|  |
| --- |
| [公司名称] |
| Javascript笔记 |
| [文档副标题] |

|  |
| --- |
| 15560903@qq.com  [日期] |

# Javascript笔记

## JavaScript 显示数据

JavaScript 可以通过不同的方式来输出数据：

* 使用 **window.alert()** 弹出警告框。
* 使用 **document.write()** 方法将内容写到 HTML 文档中。
* 使用 **innerHTML** 写入到 HTML 元素。
* 使用 **console.log()** 写入到浏览器的控制台。

创建对象方法：

*methodName : function() { code lines }*

访问对象方法：

*objectName.methodName()*

**调用带参数的函数**

在调用函数时，您可以向其传递值，这些值被称为参数。

这些参数可以在函数中使用。

您可以发送任意多的参数，由逗号 (,) 分隔：

myFunction(*argument1,argument2*)

当您声明函数时，请把参数作为变量来声明：

function myFunction(*var1*,*var2*)  
{  
*代码*  
}

**带有返回值的函数**

有时，我们会希望函数将值返回调用它的地方。

通过使用 return 语句就可以实现。

在使用 return 语句时，函数会停止执行，并返回指定的值。

**语法**

function myFunction()  
{  
    var x=5;  
    return x;  
}

var myVar=myFunction();

document.getElementById("demo").innerHTML=myFunction();

**全局 JavaScript 变量**

在函数外声明的变量是*全局*变量，网页上的所有脚本和函数都能访问它。

**JavaScript 变量的生存期**

JavaScript 变量的生命期从它们被声明的时间开始。

局部变量会在函数运行以后被删除。

全局变量会在页面关闭后被删除

**常见的HTML事件**

下面是一些常见的HTML事件的列表:

|  |  |
| --- | --- |
| **事件** | **描述** |
| onchange | HTML 元素改变 |
| onclick | 用户点击 HTML 元素 |
| onmouseover | 用户在一个HTML元素上移动鼠标 |
| onmouseout | 用户从一个HTML元素上移开鼠标 |
| onkeydown | 用户按下键盘按键 |
| onload | 浏览器已完成页面的加载 |

## 字符串属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| constructor | 返回创建字符串属性的函数 |
| length | 返回字符串的长度 |
| prototype | 允许您向对象添加属性和方法 |

## 字符串方法

更多方法实例可以参见：[JavaScript String 对象](http://www.runoob.com/jsref/jsref-obj-string.html" \t "_blank)。

|  |  |
| --- | --- |
| **方法** | **描述** |
| charAt() | 返回指定索引位置的字符 |
| charCodeAt() | 返回指定索引位置字符的 Unicode 值 |
| concat() | 连接两个或多个字符串，返回连接后的字符串 |
| fromCharCode() | 将 Unicode 转换为字符串 |
| indexOf() | 返回字符串中检索指定字符第一次出现的位置 |
| lastIndexOf() | 返回字符串中检索指定字符最后一次出现的位置 |
| localeCompare() | 用本地特定的顺序来比较两个字符串 |
| match() | 找到一个或多个正则表达式的匹配 |
| replace() | 替换与正则表达式匹配的子串 |
| search() | 检索与正则表达式相匹配的值 |
| slice() | 提取字符串的片断，并在新的字符串中返回被提取的部分 |
| split() | 把字符串分割为子字符串数组 |
| substr() | 从起始索引号提取字符串中指定数目的字符 |
| substring() | 提取字符串中两个指定的索引号之间的字符 |
| toLocaleLowerCase() | 根据主机的语言环境把字符串转换为小写，只有几种语言（如土耳其语）具有地方特有的大小写映射 |
| toLocaleUpperCase() | 根据主机的语言环境把字符串转换为大写，只有几种语言（如土耳其语）具有地方特有的大小写映射 |
| toLowerCase() | 把字符串转换为小写 |
| toString() | 返回字符串对象值 |
| toUpperCase() | 把字符串转换为大写 |
| trim() | 移除字符串首尾空白 |
| valueOf() | 返回某个字符串对象的原始值 |

