# Math

\* - Math可以用来做数学运算相关的操作

\* - Math并不是一个构造函数，我们也不需要去创建一个Math类型

\* - Math是一个工具类，它里面封装了一些数学运算相关的常量和方法

## 属性

Math中的常量（常量都大写），也叫Math对象的属性

\* Math.PI 圆周率

|  |  |
| --- | --- |
| [E](mk:@MSITStore:E:\H5\W3School.chm::/www.w3school.com.cn/jsref/jsref_e.asp) | 返回算术常量 e，即自然对数的底数（约等于2.718）。 |
| [LN2](mk:@MSITStore:E:\H5\W3School.chm::/www.w3school.com.cn/jsref/jsref_ln2.asp) | 返回 2 的自然对数（约等于0.693）。 |
| [LN10](mk:@MSITStore:E:\H5\W3School.chm::/www.w3school.com.cn/jsref/jsref_ln10.asp) | 返回 10 的自然对数（约等于2.302）。 |
| [LOG2E](mk:@MSITStore:E:\H5\W3School.chm::/www.w3school.com.cn/jsref/jsref_log2e.asp) | 返回以 2 为底的 e 的对数（约等于 1.414）。 |
| [LOG10E](mk:@MSITStore:E:\H5\W3School.chm::/www.w3school.com.cn/jsref/jsref_log10e.asp) | 返回以 10 为底的 e 的对数（约等于0.434）。 |
| [PI](mk:@MSITStore:E:\H5\W3School.chm::/www.w3school.com.cn/jsref/jsref_pi.asp) | 返回圆周率（约等于3.14159）。 |
| [SQRT1\_2](mk:@MSITStore:E:\H5\W3School.chm::/www.w3school.com.cn/jsref/jsref_sqrt1_2.asp) | 返回返回 2 的平方根的倒数（约等于 0.707）。 |
| [SQRT2](mk:@MSITStore:E:\H5\W3School.chm::/www.w3school.com.cn/jsref/jsref_sqrt2.asp) | 返回 2 的平方根（约等于 1.414）。 |

## 方法

### abs()

\* - 可以用来获取一个数的绝对值

console.log(Math.abs(-10)); //10

### ceil()

\* - 可以对一个数进行向上取整

\* - ceil() 方法执行的是向上取整计算，它返回的是大于或等于函数参数，并且与之最接近的整数。

### floor()

\* - 可以对一个数进行向下取整

\* - floor() 方法执行的是向下取整计算，它返回的是小于或等于函数参数，并且与之最接近的整数。

### round()

\* - 对一个数进行四舍五入取整

\* -对于 0.5，该方法将进行上舍入。例如，3.5 将舍入为 4，而 -3.5 将舍入为 -3。



### random();

\* - 可以生产一个0-1之间的随机数

\* Math.random();

\* 生成 0 - x 之间的随机数

\* Math.round(Math.random()\*x)

\* 生产一个 x - y之间的随机数

\* Math.round(Math.random()\*(y-x) + x)

### max()

\* - 可以从多个值中获取到最大值

### min()

\* - 可以从多个值中获取到最小值

### pow(x,y)

\* - 获取x的y次幂

### sqrt(x)

\* - 求一个数的平方根

# 包装类

通过包装类可以将一个基本数据类型包装为一个对象

\* - 在JS中一共提供了三个包装类：

\* String()

\* - 将一个基本数据类型包装为一个String对象

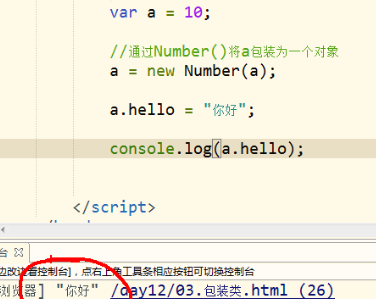
\* Boolean()

\* - 将一个基本数据类型包装为一个Boolean对象

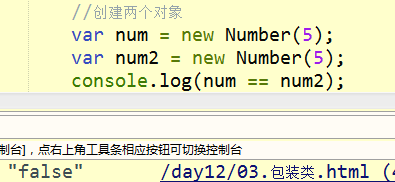
\* Number()

