第 4 章 数据类型

学习要点：

1.typeof 操作符
2.Undefined 类型
3.Null 类型
4.Boolean 类型
5.Number 类型
6.String 类型
7.Object 类型

主讲教师：李炎恢
合作网站：http://www.ibeifeng.com
讲师博客：http://hi.baidu.com/李炎恢

ECMAScript 中有 5 种简单数据类型：Undefined、Null、Boolean、Number 和 String。还有一种复杂数据类型——Object。ECMAScript 不支持任何创建自定义类型的机制，所有值都成为以上 6 中数据类型之一。

一．typeof 操作符

typeof 操作符是用来检测变量的数据类型。对于值或变量使用 typeof 操作符会返回如下字符串。

字符串	描述
undefined	未定义
boolean	布尔值
string	字符串
number	数值
object	对象或 null
function	函数

```
var box = '李炎恢';
alert(typeof box);
alert(typeof '李炎恢');
```

typeof 操作符可以操作变量，也可以操作字面量。虽然也可以这样使用：typeof(box)，但，typeof 是操作符而非内置函数。PS：函数在 ECMAScript 中是对象，不是一种数据类型。所以，使用 typeof 来区分 function 和 object 是非常有必要的。

二．Undefined 类型

Undefined 类型只有一个值，即特殊的 undefined。在使用 var 声明变量，但没有对其初始化时，这个变量的值就是 undefined。

```
var box;
alert(box);
```

PS：我们没有必要显式的给一个变量赋值为 undefined，因为没有赋值的变量会隐式的(自动的)赋值为 undefined；而 undefined 主要的目的是为了用于比较，ECMAScript 第 3 版之前并没有引入这个值，引入之后为了正式区分空对象与未经初始化的变量。

未初始化的变量与根本不存在的变量(未声明的变量)也是不一样的。

```
var box;
alert(age);     //age is not defined
```

PS：如果 typeof box，typeof age 都返回的 undefined。从逻辑上思考，他们的值，一个是 undefined，一个报错；他们的类型，却都是 undefined。所以，我们在定义变量的时候，尽可能的不要只声明，不赋值。

三．Null 类型

Null 类型是一个只有一个值的数据类型，即特殊的值 null。它表示一个空对象引用(指针)，而 typeof 操作符检测 null 会返回 object。

```
var box = null;
alert(typeof box);
```

如果定义的变量准备在将来用于保存对象，那么最好将该变量初始化为 null。这样，当检查 null 值就知道是否已经变量是否已经分配了对象引用了。

```
var box = null;
if (box != null) {
    alert('box 对象已存在！');
}
```

有个要说明的是：undefined 是派生自 null 的，因此 ECMA-262 规定对它们的相等性测试返回 true。

```
alert(undefined == null);
```

由于 undefined 和 null 两个值的比较是相等的，所以，未初始化的变量和赋值为 null 的变量会相等。这时，可以采用 typeof 变量的类型进行比较。但，建议还是养成编码的规范，不要忘记初始化变量。

```
var box;
var car = null;
alert(typeof box == typeof car)
```

四．Boolean 类型

Boolean 类型有两个值(字面量)：true 和 false。而 true 不一定等于 1，false 不一定等于 0。JavaScript 是区分大小写的，True 和 False 或者其他都不是 Boolean 类型的值。

```
var box = true;
alert(typeof box);
```

虽然 Boolean 类型的字面量只有 true 和 false 两种，但 ECMAScript 中所有类型的值都有与这两个 Boolean 值等价的值。要将一个值转换为其对应的 Boolean 值，可以使用转型函数 Boolean()。

```
var hello = 'Hello World!';
var hello2 = Boolean(hello);
alert(typeof hello);
```

上面是一种显示转换，属于强制性转换。而实际应用中，还有一种隐式转换。比如，在 if 条件语句里面的条件判断，就存在隐式转换。

```
var hello = 'Hello World!';
if (hello) {
    alert('如果条件为 true，就执行我这条！');
} else {
    alert('如果条件为 false，就执行我这条！');
}
```

以下是其他类型转换成 Boolean 类型规则

数据类型	转换为 true 的值	转换为 false 的值
Boolean	true	false
String	任何非空字符串	空字符串
Number	任何非零数字值(包括无穷大)	0 和 NaN
Object	任何对象	null
Undefined		undefined

五．Number 类型

Number 类型包含两种数值：整型和浮点型。为了支持各种数值类型，ECMA-262 定义了不同的数值字面量格式。

最基本的数值字面量是十进制整数。

```
var box = 100;          //十进制整数
```

八进制数值字面量，(以 8 为基数)，前导必须是 0，八进制序列(0~7)。

```
var box = 070;          //八进制，56
var box = 079;          //无效的八进制，自动解析为 79
var box = 08;           //无效的八进制，自动解析为 8
```

十六进制字面量前面两位必须是 0x，后面是(0~9 及 A~F)。

```
var box = 0xA;          //十六进制，10
var box = 0x1f;         //十六进制，31
```

