Function函数编程

窦连军 @八月虎baidu
北京乐美无限科技有限公司

JavaScript数据类型

	数据类型	示例
基本类型	String 字符串	‘张三’ ‘23’
	Number 数值	42 12.2 -9
	Boolean 布尔型	true false
引用类型	Object 对象	{x:12,y:34}
	Array 数组	[‘张三’,42,12.2]
	Function 函数	function(){...}

JavaScript中，函数function，跟object一样，都是一种引用的数据类型。

什么是函数对象

- JS的函数就是对象。
  - 对象字面量产生的对象通过“__proto__”属性隐含连接到Object.prototype，而函数对象隐含连接到Function.prototype，Function.prototype再隐含连接到Object.prototype。
- 函数是一个特殊的对象
  - 两个隐藏属性：
    - 函数的上下文
    - 实现函数行为的代码语句片段
  - 两个附送属性：
    - length（形式参数长度）
    - prototype：它的值是一个拥有constructor属性且constructor属性的值即为该函数的对象。
  - 函数实现体的两个缺省变量:
    - this、arguments（实参）

函数的对象用法

```
var   myFunc = function(){} ;
myFunc.myName = '张三' ;
myFunc.age = 21 ;

console.log(myFunc.myName) ;
```

判断对象的类型

- 判断一个对象是Arguments 还是 Arrays的方法： instanceof 和Object.prototype.toString

```
function bob(one, two, three) {
        var x = arguments;
        var y = [];

        console.log ( x instanceof Array ); //false
        console.log ( y instanceof Array ); //true
};
```

```
function bob(one, two, three) {
        var x = arguments;
        var y = [];

        console.log( Object.prototype.toString.call( x ) ); //[object Argume
        console.log( Object.prototype.toString.call( y ) ); //[object Array]
};
```

```
var p = new Person("Richard");
p instanceof Person //true
```

变量的作用域

- 什么是作用域（scope）
  - JavaScript的作用域完全是一个语法概念上的，一个变量的作用域基于其声明时在代码中出现的位置。
- 两种作用域
  - Global全局作用域：全局对象window
  - Functional函数作用域：位于函数内部。

全局变量 vs 局部变量

```
var message = 'Hello';
setMessage();
console.log(message);


function setMessage () {
    var message = 'Goodbye';
    console.log(message);
}
```

全局变量-
位于Windows上下文中

局部变量-
位于函数'setMessage'上下文中

```
Goodbye
Hello
```

输入参数

```
var message = 'Hello';
setMessage('Goodbye');
console.log(message);


function setMessage (message) {
    console.log(message);
}
```

'Goodbye'作为一项参数值，传递给setMessage函数

message作为参数变量名，位于函数作用域中，是本地变量。

```
Goodbye
Hello
```

返回值

```
var message = 'Hello';
console.log(setMessage());
console.log(message);

function setMessage () {
    var message = 'Goodbye';
    return message;
}
```

