各种编程语言都支持正则表达式，正则表达式能帮我们快速实现很多功能，所以了解使用正则表达式的方法非常重要。本文总结了JavaScript中使用正则表达式的方法，对其用法及解决何种需求做了介绍，辅以具体实例讲解了我们该怎么选择具体的方法，以此来展现正则表达式给我们带来的便利。

1. JS里涉及正则表达式的方法

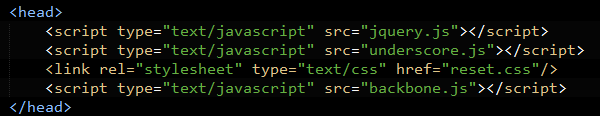
        在JS里，涉及正则表达式的方法很多，列表如下：

| **方法名** | **用法** | **解决何种需求** | **示例（=>表示执行结果）** |
| --- | --- | --- | --- |
| RegExp.prototype.exec(string) | 如果匹配失败，则返回null；   若正则表达式有global修饰符，如果匹配成功，则返回一个类数组，其下标为0的元素为符合规则的子字符串，后边紧跟着是捕获分组。数组同时定义了属性index(匹配的子字符串在整个正则表达式中的位置)，input(参数string的字符串格式)，length(类数组长度)。运行此方法可向后继续匹配符合规则的子字符串。  若正则表达式没有global修饰符，则始终返回第一个匹配的子字符串，捕获分组及相关属性。 | 常用于逐次获取子匹配项及分组匹配项及其相关属性 | var s='t1t2t3';  var r1=/t(\d)/; var r2=/t(\d)/g;  r1.exec(s); =>["t1", "1"] r1.exec(s) =>["t1", "1"] r2.exec(s) =>["t1", "1"] r2.exec(s) =>["t2", "2"] |
| RegExp.prototype.test(string) | 调用RegExp.prototype.exec(string)，若为null返回false,否则返回true. | 只需要知道匹配是否成功 | var s='t1t2t3';  var r1=/t(\d)/;  r1.test(s) =>true |
| String.prototype.match (regexp) | 若正则表达式有global标识符，则返回一个数组，该数组包含所有符合规则的子字符串。  若正则表达式没有global标识符，则返回RegExp.prototype.exec(string)的结果。 | 只获取所有的匹配的子字符串，而不需要获取捕获分组的情况下使用 | var s='t1t2t3'; var r1=/t(\d)/; var r2=/t(\d)/g; s.match(r1) =>["t1", "1"] s.match(r2) =>["t1", "t2", "t3"] |
| String.prototype.search (regexp) | 返回匹配字符串在字符串中的位置。 | 获取匹配的位置时使用 | var s='t1t2t3';  *var r1=/t(\d)/;*  *var r2=/t(\d)/g; s.search(r1) =>0 s.search(r2) =>0* |
| String.prototype.replace (searchValue, replaceValue) | searchValue可以是一个正则表达式。  若有global修饰符，则替换所有符合规则的子字符串；  若无global修饰符，则替换第一个匹配的子字符串。 | 替换匹配的字符串时使用 | 't1t2t3'.replace(/t(\d)/,'t') =>"tt2t3" 't1t2t3'.replace(/t(\d)/g,'t') =>"ttt" |
| String.prototype.split (separator, limit) | separator 可以是一个正则表达式，这个时候匹配结果作为分隔符，返回被分隔开的子字符串组成的数组。 | 分隔字符串时使用 | 'a||b||c'.split(/\|\|/) =>["a", "b", "c"] |

2. 实例说明

实例1：合并JS文件

        需求：用NodeJS实现多个JS文件的合并，生成的JS文件以多个JS文件的名称拼合而成。比如以下HTML片段：



        合并后为<script type="text/javascript" src="jqueryunderscorebackbone.js"></script>

        此正则表达式需要匹配所有script标签里的src的内容，故正则表达式必定带有global修饰符。要获取匹配结果，我们只能选用RegExp.prototype.exec(string)或者String.prototype.match (regexp)方法。若我们选择match方法，只能获取匹配的子字符串，不能获取捕获括号里的内容。比如我们尝试以以下正则表达式去匹配：

var reg=/(.+?)(?=\.js)/g;

        结果为：

["  <script type="text/javascript" src="jquery", "  <script type="text/javascript" src="underscore", "  <script type="text/javascript" src="backbone"]

        并不能取得文件名。

        因为JS不支持逆向环视(负向预匹配)，所以我们不能确定文件名前边的分号，也就没有办法仅仅去匹配文件名了。

        若我们选择exec方法呢？因为exec支持逐次读取结果，并记录捕获分组结果，所以我们可以使用以下正则表达式：

var reg=/<(script).+?src="(.+?)\.js".\*?><\/\1>/g;

        接下来我们可以逐次读取每个script里的链接：

var tempArr=[],  
res='';

while(tempArr=reg.exec(html)){  
    res+=tempArr[2];  
}

        在保证匹配script标签的情况下，它第二个捕获分组里保存了JS的文件名，我们就可以拼接成新的文件名啦！

        我们的项目在NodeJS里，在HTML里模拟效果代码如下：

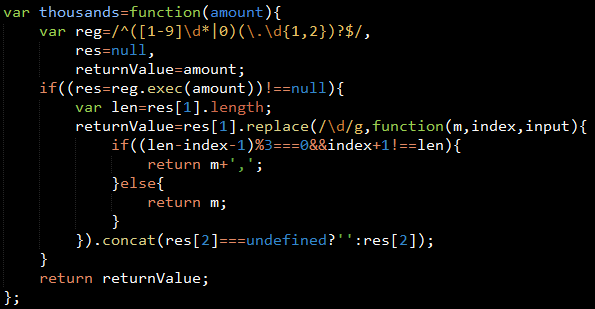


实例2：格式化数字为千分位格式

        需求：信用卡还款首页的金额格式显示为千分位格式（针对整数部分），形如：100,000.00

        这里可以用到的方法很多，如果我们用正则表达式来处理该选用哪种方法呢？因为这里不需要用到global修饰符，故mathc和exec方法都一样。我们可以用以下正则匹配到整数部分和小数部分，然后对整数部分进行处理。

        如何将整数部分格式化为千分位格式呢，这里我们可以用到replace方法，replace方法的第二个参数为函数时会将子匹配项，捕获分组匹配项，匹配位置偏移量及整个需要匹配的字符串作为实参执行，我们可以在里面做需要加逗号的逻辑处理，代码如下：



        通过replace方法，我们成功地将逗号给加入了字符串中。

3. 总结

        正则表达式能帮我们实现很多功能，在JS中，我们要了解能使用正则表达式的方法，合理选择方法来满足项目的需求。