**JS进阶课外总结**

# ch01.JavaScript发展历程及语言特点

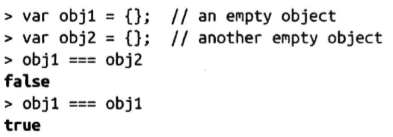
## 深入理解JavaScript第一部分

1、原始值和对象

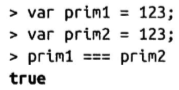
-原始值包括布尔值、数字、字符串、null和undefined。

-其他的值都是对象。

这两者之间最主要的区别在于他们的比较方式，每个对象都有唯一的标识且只（严格的）等于自己：

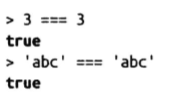


相反，所有原始值，只要编码相同，则被认为相等：

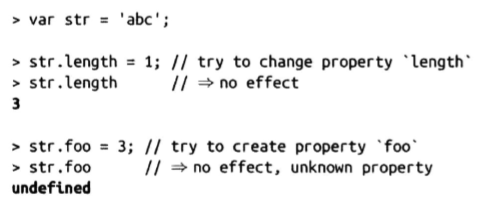


2、原始值特点

（1）按值进行比较



（2）其属性不能被改变、添加或移除，读取一个未知属性时总会返回undefined

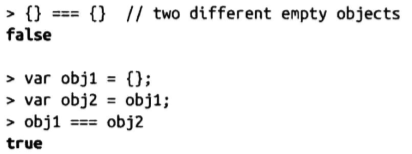


3、对象特点

所有非原始值都是对象。

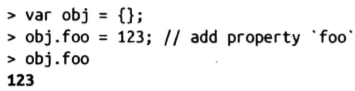
（1）按引用进行比较：

比较身份标识；每个值都有各自的身份标识



（2）默认可变

对象属性可以自由地改变、添加和移除

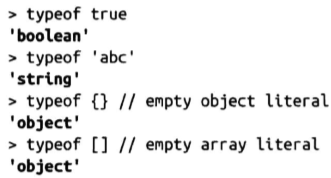


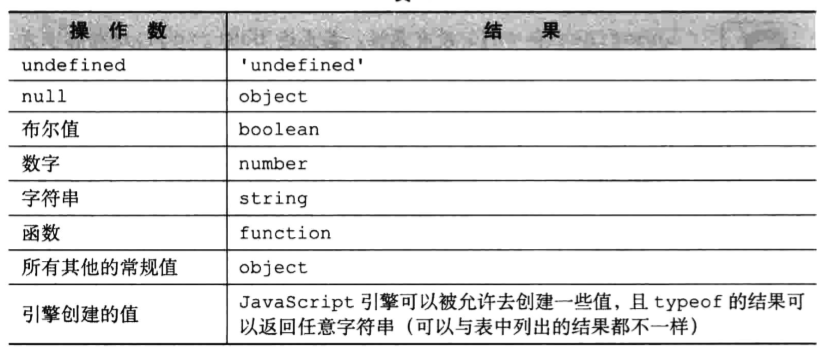
4、typeof和instanceof

typeof主要用于原始值，instanceof用于对象。

（1）typeof

typeof value：它的返回值是一个表示这个值“类型”的字符串：

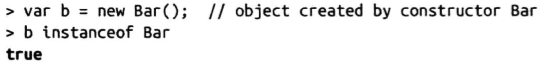
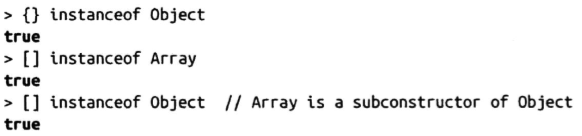




typeof null返回object是一个不能去修正的bug，因为这会破坏现有的代码，但这并不表示null是一个对象。

（2）instanceof

value instanceof Constr：如果value是一个通过Constr构造器创建的对象，则返回true：





5、等式运算符

-常规的，或“宽松的”相等（或不相等）：==，!=。

-严格的相等（或不相等）：===，!==。

6、被解释为false的值：

-undefined、null

-布尔值：false

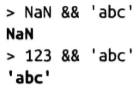
-数字：**-0**、NaN

-字符串：’’

