# javascript 简介

## javascript介绍

JavaScript 是互联网上最流行的脚本语言，这门语言可用于 HTML 和 web，更可广泛用于服务器、PC、笔记本电脑、平板电脑和智能手机等设备。

javascript因为兼容于ECMA标准，因此也称为ECMAScript。JavaScript作为一种脚本语言，已经被广泛地应用于Web页面当中，通过嵌入HTML来实现各种酷炫的动态效果，为用户提供赏心悦目的浏览效果。除此之外，也可以用于控制cookies以及基于Node.js技术进行服务器端编程。

javascript是甲骨文公司的注册商标，完整的JavaScript实现包含三个部分：ECMAScript，文档对象模型和浏览器对象模型。发展初期，JavaScript的标准并未确定，同期有Netscape的JavaScript，微软的JScript和CEnvi的ScriptEase三足鼎立。1997年，在ECMA(欧洲计算机制造商协会)的协调下，由Netscape、Sun、微软、Borland组成的工作组确定统一标准：ECMA-262。

## JavaScript 是脚本语言

JavaScript 是一种轻量级的编程语言。

JavaScript 是可插入 HTML 页面的编程代码。

JavaScript 插入 HTML 页面后，可由所有的现代浏览器执行。

JavaScript 很容易学习。

## javascript应用

1.[制作一些简单的小游戏，比如说象棋游戏、消除游戏、闯关游戏等等](https://webresourcesdepot.com/25-amazing-javascript-games-some-fun-and-inspiration/)

2.[javascript实例](http://www.w3cschool.cn/javascript/js-examples.html)

3.[利用javascript实现网页的前端逻辑](http://www.cnbeta.com/articles/75861.htm)

4.[利用javascript实现移动应用的框架](http://www.woshipm.com/it/50146.html)

## JavaScript：直接写入 HTML 输出流

<p>

JavaScript 能够直接写入 HTML 输出流中：

</p>

<script>

document.write("<h1>这是一个标题</h1>");

document.write("<p>这是一个段落。</p>");

</script>

<p>

您只能在 HTML 输出流中使用 <strong>document.write</strong>。

如果您在文档已加载后使用它（比如在函数中），会覆盖整个文档。

</p>

## JavaScript：对事件的反应

<h1>我的第一个 JavaScript</h1>

<p>

JavaScript 能够对事件作出反应。比如对按钮的点击：

</p>

<button type="button" onclick="alert('欢迎!')">点我!</button>

## JavaScript：改变 HTML 内容

<h1>我的第一段 JavaScript</h1>

<p id="demo">

JavaScript 能改变 HTML 元素的内容。

</p>

<script>

function myFunction()

{

x=document.getElementById("demo"); // 找到元素

x.innerHTML="Hello JavaScript!"; // 改变内容

}

</script>

<button type="button" onclick="myFunction()">点击这里</button>

## JavaScript：改变 HTML 样式

<h1>我的第一段 JavaScript</h1>

<p id="demo">

JavaScript 能改变 HTML 元素的样式。

</p>

<script>

function myFunction()

{

x=document.getElementById("demo") // 找到元素

x.style.color="#ff0000"; // 改变样式

}

</script>

<button type="button" onclick="myFunction()">点击这里</button>

# JavaScript 用法

## JavaScript 用法

HTML 中的脚本必须位于 <script> 与 </script> 标签之间。

脚本可被放置在 HTML 页面的 <body> 和 <head> 部分中。

## <script> 标签

如需在 HTML 页面中插入 JavaScript，请使用 <script> 标签。

<script> 和 </script> 会告诉 JavaScript 在何处开始和结束。

<script> 和 </script> 之间的代码行包含了 JavaScript:

<script>

alert("我的第一个 JavaScript");

</script>

您无需理解上面的代码。只需明白，浏览器会解释并执行位于 <script> 和 </script>之间的 JavaScript 代码

## <body> 中的 JavaScript

<!DOCTYPE html>

<html>

<body>

<script>

document.write("<h1>这是一个标题</h1>");

document.write("<p>这是一个段落</p>");