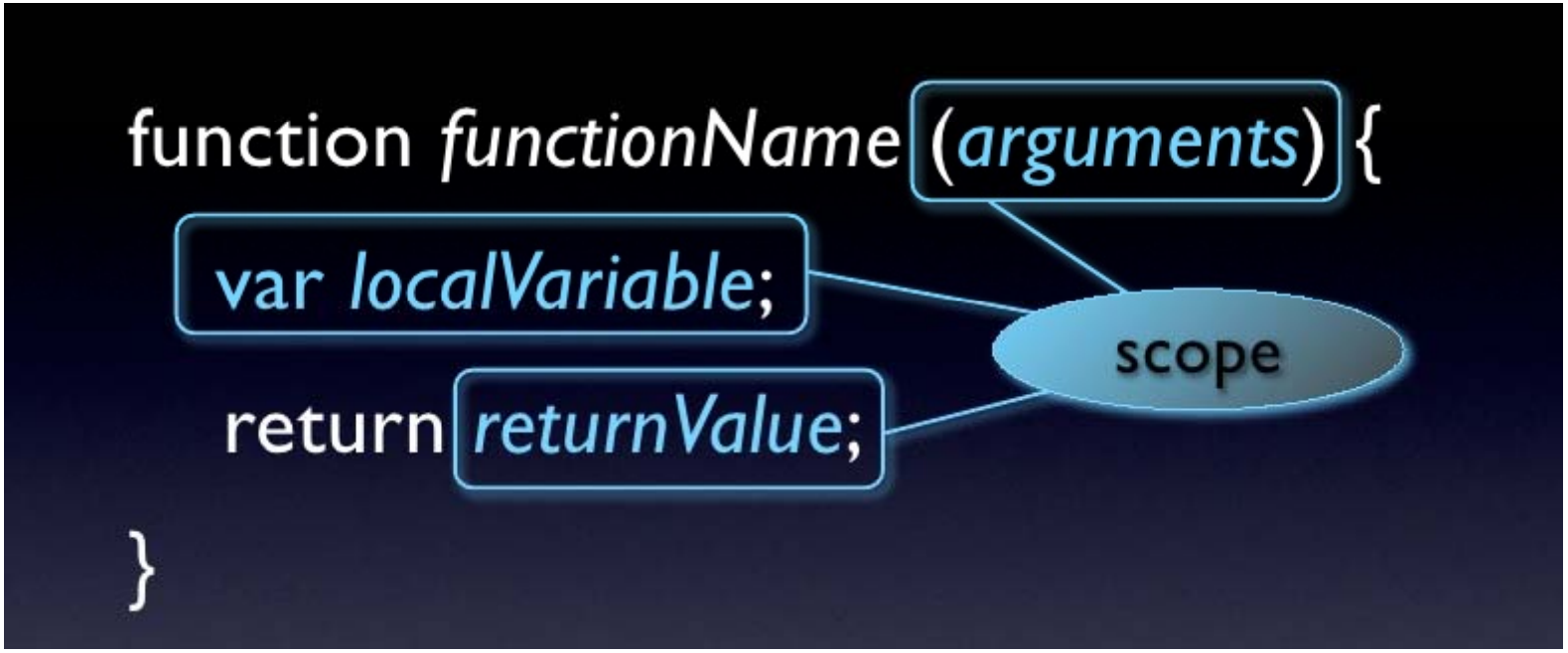
message作为函数返回值被返回时，仍然是函数作用域一部分。

message位于函数作用域，作为函数返回值。

```
Goodbye
Hello
```

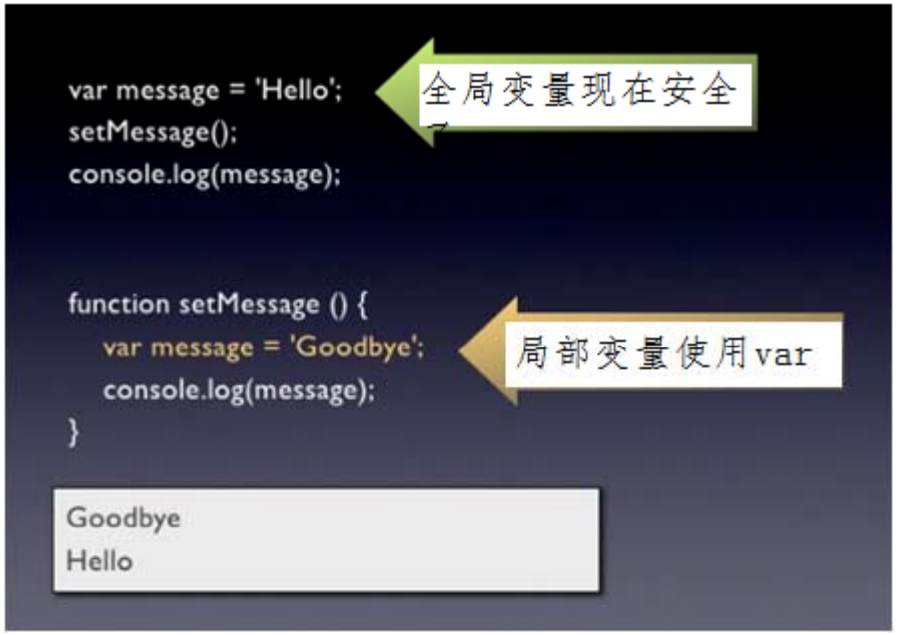
函数作用域总结

- 函数的输入参数、局部变量、返回值等均位于函数作用域中。



有无var声明的区别

- 声明变量时，应总是使用var



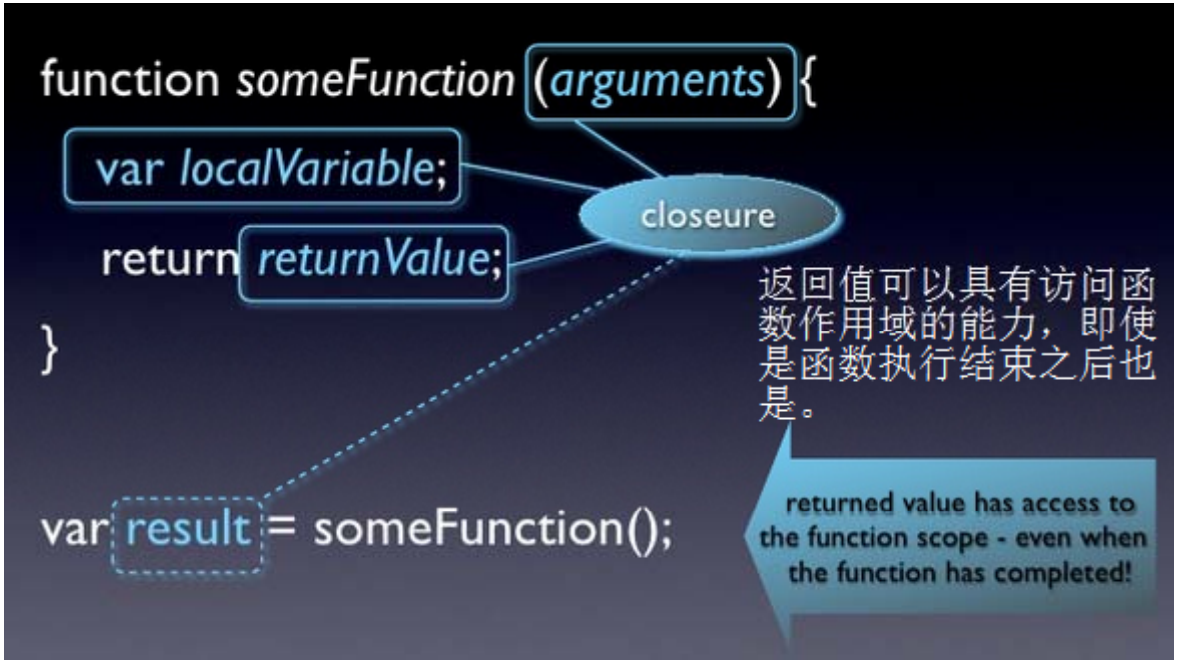
全局变量是有害的

- 全局变量容易与其他代码模块起冲突。

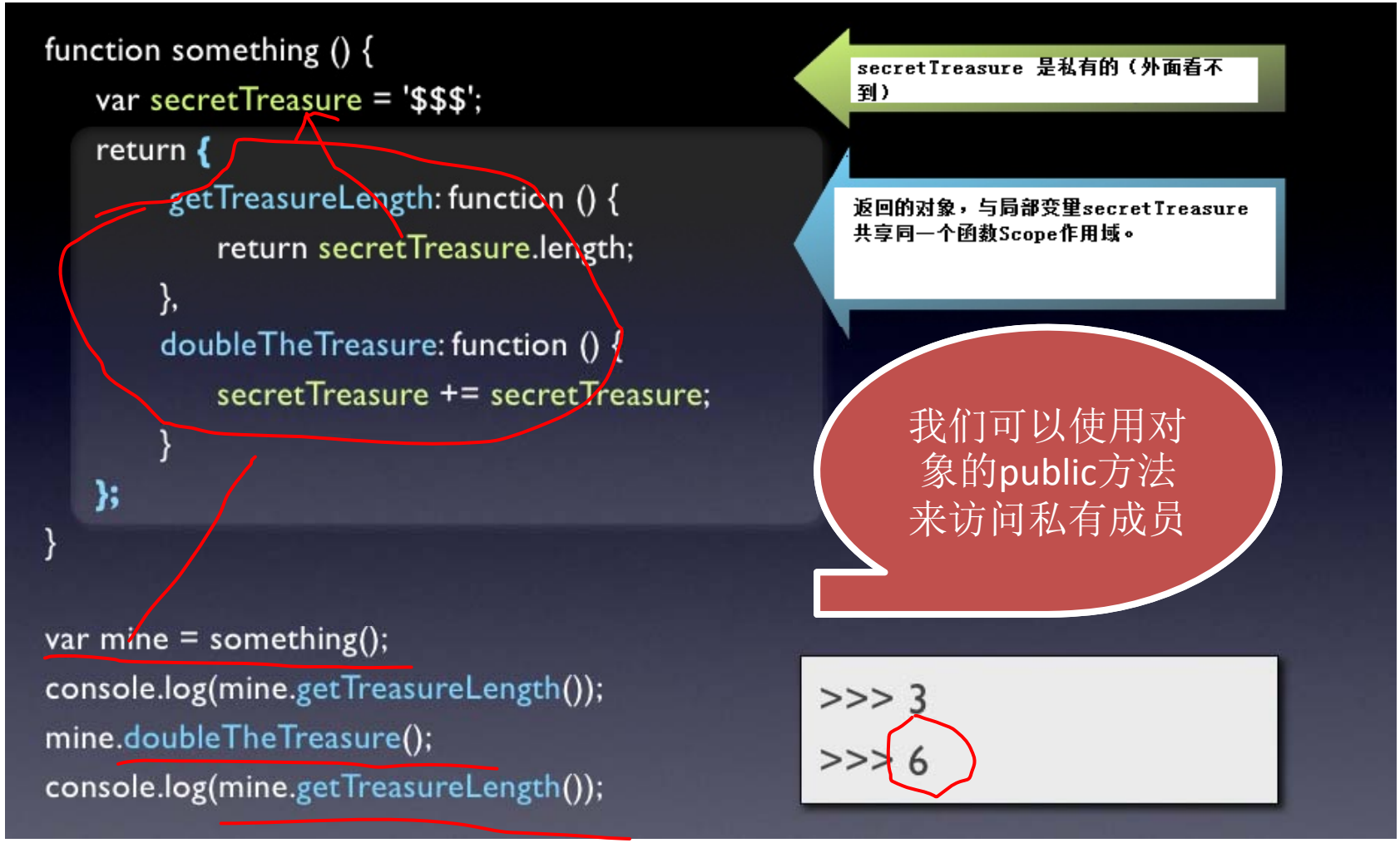
作用域总结

- 两种作用域：全局和函数
- 函数作用域：参数、局部变量和返回值
- 显式地声明变量：var
- 将全局变量保持到最小数量
- 对变量使用有意义的命名

函数作用域


function someFunction (arguments) {
var localVariable;
return returnValue;
}
closeure
返回值可以具有访问函数作用域的能力，即使是函数执行结束之后也是。
var result = someFunction();
returned value has access to the function scope - even when the function has completed!