## JavaScript 算术运算符

<="" p=""> <p给定 **y=5，下面的表格解释了这些算术运算符：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **运算符** | **描述** | **例子** | **x 运算结果** | **y 运算结果** | **在线实例** |
| + | 加法 | x=y+2 | 7 | 5 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_add) |
| - | 减法 | x=y-2 | 3 | 5 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_sub) |
| \* | 乘法 | x=y\*2 | 10 | 5 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_mult) |
| / | 除法 | x=y/2 | 2.5 | 5 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_div) |
| % | 取模（余数） | x=y%2 | 1 | 5 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_mod) |
| ++ | 自增 | x=++y | 6 | 6 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_incr) |
| x=y++ | 5 | 6 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_incr2) |
| -- | 自减 | x=--y | 4 | 4 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_decr) |
| x=y-- | 5 | 4 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_decr2) |
|  |  |  |  |  |  |

## JavaScript 赋值运算符

赋值运算符用于给 JavaScript 变量赋值。

给定 **x=10** 和 **y=5**，下面的表格解释了赋值运算符：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **运算符** | **例子** | **等同于** | **运算结果** | **在线实例** |
| = | x=y |  | x=5 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_equal) |
| += | x+=y | x=x+y | x=15 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_plusequal) |
| -= | x-=y | x=x-y | x=5 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_minequal) |
| \*= | x\*=y | x=x\*y | x=50 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_multequal) |
| /= | x/=y | x=x/y | x=2 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_divequal) |
| %= | x%=y | x=x%y | x=0 | [实例 »](http://www.runoob.com/try/try.php?filename=tryjs_oper_modequal) |

**逻辑运算符**

逻辑运算符用于测定变量或值之间的逻辑。

给定 x=6 以及 y=3，下表解释了逻辑运算符：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **例子** |
| && | and | (x < 10 && y > 1) 为 true |
| || | or | (x==5 || y==5) 为 false |
| ! | not | !(x==y) 为 true |

## 条件运算符

JavaScript 还包含了基于某些条件对变量进行赋值的条件运算符。

### 语法

variablename=(condition)?value1:value2

voteable=(age<18)?"年龄太小":"年龄已达到";

## 条件语句

通常在写代码时，您总是需要为不同的决定来执行不同的动作。您可以在代码中使用条件语句来完成该任务。

在 JavaScript 中，我们可使用以下条件语句：

* **if 语句** - 只有当指定条件为 true 时，使用该语句来执行代码
* **if...else 语句** - 当条件为 true 时执行代码，当条件为 false 时执行其他代码
* **if...else if....else 语句**- 使用该语句来选择多个代码块之一来执行
* **switch 语句** - 使用该语句来选择多个代码块之一来执行

## If 语句

只有当指定条件为 true 时，该语句才会执行代码。

### 语法

if (*condition*)  
{  
*当条件为 true 时执行的代码*  
}

## If...else 语句

请使用 if....else 语句在条件为 true 时执行代码，在条件为 false 时执行其他代码。

### 语法

if (*condition*)  
{  
*当条件为 true 时执行的代码*  
}  
else  
{  
*当条件不为 true 时执行的代码*  
}

## If...else if...else 语句

使用 if....else if...else 语句来选择多个代码块之一来执行。

### 语法

if (*condition1*)  
{  
*当条件 1 为 true 时执行的代码*  
}  
else if (*condition2*)  
{  
*当条件 2 为 true 时执行的代码*  
}  
else  
{  
*当条件 1 和 条件 2 都不为 true 时执行的代码*  
}

# JavaScript switch 语句

switch 语句用于基于不同的条件来执行不同的动作。

## JavaScript switch 语句

请使用 switch 语句来选择要执行的多个代码块之一。

## 语法

switch(n) { case 1: 执行代码块 1 break; case 2: 执行代码块 2 break; default: 与 case 1 和 case 2 不同时执行的代码 }

switch 语句的使用，当两种情况相同时，可以只在第二种情况中写要执行的代码，案例如下：

switch(l){

case1:

case2:alert(“123”)

# JavaScript for 循环

循环可以将代码块执行指定的次数。

<script>

cars=["BMW","Volvo","Saab","Ford"];

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

document.write(cars[i] + "<br>");

}

</script>

**不同类型的循环**

JavaScript 支持不同类型的循环：

* **for** - 循环代码块一定的次数
* **for/in** - 循环遍历对象的属性
* **while** - 当指定的条件为 true 时循环指定的代码块
* **do/while** - 同样当指定的条件为 true 时循环指定的代码块

## For 循环

* for 循环是您在希望创建循环时常会用到的工具。
* 下面是 for 循环的语法：
* for (*语句 1*; *语句 2*; *语句 3*)  
  {  
  *被执行的代码块*  
  }
* **语句 1** （代码块）开始前执行
* **语句 2** 定义运行循环（代码块）的条件
* **语句 3** 在循环（代码块）已被执行之后执行
* Statement 1 在循环开始之前设置变量 (var i=0)。
* Statement 2 定义循环运行的条件（i 必须小于 5）。
* Statement 3 在每次代码块已被执行后增加一个值 (i++)。