</script>

</body>

</html>

## 外部的 JavaScript

也可以把脚本保存到外部文件中。外部文件通常包含被多个网页使用的代码。

外部 JavaScript 文件的文件扩展名是 .js。

如需使用外部文件，请在 <script> 标签的 "src" 属性中设置该 .js 文件：

<h1>我的 Web 页面</h1>

<p id="demo">一个段落。</p>

<button type="button" onclick="myFunction()">点击这里</button>

<p><b>注释：</b>myFunction 保存在名为 "myscript.js" 的外部文件中。</p>

<script src="/statics/demosource/myscript.js"></script>

# JavaScript 输出

## JavaScript 输出

JavaScript 没有任何打印或者输出的函数。

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。

## 使用 window.alert()

你可以弹出警告框来显示数据：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>xxxxx </title>

</head>

<body>

<h1>我的第一个页面</h1>

<p>我的第一个段落。</p>

<script>

window.alert(5 + 6);

</script>

</body>

</html>

## 操作 HTML 元素

如需从 JavaScript 访问某个 HTML 元素，您可以使用 document.getElementById(id) 方法。

请使用 "id" 属性来标识 HTML 元素，并 innerHTML 来获取或插入元素内容：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>xxxx</title>

</head>

<body>

<h1>我的第一个 Web 页面</h1>

<p id="demo">我的第一个段落。</p>

<script>

document.getElementById("demo").innerHTML="段落已修改。";

</script>

</body>

</html>

以上 JavaScript 语句（在 <script> 标签中）可以在 web 浏览器中执行：

document.getElementById("demo") 是使用 id 属性来查找 HTML 元素的 JavaScript 代码 。

innerHTML = "Paragraph changed." 是用于修改元素的 HTML 内容(innerHTML)的 JavaScript 代码。

## 写到 HTML 文档

于测试目的，您可以将JavaScript直接写在HTML 文档中：

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>xxxxx</title>

</head>

<body>

<h1>我的第一个 Web 页面</h1>

<p>我的第一个段落。</p>

<script>

document.write(Date());

</script>

</body>

</html>

请使用 document.write() 仅仅向文档输出写内容。

如果在文档已完成加载后执行 document.write，整个 HTML 页面将被覆盖。

## 写到控制台

如果您的浏览器支持调试，你可以使用 console.log() 方法在浏览器中显示 JavaScript 值。

浏览器中使用 F12 来启用调试模式， 在调试窗口中点击 "Console" 菜单。

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>xxx </title>

</head>

<body>

<h1>我的第一个 Web 页面</h1>

<p>

浏览器中(Chrome, IE, Firefox) 使用 F12 来启用调试模式， 在调试窗口中点击 "Console" 菜单。

</p>

<script>

a = 5;

b = 6;

c = a + b;

console.log(c);

alert(c)

</script>

</body>

</html>

# JavaScript 语法

## JavaScript 语法

JavaScript 是一个程序语言。语法规则定义了语言结构。

JavaScript 是一个脚本语言。

它是一个轻量级，但功能强大的编程语言。

## JavaScript 字面量

在编程语言中，一个字面量是一个常量，如 3.14。

**数字（Number）字面量** 可以是整数或者是小数，或者是科学计数(e)。

3.14

1001

123e5

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

<script>

document.getElementById("demo").innerHTML = 123e5;

</script>

字符串（String）字面量 可以使用单引号或双引号 be written with double or single quotes:

"John Doe"  
  
'John Doe'

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

<script>

document.getElementById("demo").innerHTML = 'John Doe';

</script>

表达式字面量 用于计算：

5 + 6  
5 \* 10

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

<script>

document.getElementById("demo").innerHTML = 5 \* 10;

</script>

数组（Array）字面量 定义一个数组：

[40, 100, 1, 5, 25, 10]

对象（Object）字面量 定义一个对象：

{firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}

函数（Function）字面量 定义一个函数：

function myFunction(a, b) { return a \* b;}

## JavaScript 变量

在编程语言中，变量用于存储数据值。

JavaScript 使用关键字 **var** 来定义变量， 使用等号来为变量赋值：

var x, length  
x = 5  
length = 6

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

<script>

var length;

length = 6;

