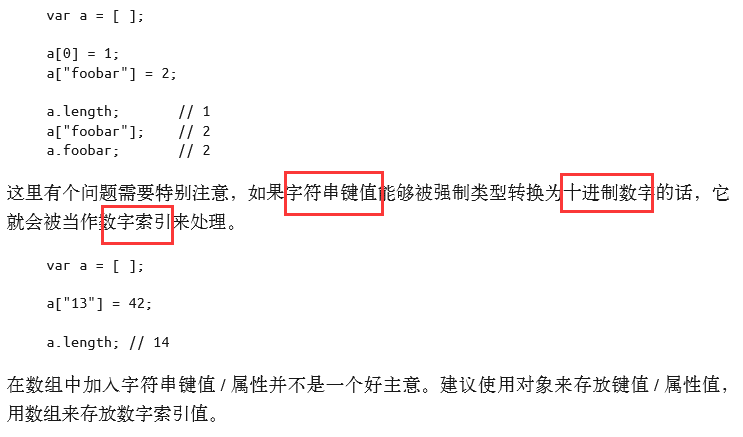
第二章 值

1. 数组
2. 在 JavaScript 中，数组可以容纳任何类型的值，可以是字符串、 数字、对象（object），甚至是其他数组（多维数组）
3. 数组通过数字进行索引，但有趣的是它们也是对象，所以也可以包含字符串键值和属性 （但这些并不计算在数组长度内）

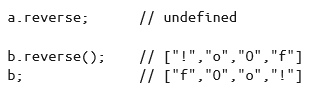


1. 类数组

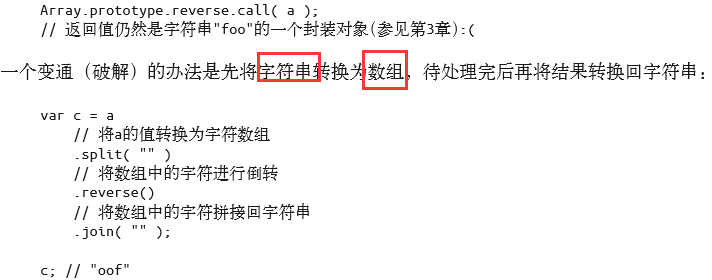
要将类数组（一组通过数字索引的值）转换为真正的数组，这一般通过数组工具函

数（如 indexOf(..)、concat(..)、forEach(..) 等）来实现。

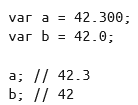
1. 字符串
2. 字符串和数组的确很相似，它们都是类数组，都有 length 属性以及 indexOf(..)和 concat(..) 方法
3. JavaScript 中字符串是不可变的，而数组是可变的。
4. 字符串不可变是指字符串的成员函数不会改变其原始值，而是创建并返回一个新的字符串。而数组的成员函数都是在其原始值上进行操作。
5. 不同点在于字符串反转（JavaScript 面试常见问题）。数组有一个字符串没有的可变更成员函数 reverse()



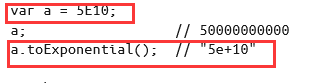
1. 使用 join("") 将字符数组转换为字符串。



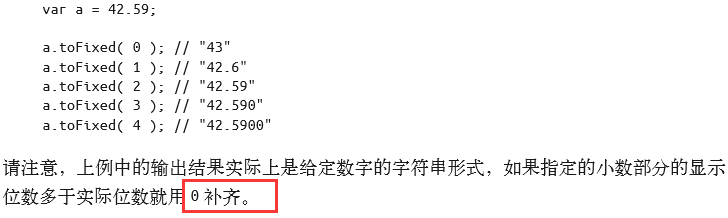
1. 数字（number）
2. 默认情况下大部分数字都以十进制显示，小数部分最后面的 0 被省略



