logo.png

javascript基础三

H5 web前端

[www.bufanui.com](http://www.bufanui.com)

# 一、Switch语句

☞语法

Switch(变量n){

case 10:

执行的代码;

break;

case 20:

执行的代码;

break;

default:

执行的代码;

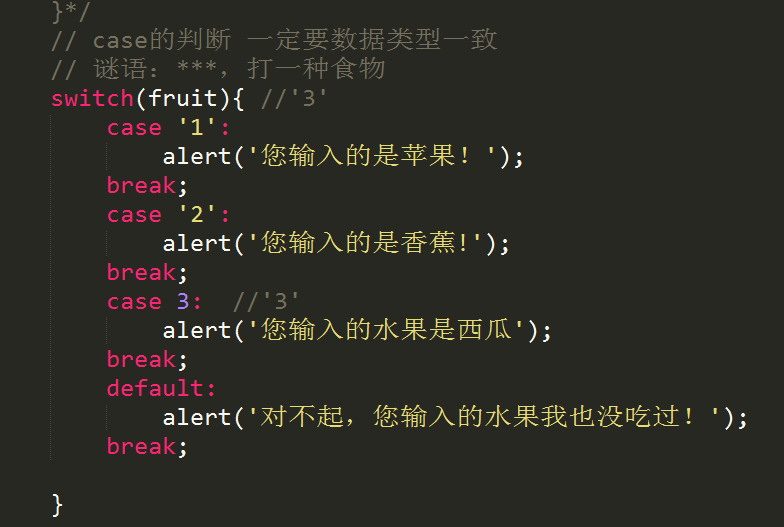
break;

}

注意：

**Switch语句后面的变量数据类型必须和case后面的数据类型保持一致。**

适合比较少的种类判断，而且我们知道多少种。



**Switch语句可以对变量进行集体判断**：



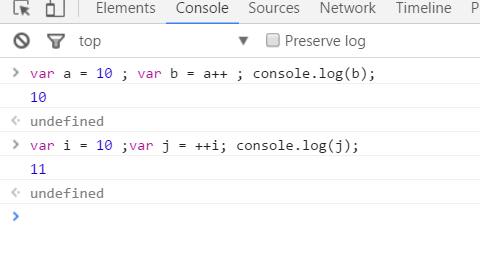
# 二、自增自减

☞ i++ ++i

◆如果变量没有直接参与运行中，i++ 和 ++i表示的含义是：在变量原来值的基础上加1。

i++ i先参与运算，再执行++ i+=6

++i ++执行，i再参与运算



◆如果i++或者++i参与到运算中。

var i=123;

var n1=i++; 该段代码的含义是： 先将i的值赋值给变量n1,然后变量i的值再加1。

var n2=++i; 该段代码的含义是： 先将i的值加1,然后变量i加1后的值赋值给变量n2。

# 三、While循环

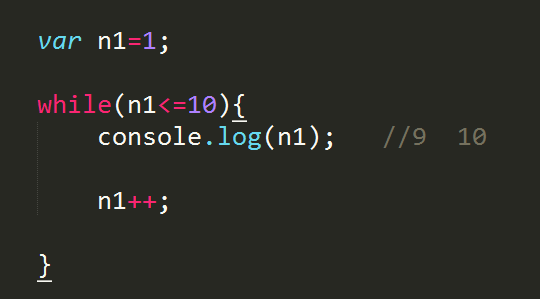
☞ 语法

while(条件表达式){

当条件表达式结果为true,会一直执行while循环体内的代 码。

当条件表达式的结果为false，while循环不再执行。

}



注意：

While循环的过程中，首先在while循环外部定义一个变量，然后判断条件。

While 循环一定要有跳出条件

# 四、Do..while循环

☞语法

Do{

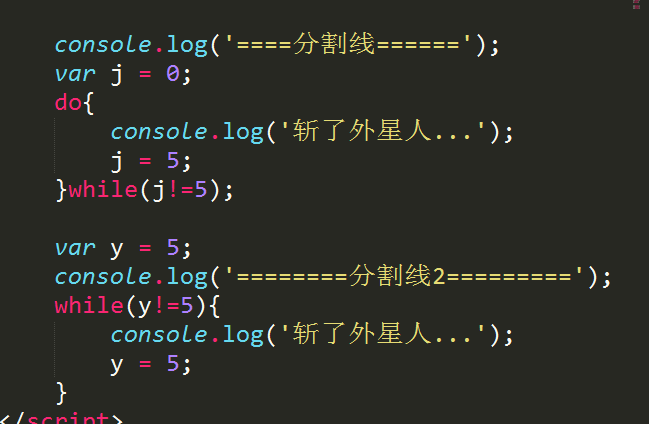
循环体;

}while(条件表达式);