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

</script>

变量可以通过变量名访问。在指令式语言中，变量通常是可变的。字面量是一个恒定的值。

变量是一个名称。字面量是一个值。

## JavaScript 操作符

JavaScript使用 算术运算符 来计算值:

(5 + 6) \* 10

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

<script>

document.getElementById("demo").innerHTML = (5 + 6) \* 10;

</script>

JavaScript使用赋值运算符给变量赋值

x = 5  
y = 6  
z = (x + y) \* 10

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

<script>

var x, y, z;

x = 5

y = 6;

z = (x + y) \* 10;

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

</script>

JavaScript语言有多种类型的运算符：



## JavaScript 关键词

JavaScript 语句通常以**关键词**为开头。 **var** 关键词告诉浏览器创建一个新的变量：

var x = 5 + 6;  
var y = x \* 10;

## JavaScript 标识符

和其他任何编程语言一样，JavaScript 保留了一些标识符为自己所用。

JavaScript 同样保留了一些关键字，这些关键字在当前的语言版本中并没有使用，但在以后 JavaScript 扩展中会用到。

JavaScript 标识符必须以字母、下划线（\_）或美元符（$）开始。

后续的字符可以是字母、数字、下划线或美元符（数字是不允许作为首字符出现的，以便 JavaScript 可以轻易区分开标识符和数字）。

以下是 JavaScript 中最​​重要的保留字（按字母顺序）：



## JavaScript 注释

JavaScript 多行注释

多行注释以 /\* 开始，以 \*/ 结尾。

下面的例子使用多行注释来解释代码：

/\*

下面的这些代码会输出

一个标题和一个段落

并将代表主页的开始

\*/

document.getElementById("myH1").innerHTML="欢迎来到我的主页";

document.getElementById("myP").innerHTML="这是我的第一个段落。";

在行末使用注释

在下面的例子中，我们把注释放到代码行的结尾处：

var x=5; // 声明 x 并把 5 赋值给它

var y=x+2; // 声明 y 并把 x+2 赋值给它

## JavaScript 数据类型

JavaScript 有多种数据类型：数字，字符串，数组，对象等等：

var length = 16; // Number 通过数字字面量赋值

var points = x \* 10; // Number 通过表达式字面量赋值

var lastName = "Johnson"; // String 通过字符串字面量赋值

var cars = ["Saab", "Volvo", "BMW"]; // Array 通过数组字面量赋值

var person = {firstName:"John", lastName:"Doe"}; // Object 通过对象字面量赋值

## JavaScript 对大小写敏感。

JavaScript 对大小写是敏感的。

当编写 JavaScript 语句时，请留意是否关闭大小写切换键。

函数 getElementById 与 getElementbyID 是不同的。

同样，变量 myVariable 与 MyVariable 也是不同的。

## 空格

JavaScript 会忽略多余的空格。您可以向脚本添加空格，来提高其可读性。下面的两行代码是等效的：

var person="Hege";

var person = "Hege";

# JavaScript 语句

## JavaScript 语法

avaScript 语句是发给浏览器的命令。

这些命令的作用是告诉浏览器要做的事情。

下面的 JavaScript 语句向 id="demo" 的 HTML 元素输出文本 "你好 Dolly" ：

document.getElementById("demo").innerHTML = "Hello Dolly.";

<h1>我的网页</h1>

<p id="demo">我的第一个段落。</p>

<script>

document.getElementById("demo").innerHTML = "你好 Dolly";

</script>

## 分号 ;

分号用于分隔 JavaScript 语句。

通常我们在每条可执行的语句结尾添加分号。

使用分号的另一用处是在一行中编写多条语句。

a = 5;  
b = 6;  
c = a + b;

a = 5; b = 6; c = a + b;

document.getElementById("demo").innerHTML="你好 Dolly";  
document.getElementById("myDIV").innerHTML="你最近怎么样?";

