网址：

<http://javascript.ruanyifeng.com/grammar/types.html>

启动后台控制器方法有：Ctrl+Shift+J、Ctrl+Shift+I和F12三种

**语法**

1. 数据类型

1、JavaScript 语言的每一个值，都属于某一种数据类型

（1）数值（number）：整数和小数（比如1和3.14）

（2）字符串（string）：字符组成的文本（比如”Hello World”）

（3）布尔值（boolean）：true（真）和false（假）两个特定值

利用Boolean转换undefined、null、false、0、NaN、“”或‘’（空字符串）会被转为false, 空数组（[]）和空对象（{}）对应的布尔值，都是true

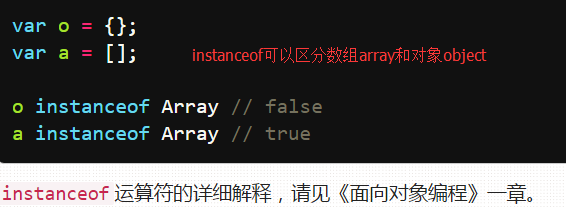
（4）undefined：表示“未定义”或不存在，即由于目前没有定义，所以此处暂时没有任何值

（5）null：表示无值，即此处的值就是“无”的状态

1. 对象（object）：各种值组成的集合
2. ES6新增的Symbol 类型的值
3. JavaScript有三种方法，可以确定一个值到底是什么类型
4. typeof运算符

返回的数据类型有number、string、boolean、function、undefined

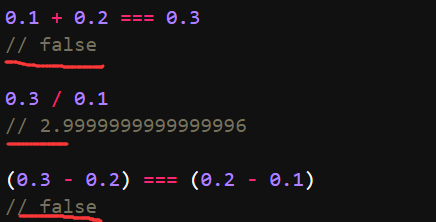
1. instanceof运算符

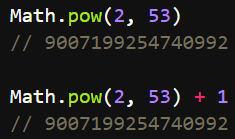
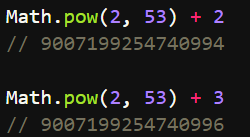
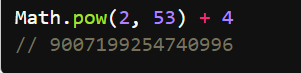


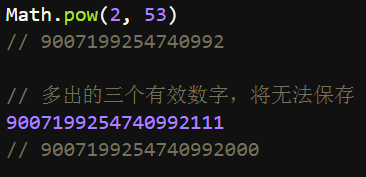
（3）Object.prototype.toString方法

二、数值

JavaScript 内部，所有数字都是以64位浮点数形式储存，即使整数也是如此。所以，1与1.0是相同的，是同一个数。也就是说JavaScript 语言的底层根本没有整数，所有数字都是小数（64位浮点数）

某些运算只有整数才能完成，此时 JavaScript 会自动把64位浮点数，转成32位整数，然后再进行运算，由于浮点数不是精确的值，所以涉及小数的比较和运算要特别小心

精度最多只能到53个二进制位，这意味着，绝对值小于2的53次方的整数，即-(253-1)到253-1，都可以精确表示，否者不能正确表示  

从上面示例可以看到，大于2的53次方以后，整数运算的结果开始出现错误。所以，大于等于2的53次方的数值，都无法保持精度

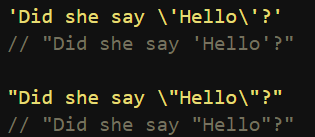
上面示例表明，大于2的53次方以后，多出来的有效数字（最后三位的111）都会无法保存，变成0

NaN不是一种独立的数据类型，而是一种特殊数值，它的数据类型依然属于Number ，0除以0也会得到NaN，NaN不等于任何值，包括它本身,NaN在布尔运算时被当作false, NaN与任何数（包括它自己）的运算，得到的都是NaN

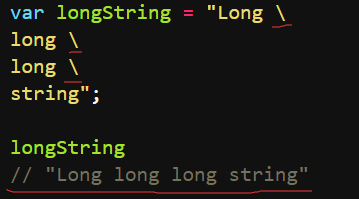
三、字符串

定义

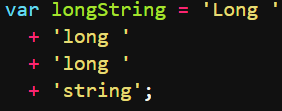
如果要在单引号字符串的内部，使用单引号（或者在双引号字符串的内部，使用双引号），就必须在内部的单引号（或者双引号）前面加上反斜杠，用来转义