闭包


function something () {
var secretTreasure = '$$$';
return {
getTreasureLength: function () {
return secretTreasure.length;
},
doubleTheTreasure: function () {
secretTreasure += secretTreasure;
}
};
}
var mine = something();
console.log(mine.getTreasureLength());
mine.doubleTheTreasure();
console.log(mine.getTreasureLength());
secretTreasure 是私有的（外面看不到）
返回的对象，与局部变里secretTreasure共享同一个函数Scope作用域。
我们可以使用对象的public方法来访问私有成员
>>> 3
>>> 6

闭包技术实现私有属性封装

- OO应可以支持封装，可以封装private私有成员；
- 但Object无法直接封装private成员，不过通过闭包技术可以实现私有属性的封装。

```
var radio = {
    volume: 0,
    frequency: 88.0,

    changeVolume: function (direction) {
        if (direction === 'up') this.volume += 1;
        else this.volume -= 1;
    },
    changeTuner: function (direction) {
        if (direction === 'up') this.frequency += 4;
        else this.frequency -= 4;
    }
};
```

编程模式：module pattern

```
var Module = (function(){
    var privateProperty = 'foo';

    function privateMethod(args){
        //do something
    }

    return {

        publicProperty: "",

        publicMethod: function(args){
            //do something
        },

        privilegedMethod: function(args){
            privateMethod(args);
        }
    }
})();
```

总结

- Object对象的属性是公共（public）的。
- function函数体中的局部变量是私有（private）的。
- 在函数执行完毕后，函数的闭包提供访问局部变量的能力。
- function函数可以提供带有私有数据成员的模块(module)