## JavaScript 代码块

JavaScript 可以分批地组合起来。

代码块以左花括号开始，以右花括号结束。

代码块的作用是一并地执行语句序列。

本例向网页输出一个标题和两个段落：



## JavaScript 代码

JavaScript 代码是 JavaScript 语句的序列。

浏览器按照编写顺序依次执行每条语句。

本例向网页输出一个标题和两个段落：

# JavaScript 变量

## JavaScript 变量

变量是用于存储信息的"容器"。

var x=5;  
var y=6;  
var z=x+y;

<script>

var x=5;

var y=6;

var z=x+y;

document.write(x + "<br>");

document.write(y + "<br>");

document.write(z + "<br>");

</script>

就像代数那样

x=5

y=6

z=x+y

在代数中，我们使用字母（比如 x）来保存值（比如 5）。

通过上面的表达式 z=x+y，我们能够计算出 z 的值为 11。

在 JavaScript 中，这些字母被称为变量。

您可以把变量看做存储数据的容器。

与代数一样，JavaScript 变量可用于存放值（比如 x=5）和表达式（比如 z=x+y）。

变量可以使用短名称（比如 x 和 y），也可以使用描述性更好的名称（比如 age, sum, totalvolume）。

变量必须以字母开头

变量也能以 $ 和 \_ 符号开头（不过我们不推荐这么做）

变量名称对大小写敏感（y 和 Y 是不同的变量）

JavaScript 语句和 JavaScript 变量都对大小写敏感。

## JavaScript 数据类型

JavaScript 变量还能保存其他数据类型，比如文本值 (name="Bill Gates")。

在 JavaScript 中，类似 "Bill Gates" 这样一条文本被称为字符串。

JavaScript 变量有很多种类型，但是现在，我们只关注数字和字符串。

当您向变量分配文本值时，应该用双引号或单引号包围这个值。

当您向变量赋的值是数值时，不要使用引号。如果您用引号包围数值，该值会被作为文本来处理。

var pi=3.14;  
var person="John Doe";  
var answer='Yes I am!';

<script>

var pi=3.14;

var name="Bill Gates";

var answer='Yes I am!';

document.write(pi + "<br>");

document.write(name + "<br>");

document.write(answer + "<br>");

</script>

## 声明（创建） JavaScript 变量

在 JavaScript 中创建变量通常称为"声明"变量。

我们使用 var 关键词来声明变量：

var carname; // variable

变量声明之后，该变量是空的（它没有值）。

如需向变量赋值，请使用等号：

carname="Volvo";

不过，您也可以在声明变量时对其赋值

var carname="Volvo";

一个好的编程习惯是，在代码开始处，统一对需要的变量进行声明。

## 一条语句，多个变量

您可以在一条语句中声明很多变量。该语句以 var 开头，并使用逗号分隔变量即可：

var lastname="Doe", age=30, job="carpenter";

声明也可横跨多行：

var lastname="Doe",

age=30,

job="carpenter";

## Value = undefined

在计算机程序中，经常会声明无值的变量。未使用值来声明的变量，其值实际上是 undefined。

在执行过以下语句后，变量 carname 的值将是 undefined：

var carname;

## 重新声明 JavaScript 变量

如果重新声明 JavaScript 变量，该变量的值不会丢失：

在以下两条语句执行后，变量 carname 的值依然是 "Volvo"：

var carname="Volvo";

var carname;

# JavaScript 数据类型

## JavaScript 数据类型

字符串（String）、数字(Number)、布尔(Boolean)、数组(Array)、对象(Object)、空（Null）、未定义（Undefined）。

JavaScript 拥有动态类型

JavaScript 拥有动态类型。这意味着相同的变量可用作不同的类型：

var x; // x 为 undefined

var x = 5; // 现在 x 为数字

var x = "John"; // 现在 x 为字符串

## JavaScript 字符串

字符串是存储字符（比如 "Bill Gates"）的变量。

字符串可以是引号中的任意文本。您可以使用单引号或双引号：

var carname="Volvo XC60";

var carname='Volvo XC60';

您可以在字符串中使用引号，只要不匹配包围字符串的引号即可：

var answer="It's alright";

var answer="He is called 'Johnny'";

var answer='He is called "Johnny"';