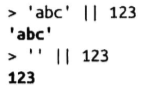
7、二元逻辑运算符（与假或真）

短路。**如果第一个运算数就足以确定结果的话，则永运不会第二个运算数做评估。**

（1）与（&&）：如果第一个运算数是假值，返回它。否则返回第二个运算数。



（2）或（||）：如果第一个运算数是真值，返回它。否则返回第二个运算数。



8、数字

JS中所有的数字都是浮点数：

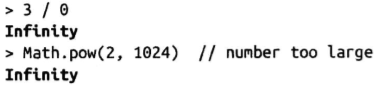


也包含一些特殊的数字：

-NaN（”not a number”） 一个错误的值：



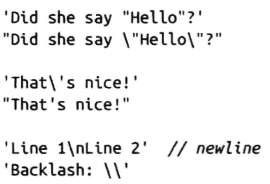
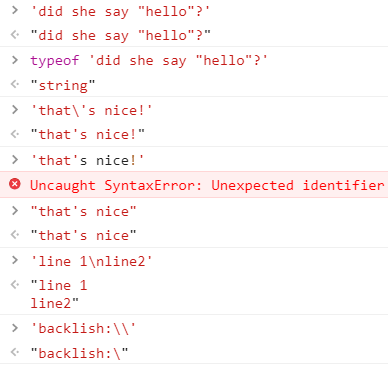
-Infinity 多数情况下也是一个错误的值：



Infinity比任何一个数都要大（NaN除外）。同样的，-Infinity比任何一个数都要小（NaN除外）。这使得这两个数字常用来作为默认值（比如，当你需要一个最大值和最小值的时候）。

9、 字符串-转义字符（\）

反斜杠（\）用于转义字符及产生一些控制字符：

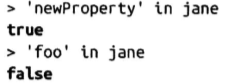
 

10、break：跳离循环

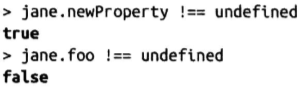
continue：开始一个新的迭代循环

11、对象

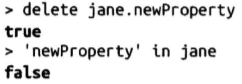
（1）使用in运算符检查属性是否存在：



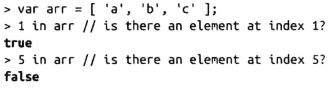
（2）如果读取一个不存在的属性，会得到undefined：



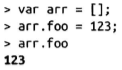
（3）使用delete运算符移除属性：



12、数组in操作符

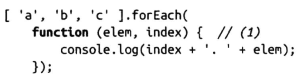


注意数组是对象，可以拥有对象属性。



13、遍历数组

（1）forEach迭代数组并且将当前的元素和元素的index扔到一个函数中：



输出结果：



(1)处的函数可以被随意地忽略参数。例如，它可以只包含一个elem参数。

（2）map通过一个函数映射到现有的数组的每个已经存在的元素创建一个新的数组：

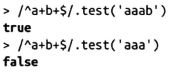
14、正则表达式

JavaScript内置的支持正则表达式，它们使用斜线分割：

/^abc$/

/[A-Za-z0-9]+/

（1）test()方法：匹配吗



（2）exec()方法：匹配以及捕获分组



返回的数组包含完整的匹配结果，它的**索引从0开始，第一组捕获的内容的索引是1**，以此类推。

（3）replace()方法：搜索和替换



replace的第一个参数必须是一个带着/g标志的正则表达式；否则将只替换第一次出现的内容。

# ch02.数据类型以及存储和转换

## 《你不知道的JavaScript（中卷）》第1/2/4章

1-1、JavaScript七种内置类型：

-空值（null）

-未定义（undefined）

-布尔值（boolean）

-数字（number）

-字符串（string）

-对象（object）

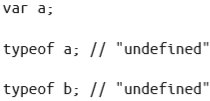
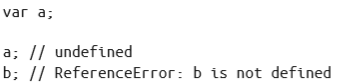
-符号（symbol，ES6中新增）

除对象之外，其他统称为“基本类型”。

1-2、JS中变量没有类型，只有值才有。变量可以随时持有任何类型的值。

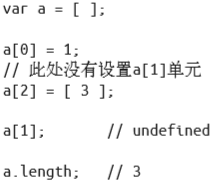
1-3、undefined：已在作用域中声明但还没有赋值的变量。

undeclared：还没有在作用域中声明过得变量。



2-1、稀疏数组

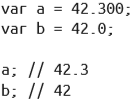
在创建“稀疏”数组（sparse array，即含有空白或空缺单元的数组）时要特别注意：