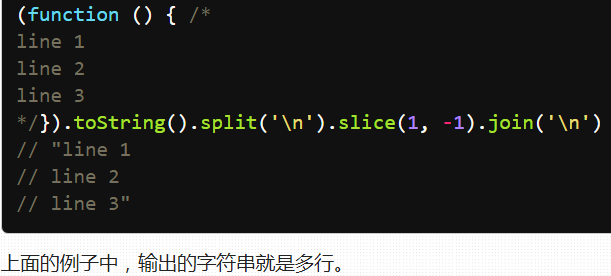
如果长字符串必须分成多行，可以在每一行的尾部使用反斜杠，输出的时候还是单行，注意：反斜杠的后面必须是换行符，而不能有其他字符（比如空格），否则会报错



连接运算符（+）可以连接多个单行字符串，将长字符串拆成多行书写，输出的时候也是单行

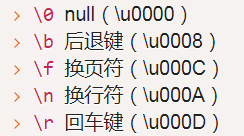
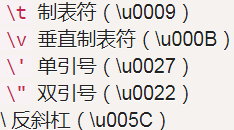


如果想输出多行字符串，有一种利用多行注释的变通方法



转义

反斜杠（\）在字符串内有特殊含义，用来表示一些特殊字符，所以又称为转义符，需要用反斜杠转义的特殊字符，主要有下面这些

反斜杠三种特殊用法：

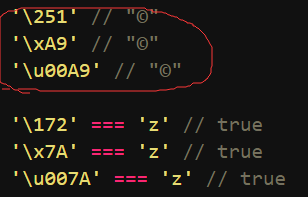
（1）\HHH

反斜杠后面紧跟三个八进制数（000到377），代表一个字符。HHH对应该字符的Unicode码点，比如\251表示版权符号。显然，这种方法只能输出256种字符

（2）\xHH

\x后面紧跟两个十六进制数（00到FF），代表一个字符。HH对应该字符的Unicode码点，比如\xA9表示版权符号。这种方法也只能输出256种字符

（3）\uXXXX

\u后面紧跟四个十六进制数（0000到FFFF），代表一个字符。HHHH对应该字符的Unicode码点，比如\u00A9表示版权符号

在非特殊字符前面使用反斜杠，则反斜杠会被省略

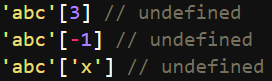
如果在字符串的正常内容之中，需要包含反斜杠，则反斜杠前面需要再加一个反斜杠，用来对自身转义



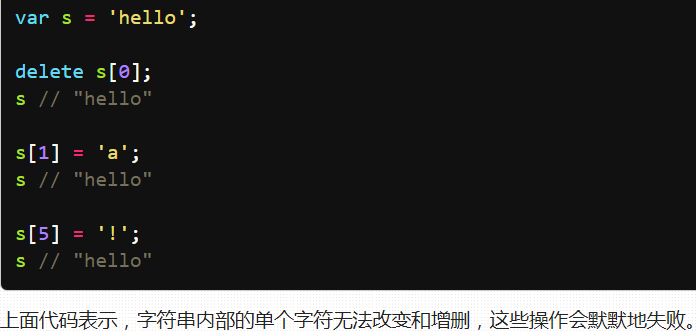
字符串与数组

字符串可以被视为字符数组，因此可以使用数组的方括号运算符，用来返回某个位置的字符（位置编号从0开始）

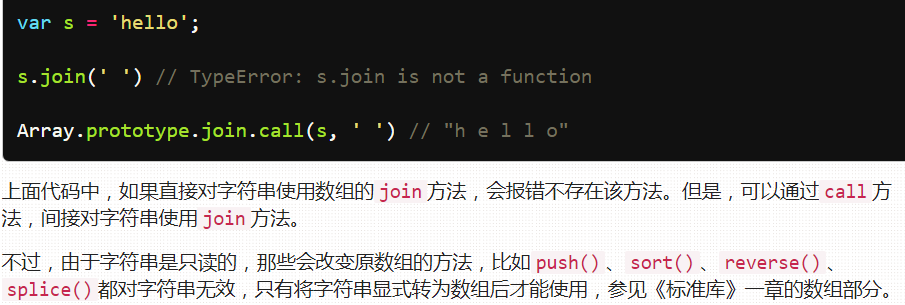
如果方括号中的数字超过字符串的长度，或者方括号中根本不是数字，则返回undefined



