JS函数/分支结构

2020年4月6日 14:41

函数：（function）

函数(function)，也可以被称之为方法(method),或者过程(procedure)
是一段预定义好，并可以被反复使用的代码块。其中可以包含多条可执行语句
预定义好:事先声明好，但不被执行
反复使用：允许被多个地方(元素，函数中)所应用
代码块：允许包含多条可执行的代码
函数本质上是功能完整的对象

- 函数的声明：

语法：

```
function 函数名( ){
        可执行语句;
}
```

ex ：创建一个函数 名称:sayHello,主体功能为，向控制台上 输出一句 Hello World

```
function printHello( ){
        console.log( "hello" );
        console.log( "world" );
```

- 函数的调用：

执行函数中的内容
任何 JS 的合法位置处，都允许调用函数
语法：函数名称();

```
function sayHello( ){
        console.log( "hello" );
        console.log( "world" );
}

sayHello( );
```

- 定义带参数函数：

```
function 函数名(参数列表声明){
    //代码块(函数体，功能体，方法体)
}
```

参数列表：由一或多个 变量名称来组成
声明函数时定义的参数，可以称之为叫作 "形参"(形式参数)
ex:

```
function printInfo(userName, ueserPwd){
    console.log('用户名:'+userName + '密码:'+userPwd);
}

printInfo('Tom' , '123');
```

在调用函数时所传递的参数值，被称之为"实参" (实 际参数)

- 带返回值的函数：

4.2、带返回值的函数

1、声明

```
function 函数名(0或多个参数){
    //代码块;
    return 值;
}
```

2、调用

```
var 变量 = 函数名(参数);
```

```
function add( num1, num2){
        return num1 + num2 ;
}

var result = add( 10, 20 );
```

- 变量的作用域：
  1. 函数作用域：只在当前函数内可访问
  2. 全局作用域：一经定义，代码任何位置都可以访问

- 函数作用域中的变量：（离开当前函数就不可以访问了）

```
function add(){
    var sum = 1 + 2;                    //局部变量
    console.log( sum );                 //正确
}
console.log( sum ) ;                    //脚本错误
```

- 全局作用域中的变量：

全局作用域中的变量，称之为 “全局变量”，
在代码的任何位置处都能访问

```
var sum = 0;                            //全局变量
function add(){
    sum = 1 + 2;
    console.log( sum );                 //正确
}
console.log( sum ) ;                    //正确
```

不推荐使用

```
function add(){
    sum = 1 + 2;                        //全局变量
}
add();                                  //必须调用一次
console.log( sum ) ;                    //正确
```

var

没加var就是全局变量

- 声明提前：JS正式执行前，会将所有var声明的变量和function声明的函数，预读到作用域的顶部

```
console.log(a); //不会出错，输出undefined
var a=100;
console.log(a); //100 ;
```

未附值

```
var a;  //仅声明提前
console.log(a);  //undefined
a=100; //赋值仍保留在原位置
console.log(a);  //100
```

- 按值传递：

传参时，实际上是将 实参 复制了一份副本传给了函数。在 函数体内 对变量进行修改，实际上是不会影响到外部的实参变量的

```
var n=100; //全局变量n
function fun(n){ //参数变量也是局部变量
    n-=3;  //修改的是局部变量n
    console.log(n); //输出的是局部变量n
}
fun(n);  //按值传递，方法内输出97;
console.log(n);  //输出全局变量的值：100
```

函数声明时的参数叫形参
函数调用的参数叫实参

```
        return num1 + num2 ;
}

var result = add( 10, 20 );
console.log( result );          //输出30
```

函数声明时的参数叫形参

函数调用的参数叫实参

程序的三种结构：

- 顺序结构
- 分支结构
- 循环结构

分支结构：

- if-结构：

```
if(条件表达式){
    语句块;
}
```

注意

1、if语句，条件位置处，必须为 boolean的值/表达式/变量

如果条件不是boolean 类型的话，JS会自动进行转换

以下情况，if 都会认为是 false

if(0/0.0/""/null/undefined/NaN){}

除以上情况外，一律为真

```
if(1){
        console.log("真!");
}
if("我帅吗"){
        console.log("真!!!!");
}
```

2、if语句块的{} ，可以被省略的

如果省略{},那么if只控制它下面的第一条语句

- if-else结构：

2、语法

```
if(条件1){
        语句块1;
}else if(条件2){
        语句块2;
}else if(条件3){
        语句块3;
}...else{
        语句块n;
}
```

注：最后的 else 模块，可以选择性添加

- switch-case结构：

```
switch (表达式) {
    case 值1 :
        语句1 ;
        语句2 ;
    case 值2 :
        语句3 ;
        ... ... ...
    default:
        语句n ;
}
```

```
switch (表达式) {
    case 值1 :
        语句1 ;
        语句2 ;
        break;
    case 值2 :
        语句3 ;
        break;
    default:
        语句n ;
}
```

break语句的作用在于跳出switch结构

商品优惠实验：alert（弹出内容）

程序卸载body中的script中

```
// 1、定义输入变量：单价、数量、收款金额
var price,count,money;
// 2、输出变量：产品总价、找零
var total,change;

// 3、为输入变量赋值
price = prompt("请输入商品单价");
count = prompt("请输入商品数量");
// 4、根据单价和数量计算出商品总价，输出产品总价
total = price*count;
// alert(total);
// 5、判断，判断商品总价>=500,享受八折优惠
if(total>=500){
    alert("您的消费金额已经大于等于500元，可以享受8折优惠");
    total = total*0.8;
}
alert("您此次的消费金额是:"+total+"元");
// 6、判断应收金额 total 和收款金额money之间的关系
money = prompt("收款:");
if(money >= total){
    // 计算找零
    change = money - total;
    alert("共找您"+change+"元");
}else{
    alert("您给的钱不够，请多给点");
}
```

结束机制：

switch-case 的结束机制：

1、碰到 break 结束

2、整个结构都执行完毕 结束直落

两个case 或 多个 case之间，没有任何的可执行代码，那么就以最后一块的case为主