JavaScript 对象 JavaScript 运算符

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。

实例

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>测试实例</title>
<script>
function myFunction()
{
alert("Hello World!");
}
</script>
</head>
<body>
<button onclick="myFunction()">点我</button>
</body>
</html>
```

尝试一下 »

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。

实例

```
<p>点击这个按钮，来调用带参数的函数。</p>
<button onclick="myFunction('Harry Potter','Wizard')">点击这里</button>
<script>
function myFunction(name,job){
alert("Welcome " + name + ", the " + job);
}
</script>
```

尝试一下 »

上面的函数在按钮被点击时会提示 "Welcome Harry Potter, the Wizard"。

函数很灵活，您可以使用不同的参数来调用该函数，这样就会给出不同的消息：

实例

```
<button onclick="myFunction('Harry Potter','Wizard')">点击这里</button>
<button onclick="myFunction('Bob','Builder')">点击这里</button>
```

尝试一下 »

根据您点击的不同的按钮，上面的例子会提示 "Welcome Harry Potter, the Wizard" 或 "Welcome Bob, the Builder"。

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;
}
```

上面的函数会返回值 5。

注意： 整个 JavaScript 并不会停止执行，仅仅是函数。JavaScript 将继续执行代码，从调用函数的地方。

函数调用将被返回值取代：

```
var myVar=myFunction();
```

myVar 变量的值是 5，也就是函数 "myFunction()" 所返回的值。

即使不把它保存为变量，您也可以使用返回值：

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

"demo" 元素的 innerHTML 将成为 5，也就是函数 "myFunction()" 所返回的值。

您可以使返回值基于传递到函数中的参数：

实例

计算两个数字的乘积，并返回结果：

```
function myFunction(a,b)
{
return a*b;
}
document.getElementById("demo").innerHTML=myFunction(4,3);
```

"demo" 元素的 innerHTML 将是：

```
12
```

尝试一下 »

在您仅仅希望退出函数时 ，也可使用 return 语句。返回值是可选的：

```
function myFunction(a,b)
{
if (a>b)
{
return;
}
x=a+b
}
```

如果 a 大于 b，则上面的代码将退出函数，并不会计算 a 和 b 的总和。

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 window 的一个属性。

这条语句：

```
carname="Volvo";
```

将声明 window 的一个属性 carname。

非严格模式下给未声明变量赋值创建的全局变量，是全局对象的可配置属性，可以删除。

```
var var1 = 1; // 不可配置全局属性
var2 = 2; // 没有使用 var 声明，可配置全局属性

console.log(this.var1); // 1
console.log(window.var1); // 1

delete var1; // false 无法删除
console.log(var1); //1

delete var2;
console.log(delete var2); // true
console.log(var2); // 已经删除 报错变量未定义
```

← JavaScript 对象 JavaScript 运算符 →