\* - 将一个基本数据类型保存一个Number对象

\* - 通过这三个包装类，包装过后基本数据类型都会变成对象，会具有对象的一切的功能 （其实就是调了一个构造函数）

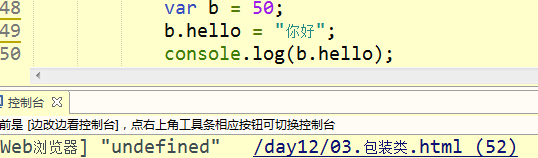


在开发中，不要使用包装类来创建基本数据类型的对象：



（两个对象比的是地址，所以返回false，而且如果输出num和nmu2，会得到”5”，不知道到底是什么类型的）

当我们去使用一个基本数据类型的属性，或是调用一个基本数据类型的方法时，JS解析器会临时将基本数据类型包装为一个对象，然后去操作这个对象的属性或方法，操作完成以后将对象立即销毁



上图中创建了一个Number类型的b，当执行到49行时，将b包装为一个对象，创建一个属性为hello并赋值为”你好”，然后销毁，当执行到50行时，又将b包装为一个对象，然后调这个对象的hello属性，由于上面的已经销毁，所以返回undefined

这也就解释了为什么nulll和undefined调不了toString方法，会报错，因为null和undefined没有包装类，转换不了对象

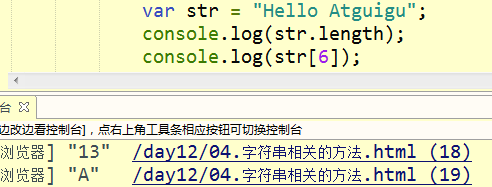
包装类的作用是给JS解析器使用的

所以Number、Boolean、String中的属性、方法可以直接通过一个基本数据类型去调

# 字符串的相关方法

字符串的所有方法都不会对原字符串产生影响

在解析器的底层，保存字符串时，实际上是以字符数组的形式保存的，所以字符串的操作方法和数组都是非常相似的



## charAt()

\* - 可以根据索引来获取指定位置的字符

str=”Hello World”;

str.charAt(4)和str[4]是一样的

## charCodeAt()

\* - 可以根据索引来获取指定位置的字符的字符编码

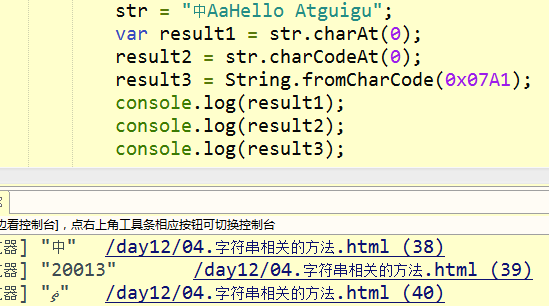
## fromCharCode()

需要通过构造函数去调用

String.fromCharCode();