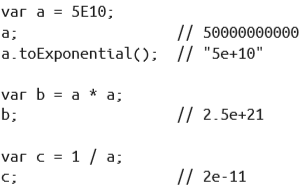
上方代码中a[1]的值为undefined，但这与将其显示赋值为undefined（a[1]=undefined）有所区别。

2-2、数字位数

默认情况下大部分数字都以十进制显示，小数部分最后面的0被省略：

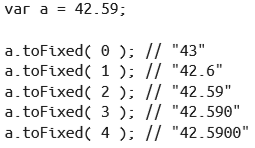


特别大和特别小的数字默认用指数格式显示，与toExponential()函数的输出结果相同：

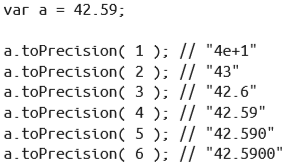


由于数字值可以使用 Number 对象进行封装，因此数字值可以调用 Number.prototype 中的方法。例如，

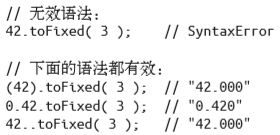
tofixed(..) 方法可指定小数部分的显示位数：

（输出结果为字符串形式）

toPrecision(..)方法用来指定有效数位的显示数位：



注意：





2-3、较小的数值

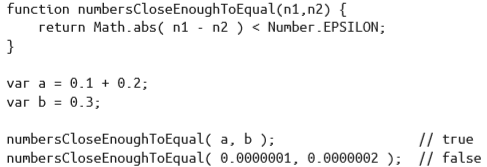
二进制浮点数最大问题：

简单来说，二进制浮点数中的 0.1 和 0.2 并不是十分精确，它们相加的结果并非刚好等于 0.3，而是一个比较接近的数字 0.30000000000000004，所以条件判断结果为 false。

机器精度值通常是：2^-52（2.220446049250313e-16）。从ES6开始，该值定义在Number.EPSILON中，可以直接用，也可以为ES6之前的版本写polyfill：



可以使用Number.EPSILON来比较两个数字是否相等（在指定误差范围内）：



2-4、整数安全范围

能被“安全”呈现的最大整数：2^53-1，即9007199254740991，在ES6中被定义为Number.MAX\_SAFE\_INTEGER；

最小整数：-9007199254740991，在ES6中被定义为Number.MIN\_SAFE\_INTEGER。

JS的数字类型无法精确呈现64位数值，所以必须将他们保存（转换）为字符串。

2-5、null和undefined

null指空值（曾赋过值，但目前没有指）

undefined指没有指（从未赋值）

null是一个特殊关键字，不是标识符，不能将其当作变量来赋值。然而undefined是一个标识符，可以被当作变量来使用和赋值。

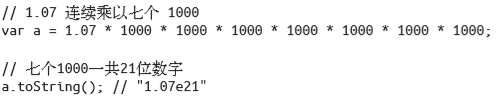
2-6、void运算符

表达式void \_没有返回值，因此返回结果是undefined。void并不改变表达式的结果，只是让表达式不返回值：



4-1、ToString：处理非字符串到字符串的强制类型转换。

数字的字符串化则遵循通用规则，不过极大和极小的数字使用指数形式：



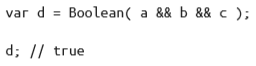
数组的默认toString()方法经过了重新定义，将所有单元字符串化以后再用“,”链接起来：



4-2、假值对象

假值对象并非封装了假值的对象：

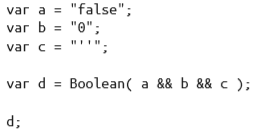


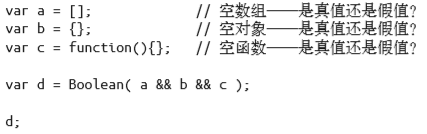


d为true说明a、b、c都为true。

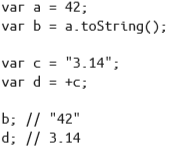
假值对象看起来和普通对象并无二致（都有属性，等等），但将它们强制类型转换为布尔值时结果为false。