浮点类型，就是该数值中必须包含一个小数点，并且小数点后面必须至少有一位数字。

```
var box = 3.8;
var box = 0.8;
var box = .8;           //有效，但不推荐此写法
```

由于保存浮点数值需要的内存空间比整型数值大两倍，因此 ECMAScript 会自动将可以转换为整型的浮点数值转成为整型。

```
var box = 8.;           //小数点后面没有值，转换为 8
var box = 12.0;         //小数点后面是 0，转成为 12
```

对于那些过大或过小的数值，可以用科学技术法来表示(e 表示法)。用 e 表示该数值的前面 10 的指数次幂。

```
var box = 4.12e9;       //即 4120000000
var box = 0.00000000412;    //即 4.12e-9
```

虽然浮点数值的最高精度是 17 位小数，但算术运算中可能会不精确。由于这个因素，做判断的时候一定要考虑到这个问题(比如使用整型判断)。

```
alert(0.1+0.2);         //0.30000000000000004
```

浮点数值的范围在：Number.MIN_VALUE ~ Number.MAX_VALUE 之间。

```
alert(Number.MIN_VALUE);        //最小值
alert(Number.MAX_VALUE);        //最大值
```

如果超过了浮点数值范围的最大值或最小值，那么就先出现 Infinity(正无穷)或者 -Infinity(负无穷)。

```
var box = 100e1000;             //超出范围，Infinity
var box = -100e1000;            //超出范围，-Infinity
```

也可能通过 Number.POSITIVE_INFINITY 和 Number.NEGATIVE_INFINITY 得到 Infinity(正无穷)及-Infinity(负无穷)的值。

```
alert(Number.POSITIVE_INFINITY);  //Infinity(正无穷)
alert(Number.NEGATIVE_INFINITY);//-Infinity(负无穷)
```

要想确定一个数值到底是否超过了规定范围，可以使用 isFinite()函数。如果没有超过，返回 true，超过了返回 false。

```
var box = 100e1000;
alert(isFinite(box));                    //返回 false 或者 true
```

NaN，即非数值(Not a Number)是一个特殊的值，这个数值用于表示一个本来要返回数值的操作数未返回数值的情况(这样就不会抛出错误了)。比如，在其他语言中，任何数值除以 0 都会导致错误而终止程序执行。但在 ECMAScript 中，会返回出特殊的值，因此不会影响程序执行。

```
var box = 0 / 0;                         //NaN
var box = 12 / 0;                        //Infinity
var box = 12 / 0 * 0;                    //NaN
```

可以通过 Number.NaN 得到 NaN 值，任何与 NaN 进行运算的结果均为 NaN，NaN 与自身不相等(NaN 不与任何值相等)。

```
alert(Number.NaN);                       //NaN
alert(NaN+1);                            //NaN
alert(NaN == NaN)                        //false
```

ECMAScript 提供了 isNaN()函数，用来判断这个值到底是不是 NaN。isNaN()函数在接收到一个值之后，会尝试将这个值转换为数值。

```
alert(isNaN(NaN));                       //true
alert(isNaN(25));                        //false，25 是一个数值
alert(isNaN('25'));                      //false，'25'是一个字符串数值，可以转成数值
alert(isNaN('Lee'));                     //true，'Lee'不能转换为数值
alert(isNaN(true));                      //false     true 可以转成成 1
```

isNaN()函数也适用于对象。在调用 isNaN()函数过程中，首先会调用 valueOf()方法，然后确定返回值是否能够转换成数值。如果不能，则基于这个返回值再调用 toString()方法，再测试返回值。

```
var box = {
    toString : function () {
        return '123';                    //可以改成 return 'Lee'查看效果
    }
};
alert(isNaN(box));                       //false
```

有 3 个函数可以把非数值转换为数值：Number()、parseInt()和 parseFloat()。Number()函数是转型函数，可以用于任何数据类型，而另外两个则专门用于把字符串转成数值。

```
alert(Number(true));                     //1，Boolean 类型的 true 和 false 分别转换成 1 和 0
alert(Number(25));                       //25，数值型直接返回
alert(Number(null));                     //0，空对象返回 0
alert(Number(undefined));                //NaN，undefined 返回 NaN
```

如果是字符串，应该遵循一下规则：

1.只包含数值的字符串，会直接转成成十进制数值，如果包含前导 0，即自动去掉。

```
alert(Number('456'));          //456
alert(Number('070'));          //70
```

2.只包含浮点数值的字符串，会直接转成浮点数值，如果包含前导和后导 0，即自动去掉。

```
alert(Number('08.90'));        //8.9
```

3.如果字符串是空，那么直接转成成 0。

```
alert(Number(''));             //0
```

4.如果不是以上三种字符串类型，则返回 NaN。

```
alert('Lee123');               //NaN
```

5.如果是对象，首先会调用 valueOf()方法，然后确定返回值是否能够转换成数值。如果转换的结果是 NaN，则基于这个返回值再调用 toString()方法，再测试返回值。

```
var box = {
    toString : function () {
        return '123';          //可以改成 return 'Lee'查看效果
    }
};
alert(Number(box));            //123
```

