第三章

创建时间： 2021/1/6 17:48 **更新时间：** 2021/1/11 8:44

类型、值和变量

原始类型和**对象类型**
原始类型：数字、字符和布尔值
其中null和undefined，不是原始类型，它们通常分别代表各自特殊类型的唯一的成员。
对象：属性的集合，每个属性都有“名/值对”；数字
函数：具有与它相关联的可执行代码的对象，通过调用函数来运行可执行代码，并返回运算结果。
如果函数用new运算符新建一个对象，我们称之为构造函数

JavaScript语言核心定义了其他三种有用的类。日期(Date)类、正则(RegExp)类、错误(Error)类。

JavaScript解释器有自己的内存管理机制，可以自动对内存进行垃圾回收。这意味着程序可以按需创建对象，程序员则不必担心这些对象销毁和内存回收。当不再有任何引用指向一个对象，解释器就会知道这个对象没有用，然后自动回收它所占用的内存资源。

JavaScript是一种面向对象的语言。在JavaScript中只有null和undefined是无法拥有方法的值。

JavaScript类型可以分为：原始类型和对象类型，也可以分为拥有方法的类型和不能拥有方法的类型，同样也可以分为可变和不可变类型。可变类型的值是可修改的。对象和数组属于可变类型；数组、布尔值、null和undefined属于不可变类型。

JavaScript可以自由的进行数据类型转换，会进行自动转换。但是这种灵活的类型转换规则对“判断相等”的定义有一定的影响。

JavaScript变量是无类型的，变量可以被赋予任何类型的值，同样一个变量也可以重新赋予不同类型的值。
使用var关键字来声明，不在任何函数内声明的变量称作全局变量，在函数内声明的变量具有函数作用域。

3.1 数字
JavaScript不区分整数和浮点数值，JavaScript中所有数字均用浮点数值表示。
最大值是1.7976931348623157 x 10^308，最小值是5 x 10^-324

当一个数字直接出现在JavaScript程序中，我们称之为数字直接量。

3.1.1 整型直接量

十六进制：0x/0X为前缀
八进制：以0开始

3.1.2 浮点型直接量

3.1.3 JavaScript中的算术运算
+ - * / %
Math对象定义了许多的方法，也是可以进行计算的。

JavaScript的算术运算在溢出、下溢或被零整除时不会报错。无穷大：Infinity，负无穷大：-Infinity。NaN：非数字值。

isNaN()，如果参数是NaN或者是一个非数字值，则返回true。
isFinite()，在参数不是NaN、Infinity或-Infinity的时候返回true。

3.1.5 日期和时间

```
var then = new Date(2020,0,1);    //2020年1月1日
var later = new Date(2020,0,1,17,10,30);    //2020年1月1日 5:10:30pm
var now = new Date();    //当前日期和时间
var elapsed = now - then;    //日期减法：计算时间间隔的毫秒数
later.getFullYear()  //2020
later.getMounth()  //0 从0开始计数的月份
```

3.2.1 字符串直接量

在JavaScript程序中字符串直接量，是有单引号和双引号括起来的字符序列。
使用单引号的时候需要注意英文单词的缩写，can't 其中'和单引号是同一个字符，所以必须使用反斜杠(\)来转义。

3.2.3 字符串的使用

字符串的方法：

```
length    //长度
charAt(0)    //返回第一个字符
subString(1,4)    //第2~4个字符
slice(1,4)    //第2~4个字符
slice(-3)    //最后三个字符
indexOf("1")    //第一次出现1的位置
lastIndexOf("1",3)    //在位置3及之后首次出现字符1的位置
split(", ")    //根据,进行分割
replace("h", "H")    //替换字符h为H
toUpperCase()    //小写转大写

其中replace、toUpperCase是返回一个新的字符串，原本的字符串没有改变
在ES5中可以使用charAt()或者使用[](类似于数组)来读取字符串某个位置的值
```

3.2.4 模式匹配

使用RegExp(),也就是使用正则表达式实现一些字符串的操作。

```
var text = "testing: 1, 2, 3";
var pattern = /\d+/g    //匹配所有包含一个或多个数字的实例
text.match(pattern)  //["1","2","3"]：所有匹配组成的数组
text.replace(pattern,"#")  // “testing：#,#,#”
```

3.3 布尔值

布尔值指代真或假。
JavaScript程序中的比较语句的结果通常都是布尔值，例如
a==4

JavaScript的值都可以转换为布尔值，下面这些值会被转换成false：
undefined
null
0
-0
NaN
"" //空字符串
其他值，包括对象(数组)都会转换成true。

3.4 null和undefined

null ---> object

3.4 全局对象

- **全局属性：**undefined、Infinity、NaN
- **全局函数：**isNaN()、parseInt()、eval()
- **构造函数：**Date()、RegExp()、String()、Object()、Array()
- **全局对象：**Math、JSON