Do while 循环执行顺序：

程序运行开始，直接进入循环体内，执行完第一遍循环体内代码后，进行while条件判断，如果while条件结果为true,那么继续执行do中的循环体代码。如果while条件结果为false,那么do 中的循环体代码不再执行（跳出循环）

◆do while循环在条件不满足的情况下会比while循环多执行一次代码



# 五、For循环

☞用法

for(var i=0; i<=10; i++){  
 循环体代码

}

执行顺序：

◆首先进行变量初始化，并进行条件判断

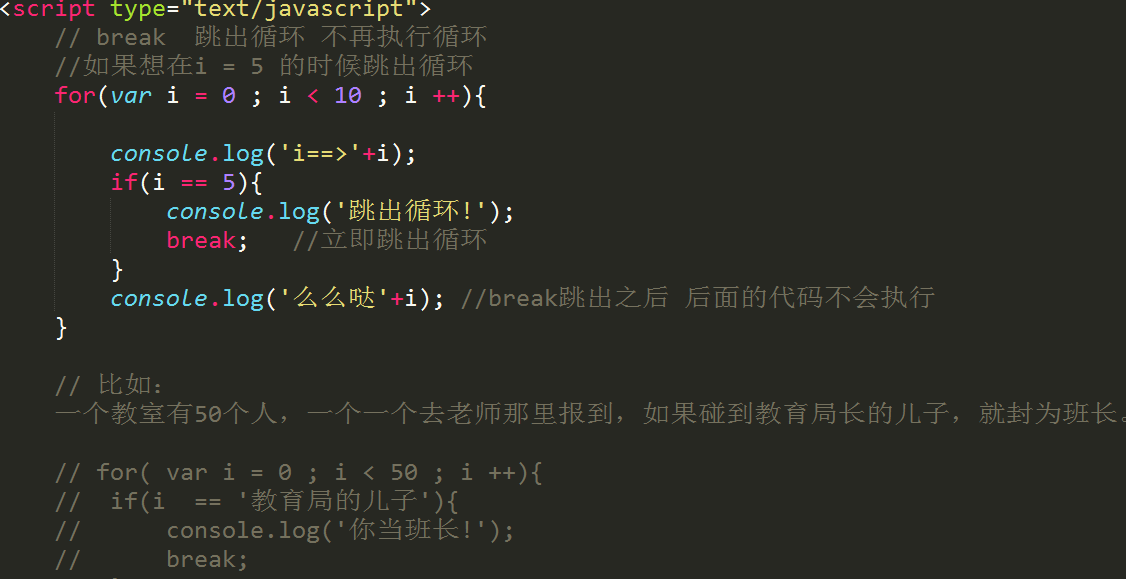
◆如果条件结果为true，那么执行循环体内的代码，然后执行 i++;

◆判断条件是否为true,继续执行循环体内代码。否则跳出循环

# 六、Break语句

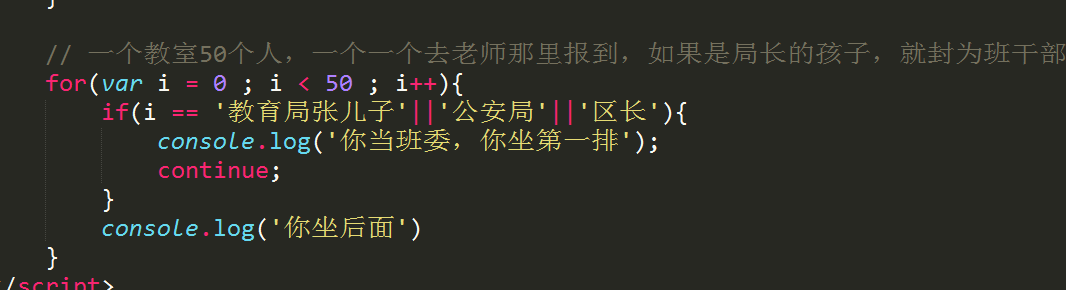
在循环体内，只要代码遇到break,程序立马结束当前循环。

当前循环指的是break语句所在的循环体。



# 七、Continue语句

Continue语句指的是跳出本次循环，该语句后面的代码不再执行。

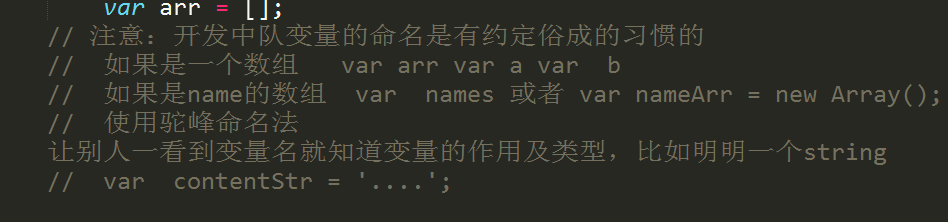


# 八、数组（定义，赋值，取值）

## 8.1 定义

var arr = new Array(); //通过对象方式创建数组

var arr = [] //直接创建数组



## 8.2 赋值

☞ 数组中通过下标的方式进行赋值。**下标从0开始**。

arr[0] = 123;