## JavaScript 数字

JavaScript 只有一种数字类型。数字可以带小数点，也可以不带：

var x1=34.00; // 使用小数点来写

var x2=34; // 不使用小数点来写

极大或极小的数字可以通过科学（指数）计数法来书写：

var y=123e5;      // 12300000  
var z=123e-5;     // 0.00123

<script>

var x1=34.00;

var x2=34;

var y=123e5;

var z=123e-5;

document.write(x1 + "<br>")

document.write(x2 + "<br>")

document.write(y + "<br>")

document.write(z + "<br>")

</script>

JavaScript 布尔

布尔（逻辑）只能有两个值：true 或 false。

var x=true;

var y=false;

## JavaScript 数组

下面的代码创建名为 cars 的数组：

var cars=new Array();

cars[0]="Saab";

cars[1]="Volvo";

cars[2]="BMW";

或者 (condensed array):

var cars=new Array("Saab","Volvo","BMW");

var cars=new Array("Saab","Volvo","BMW");

或者 (literal array):

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

var i;

var cars = new Array();

cars[0] = "Saab";

cars[1] = "Volvo";

cars[2] = "BMW";

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

{

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

}

数组下标是基于零的，所以第一个项目是 [0]，第二个是 [1]，以此类推。

## JavaScript 对象

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

var person={firstname:"John", lastname:"Doe", id:5566};

上面例子中的对象 (person) 有三个属性：firstname、lastname 以及 id。

空格和折行无关紧要。声明可横跨多行：

var person={

firstname : "John",

lastname : "Doe",

id : 5566

};

对象属性有两种寻址方式：

name=person.lastname;

name=person["lastname"];

<script>

var person=

{

firstname : "John",

lastname : "Doe",

id : 5566

};

document.write(person.lastname + "<br>");

document.write(person["lastname"] + "<br>");

</script>

Undefined 和 Null

Undefined 这个值表示变量不含有值。

可以通过将变量的值设置为 null 来清空变量。

cars=null;  
person=null;

<script>

var person;

var car="Volvo";

document.write(person + "<br>");

document.write(car + "<br>");

var car=""

document.write(car + "<br>");

</script>

# JavaScript 运算符

## JavaScript 运算符

**运算符 = 用于赋值。**

**运算符 + 用于加值。**

1. 数值相加
2. 字符串相加
3. 数值和字符串混合相加

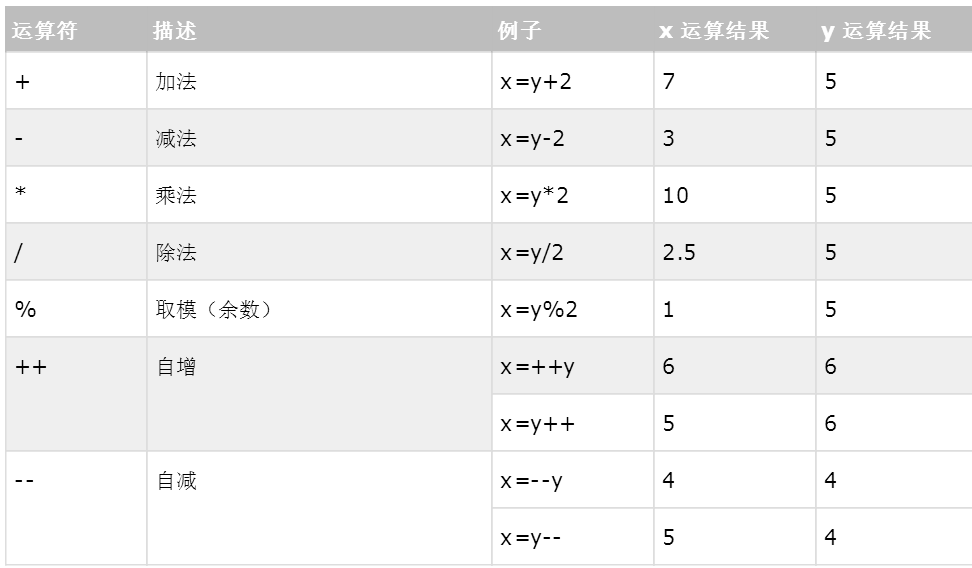
运算符 = 用于给 JavaScript 变量赋值。

算术运算符 + 用于把值加起来。

y=5;  
z=2;  
x=y+z;

