正则表达式：一个具有特殊字符的字符串

正则的作用：

1、表单验证（检测用户名、密码等是否符合规范）

2、查找字符串

3、查找替换并替换

正则的学习：

定界符 /正则规则/

原子

```
1、每一个字符都称为一个原子（数字、、字母、下划线以及任何符号，包括空格）
2、每一个[]代表一个原子，中括号内为可选字符
[]号内的简写：
 [0-9]代表0-9的任意一个数
 [0-9A-Za-z_]代表数字、字母、下划线中的任意一个字符
 注意：在[]内出现^，代表除了[]内以外的字符

3、{}代表重复匹配前面的原子
 {m}重复匹配前面的原子m次
  {m,n}重复匹配前面的原子至少m次，至多n次
  {m,}重复匹配前面的原子至少m次

特殊的原子：
  \d 代表0-9的任意一个数 相当于[0-9]
  \D 代表除了0-9以外的任意字符 相当于[^0-9]

  \w 代表数字、字母、下划线中的任意一个字符 相当于[0-9A-Za-z_]
  \W 代表除了数字、字母、下划线以外的任意一个字符 相当于[0-9A-Za-z_]

  \s 代表所有的空白符（如：空格、换行、制表符）
  \S 代表除了空白符以外的任意字符
```

元字符

```
. 代表任意一个字符
```

```
* 代表重复匹配前面的原子任意次 相当于{0,}
? 代表重复匹配前面的原子至少0次，至多1次 相当于{0,1}

+ 代表重复匹配前面的原子至少一次 相当于{1,}
```

模式修正符

```
i 不区分大小写
g 全局匹配
m 识别换行符
注意：模式修正符放置在定界符之外
```

定义正则表达式

```
// 方式一:通过定界符定义（推荐使用）
var pattern = /aaa/;

// 方式二:通过对象原型定义
var pattern = new RegExp('aaa');
```

严格模式

```
^必须以指定的字符开头 $必须以指定的字符结尾

必须完全符合正则表达式规则，才会验证通过
var pattern = /^[0-9A-Za-z_]{6}$/
```

贪婪模式

```
var pattern = /<li>.*<\/li>/;
var str = '<li>百事可乐</li><li>珍珠奶茶</li>';
var res = pattern.exec(str);
console.log(res)//<li>百事可乐</li><li>珍珠奶茶</li>

// 取消贪婪模式 .*?
var pattern = /<li>.*?<\/li>/;
var str = '<li>百事可乐</li><li>珍珠奶茶</li>';
var res = pattern.exec(str);
 console.log(res)//<li>百事可乐</li>
```

正则的函数

```
exec()检测是否符合正则规则，将符合的部分以数组返回，不符合则返回null
语法：pattern.exec(str)

*test() 检测是否符合正则规范，返回Boolean值 符合->true 不符合->false
语法：pattern.test(str)

search()验证是否符合正则规则，返回字符第一次出现的下标
语法：str.search(pattern)

 match()检测是否符合正则规范，将所有符合的内容全部以数组方式返回（配合模式修正符g使用）
 语法：str.match(pattern)

 replace()将所有符合正则规则的内容进行替换
 语法：str.replace(pattern,替换后的内容)
```

中文匹配

```
\u4e00-\u9fa5 中文的范围

var pattern = /[\u4e00-\u9fa5]/;代表任意的一个中文

```