## 8.3 属性介绍

获取数组中的元素个数

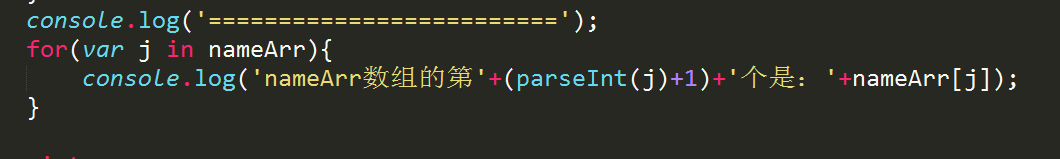
通过数组名.length获取数组长度（元素个数）

## 8.4 数组遍历

两种方法遍历：

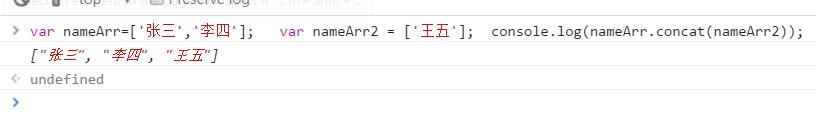


另一种遍历方式：



## 8.5 concat()数组合并

concat()



## 8.6 join() 把数组转换为字符串

join()

作用是将数组各个元素是通过指定的分隔符进行连接成为一个字符串。

语法

arrayObject.join(separator)

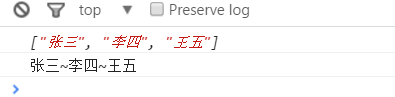
数组名.join(符号)

数组转换为字符串

参数 separator 可选。指定要使用的分隔符。如果省略该参数，则使用逗号作为分隔符。

var arr = [1,2,3];

console.log(arr.join(“-”)) 结果就是： 1-2-3 字符串



## 8.7 把字符串转换为数组 split()

join <=> split

split() 方法用于把一个字符串分割成字符串数组

语法

stringObject.split(separator,howmany)

参数 separator 可选。指定要使用的分隔符。如果省略该参数，则使用逗号作为分隔符。

howmany 可选。该参数可指定返回的数组的最大长度



## 8.8 添加数组

任何一个对象的操作无非是 增删改查

任何一个系统 也是 增删改查

每一行代码 都跑不出增删改查

var arr = [1, 3, 5];

我们想要把7这个数字放到这个数组的后面

变成[1, 3, 5, 7];

1. push() \*\*\*\*\* 从后面推进去

push方法可以向数组的末尾添加一个或者多个元素，并返回新的长度

1. unshift \*\*\*\*\* 从数组的前面放入

unshift 方法向数组的开头添加一个或者多个元素，并返回新的长度



## 8.9删除数组元素

1. pop 删除最后一个元素

pop移除最后一个元素，并且返回最后一个删除的值

1. shift 删除第一个元素

Shift移除第一个元素，并且返回第一个删除的值

## 8.10冒泡算法和选择排序

冒泡排序算法的运作如下：（从后往前）

1. 比较相邻的元素。如果第一个比第二个大，就交换他们两个。
2. 对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。
3. 针对所有的元素重复以上的步骤，除了最后一个。
4. 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

|  |
| --- |
| <title>冒泡</title>  <script type="text/javascript">  var a = [55,33,12,31,9];  console.log('ol==>'+a);    for(var i = 0 ; i < a.length -1 ; i ++){  for(var j = 0 ; j < a.length-i-1 ; j ++){  if(a[j]>a[j+1]){  var temp = a[j];  a[j] = a[j+1];  a[j+1] = temp;  }  }  console.log(i+'==>'+a)    }  console.log(a)  </script> |

选择排序（Selection sort）是一种简单直观的[排序算法](http://baike.baidu.com/item/%E6%8E%92%E5%BA%8F%E7%AE%97%E6%B3%95" \t "_blank)。它的工作原理是每一次从待排序的[数据元素](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%85%83%E7%B4%A0" \t "_blank)中选出最小（或最大）的一个元素，存放在序列的起始位置，直到全部待排序的数据元素排完。

|  |
| --- |
| var minIndex = 0 ;  for(var j = 0 ; j < a.length ; j ++){  minIndex = j;  for(var i = j+1 ; i < a.length ; i ++){  if(a[i]<a[minIndex]){  minIndex = i;  }  }  if(j != minIndex){  var temp = a[j];  a[j] = a[minIndex];  a[minIndex] = temp;  }  console.log(i+'==>'+a);    } |

