JavaScript 函数调用 JavaScript 正则表达式

# JavaScript 闭包

JavaScript 变量可以是局部变量或全局变量。

私有变量可以用到闭包。

## 全局变量

函数可以访问 由函数内部定义的变量，如：

**实例**

```
function myFunction() {
var a = 4;
return a * a;
}
```

尝试一下 »

函数也可以访问函数外部定义的变量，如：

**实例**

```
var a = 4;
function myFunction() {
return a * a;
}
```

尝试一下 »

后面一个实例中， **a** 是一个 **全局** 变量。

在web页面中全局变量属于 window 对象。

全局变量可应用于页面上的所有脚本。

在第一个实例中， **a** 是一个 **局部** 变量。

局部变量只能用于定义它函数内部。对于其他的函数或脚本代码是不可用的。

全局和局部变量即便名称相同，它们也是两个不同的变量。修改其中一个，不会影响另一个的值。

变量声明时如果不使用 **var** 关键字，那么它就是一个全局变量，即便它在函数内定义。

## 变量生命周期

全局变量的作用域是全局性的，即在整个JavaScript程序中，全局变量处处都在。

而在函数内部声明的变量，只在函数内部起作用。这些变量是局部变量，作用域是局部性的；函数的参数也是局部性的，只在函数内部起作用。

# 计数器困境

设想下如果你想统计一些数值，且该计数器在所有函数中都是可用的。

你可以使用全局变量，函数设置计数器递增：

**实例**

```
var counter = 0;
function add() {
return counter += 1;
}
add();
add();
add();
// 计数器现在为 3
```

尝试一下 »

计数器数值在执行 add() 函数时发生变化。

但问题来了，页面上的任何脚本都能改变计数器，即便没有调用 add() 函数。

如果我在函数内声明计数器，如果没有调用函数将无法修改计数器的值：

**实例**

```
function add() {
var counter = 0;
return counter += 1;
}
add();
add();
add();
// 本意是想输出 3, 但事与愿违，输出的都是 1 !
```

尝试一下 »

以上代码将无法正确输出，每次我调用 add() 函数，计数器都会设置为 1。

**JavaScript 内嵌函数可以解决该问题。**

---

# JavaScript 内嵌函数

所有函数都能访问全局变量。

实际上，在 JavaScript 中，所有函数都能访问它们上一层的作用域。

JavaScript 支持嵌套函数。嵌套函数可以访问上一层的函数变量。

该实例中，内嵌函数 **plus()** 可以访问父函数的 **counter** 变量：

**实例**

```
function add() {
var counter = 0;
function plus() {counter += 1;}
plus();
```

```
return counter;
}
```

尝试一下 »

如果我们能在外部访问 **plus()** 函数，这样就能解决计数器的困境。

我们同样需要确保 **counter = 0** 只执行一次。

**我们需要闭包。**

## JavaScript 闭包

还记得函数自我调用吗？该函数会做什么？

**实例**

```
var add = (function () {
var counter = 0;
return function () {return counter += 1;}
})();
add();
add();
add();
// 计数器为 3
```

尝试一下 »

## 实例解析

变量 **add** 指定了函数自我调用的返回字值。

自我调用函数只执行一次。设置计数器为 0。并返回函数表达式。

add变量可以作为一个函数使用。非常棒的部分是它可以访问函数上一层作用域的计数器。

这个叫作 JavaScript **闭包。**它使得函数拥有私有变量变成可能。

计数器受匿名函数的作用域保护，只能通过 add 方法修改。

闭包是可访问上一层函数作用域里变量的函数，即便上一层函数已经关闭。

← JavaScript 函数调用 JavaScript 正则表达式 →