JS 权威指南 第三章

类型、值和变量

JS的数据类型分为两类：原始类型和对象类型。

JS中除了数字、字符串、布尔值、null和undefined之外就是对象了。

其中有一个比较特殊的对象——全局对象；JS同样定义了一种特殊对象——数组；还有一种特殊对象——函数。

如果函数用来初始化（使用new运算符）一个新建的对象，我们称之为构造函数。每一个构造函数定义了一类对象——由构造函数初始化的对象组成的集合。类可以看作是对象类型的子类型。除了数组（array）类和函数（function）类之外，JS语言核心定义了其它三种有用的类：日期（date）、正则（regexp）、错误（error）。

JS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，当不再有任何一个引用指向对象，解释器就会知道这个对象没用了，然后自动回收它所占用的内存资源。

JS是一种面向对象的语言，不严格的讲，这意味着我们不用全局的定义函数去操作不同类型的值，数据类型本身可以定义方法来使用值。

在JS中，只有null和undefined是无法拥有方法的值。

JS的类型可以分为可变类型和不可变类型。可变类型的值是可以修改的，对象和数组就属于可变类型……

JS变量是无类型的，变量可以被赋予任何类型的值，同样一个变量可以重新赋予不同类型的值。使用var关键字声明。

1. 数字

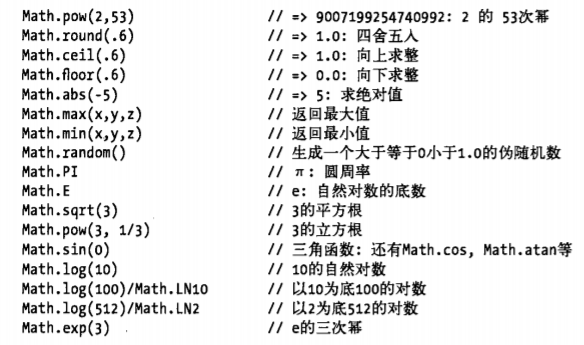
JS不区分整数值和浮点数值，JS中的所有数字均用浮点数值表示。采用64位浮点格式表示数字。需要注意的是，JS中实际的操作（如数组索引）则是基于32位整数。

有一个数字直接出线在JS的程序中，我们称之为数字直接量。

整型直接量：

浮点型直接量：

JS中的算术运算：基本的运算符 还有复杂运算符，通过Math对象的属性定义的函数和常量来实现。



JS中的算术运算在溢出、下溢或被零整除时不会报错。会被返回Infinity 或者 -Infinity。

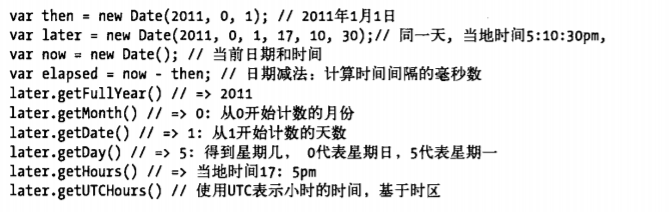
0/0会被返回NaN。无穷大除以无穷大、给任意负数作为开方运算或者算术运算符与不是数字或无法转换为数字的操作数一起使用时都将返回NaN。

NaN ！= NaN。没有办法通过x== NaN来判断变量x是否是NaN，应当用x！=x来判断。会返回true。函数is NaN（）的作用与此类似。

0===-0；

二进制浮点数和四舍五入错误：

日期和时间：

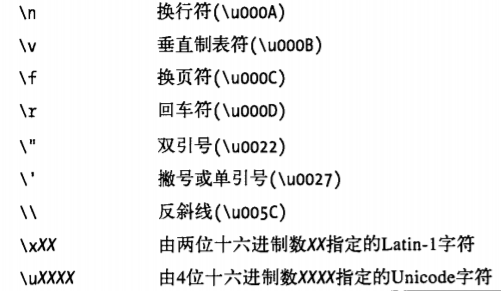


1. 文本

字符串是一组由16位值组成的不可变的有序序列，每个字符通常来自于Unicode字符集。JS通过字符串类型表示文本。字符串的长度是其所含16位值的个数。JS字符串（和其数组）的索引从0开始，空字符串的长度为0。

字符串直接量：

转义字符：



字符串的使用：加号运算符用于数字，表示两数相加。但将它们作用于字符串，则表示字符串连接。

得到字符串的长度：length属性

在JS中字符串是固定不变的，类似replace（）的方法都返回新字符串，原字符串本身并没有发生改变。

在ES5中，字符串可以当作只读数组，除了使用charAt()方法，也可以使用[ ]来访问字符串中的单个字符。

模式匹配：JS定义了RegExp（）构造函数，用来创建表示文本匹配模式的对象，这些对象称为“正则表达式”。？？？？？？？不懂

1. 布尔值

False undefined null 0 -0 NaN “” 被称为假值，除此之外全都为真值。

1. null和undefined

null常用来描述空值，对null执行typeof预算，结果返回“object”。

Undefined表明变量没有初始化，如果查询对象属性或数组元素的值时返回undefined则表明该属性或元素不存在，如果函数没有返回任何值，也返回undefined。引用没有提供实参的函数形参的值也会得到undefined。

