# 正则表达式(RegExp)

**Regular Expression**

用于匹配字符串中符合自定义的正则的表达式，在js中正则表达式是一个对象

## 正则表达式的创建

1. 使用一个正则表达式字面量，其由包含在斜杠之间的模式组成

例：var re = /ab+c/;//re就是一个正则表达式

1. 调用RegExp对象的构造函数

例：var re = new RegExp("ab+c");

## 元字符

|  |  |
| --- | --- |
| **元字符** | **描述** |
| \ | 转义字符 声明在\后面的字符为特殊字符 |
| ^ | 匹配字符串行的开头，若多行则匹配每一行开头,**加在目标字符之前**  例如，/^A/ 并不会匹配 "an A" 中的 'A'，但是会匹配 "An E" 中的 'A'。 |
| $ | 匹配字符串行的末尾，若多行则匹配每一行末尾  匹配字符串行的开头，若多行则匹配每一行开头  例如，/t$/ 并不会匹配 "eater" 中的 't'，但是会匹配 "eat" 中的 't'。 |
| \* | 匹配前一个表达式 0 次或多次。  例：/bo\*/ 会匹配 "A ghost boooooed" 中的 'booooo' 和 "A bird warbled" 中的 'b' //如果字符串没有b后紧接o |
| + | 匹配前一个表达式1次或多次。 |
| ? | 匹配前一个表达式0次或1次。  例如，/e?le?/ 匹配 "angele" 中的 'ele'。  对每一个?都执行0或1次，整个正则只执行一次 |
| . | 查找单个字符，除了换行和行结束符  例：/.n/匹配在n之前的字符和n |
| () | 捕获括号，匹配并记住括号内的内容 |
| (?:x) | 非捕获括号，匹配但不记住括号内的内容 |
| | | 或 例：x|y 匹配x或y |
| - |  |
| {n} | n为正整数，匹配出现了n次的字符，若多余n次则匹配最先出现的n次字符  例：/a{2}/ 不会匹配“candy”中的'a',但是会匹配“caandy”中所有的 a，以及“caaandy”中的前两个'a'。 |
| \w | 匹配基本拉丁字母的任何数字字母字符 相当于 [A-Za-z0-9\_]  //因此法语等语言的如É字符不会被匹配 |
| \W | 匹配任何非数字字母字符 相当于 [^A-Za-z0-9\_] |
| \d | 匹配数字 |
| \D | 匹配非数字字符 |
| \s | 匹配空白字符 |
| \S | 匹配非空白字符 |
| \b | 匹配单词边界 |
| \B | 匹配非单词边界 |
| \0 | 匹配 NUL字符 |
| \n | 匹配换行符 |
| \f | 匹配换页符 |
| \r | 匹配回车符 |
| \t | 匹配制表符 |
| \v | 匹配垂直制表符 |
| \xxx | 匹配以八进制数 xxxx 规定的字符 |
| \xdd | 匹配以十六进制数 dd 规定的字符 |
| \uxxxx | 匹配以十六进制 xxxx规定的 Unicode 字符 |

### 插入语

## 正则表达式的使用方法

在 JavaScript中，正则表达式也是对象。这些模式被用于 **RegExp 的 exec 和 test 方法, 以及 String 的 match、matchAll、replace、search 和 split 方法。**

区别：RegExp为正则表达式对象，String为字符串对象；

例：var patt = /e/;//patt为一个正则表达式对象

patt.test("The best things in life are free!");//在对象的方法中添加字符串，此例为检测字符串是否含有e字符。

也可以直接使用/e/.test("The best things in life are free!")；

**若使用字符串方法如search则为**

var str=’ The best things in life are free!’;

var patt=/e/;

str.search(patt);

### test()

test() 方法用于检测一个字符串是否匹配某个模式，如果字符串中含有匹配的文本，则返回 true，否则返回 false。

### exec()

exec() 方法用于检索字符串中的正则表达式的匹配。

该函数**返回一个数组，其中存放匹配的结果。**如果未找到匹配，则返回值为 null。

### search()

一个在字符串中测试匹配的String方法，它**返回匹配到的位置索引**，或者在失败时返回-1

### march()

执行查找匹配的String方法，它返回一个数组，在未匹配到时会返回 null。

### marchAll()

在字符串中执行查找所有匹配的String方法，它返回一个迭代器（iterator）。

迭代器中每个对象包含了执行一个方法得到的结果

例：const regexp = /t(e)(st(\d?))/g;

const str = 'test1test2';

const array = [...str.matchAll(regexp)];

console.log(array[0]);// expected output: Array ["test1", "e", "st1", "1"]

## 高级搜索标志

正则表达式有六个可选参数 (flags) 允许全局和不分大小写搜索等。这些参数既可以单独使用也能以任意顺序一起使用, 并且被包含在正则表达式实例中。

语法：var re = /pattern/flags;或者var re = new RegExp("pattern", "flags");

例：var re = /\w+\s/g;

| **标志** | **描述** |
| --- | --- |
| g | 全局搜索。 |
| i | 不区分大小写搜索。 |
| m | 多行搜索。 |
| s | 允许 . 匹配换行符。 |
| u | 使用unicode码的模式进行匹配。 |
| y | 执行“粘性(sticky)”搜索,匹配从目标字符串的当前位置开始。 |