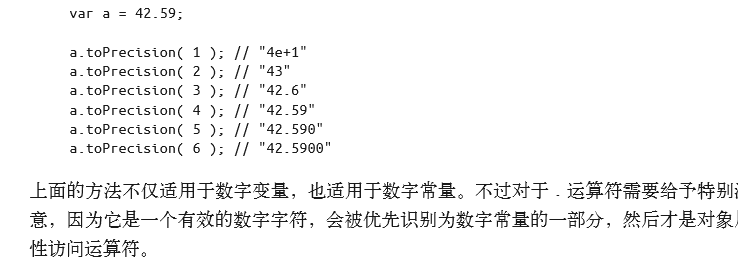
1. 特别大和特别小的数字默认用指数格式显示，与 toExponential() 函数的输出结果相同。

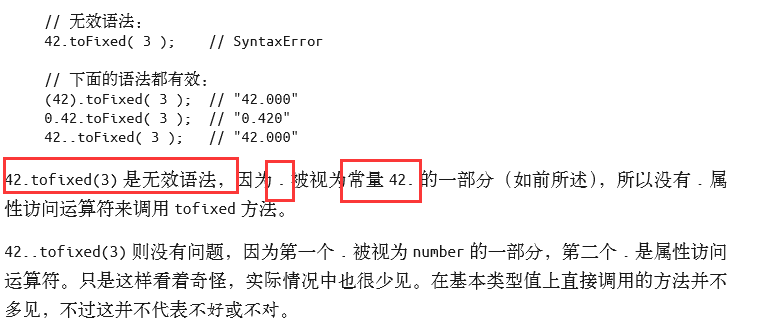


1. tofixed(..) 方法可指定小数部分的显示位数



1. toPrecision(..) 方法用来指定有效数位的显示位数：





1. 较小的数值

二进制浮点数最大的问题（不仅 JavaScript，所有遵循 IEEE 754 规范的语言都是如此），是 会出现如下情况：

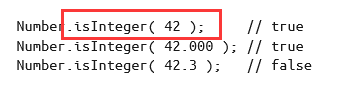
* 1. + 0.2 === 0.3; // false

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

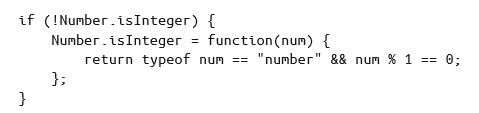
可以使用 Number.EPSILON 来比较两个数字是否相等（在指定的误差范围内）

1. 整数检测

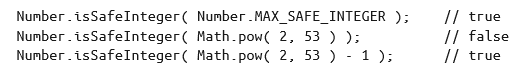
要检测一个值是否是整数，可以使用 ES6 中的 Number.isInteger(..) 方法：



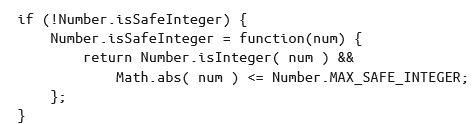
也可以为 ES6 之前的版本 polyﬁll Number.isInteger(..) 方法：



