17.9.1 使用 constructor 检测数据类型

对于对象、数组等复杂数据，可以使用 Object 对象的 constructor 属性进行检测。constructor 表示构造器，该属性值引用的是构造当前对象的函数。

【示例 1】下面代码可以检测对象直接量和数组直接量的类型。

```
var o = {};
var a = [];
alert(o.constructor == Object);          // 返回 true
alert(a.constructor == Array);           // 返回 true
```

通过上面方法，可以准确判断复杂数据是对象，还是数组。如果结合 typeof 运算符和 constructor 属性，用户基本能够完成数据类型的检测，如表 E17.15 所示列举了不同类型数据的检测结果。测试代码：

```
var value = 1;                           // 输入不同类型的值（第一列）
alert(typeof value);                     // 返回 typeof 运算符返回的字符串（第二列）
alert(value.constructor);                // 返回 constructor 属性返回的对象（第三列）
```

表 E17.15 数据类型检测

值（value）	typeof value（表达式返回值）	value.constructor（构造函数的属性值）
var value = 1	"number"	Number
var value = "a"	"string"	String
var value = true	"boolean"	Boolean
var value = {}	"object"	Object
var value = new Object()	"object"	Object
var value = []	"object"	Array
var value = new Array()	"object"	Array
var value = function(){}	"function"	Function
function className(){}; var value = new className();	"object"	className

【示例 2】

使用 constructor 属性可以检测绝大部分数据的类型，但对于 undefined 和 null 特殊值，就不能够使用 constructor 属性，否则会抛出异常。这时可以先把值转换为布尔值，如果为 true，则说明是存在值的，然后再调用 constructor 属性。

```
var value = undefined;
alert(typeof value);                     // 返回字符串"undefined"
alert(value && value.constructor);       // 返回 undefined
var value = null;
alert(typeof value);                     // 返回字符串"object"
alert(value && value.constructor);       // 返回 null
```

另外，对于数值直接量也不能够直接使用 constructor 属性，下面代码将会提示语法错误：

```
alert(10.constructor);
```

但是如果加上一个小括号，则可以检测：

```
alert((10).constructor);
```

这是因为小括号运算符能够把数值转换为对象。