\* - 可以根据字符编码来获取字符

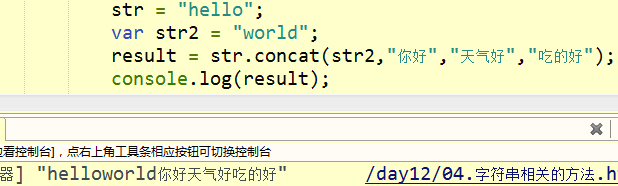
默认是使用十进制编码，如果想使用十六进制，前面加0x



## concat()

\* - 可以用来连接两个或多个字符串

\* - 作用和 + 一样



## indexOf()

\* - 检索指定字符串在当前字符串中的位置

\* - 参数：

\* 第一个参数，要查询的字符串

\* 第二个参数，查找开始的位置，合法值从0到length-1，如果不写，则从第一个开始查找

\* - 返回值：

\* 默认会返回字符串第一次出现的位置的索引，以后的不会返回

\* 如果没有找到该内容，则返回-1

## lastIndexOf()

\* - 检索指定字符串在当前字符串中的位置

\* - 该方法和indexOf()一样，不同的是indexOf()是从前往后找

\* 而lastIndexOf()是从后往前找

## slice()

\* - 可以用来截取一个字符中的内容

\* - 参数：

\* 第一个参数，截取开始位置的索引（包括开始位置）

\* 第二个参数，截取结束位置的索引（不包括结束位置）

\* - 可以接收一个负值作为参数，如果是负值，则从后往前数

\* - 也可以省略第二个参数，如果省略则一直截取到最后

当第一个参数在字符串中的位置比第二个参数的位置靠前，则正常输出，否则输出空串

## substring()

\* - 用来截取字符串的

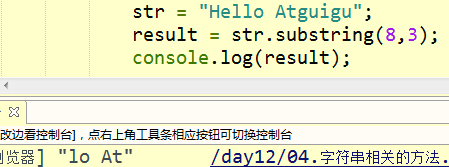
\* - 参数：

\* 第一个参数，截取开始位置的索引（包括开始位置）

\* 第二个参数，截取结束位置的索引（不包括结束位置）

\* - 该方法中如果第二个参数小于第一个，则会调整参数的位置

\* - 该方法不接收负值，如果第一个参数传递了一个负值，第二个非负值，则第一个默认会设置为0；第一个非负值，第二个设置为负值，则第二个小于第一个，调整位置，和第一种情况一样处理；两个都是负值，则返回空串。



## substr()

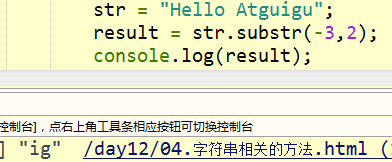
\* - 用来截取字符串

\* - 参数：

\* 第一个参数，截取开始位置的索引

\* - 可以是负值，如果是负值从后往前数，但截取还是从前往后

\* 第二个参数，截取的个数

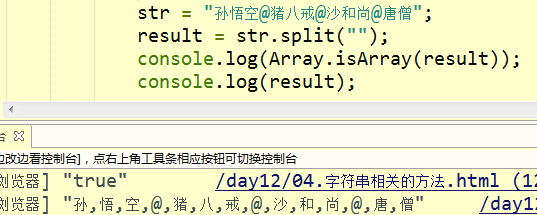


## split()

\* - 可以将一个字符串拆分为一个数组

\* - 它需要一个字符串作为参数，将会根据该字符串对字符串进行拆分，如果传一个空串，则每个字符串都拆分

\* 并且将拆分的后的值保存到一个数组中返回



## toLowerCase()

\* - 将一个字符串转换为小写

## toUpperCase()

\* - 将一个字符串转换为大写

## 练习

将一个字符串以相反的顺序输出。例如：abc ---> cba

//var str = "abc";

//str.reverse();由于reverse是数组的方法，在字符串中不可用

//将字符串拆分为一个数组

//var arr = str.split("");

//将数组进行反转

//arr.reverse();

//将数组转换为字符串

//str = arr.join("");

结果：str = str.split("").reverse().join("");

console.log(str);

# 正则表达式

## 正则表达式

\* - 正则表达式用来定义一些规则，计算机可以根据这些规则来判断一个字符串是否符合规则，或者从一个字符串中将符合规则的内容提取出来

创建一个正则表达式的对象

//语法：var 变量 = new RegExp("正则表达式","匹配模式");

//在正则的构造函数中，可以传递一个匹配模式的字符串作为第二个参数g 全局匹配模式 i 忽略大小写匹配

匹配模式可以不写，也可以同时写多个，并且顺序不限

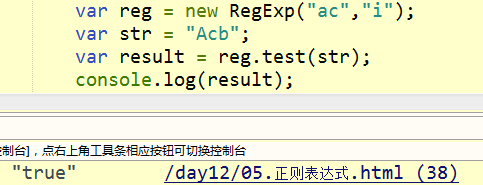
使用typeof来检查正则，会返回object

正则表达式中有一个方法

### test()

\* 他可以用来检查一个字符串是否符合正则表达式的要求

\* 如果符合则返回true，否则返回false



创建正则表达式方式二（字面量）：

\* var 变量 = /正则表达式/匹配模式;