1. 全局变量

全局对象的属性是全局定义的符号，JS程序可以直接使用。

全局属性：如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

在代码的最顶级——不在任何函数内的js代码——可以使用JS关键字this来引用全局对象。

1. 包装对象

JS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。

1. 不可变的原始值和可变的对象引用

JS中的原始值与对象有着根本的区别。原始值是不可更改的：任何方法都无法更改一个原始值。字符串中所有的方法看上去返回了一个修改后的字符串，实际上返回的是一个新的字符串值。

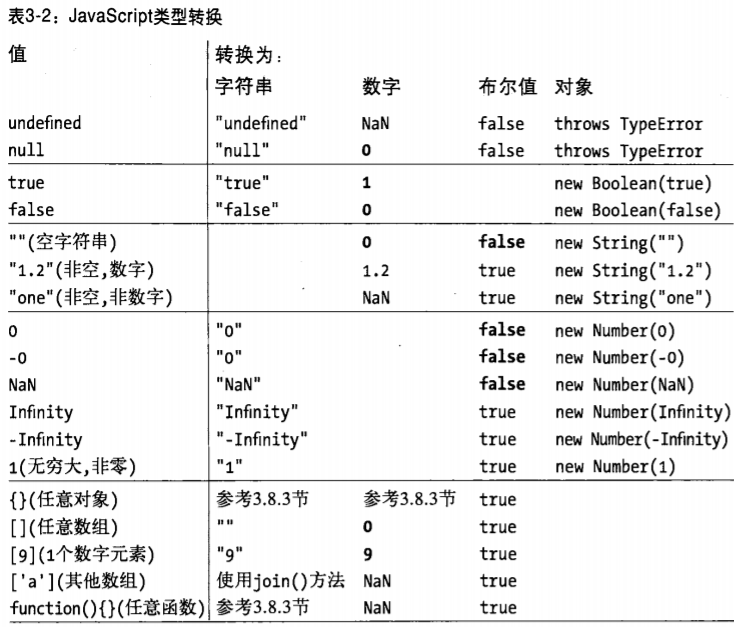
原始值的比较是值的比较：只有在他们值相等的时候他们才相等。

对象和原始值不同，它们的值是可以修改的。

对象的比较并非值的比较，即使两个对象包含相同的属性及相同的值，他们也是不相等的。各个索引元素完全相等的两个数组也不相等（比较的是地址）。

我们通常将对象称为引用类型，以此来和JS中的基本类型区分开。按照术语的叫法，对象值都是引用，对象的比较均是引用的比较，当且仅当他们引用同一个基对象时，他们才相等。

1. 类型转换



转换和相等性：

显式类型转换：

显示类型转换最简单的方法就是使用Boolean（）、Number（）、String()或Object（）函数……

对象转换为原始值：

对象到布尔值的转换非常简单，所有的对象（包括数组和函数）都转换为true。包装对象也是如此。

使用toString() valueof()方法，就可以将对象转化到字符串和数字了。

……

1. 变量声明

Var

1. 变量作用域

函数作用域和声明提前：在一些类似C语言的编程语言中，花括号内的每一段代码都具有各自的作用域，而且变量在声明他们的代码段之外是不可见的，我们称之为块级作用域。而JS没有块级作用域。JS取而代之的使用了函数作用域，变量在声明他们的函数体以及这个函数体嵌套的任意函数体内都是由定义的。

JS的函数作用域是指在函数内声明的所有变量在函数体内始终是可见的。这代表着变量在声明之前甚至已经可用。JS的这个特性被非正式地称为声明提前，即JS函数里声明地所有变量（但不涉及赋值）都被称为提前至函数的顶部。

由于函数作用域的特性，局部变量在整个函数体始终是有定义的，也就是说，在函数体内局部变量遮盖了同名全局变量。尽管如此，只有在程序执行到var语句时，局部变量才会被真正赋值。因此，上述过程等价于：将函数内的变量声明提前到函数体顶部，同时变量初始化留在原来的位置。

作为属性的变量：当声明一个JS全局变量时，实际上是定义了全局对象的一个属性。当使用var声明一个变量时，创建的这个属性是不可配置的，也就是说这个变量无法通过delete运算符删除。如果没有使用严格模式并给一个未声明的变量赋值的话，JS会自动创建一个全局变量。以这种方式创建的变量是全局变量的正常的可匹配属性，并可以删除它们。

JS全局变量是全局对象的属性。JS中可以允许使用this关键字来引用全局对象，却没有办法可以引用局部变量中存放的对象。

作用域链：？？？