## 8.11~~案例 隔行变色 dom~~

## 8.12求平均值

## [33,21,1,40,12,5]

# 函数function（函数的定义和调用）

## 9.1 函数声明方式

### 9.1.1 自定义函数声明

function a(){  
 alert("我是自定义函数")  
}  
a(); // 函数不调用，自己不执行

总结：

　　1：此时a函数是变量的结果，默认存储在全局上下文的变量中(可用 window.函数名 来验证)

~~2：此方式为函数声明，在进入全局上下文阶段创建。~~

　　3：它可以影响变量对象（仅影响存储在上下文中的变量）

### 9.1.2 匿名函数声明 -- 函数表达式

var a = function(){  
 alert("直接量声明")  
}  
a(); 也需要调用

总结：

　　1：首先变量本身不做为一个函数存在，而是一个匿名函数的引用（值类型的不属于引用）

　　 2：在代码执行阶段，初始化全局上下文时，它没有被做为全局的一个属性而存在，所以不会造成变量对象的污染

　　3：该类型的声明一般在插件的开发比较常见，也可做为闭包中回调函数的调用

### 9.1.3~~利用Function 关键字声明~~

var fun2 = new **Function**("var a = 10; var b = 20; alert(a+b)");  
fun2();

## 9.2 初步认识函数

|  |
| --- |
| 1. <script type="text/javascript"> 2. console.log("你好"); 3. **sayHello();** //调用函数 4. //定义函数： 5. function sayHello(){ 6. console.log("欢迎"); 7. console.log("welcome"); 8. } 9. </script> 10. </body> |



**函数，是一种封装。就是将一些语句，封装到函数里面。通过调用的形式，执行这些语句。**

函数的使用，是两个步骤，**第一步，函数的定义**：

语法：

|  |
| --- |
| 1. function 函数名字(){ 3. } |

function就是英语“函数”、“功能”的意思。顾名思义，将一些功能封装到函数里面。

function是一个关键字，和var、typeof一样，都是关键字，后面要加空格。

函数名字的命名规定，和变量的命名规定一样。只能是字母、数字、下划线、美元符号，不能以数字开头。

后面有一对儿空的小括号，里面是放参数用的，下午介绍。大括号里面，是这个函数的语句。

常见错误：

不能小括号包裹大括号：

|  |
| --- |
| 1. function sayHello({ 3. }) |

不能忘了小括号对儿：

|  |
| --- |
| 1. function sayHello{ 2. } |

**第二步，函数的调用。**

**函数如果不调用，等于白写。**

调用一个函数，太简单了，就是这个函数的名字后面加小括号对儿。

语法：

|  |
| --- |
| 1. 函数名字(); |

定义函数，可以在调用的后面：这是JS的语法特性，函数声明头的提升。知道就行了。

|  |
| --- |
| 1. console.log("你好"); 2. sayHello(); //调用函数 3. //定义函数： 4. function sayHello(){ 5. console.log("欢迎"); 6. console.log("welcome"); 7. } |

函数的功能、好处：

1） 将会被大量重复的语句写在函数里面，这样以后需要这些语句的时候，直接调用函数，不用重写那些语句。

2） 简化编程，让编程变的模块化。

## 9.3 函数的参数

实际参数和形式参数的个数，要相同。



|  |
| --- |
|  |

\*javascritp中不支持函数的重载

\*arguments -- 得到实参的个数

## 9.4 函数的返回值

|  |
| --- |
|  |

return就是英语“返回”的意思，

● 函数里面可以没有return，如果有，只能有一个。不能有多个return；

● 函数里面，return后面不允许书写程序了，也就是说写在后面的程序无效；

## 9.5 变量声明提升 -- Hoisting(涉及名词:scoping)

什么是变量提升

1. function fun(){
2. console.log(num);
3. var num = 20;
4. }

相当于 ---

1. function fun(){
2. var num;
3. console.log(num);
4. Num = 20;
5. }

在函数体内部，声明变量，会把该声明提升到函数体的最顶端。 只提升变量声明，不赋值。

var a = 18;  
f1();  
function f1(){  
 var b=9;  
 **console**.log(a);  
 **console**.log(b);  
 var a = '123';  
}

结果是： undefined 9

# 十、变量的作用域

根据变量的作用范围 可以分为 全局变量 和 局部变量

全局变量：

1. 在最外层声明的变量。

2. 在函数体内部，但是没有声明var 的变量也是全局变量

局部变量：

在函数体内部的 声明的变量

小知识点：

隐式的全局变量

1. 在函数体内部，但是没有声明var 的变量也是全局变量。
2. var a = 1
3. function func() {
4. a = b = 2
5. }
6. func()
7. alert(a)
8. alert(b)