## JavaScript 算术运算符

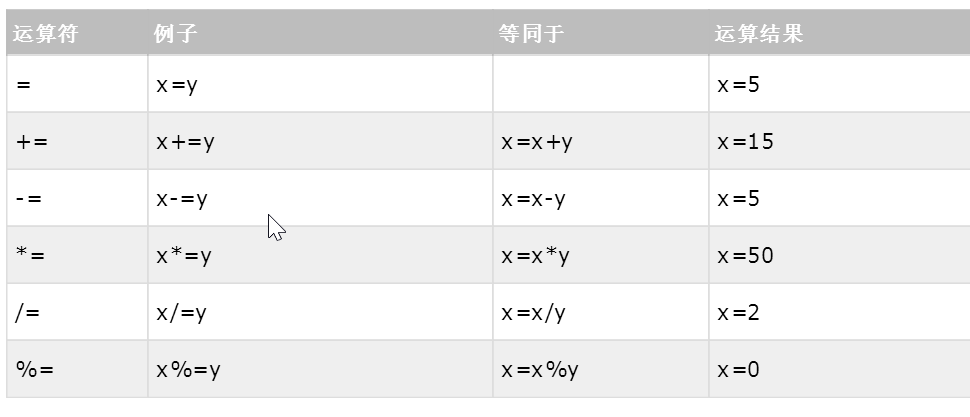
算术运算符用于执行变量与/或值之间的算术运算。 给定**y=5**，下面的表格解释了这些算术运算符：



## 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：



## 用于字符串的 + 运算符

+ 运算符用于把文本值或字符串变量加起来（连接起来）。

如需把两个或多个字符串变量连接起来，请使用 + 运算符。

txt1="What a very";  
txt2="nice day";  
txt3=txt1+txt2;

txt3 运算结果如下:

What a verynice day

txt1="What a very";

txt2="nice day";

txt3=txt1+txt2;

document.getElementById("demo").innerHTML=txt3;

## 对字符串和数字进行加法运算

两个数字相加，返回数字相加的和，如果数字与字符串相加，返回字符串，如下实例：

x=5+5;

y="5"+5;

z="Hello"+5;