## 语句 1

通常我们会使用语句 1 初始化循环中所用的变量 (var i=0)。

语句 1 是可选的，也就是说不使用语句 1 也可以。

您可以在语句 1 中初始化任意（或者多个）值

## 语句 2

通常语句 2 用于评估初始变量的条件。

语句 2 同样是可选的。

如果语句 2 返回 true，则循环再次开始，如果返回 false，则循环将结束。

## 语句 3

通常语句 3 会增加初始变量的值。

语句 3 也是可选的。

语句 3 有多种用法。增量可以是负数 (i--)，或者更大 (i=i+15)。

语句 3 也可以省略（比如当循环内部有相应的代码时）：

## For/In 循环

JavaScript for/in 语句循环遍历对象的属性：

## 实例

var person={fname:"John",lname:"Doe",age:25}; for (x in person) // x 为属性名 { txt=txt + person[x]; }

**JavaScript while 循环**

只要指定条件为 true，循环就可以一直执行代码块。

**while 循环**

while 循环会在指定条件为真时循环执行代码块。

**语法**

while (*条件*)  
{  
*需要执行的代码*  
}

## do/while 循环

do/while 循环是 while 循环的变体。该循环会在检查条件是否为真之前执行一次代码块，然后如果条件为真的话，就会重复这个循环。

### 语法

do  
{  
*需要执行的代码*}  
while (条件);

别忘记增加条件中所用变量的值，否则循环永远不会结束！

# JavaScript Break 和 Continue 语句

break 语句用于跳出循环。

continue 用于跳过循环中的一个迭代。

## JavaScript 标签

正如您在 switch 语句那一章中看到的，可以对 JavaScript 语句进行标记。

如需标记 JavaScript 语句，请在语句之前加上冒号：

label: statements

break 和 continue 语句仅仅是能够跳出代码块的语句。

语法:

break labelname; continue labelname;

continue 语句（带有或不带标签引用）只能用在循环中。

break 语句（不带标签引用），只能用在循环或 switch 中。

通过标签引用，break 语句可用于跳出任何 JavaScript 代码块

break 的作用是跳出代码块, 所以 break 可以使用与循环和 switch 等

continue 的作用是进入下一个迭代, 所以 continue 只能用于循环的代码块。

代码块: 基本上是｛｝大括号之间

## typeof 操作符

你可以使用 typeof 操作符来检测变量的数据类型。

## null

在 JavaScript 中 null 表示 "什么都没有"。

null是一个只有一个值的特殊类型。表示一个空对象引用。

## undefined

在 JavaScript 中, **undefined** 是一个没有设置值的变量。

**typeof** 一个没有值的变量会返回 **undefined**。

任何变量都可以通过设置值为 **undefined** 来清空。 类型为 **undefined**.

## JavaScript 数据类型

在 JavaScript 中有 5 种不同的数据类型：

* string
* number
* boolean
* object
* function

3 种对象类型：

* Object
* Date
* Array

2 个不包含任何值的数据类型：

* null
* undefined

请注意：

* NaN 的数据类型是 number
* 数组(Array)的数据类型是 object
* 日期(Date)的数据类型为 object
* null 的数据类型是 object
* 未定义变量的数据类型为 undefined

## constructor 属性

**constructor** 属性返回所有 JavaScript 变量的构造函数。

你可以使用 constructor 属性来查看对象是否为数组 (包含字符串 "Array"):

你可以使用 constructor 属性来查看对象是否为日期 (包含字符串 "Date"):

<p> constructor 属性返回变量或对象的构造函数。</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML =

"john".constructor + "<br>" +

(3.14).constructor + "<br>" +

false.constructor + "<br>" +

[1,2,3,4].constructor + "<br>" +

{name:'john', age:34}.constructor + "<br>" +

new Date().constructor + "<br>" +

function () {}.constructor;

</script>

# IsArray 函数

返回一个**Boolean**值，该值变量数组。

**语法**

**IsArray** **(** ***varname* )**

需要的***varname***参数是指定一个变量的标识符。

**备注**

**IsArray**返回**True**变量是数组; 如果否则，返回**False**。包含数组的变体**IsArray**尤其有用。