## 方括号

//创建一个正则表达式，检查一个字符串中是否含有a或b或c

// | 表示或者

reg = /a|b|c/;

//创建一个正则表达式，表示任意的小写字母

reg = /[a-z]/;

// [] 中括号中的内容也表示或

// [a-z] 任意的小写字母

// [A-Z] 任意的 大写字母

// [A-z] 任意字母

// [0-9] 任意数字

// [^ ] 表示除了括号中的内容

//除了小写字母和数字

reg = /[^a-z0-9]/;

//检查一个字符串中是否含有abc 或 adc 或aec

// reg = /abc|adc|aec/;

reg = /a[bde]c/;

## 量词

\* - 量词用来表示一个内容出现的次数，只对前面一个的起作用，如果希望对前面几个起作用，则用（）把前面的括起来

\* {n} 正好出现n次

\* {m,n} 出现m到n次

\* {n,} 至少n次

\* + 至少出现1次，相当于{1,}

\* \* 0个或多个，相当于{0,}

\* ? 0个或1个，相当于{0,1}

\* ^ 表示开头

\* $ 表示结尾

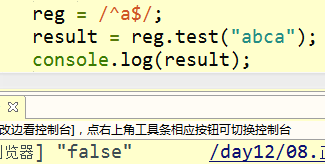
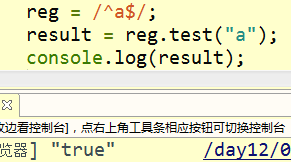
//创建一个正则表达式，检查一个字符串是否以a开头

reg = /^a/;

//创建一个正则表达式，检查一个字符串是否以a结尾

reg = /a$/;

//使用^开头 $结尾的正则表达式，必须字符串和正则表达式完全匹配才会返回true



^a$表示以a开头并且以那个a结尾，只能是a；

### 练习

检查一个字符串是否是一个有效的手机号



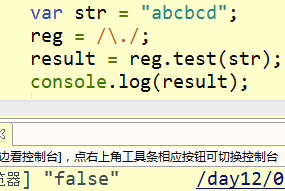
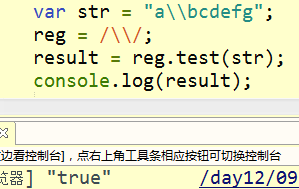
## 元字符

**.** 在正则中代表任意字符

在正则表达式中，可以使用 \ 作为转义字符

\* \**.** 来表示.

\* \\ 表示 \

当使用构造函数来创建正则表达式时，由于它的正则是以字符串的形式传递的，而\在字符串中也是转义字符串,所以这里需要使用\\来表示一个\

reg = new RegExp("\\."); == reg=/\./;

reg=new RegExp("\\\\"); == reg=/\\/;

\*\w

\* - 相当于[A-z0-9\_]

\* - 任意字母数字下划线

\* \W

\* - 相当于[^A-z0-9\_]