10

55

Hello5

# JavaScript 比较和逻辑运算符

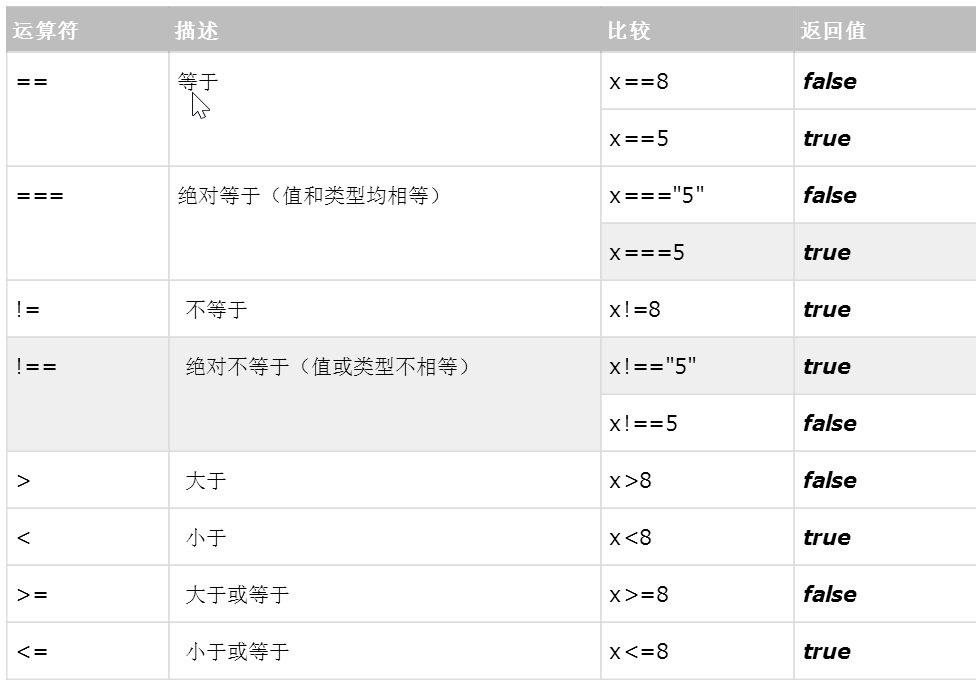
## JavaScript 比较 和 逻辑运算符

JavaScript 比较 和 逻辑运算符

## 比较运算符

比较运算符在逻辑语句中使用，以测定变量或值是否相等。

<p给定**x=5，下面的表格解释了比较运算符**



## 如何使用

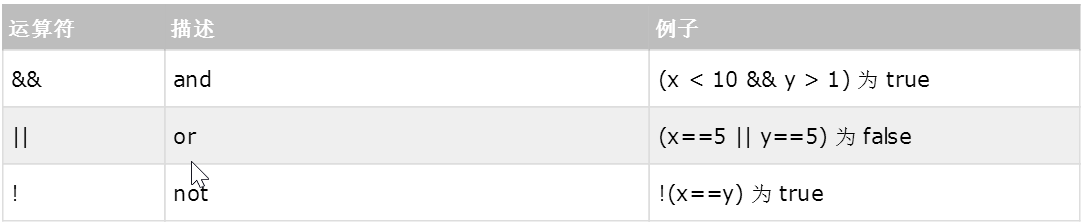
可以在条件语句中使用比较运算符对值进行比较，然后根据结果来采取行动：

if (age<18) x="Too young";

## 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：



## 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

如果变量 age 中的值小于 18，则向变量 voteable 赋值 "年龄太小"，否则赋值 "年龄已达到"。

voteable=(age<18)?"年龄太小":"年龄已达到";

# If…Else 语句

## 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
* **JavaScript三目运算 -**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 (time<20)  
  {  
  x="Good day";  
  }

## 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。

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 (time<20)  
  {  
  x="Good day";  
  }  
else  
  {  
  x="Good evening";  
  }

## 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。

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

}

if (time<10)  
  {  
  x="Good morning";  
  }  
else if (time>=10 && time<20)  
  {  
  x="Good day";  
  }  
else  
  {  
  x="Good evening";  
  }

# JavaScript while 循环

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

}

### 实例

本例中的循环将继续运行，只要变量 i 小于 5：

while (i<5)  
 {  
  x=x + "The number is " + i + "<br>";  
  i++;  
 }

如果您忘记增加条件中所用变量的值，该循环永远不会结束。这可能导致浏览器崩溃。

## 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);

下面的例子使用 do/while 循环。该循环至少会执行一次，即使条件为 false 它也会执行一次，因为代码块会在条件被测试前执行：

do

{

x=x + "The number is " + i + "<br>";

i++;

}

while (i<5);

# 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。

**Break 语句**

我们已经在本教程之前的章节中见到过 break 语句。它用于跳出 switch() 语句。

break 语句可用于跳出循环。

break 语句跳出循环后，会继续执行该循环之后的代码（如果有的话）：

while(x >10){

if(x== 5){

break;

}else{

Console.log(x);

}

}

## Continue 语句

**continue 语句**中断循环中的迭代，如果出现了指定的条件，然后继续循环中的下一个迭代。

while(x >10){

if(x== 5){

continue

}else{

Console.log(x);

}

}

# JavaScript for 循环

## JavaScript 循环

如果您希望一遍又一遍地运行相同的代码，并且每次的值都不同，那么使用循环是很方便的。

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

console.log(i);

}

# JavaScript 函数

函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。

<!DOCTYPE html>  
<html>  
<head>  
<script>  
function myFunction()  
{  
alert("Hello World!");  
}  
</script>  
</head>  
  
<body>  
<button onclick="myFunction()">Try it</button>  
</body>  
</html>

## JavaScript 函数语法

函数就是包裹在花括号中的代码块，前面使用了关键词 function：

function functionname()