要检测一个值是否是安全的整数，可以使ES6中的 Number.isSafeInteger(..) 方法

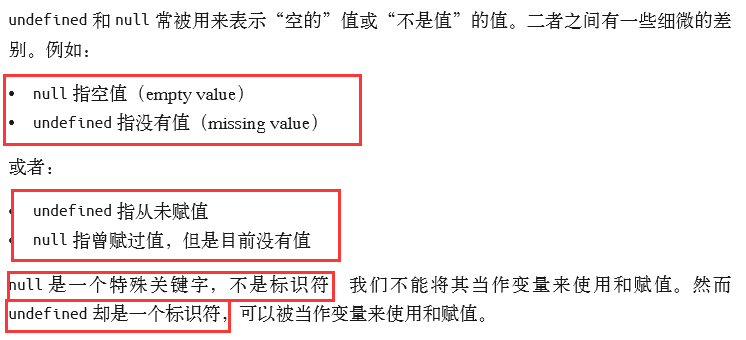


可以为 ES6 之前的版本 polyﬁll Number.isSafeInteger(..) 方法：



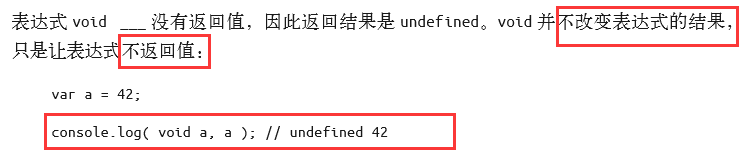
1. 特殊数值
2. 不是值的值

undefined 类型只有一个值，即 undefined。null 类型也只有一个值，即 null。它们的名 称既是类型也是值。



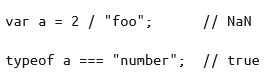
1. Undefined

undefined 是一个内置标识符（除非被重新定义，见前面的介绍），它的值为 undefined， 通过 void 运算符即可得到该值。



1. 特殊的数字
   1. 不是数字的数字

NaN 意指“不是一个数字”（not a number），“不是数字的数字”仍然是数字类型



NaN 是一个特殊值，它和自身不相等，是唯一一个非自反（自反，reﬂexive，即 x === x 不 成立）的值。 NaN != NaN 为 true

Number.isNaN(..)方法

* 1. 无穷数

var a = 1 / 0; 在JavaScript 中的结果为Infinity（即 Number.POSITIVE\_INfiNITY）。



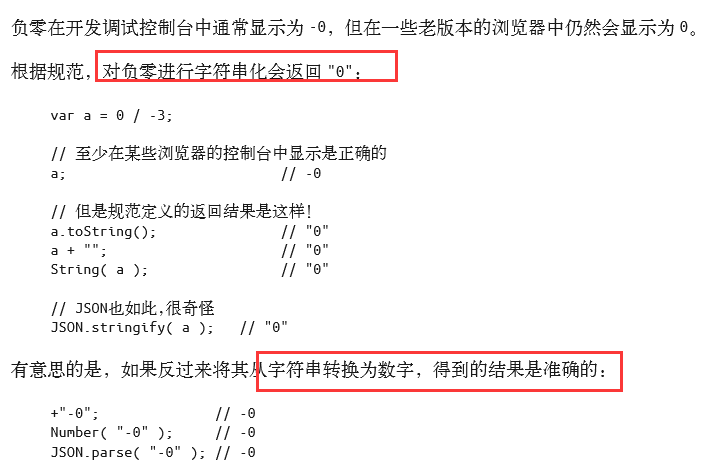
Infinity/ Infinity 是一个未定义操作，结果为 NaN

有穷正数除以 Infinity 的结果是 0。

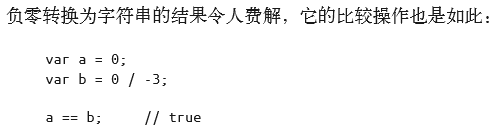
* 1. 零值

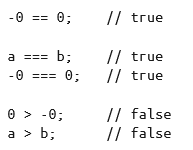


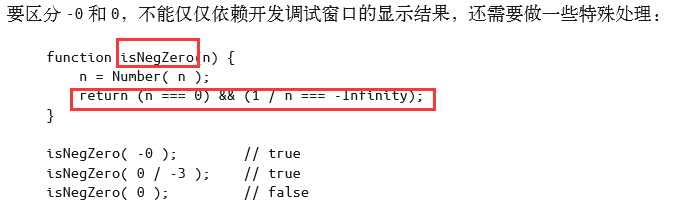
加法和减法运算不会得到负零（negative zero）。



JSON.stringify(-0) 返回 "0"，而 JSON.parse("-0") 返回 -0。







1. 特殊等式

ES6 中新加入了一个工具方法 Object.is(..) 来判断两个值是否绝对相等

1. 值和引用

JavaScript 对值和引用的赋值 / 传递在语法上没有区别，完全根据值的类型来决定。

简单值（即标量基本类型值，scalar primitive）总是通过值复制的方式来赋值 / 传递，包括 null、undefined、字符串、数字、布尔和 ES6 中的 symbol。

复合值（compound value）——对象（包括数组和封装对象）和函数，则总是通过引用复制的方式来赋值 / 传递。