\* - 除了字母数字下划线

\* \d

\* - 任意数字

\* \D

\* - 除了数字

\* \s

\* - 空格

\* \S

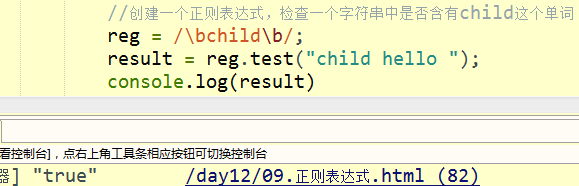
\* - 除了空格

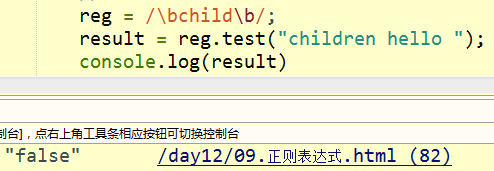
\* \b

\* - 单词边界

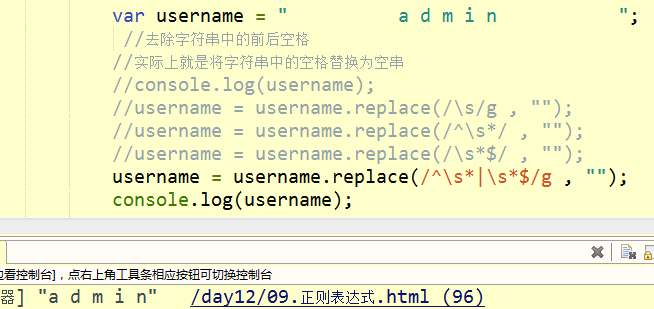
\* \B

\* - 除了单词边界





### 练习



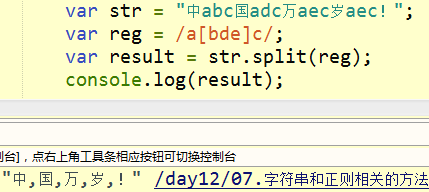
## 支持正则表达式的String对象的方法

### split()

\* - 可以将一个字符串拆分为一个数组

\* - 这个方法中可以传递一个正则表达式作为参数，这样将会根据正则表达式去拆分字符串

\* - split()拆分字符串，无论是否指定全局匹配，都会全都拆



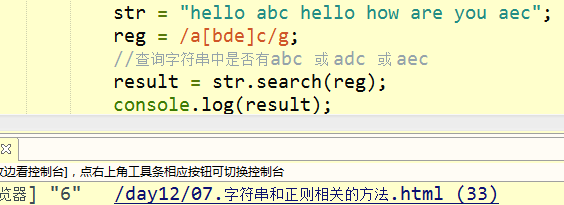
### search()

\* - 可以用来搜索一个字符串中是否含有指定内容

\* - 如果查询到了，会返回它第一次出现的索引，如果没找到则返回-1

\* - 也可以接收一个正则表达式作为参数，将会根据正则表达式去字符串中查询内容

\* - search()只会匹配第一个，即使加了全局匹配



### match()

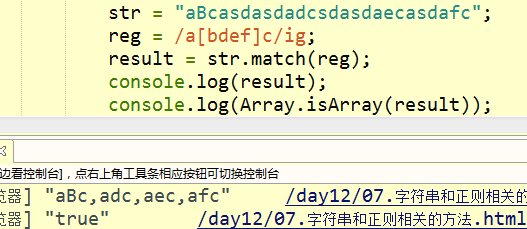
\* - 根据正则表达式将字符串中符合要求的内容提取出来

\* - 默认情况match()只会匹配到第一个符合条件的内容，

\* 如果需要匹配所有的内容，则需要设置全局匹配模式

\* - match()方法会将匹配到的内容封装到一个数组中返回，

\* 即使只匹配到了一个，也会保存到数组中



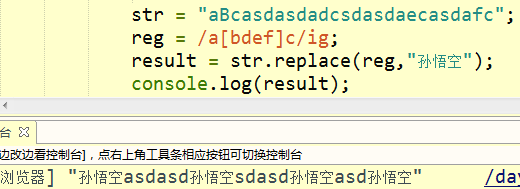
### replace()

\* - 可以将字符串的指定内容，替换为新的内容

\* - 参数：

\* 第一个，被替换的内容,可以是一个正则表达式

\* 第二个，新的内容



## 邮件的正则表达式

abc .hello @ abc .com .cn

开头 字母 数字 下划线 一位以上 ^\w+

\.任意字母数字下划线 一位以上(有没有都行) (\.\w+)\*

@ @

字母数字一位以上 [A-z0-9]+

\.字母（2-5位） \.[A-z]{2,5}

\.字母（2-5位）结尾 \.[A-z]{2,5}$

最后两个相同可以合并为一个：(\.[A-z]{2,5}){1,2}$

结果var emailReg = /^\w+(\.\w+)\*@[A-z0-9]+(\.[A-z]{2,5}){1,2}$/;