4-3、真值（假值列表之外的值）

d为true

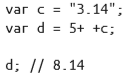
d为true

4-4、



上例中+c是+运算符的一元形式（即只有一个操作数）。+运算符显示地将c转换为数字，而非数字加法运算（也不是字符串拼接）。

在 JavaScript 开源社区中，一元运算 + 被普遍认为是显式强制类型转换。不过这样有时候也容易产生误会。例如：



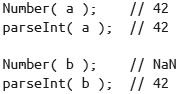
一元运算符 - 和 + 一样，并且它还会反转数字的符号位。由于 -- 会被当作递减运算符来处 理，所以我们不能使用 -- 来撤销反转，而应该像 - -"3.14" 这样，在中间加一个空格，才 能得到正确结果 3.14。

特例：

尽量不要把一元运算符 +（还有 -）和其他运算符放在一起使用。上面的代码可以运行， 但非常糟糕。此外 d = +c（还有 d =+ c）也容易和 d += c 搞混，两者天壤之别。

4-5、显示解析数字字符串

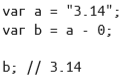




解析允许字符串中含有非数字字符，解析按从左到右的顺序，如果遇到非数字字符就停 止。而转换不允许出现非数字字符，否则会失败并返回 NaN。

解析和转换之间不是相互替代的关系。它们虽然类似，但各有各的用途。如果字符串右边 的非数字字符不影响结果，就可以使用解析。而转换要求字符串中所有的字符都是数字， 像 "42px" 这样的字符串就不行。

4-6、字符串强制类型转换为数字

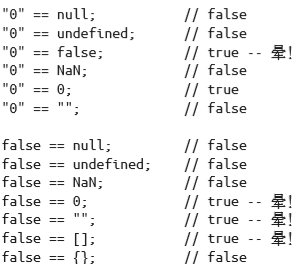


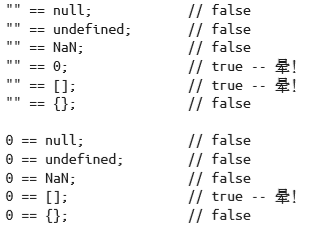
- 是数字减法运算符，因此 a - 0 会将 a 强制类型转换为数字。也可以使用 a \* 1 和 a / 1，因为这两个运算符也只适用于数字，只不过这样的用法不太常见。

4-7、NaN不等于NaN。

+0等于-0。

4-8、假值相等比较



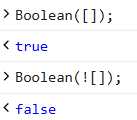


以上 24 种情况中有 17 种比较好理解。比如我们都知道 "" 和 NaN 不相等，"0" 和 0 相等。

然而有 7 种我们注释了“晕！”，因为它们属于假阳（false positive）的情况，里面坑很多。 "" 和 0 明显是两个不同的值，它们之间的强制类型转换很容易搞错。请注意这里不存在假 阴（false negative）的情况。

极端情况：

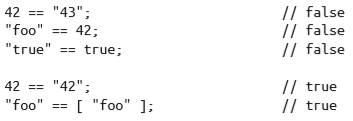


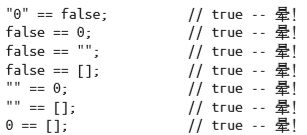


根据 ToBoolean 规则，它会进行布尔值的显式强制类型转换（同时反转奇偶校验位）。所以 [] == ![] 变成了 [] == false。又因为false == []，所以[] == ![]。



""、"\n"（或者 " " 等其他空格组合）等空字符串被 ToNumber 强制类型转换 为 0。







# ch03.包装对象和数据类型转换

## 《JavaScript权威指南》第三章

1、除了十进制的整型直接量，JavaScript同样能识别十六进制（以16为基数）值。所谓十六进制的直接量是以“0x”或“0X”为前缀，其后跟随十六进制数串的直接量。



2、Math



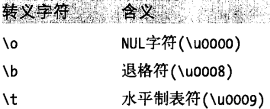
3、在BCMAScript 3中，字符串直接量必须写在一行中，而在ECMAScript 5中，字符串直