{

执行代码

}

当调用该函数时，会执行函数内的代码。

可以在某事件发生时直接调用函数（比如当用户点击按钮时），并且可由 JavaScript 在任何位置进行调用。

JavaScript 对大小写敏感。关键词 function 必须是小写的，并且必须以与函数名称相同的大小写来调用函数。

## 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

}

变量和参数必须以一致的顺序出现。第一个变量就是第一个被传递的参数的给定的值，以此类推。

<button onclick="myFunction('Harry Potter','Wizard')">Try it</button>  
  
<script>  
function myFunction(name,job)  
{  
alert("Welcome " + name + ", the " + job);  
}  
</script>

## 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。

function myFunction()

{

var x=5;

return x;

}

# 局部 JavaScript 变量

在 JavaScript 函数内部声明的变量（使用 var）是局部变量，所以只能在函数内部访问它。（该变量的作用域是局部的）。

您可以在不同的函数中使用名称相同的局部变量，因为只有声明过该变量的函数才能识别出该变量。

只要函数运行完毕，本地变量就会被删除。

# 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。

# 向未声明的 JavaScript 变量分配值

如果您把值赋给尚未声明的变量，该变量将被自动作为全局变量声明。

# JavaScript 作用域

作用域可访问变量的集合。

在 JavaScript 中, 对象和函数同样也是变量。

**在 JavaScript 中, 作用域为可访问变量，对象，函数的集合。**

JavaScript 函数作用域: 作用域在函数内修改。

## JavaScript 局部作用域

变量在函数内声明，变量为局部作用域。

局部变量：只能在函数内部访问。

// 此处不能调用 carName 变量  
  
function myFunction() {  
    var carName = "Volvo";  
  
    // 函数内可调用 carName 变量  
  
}

因为局部变量只作用于函数内，所以不同的函数可以使用相同名称的变量。

局部变量在函数开始执行时创建，函数执行完后局部变量会自动销毁。

## JavaScript 全局变量

变量在函数外定义，即为全局变量。

全局变量有 **全局作用域**: 网页中所有脚本和函数均可使用。

var carName = " Volvo";  
  
// 此处可调用 carName 变量  
  
function myFunction() {  
  
    // 函数内可调用 carName 变量   
  
}

如果变量在函数内没有声明（没有使用 var 关键字），该变量为全局变量。

以下实例中 carName 在函数内，但是为全局变量。

// 此处可调用 carName 变量  
  
function myFunction() {  
    carName = "Volvo";  
  
    // 此处可调用 carName 变量  
  
}

## JavaScript 变量生命周期

JavaScript 变量生命周期在它声明时初始化。

局部变量在函数执行完毕后销毁。

全局变量在页面关闭后销毁。

## 函数参数

函数参数只在函数内起作用，是局部变量。

## switch 语句

JavaScript switch 语句

请使用 switch 语句来选择要执行的多个代码块之一。

switch(n)

{

case 1:

执行代码块 1

break;

case 2:

执行代码块 2

break;

default:

n 与 case 1 和 case 2 不同时执行的代码

}

工作原理：首先设置表达式 n（通常是一个变量）。随后表达式的值会与结构中的每个 case 的值做比较。如果存在匹配，则与该 case 关联的代码块会被执行。请使用 break 来阻止代码自动地向下一个 case 运行。

显示今天的星期名称。请注意 Sunday=0, Monday=1, Tuesday=2, 等等：

var day=new Date().getDay();  
switch (day)  
{  
case 0:  
  x="Today it's Sunday";  
  break;  
case 1:  
  x="Today it's Monday";  
  break;  
case 2:  
  x="Today it's Tuesday";  
  break;  
case 3:  
  x="Today it's Wednesday";  
  break;  
case 4:  
  x="Today it's Thursday";  
  break;  
case 5:  
  x="Today it's Friday";  
  break;  
case 6:  
  x="Today it's Saturday";  
  break;  
}

## default 关键词

请使用 default 关键词来规定匹配不存在时做的事情：

如果今天不是星期六或星期日，则会输出默认的消息：

var day=new Date().getDay();  
switch (day)  
{  
case 6:  
  x="Today it's Saturday";  
  break;  
case 0:  
  x="Today it's Sunday";  
  break;  
default:  
  x="Looking forward to the Weekend";  
}