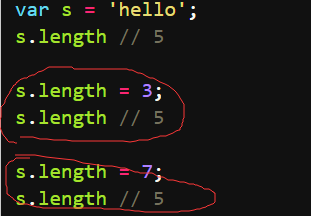
字符串与数组的相似性仅此而已。实际上，无法改变字符串之中的单个字符



字符串也无法直接使用数组的方法，必须通过call方法间接使用



length属性返回字符串的长度，该属性也是无法改变的，如果对字符串长度重新赋值也不会报错

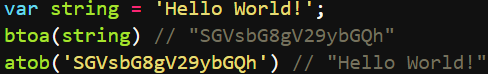


Base64转码

Base64是一种编码方法，可以将任意字符转成可打印字符。使用这种编码方法，主要不是为了加密，而是为了不出现特殊字符，简化程序的处理

JavaScript原生提供两个Base64相关方法

1. btoa()：字符串或二进制值转为Base64编码
2. atob()：Base64编码转为原来的编码

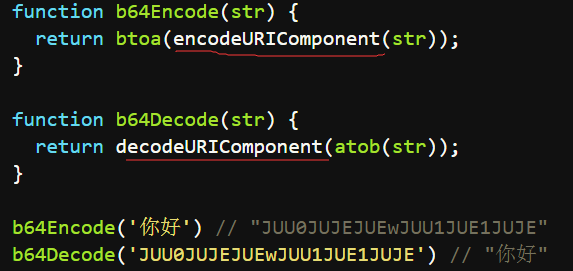


上面这两个方法不适合非ASCII码的字符，会报错

要将非ASCII码字符转为Base64编码，必须中间插入一个转码环节，再使用这两个方法

encodeURIComponent（）方法是将非ASCII码转为ASCII码

decodeURIComponent（）方法是将ASCII码转为非ASCII码



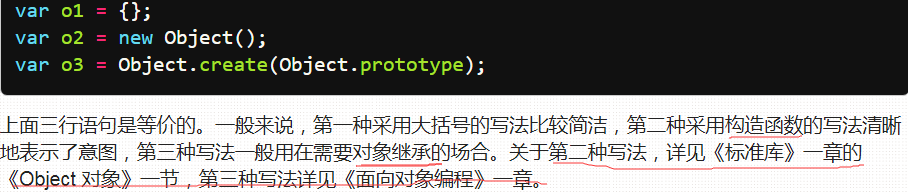
四、对象

对象概述

对象生成，说明

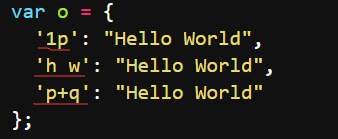
对象（object）是JavaScript的核心概念，也是最重要的数据类型。JavaScript的所有数据都可以被视为对象。所谓对象，就是一种无序的数据集合，由若干个“键值对”（key-value）构成。如果对象内部包含多个键值对，每个键值对之间用逗号分隔，最后一个有逗号和没有逗号都不报错



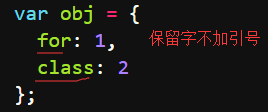
对象的生成方法，通常有三种方法。除了像上面那样直接使用大括号生成（{}），还可以用new命令生成一个Object对象的实例，或者使用Object.create方法生成

对象键名

对象的所有键名都是字符串，所以加不加引号都可以。如果键名是数值，会被自动转为字符串。但是，如果键名不符合标识名的条件（比如第一个字符为数字，或者含有空格或运算符），也不是全数字，则必须加上引号，否则会报错

