* Number
* JavaScript不区分整数和浮点数，统一用Number表示，以下都是合法的Number类型：



* Number可以直接做四则运算。
* 字符串
* 字符串是以单引号'或双引号"括起来的任意文本，比如'abc'，"xyz"等等。请注意，''或""本身只是一种表示方式，不是字符串的一部分，因此，字符串'abc'只有a，b，c这3个字符。
* 布尔值
* 布尔值和布尔代数的表示完全一致，一个布尔值只有true、false两种值，要么是true，要么是false，可以直接用true、false表示布尔值，也可以通过布尔运算计算出来。
* &&运算是与运算，只有所有都为true，&&运算结果才是true。
* ||运算是或运算，只要其中有一个为true，||运算结果就是true。
* !运算是非运算，它是一个单目运算符，把true变成false，false变成true。
* 比较运算符
* JavaScript允许对任意数据类型做比较：
* JavaScript在设计时，有两种比较运算符：
  + 第一种是==比较，它会自动转换数据类型再比较，很多时候，会得到非常诡异的结果；
  + 第二种是===比较，它不会自动转换数据类型，如果数据类型不一致，返回false，如果一致，再比较。
* 由于JavaScript这个设计缺陷，不要使用==比较，始终坚持使用===比较。
* 另一个例外是NaN这个特殊的Number与所有其他值都不相等，包括它自己：



* **唯一能判断NaN的方法是通过isNaN()函数**：



* 注意浮点数的相等比较，浮点数在运算过程中会产生误差，因为计算机无法精确表示无限循环小数。要比较两个浮点数是否相等，只能计算它们之差的绝对值，看是否小于某个阈值：



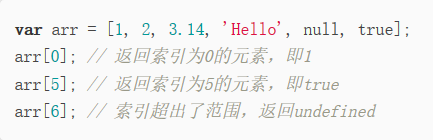
* null和undefined
* null表示一个“空”的值，它和0以及空字符串''不同，0是一个数值，''表示长度为0的字符串，而**null表示“空”**。
* 在其他语言中，也有类似JavaScript的null的表示，例如Java也用null，Swift用nil，Python用None表示。但是，在JavaScript中，还有一个和null类似的**undefined，它表示“未定义”**。
* JavaScript的设计者希望用null表示一个空的值，而undefined表示值未定义。事实证明，这并没有什么卵用，区分两者的意义不大。**大多数情况下，我们都应该用null。undefined仅仅在判断函数参数是否传递的情况下有用**。
* 数组
* 数组是一组按顺序排列的集合，集合的每个值称为元素。JavaScript的数组可以**包括任意数据类型**。例如：



* 另一种创建数组的方法是通过Array()函数实现：



* 然而，出于代码的可读性考虑，强烈建议直接使用[]。数组的元素可以通过索引来访问。请注意，索引的起始值为0：



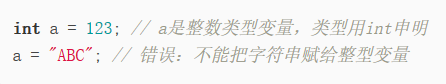
* 对象
* JavaScript的**对象是一组由键-值组成的无序集合**，例如：



* JavaScript对象的键都是字符串类型，值可以是任意数据类型。上述person对象一共定义了6个键值对，其中每个键又称为对象的属性，例如，person的name属性为'Bob'，zipcode属性为null。
* 要获取一个对象的属性，我们用对象变量.属性名的方式：



* 变量
* 变量在JavaScript中就是用一个变量名表示，变量名是大小写英文、数字、$和\_的组合，且不能用数字开头。变量名也不能是JavaScript的关键字，如if、while等。申明一个变量用var语句。
* 在JavaScript中，使用等号=对变量进行赋值。可以把任意数据类型赋值给变量，同一个变量可以反复赋值，而且可以是不同类型的变量，但是要注意**只能用var申明一次**。
* 这种变量本身类型不固定的语言称之为**动态语言**，与之对应的是静态语言。静态语言在定义变量时必须指定变量类型，如果赋值的时候类型不匹配，就会报错。例如Java是静态语言，赋值语句如下：



* 要显示变量的内容，可以用**console.log(x)**，打开Chrome的控制台就可以看到结果。
* strict模式
* JavaScript在设计之初，为了方便初学者学习，并不强制要求用var申明变量。这个设计错误带来了严重的后果：**如果一个变量没有通过var申明就被使用，那么该变量就自动被申明为全局变量**：



* 在同一个页面的不同的JavaScript文件中，如果都不用var申明，恰好都使用了变量i，将造成变量i互相影响，产生难以调试的错误结果。
* 使用var申明的变量则不是全局变量，它的范围被限制在该变量被申明的函数体内（函数的概念将稍后讲解），同名变量在不同的函数体内互不冲突。
* 为了修补JavaScript这一严重设计缺陷，ECMA在后续规范中推出了strict模式，在**strict模式下运行的JavaScript代码，强制通过var申明变量，未使用var申明变量就使用的，将导致运行错误**。
* 启用strict模式的方法是在JavaScript代码的第一行写上：



* 这是一个字符串，不支持strict模式的浏览器会把它当做一个字符串语句执行，支持strict模式的浏览器将开启strict模式运行JavaScript。