3.6 包装对象

JavaScript对象是一种复合值：它是属性或已命名值的集合。通过"."符号来引用属性值。当属性值是一个函数的时候，称其为方法。通过o.m()来调用对象o中的方法。

```
var s = "hello world";
var word = s.substring(s.indexOf(" ") + 1, s.length);
```

那字符串既然不是对象，为什么会有属性呢?
原因是只要引用字符串s的属性，JavaScript就会将字符串值通过new String(s)的方式转换成对象，这个对象继承了字符串的方法，并被用来处理属性的引用。一旦属性引用结束，这个新建的对象就会销毁。

同字符串一样，数字和布尔值也具有各自的方法：通过Number()和Boolean()构造函数创建临时对象。null和undefined没有包装对象：访问他们的属性会造成一个类型错误。

```
var s = "test";
s.len = 4;
var t = s.len;
```

上面这段代码，t的值为undefined。第二行代码创建一个临时对象，并给它的属性len赋值为4，执行完这条语句后对象会被销毁，所以在第三条语句中再次获取s的属性len时，字符串s会再次新建一个对象，而这次新建的对象和上次的对象显然不是同一个对象，所以当获取属性len的值时会是undefined。

在使用字符串、数字、布尔的属性时会创建一个临时对象，这个临时对象称为包装对象。
当然字符串、数字、布尔也可以直接调用构造函数来显示创建包装对象。
使用“==”会认为包装对象和原始值相等，而“===”则判断结果为false

3.7 不可变的原始值和可变的对象引用

原始值(undefined、null、布尔值、数字和字符串)是不可更改的。其中字符串我们往往认为是可以改变的，但是其改变的本质是返回一个新的字符串，原始的字符串还是没有改变。
原始值的比较是值的比较。

对象（数组、函数）是可以更改的。
对象的比较并非值的比较，而是引用的比较；当且仅当它们引用同一个基对象时，它们才相等。

```
var a = [];
var b = a;
b[0] = 1;
a[0];    //1
a === b;    //true
```

如果想要赋值一个对象，应该是逐个复制他的值，而不是引用。同样如果我们想要比较两个单独的对象或数组，则必须比较它们的属性或元素。

3.8 类型转换

表3-2：JavaScript类型转换

值	转换为：			
	字符串	数字	布尔值	对象
undefined	"undefined"	NaN	false	throws TypeError
null	"null"	0	false	throws TypeError
true	"true"	1		new Boolean(true)
false	"false"	0		new Boolean(false)
""(空字符串)		0	false	new String("")
"1.2"(非空,数字)		1.2	true	new String("1.2")
"one"(非空,非数字)		NaN	true	new String("one")
0	"0"		false	new Number(0)
-0	"0"		false	new Number(-0)
NaN	"NaN"		false	new Number(NaN)
Infinity	"Infinity"		true	new Number(Infinity)
-Infinity	"-Infinity"		true	new Number(-Infinity)
1(无穷大,非零)	"1"		true	new Number(1)
{}(任意对象)	参考3.8.3节	参考3.8.3节	true	
[](任意数组)	""	0	true	
[9](1个数字元素)	"9"	9	true	
['a'](其他数组)	使用join()方法	NaN	true	
function(){}(任意函数)	参考3.8.3节	NaN	true	

3.8.1 转换和相等性

当一个值转换为另一个值并不代表两个值相等。比方在if语句中使用undefined，会默认将undefined转换为false，但是这并不意味着undefined==false.

3.9.2 显示类型转换

尽管JavaScript可以自动做许多类型转换，但有时仍需要做显示转换，或者为了使代码变得清晰易读而做显示转换。

做显示转换最简单的方法就是使用Boolean()、Number()、String()或Object()函数，当不通过new运算符去调用这些函数的时候，他们会被当做类型转换函数，进行类型的转换

```
Number("3") //=>3
String(false)    //=>"false" 或者可以使用false.toString()
Boolean([])    //=>true
Object(3)    //=>new Object(3)
```

隐式类型转换

```
x + ""    //等价于String(x)
+x    //等价于Number(x),也可以写成x-0
!!x    //等价于Bealon(x),注意是双叹号
```

Number类定义了toString()方法，如果不传参默认转换为十进制，也可以将数字转换为其他进制数。

```
var n = 17;
binary_string = n.string(2);    //转换为"10001"
```

toFixed() //根据小数点后的指定位数将数字转换为字符串
toExponential() //使用指数记数法，将数字转换为指数形式的字符串，小数点前只有一位

toPrecision()　//根据指定的有效数字位数将数字转换成字符串。

```
var n = 123456.789;
n.toFixed(0);    //123456
n.toFixed(2);    //123456.78
n.toExponential(3);    1.234e+5
```