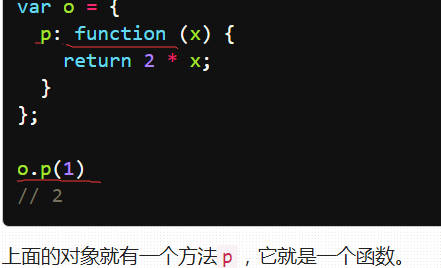


注意，JavaScript的保留字可以不加引号当作键名



对象属性

对象的每一个“键名”又称为“属性”（property），它的“键值”可以是任何数据类型。如果一个属性的值为函数，通常把这个属性称为“方法”，它可以像函数那样调用

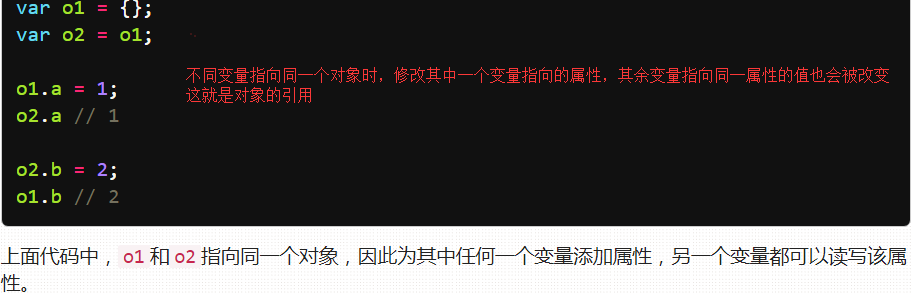


对象的属性之间用逗号分隔，最后一个属性后面可以加逗号（trailing comma），也可以不加。属性可以动态创建，不必在对象声明时就指定

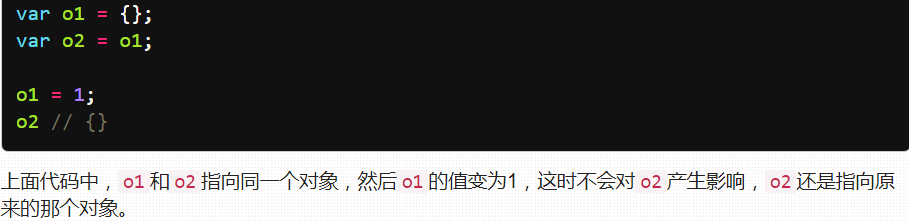


对象引用

如果不同的变量名指向同一个对象，那么它们都是这个对象的引用，也就是说指向同一个内存地址。修改其中一个变量，会影响到其他所有变量



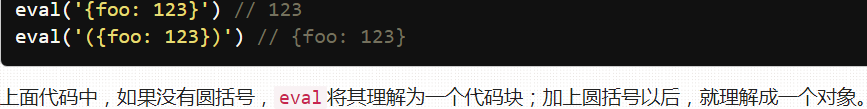
如果取消某一个变量对于原对象的引用，不会影响到另一个变量。



区分是表达式还是语句

JavaScript规定，如果行首是大括号，一律解释为语句（即代码块）。如果要解释为表达式（即对象），必须在大括号前加上圆括号，

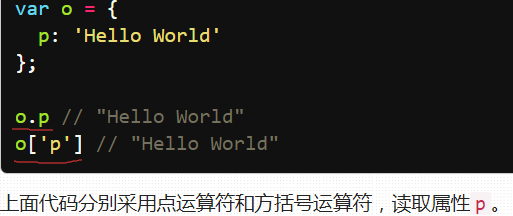
这种差异在eval语句中反映得最明显



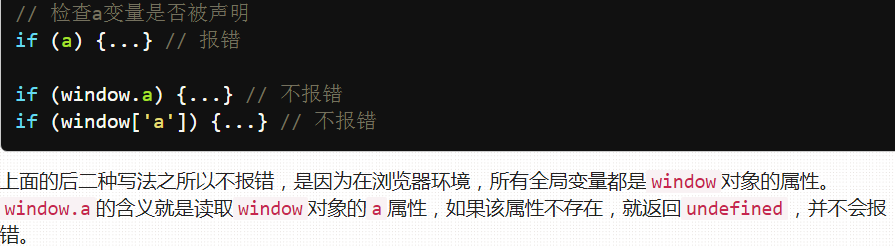
对象属性的操作

读取属性

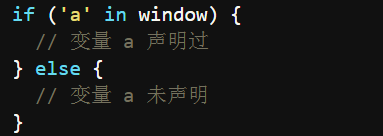
读取对象的属性，有两种方法，一种是使用点运算符，还有一种是使用方括号运算符。注意，如果使用方括号运算符，键名必须放在引号里面，否则会被当作变量处理。注意：数字键可以不加引号，因为会被当作字符串处理并且数值键名不能使用点运算符（因为会被当成小数点），只能使用方括号运算符



