

rabbit(n) = rabbit(n - 1) + rabbit(n - 3);

\*/

function rabbit(month){

if(month < 4){

return 1;

}

return rabbit(month - 1) + rabbit(month - 3);

}

alert(rabbit(11));