JavaScript 函数定义 JavaScript 函数调用

JavaScript 函数参数

JavaScript 函数对参数的值没有进行任何的检查。

函数显式参数(Parameters)与隐式参数(Arguments)

在先前的教程中，我们已经学习了函数的显式参数:

```
functionName(parameter1, parameter2, parameter3) {
// 要执行的代码.....
}
```

函数显式参数在函数定义时列出。

函数隐式参数在函数调用时传递给函数真正的值。

参数规则

JavaScript 函数定义显式参数时没有指定数据类型。

JavaScript 函数对隐式参数没有进行类型检测。

JavaScript 函数对隐式参数的个数没有进行检测。

默认参数

ES5 中如果函数在调用时未提供隐式参数，参数会默认设置为： **undefined**

有时这是可以接受的，但是建议最好为参数设置一个默认值：

实例(ES5)

```
function myFunction(x, y) {
if (y === undefined) {
y = 0;
}
}
```

尝试一下 »

或者，更简单的方式：

实例(ES5)

```
function myFunction(x, y) {
y = y || 0;
}
```

尝试一下 »

如果y已经定义 ， y || 返回 y, 因为 y 是 true, 否则返回 0, 因为 undefined 为 false。

如果函数调用时设置了过多的参数，参数将无法被引用，因为无法找到对应的参数名。 只能使用 arguments 对象来调用。

ES6 函数可以自带参数

ES6 支持函数带有默认参数，就判断 undefined 和 || 的操作：

实例 (ES6)

```
function myFunction(x, y = 10) {
// y is 10 if not passed or undefined
return x + y;
}
myFunction(0, 2) // 输出 2
myFunction(5); // 输出 15, y 参数的默认值
```

尝试一下 »

Arguments 对象

JavaScript 函数有个内置的对象 arguments 对象。

argument 对象包含了函数调用的参数数组。

通过这种方式你可以很方便的找到最大的一个参数的值：

实例

```
x = findMax(1, 123, 500, 115, 44, 88);
function findMax() {
var i, max = arguments[0];
if(arguments.length < 2) return max;
for (i = 0; i < arguments.length; i++) {
if (arguments[i] > max) {
max = arguments[i];
}
}
return max;
}
```

尝试一下 »

或者创建一个函数用来统计所有数值的和：

实例

```
x = sumAll(1, 123, 500, 115, 44, 88);
function sumAll() {
var i, sum = 0;
for (i = 0; i < arguments.length; i++) {
sum += arguments[i];
}
return sum;
}
```

尝试一下 »

通过值传递参数

在函数中调用的参数是函数的隐式参数。

JavaScript 隐式参数通过值来传递：函数仅仅只是获取值。

如果函数修改参数的值，不会修改显式参数的初始值（在函数外定义）。

隐式参数的改变在函数外是不可见的。

通过对象传递参数

在JavaScript中，可以引用对象的值。

因此我们在函数内部修改对象的属性就会修改其初始的值。

修改对象属性可作用于函数外部（全局变量）。

修改对象属性在函数外是可见的。

JavaScript 函数定义 JavaScript 函数调用