检查变量是否声明

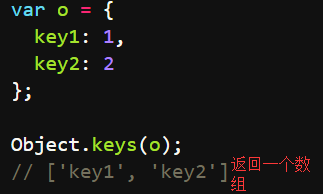
如果读取一个不存在的键（属性），会返回undefined，而不是报错。可以利用这一点，来检查一个全局变量是否被声明

后二种写法有漏洞，如果a属性是一个空字符串（或其他对应的布尔值为false的情况），则无法起到检查变量是否声明的作用。正确的做法是可以采用下面的写法



查看对象所有的属性

查看一个对象本身的所有属性，可以使用Object.keys方法，返回一个数组

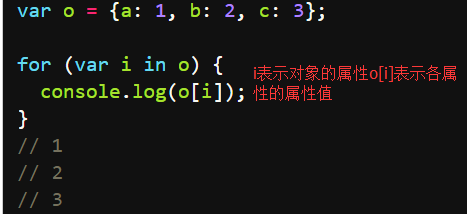


对象的delete命令

delete命令用于删除对象的属性，删除成功后返回true。delete命令删除对象的某个属性。删除后，再读取这个属性就会返回undefined，而且Object.keys方法的返回值中，此对象也不再包括该属性。delete命令不能删除var命令声明的变量，只能用来删除对象的属性，因为var声明的全局变量都是顶层对象（window对象）的属性，而且默认不得删除。

对象for in循环

for...in循环用来遍历一个对象的全部属性



for...in循环有两个使用注意点

1. 它遍历的是对象所有可遍历（enumerable）的属性，会跳过不可遍历的属性
2. 它不仅遍历对象自身的属性，还遍历继承的属性

五、数组

判断数组的方法：Array.isArray()方法可以判定一个值是否是数组，typeof运算符只能显示数组的类型是Object，Array.isArray()方法可以对数组返回true

var a = [1, 2, 3];

typeof a // "object"

Array.isArray(a) // true

Array实例的方法：

valueOf()方法返回数组本身

toString()方法返回数组的字符串形式

push()方法在数组末端添加一个或多个元素，并返回添加新元素后的数组长度。注意，该方法会改变原数组

pop()方法用于删除数组的最后一个元素，并返回该元素。注意，该方法会改变原数组

join()方法以参数作为分隔符，将所有数组成员组成一个字符串返回。如果不提供参数，默认用逗号分隔

concat()方法，用于多个数组的合并。它将新数组的成员，添加到原数组的尾部，然后返回一个新数组，原数组不变

shift()方法用于删除数组的第一个元素，并返回该元素。注意，该方法会改变原数组

unshift()方法用于在数组的第一个位置添加元素，并返回添加新元素后的数组长度。注意，该方法会改变原数组

reverse()方法，用于颠倒数组中元素的顺序，返回改变后的数组。注意，该方法将改变原数组

slice()方法用于提取原数组的一部分，返回一个新数组，原数组不变。第一个参数为起始位置（从0开始），第二个参数为终止位置（但该位置的元素本身不包括在内）。如果省略第二个参数，则一直返回到原数组的最后一个成员

splice()方法用于删除原数组的一部分成员，并可以在被删除的位置添加入新的数组成员，返回值是被删除的元素。注意，该方法会改变原数组。第一个参数是删除的起始位置，第二个参数是被删除的元素个数。如果后面还有更多的参数，则表示这些就是要被插入数组的新元素；如果只是单纯地插入元素，splice方法的第二个参数可以设为0；如果只提供第一个参数，等同于将原数组在指定位置拆分成两个数组

sort()方法对数组成员进行排序，默认是按照字典顺序排序。排序后，原数组将被改变。想让sort方法按照自定义方式排序，可以传入一个函数作为参数，表示按照自定义方法进行排序。该函数本身又接受两个参数，表示进行比较的两个元素。如果返回值大于0，表示第一个元素排在第二个元素后面；其他情况下，都是第一个元素排在第二个元素前面

