正则：

1. 字符^表示字符串开始位置匹配
2. 字符$表示字符串结束位置匹配
3. 字符\*表示字符串任意位置匹配
4. 字符i表示字符串不区分大小写，默认是大小写匹配的
5. 字符g表示字符串全局匹配
6. 字符m：multiple lines表示多行搜索

Demo1：匹配链接中/project

/\/project\/[a-z]+\//g

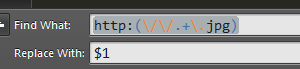
慕课网学习例子和知识点整理：

1. sublime中Ctrl+h是替换功能
2. \bis\b匹配一个is整个单词,\b表示单词边界

3.

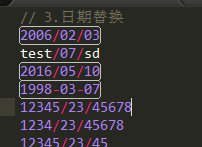


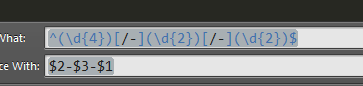
* http:\/\/.+\.jpg，其中的点表示任意字符，+在这里表示必须有一个字符。后面的.jpg中的.需用反斜杠转义()，http://里的反斜杠也需要转义。
* http:(\/\/.+\.jpg)，找到对应字符以后，还要截取字符串协议头后面的部分。用括号对其进行分组，$1表示内容

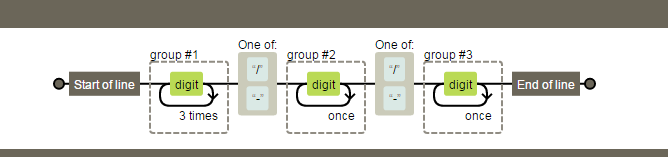


1. 将图中的yyyy-(/)mm-(/)-dd格式提取出来，变成mm-dd-yyyy格式

* ^(\d{4})[/-](\d{2})[/-](\d{2})$
* \d表示数字，{}表示量词，或者的含义用中括号[]表示，^表示开头，$表示结尾







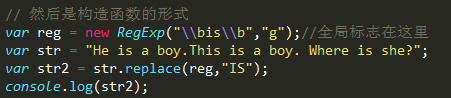
详细：

先是字面量的形式：

1. 打印的结果是 He IS a boy.This is a boy. Where is she?，但是后面两个is没有替换。
2. 这样就可以全部替换啦



这是构造函数的形式



* 元字符

1. 原义文本字符和元字符（在正则表达式有特殊含义的非字母字符）

* [ ]构建一个简单的类
* ^创建反向类，[^abc]表示不是a或b或c的内容
* 范围类：[a-z]表示a到z之间的任意字符，这是一个闭区间，[a-zA-Z],大小写字母都可以

如果匹配横线，这样写就可以



* 预定义类：

.表示除了回车换行外的任意字符[^\r\n]

\d是数字

\D是非数字

\s空白符

\S非空白符

\w单词字符（字母，下划线）

\W非单词字符

* 边界：^开始，$结束,\b单词边界，\B非单词边界
* 量词：

?:出现一次或0次

+：出现至少一次

\*：任意次

{n}:出现n次

{n,m}:出现n-m次

{n,}:至少出现n次

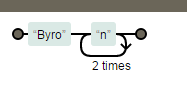
* 贪婪模式和非贪婪模式
* 这里会匹配6个而不是3个

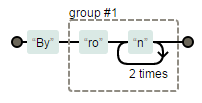


* 非贪婪模式，加个问号就可以



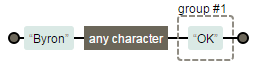
* 分组：

Byron{3} =》，说明量词是作用于紧挨着的字母，如何实现单词匹配呢？需要分组

By(ron{3}) =》

* 或：|
* 反向引用：

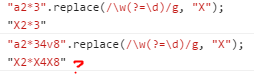


* 忽略分组(?:Byron).(OK) =》
* 前瞻：
* 正则表达式从文本头部向尾部开始解析，文本尾部方向为前
* 前瞻就是在正则表达式匹配到规则的时候，向前检查是否符合断言

后顾/后瞻方向相反（js不支持）

* 符合和不符合特定断言称为肯定/正向匹配和否定/负向匹配
* 正向前瞻：exp(?=assert)

负向前瞻：exp(?!assert)



* 对象属性：

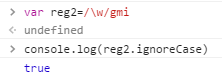
lastIndex:是当前匹配内容的最后一个字符的下一个位置（即下一次开始检索的位置）

source:正则表达式的文本字符串

global:是否全文搜素，默认false

ignoreCase:是否大小写敏感,默认false

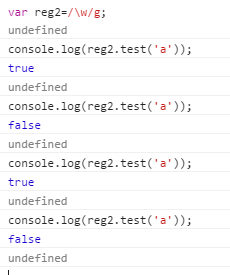
multiline:多行搜素，默认false



都是只读

* 正则表达式的方法：
* RegExp.prototype.test(str):用于测试字符串参数中是否存在匹配正则表达式模式的字符串，存在返回true，不存在返回false；

下面这张图中只有奇数次是好使的，其实是reg2的lastIndex属性在作怪，所以说尽量不要用/g,



* RegExp.protype.exec(str):使用正则表达式模式对字符执行搜索，并将更新全局RegExp对象的属性以返回匹配结果，没有匹配的文本返回null，苟泽返回一个结果数组

Index声明匹配文本的第一个字符的位置

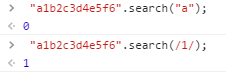
Input存放被检索的字符串string



非全局下lastIndex不生效，第一个匹配字符是0位置，[与正则表达式相匹配的文本，RegExpObject的第一个子表达式相匹配的文本（有的话）]



* 字符串中与正则表达式相关的方法
* String.prototype.search(reg):用于检索字符串中指定的子字符串，或检索与正则表达式相匹配的子字符串，有则返回第一个匹配结果，查找不到返回-1。（不执行全局匹配，并且总是从字符串开始检索）

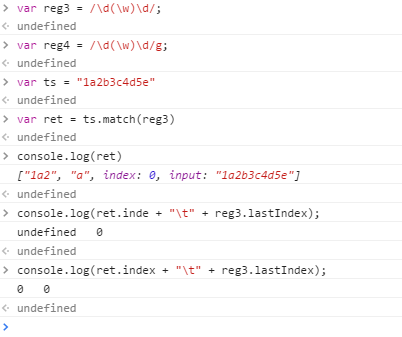


* String.prototype.match(reg):将检索字符串，以找到一个或多个与regexp匹配的文本。

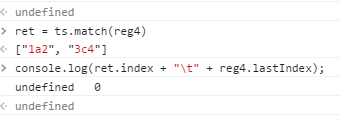
Regexp是否有g标志对结果影响很大。

（用起来和exec好像）

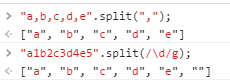
非全局：



全局：好像比exec弱一些



* String.prototype.spilt(reg):把字符串分割为字符数组,里面用逗号分隔也是转换成正则表达式。



* String.prototype.replace：