由于 Number()函数在转换字符串时比较复杂且不够合理，因此在处理整数的时候更常用的是 parseInt()。

```
alert(parsetInt('456Lee'));          //456，会返回整数部分
alert(parsetInt('Lee456Lee'));       //NaN，如果第一个不是数值，就返回 NaN
alert(parseInt('12Lee56Lee'));       //12，从第一数值开始取，到最后一个连续数值结束
alert(parseInt('56.12'));            //56，小数点不是数值，会被去掉
alert(parseInt(''));                 //NaN，空返回 NaN
```

parseInt()除了能够识别十进制数值，也可以识别八进制和十六进制。

```
alert(parseInt('0xA'));              //10，十六进制
alert(parseInt('070'));              //56，八进制
alert(parseInt('0xALee'));           //100，十六进制，Lee 被自动过滤掉
```

ECMAScript 为 parseInt()提供了第二个参数，用于解决各种进制的转换。

```
alert(parseInt('0xAF'));             //175，十六进制
alert(parseInt('AF',16));            //175，第二参数指定十六进制，可以去掉 0x 前导
alert(parseInt('AF'));               //NaN，理所当然
alert(parseInt('101010101',2));      //314，二进制转换
alert(parseInt('70',8))              //56，八进制转换
```

parseFloat()是用于浮点数值转换的，和 parseInt()一样，从第一位解析到非浮点数值位置。

```
alert(parseFloat('123Lee'));        //123，去掉不是别的部分
alert(parseFloat('0xA'));           //0，不认十六进制
alert(parseFloat('123.4.5'));       //123.4，只认一个小数点
alert(parseFloat('0123.400'));      //123.4，去掉前后导
alert(parseFloat('1.234e7'));       //12340000，把科学技术法转成普通数值
```

六．String 类型

String 类型用于表示由于零或多个 16 位 Unicode 字符组成的字符序列，即字符串。字符串可以由双引号(")或单引号(')表示。

```
var box = 'Lee';
var box = "Lee";
```

PS：在某些其他语言(PHP)中，单引号和双引号表示的字符串解析方式不同，而 ECMAScript 中，这两种表示方法没有任何区别。但要记住的是，必须成对出现，不能穿插使用，否则会出错。

```
var box = '李炎恢";                 //出错
```

String 类型包含了一些特殊的字符字面量，也叫转义序列。

字面量	含义
\n	换行
\t	制表
\b	空格
\r	回车
\f	进纸
\\	斜杠
\'	单引号
\"	双引号
\xnn	以十六进制代码 nn 表示的一个字符(0~F)。例：\x41
\unnn	以十六进制代码 nnn 表示的一个 Unicode 字符(0~F)。例：\u03a3

ECMAScript 中的字符串是不可变的，也就是说，字符串一旦创建，它们的值就不能改变。要改变某个变量保存的字符串，首先要销毁原来的字符串，然后再用另一个包含新值的字符串填充该变量。

```
var box = 'Mr.';
box = box + ' Lee';
```

toString()方法可以把值转换成字符串。

```
var box = 11;
```

```
var box = true;
alert(typeof box.toString());
```

toString()方法一般是不需要传参的，但在数值转成字符串的时候，可以传递进制参数。

```
var box = 10;
alert(box.toString());          //10，默认输出
alert(box.toString(2));         //1010，二进制输出
alert(box.toString(8));         //12，八进制输出
alert(box.toString(10));        //10，十进制输出
alert(box.toString(16));        //a，十六进制输出
```

如果在转型之前不知道变量是否是 null 或者 undefined 的情况下，我们还可以使用转型函数 String()，这个函数能够将任何类型的值转换为字符串。

```
var box = null;
alert(String(box));
```

PS：如果值有 toString()方法，则调用该方法并返回相应的结果；如果是 null 或者 undefined，则返回"null"或者"undeinfed"。

七．Object 类型

ECMAScript 中的对象其实就是一组数据和功能的集合。对象可以通过执行 new 操作符后跟要创建的对象类型的名称来创建。

```
var box = new Object();
```

Object()是对象构造，如果对象初始化时不需要传递参数，可以不用写括号，但这种方式我们是不推荐的。

```
var box = new Object;
```

Object()里可以任意传参，可以传数值、字符串、布尔值等。而且，还可以进行相应的计算。

```
var box = new Object(2);        //Object 类型，值是 2
var age = box + 2;              //可以和普通变量运算
alert(age);                     //输出结果，转型成 Number 类型了
```

既然可以使用 new Object()来表示一个对象，那么我们也可以使用这种 new 操作符来创建其他类型的对象。

```
var box = new Number(5);        //new String('Lee')、new Boolean(true)
alert(typeof box);              //Object 类型
```

PS：面向对象是 JavaScript 课程的重点，这里我们只是简单做个介绍。详细的课程将在以后的章节继续学习。

感谢收看本次教程！

本课程是由北风网(ibeifeng.com)

瓢城 Web 俱乐部(yc60.com)联合提供：

本次主讲老师：李炎恢

我的博客：hi.baidu.com/李炎恢/

我的邮件：yc60.com@gmail.com