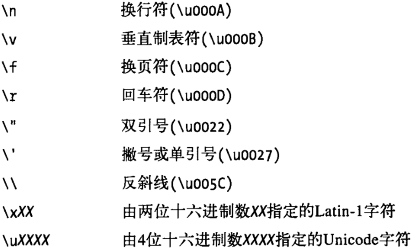
量可以拆分成数行，每行必须以反斜线(\) 结束，反斜线和行结束符都不算是字符串直接量的内容。如果希望在字符串直接量中另起一行，可以使用转义字符\n。



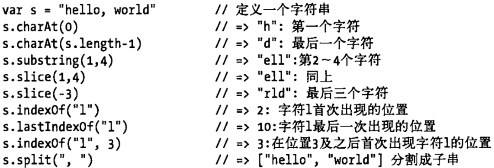


4、转义字符





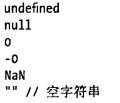
5、字符串属性





s.length

6、假值



false和上面6个可以转换成false的值有时称作“假值”，其他值称作真值。

7、全局对象

全局对象的属性时全局定义的符号，JS程序可以直接使用。当JS解释器启动时（或者任何Web浏览器加载新页面的时候），它将创建一个新的全局对象，并给它一组定义的初始属性：

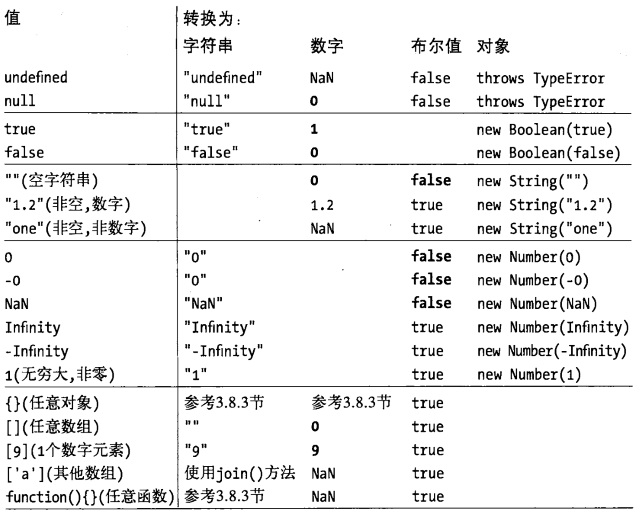
全局属性，比如undefined、Infinity和NaN

全局函数，比如isNaN()、parseInt()和eval()

构造函数，比如Data()、RegExp()、String()、Object()和Array()

全局对象，比如Math和JSON

8、类型转换



## 《深入理解JavaScript》第八章

1、undefined和null出现场景

（1）undefined出现场景

未初始化的变量



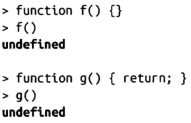
缺失的参数



访问一个不存在的属性会返回undefined



如果函数中没有显示地返回任何值，函数会隐式返回undefined



（2）null出现的场景

null是原型链最顶端的元素



当字符串中没有匹配到正则表达式的结果是，RegExp.prototype.exec()会返回null



2、算法：ToPrimitive()——将值转换为原始值

要将任意值转换为数字或者字符串，首先会被转换为任意的原始值，然后再转换为

最终的结果。



可选参数PreferredType表明转换后的类型:它可以是Number或String,具体取决于ToPrimitive的结果是希望转换成数字还是字符串。

如果PreferredType是Number,会执行以下步骤。

(1)如果input是原始值，返回这个值(没有其他需要做的)。

(2)否则，如果input是对象，调用input. value0f()。 如果结果是原始值，

返回结果。

(3)否则，调用input. toString()。如果结果是原始值，返回结果。

(4)否则，抛出一个TypeError (说明将输入转换为原始值出错了)。

如果PreferredType 是字符串，第二步和第三步会进行交换。PreferredType 也可以

被省略，这种情况下，日期会被认为是String而其他值会被认为是Number。因此，

+运算符和==运算符可以操作ToPrimitive()。

# ch04.标识符与表达式

## 《JavaScript权威指南》第4、5章

## 《深入理解JavaScript》第9、13章

# ch05.函数定义和调用形式

## 《JavaScript权威指南》第8章

## 《深入理解JavaScript》第15章

# ch06.函数定义和调用过程

## 《JavaScript权威指南》第8章

## 《深入理解JavaScript》第15、16章

# ch07.闭包和this

## 你不知道的JavaScript（上卷）

1-1、