JavaScript 调试 JavaScript 函数参数

JavaScript 函数定义

JavaScript 使用关键字 **function** 定义函数。

函数可以通过声明定义，也可以是一个表达式。

函数声明

在之前的教程中，你已经了解了函数声明的语法：

```
function functionName(parameters) {
  执行的代码
}
```

函数声明后不会立即执行，会在我们需要的时候调用到。

实例

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

尝试一下 »

分号是用来分隔可执行JavaScript语句。
由于函数声明不是一个可执行语句，所以不以分号结束。

函数表达式

JavaScript 函数可以通过一个表达式定义。

函数表达式可以存储在变量中：

实例

```
var x = function (a, b) {return a * b};
```

尝试一下 »

在函数表达式存储在变量后，变量也可作为一个函数使用：

实例

```
var x = function (a, b) {return a * b};
var z = x(4, 3);
```

尝试一下 »

以上函数实际上是一个 **匿名函数** (函数没有名称)。

函数存储在变量中，不需要函数名称，通常通过变量名来调用。

上述函数以分号结尾，因为它是一个执行语句。

Function() 构造函数

在以上实例中，我们了解到函数通过关键字 **function** 定义。

函数同样可以通过内置的 JavaScript 函数构造器（Function()）定义。

实例

```
var myFunction = new Function("a", "b", "return a * b");

var x = myFunction(4, 3);
```

尝试一下 »

实际上，你不必使用构造函数。上面实例可以写成：

实例

```
var myFunction = function (a, b) {return a * b}

var x = myFunction(4, 3);
```

尝试一下 »

在 JavaScript 中，很多时候，你需要避免使用 **new** 关键字。

函数提升（Hoisting）

在之前的教程中我们已经了解了 "hoisting(提升)"。

提升（Hoisting）是 JavaScript 默认将当前作用域提升到前面去的的行为。

提升（Hoisting）应用在变量的声明与函数的声明。

因此，函数可以在声明之前调用：

```
myFunction(5);

function myFunction(y) {

```

```
    return y * y;
}
```

使用表达式定义函数时无法提升。

自调用函数

函数表达式可以 "自调用"。

自调用表达式会自动调用。

如果表达式后面紧跟 () ，则会自动调用。

不能自调用声明的函数。

通过添加括号，来说明它是一个函数表达式：

实例

```
(function () {
    var x = "Hello!!";      // 我将调用自己
})();
```

尝试一下 »

以上函数实际上是一个 **匿名自我调用的函数** (没有函数名)。

函数可作为一个值使用

JavaScript 函数作为一个值使用：

实例

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

var x = myFunction(4, 3);
```

尝试一下 »

JavaScript 函数可作为表达式使用：

实例

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

var x = myFunction(4, 3) * 2;
```

尝试一下 »

函数是对象

在 JavaScript 中使用 **typeof** 操作符判断函数类型将返回 "function" 。

但是JavaScript 函数描述为一个对象更加准确。

JavaScript 函数有 **属性** 和 **方法**。

arguments.length 属性返回函数调用过程接收到的参数个数：

实例

```
function myFunction(a, b) {
    return arguments.length;
}
```

尝试一下 »

toString() 方法将函数作为一个字符串返回:

实例

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

var txt = myFunction.toString();
```

尝试一下 »

函数定义作为对象的属性，称之为对象方法。
函数如果用于创建新的对象，称之为对象的构造函数。

箭头函数

ES6 新增了箭头函数。

箭头函数表达式的语法比普通函数表达式更简洁。

```
(参数1, 参数2, …, 参数N) => { 函数声明 }

(参数1, 参数2, …, 参数N) => 表达式(单一)
// 相当于：(参数1, 参数2, …, 参数N) =>{ return 表达式; }
```

当只有一个参数时，圆括号是可选的：

```
(单一参数) => {函数声明}
单一参数 => {函数声明}
```

没有参数的函数应该写成一对圆括号:

```
() => {函数声明}
```

实例

```
// ES5
var x = function(x, y) {
return x * y;
}
// ES6
const x = (x, y) => x * y;
```

尝试一下 »

有的箭头函数都没有自己的 **this**。 不适合顶一个 **对象的方法**。

当我们使用箭头函数的时候，箭头函数会默认帮我们绑定外层 this 的值，所以在箭头函数中 this 的值和外层的 this 是一样的。

箭头函数是不能提升的，所以需要在使用之前定义。

使用 **const** 比使用 **var** 更安全，因为函数表达式始终是一个常量。

如果函数部分只是一个语句，则可以省略 return 关键字和大括号 {}，这样做是一个比较好的习惯:

实例

```
const x = (x, y) => { return x * y };
```

尝试一下 »

注意：IE11 及更早 IE 版本不支持箭头函数。

JavaScript 调试 JavaScript 函数参数