[10111, 1101, 111].sort(function (a, b) {

return a - b;

})

// [111, 1101, 10111]

[

{ name: "张三", age: 30 },

{ name: "李四", age: 24 },

{ name: "王五", age: 28 }

].sort(function (o1, o2) {

return o1.age - o2.age;

})

// [

// { name: "李四", age: 24 },

// { name: "王五", age: 28 },

// { name: "张三", age: 30 }

// ]

map()方法对数组的所有成员依次调用一个函数，根据函数结果返回一个新数组，该函数接受三个参数，分别是当前成员（element）、当前位置（index）、数组本身（arr）

[1, 2, 3].map(function(elem, index, arr) {

return elem \* index;

});

// [0, 2, 6]

map方法还可以接受第二个参数，表示回调函数执行时this所指向的对象，下面代码通过map方法的第二个参数，将回调函数内部的this对象，指向arr数组

var arr = ['a', 'b', 'c'];

[1, 2].map(function(e){

return this[e];

}, arr)

// ['b', 'c']

forEach()方法与map方法很相似，也是遍历数组的所有成员，执行某种操作，但是forEach方法一般不返回值，只用来操作数据。如果需要有返回值，一般使用map方法。forEach方法的参数与map方法一致，也是一个函数，数组的所有成员会依次执行该函数。它接受三个参数，分别是当前位置的值（element）、当前位置的编号（index）和整个数组（arr）。forEach方法也可以接受第二个参数，用来绑定回调函数的this关键字

注意：forEach方法无法中断执行，总是会将所有成员遍历完。如果希望符合某种条件时，就中断遍历，要使用for循环

var out = [];

[1, 2, 3].forEach(function(elem) {

this.push(elem \* elem);

}, out);

out // [1, 4, 9]

filter()方法的参数是一个函数，所有数组成员依次执行该函数，返回结果为true的成员组成一个新数组返回。该方法不会改变原数组。filter方法还可以接受第二个参数，指定测试函数所在的上下文对象（即this对象）

some()方法是只要有一个数组成员的返回值是true，则整个some方法的返回值就是true，否则false

every()方法则是所有数组成员的返回值都是true，才返回true，否则false

注意，对于空数组，some方法返回false，every方法返回true，回调函数都不会执行

注意：some和every方法还可以接受第二个参数，用来绑定函数中的this关键字

Reduce()方法和reduceRight()方法依次处理数组的每个成员，最终累计为一个值；它们的差别是，reduce是从左到右处理（从第一个成员到最后一个成员），reduceRight则是从右到左（从最后一个成员到第一个成员）， 这两个方法的第一个参数都是一个函数。该函数接受以下四个参数

1. 累积变量，默认为数组的第一个成员（必须）
2. 当前变量，默认为数组的第二个成员（必须）
3. 当前位置（从0开始）（可选）
4. 原数组（可选）

运算符

1. 加法运算符（+）

可以完成两种运算，一种是处理算数的加法，另一种是做字符串的连接

// 加法

1 + 1 // 2

true + true // 2

1 + true // 2

// 字符串连接

'1' + '1' // "11"

'1.1' + '1.1' // "1.11.1"

算法步骤：

如果运算子是对象，先自动转成原始类型的值（即先执行该对象的valueOf方法，如果结果还不是原始类型的值，再执行toString方法；如果对象是Date实例，则先执行toString方法）。

两个运算子都是原始类型的值以后，只要有一个运算子是字符串，则两个运算子都转为字符串，执行字符串连接运算。

否则，两个运算子都转为数值，执行加法运算

'1' + {foo: 'bar'} // "1[object Object]"

'1' + 1 // "11"

'1' + true // "1true"

'1' + [1] // "11"

这种由于参数不同，而改变自身行为的现象，叫做“重载”（overload）。由于加法运算符是运行时决定到底执行那种运算，使用的时候必须很小心

'3' + 4 + 5 // "345"

3 + 4 + '5' // "75"

运算结果由于字符串的位置不同而不同