什么是正则表达式?

正则表达式在不同编程语言中都存在，通常用于被检索、验证符合某一规则的文本。

比如，注册账号时，要求密码长度要超过8位，只能包括英文大小写和数字。这时就需要设置一个规则进行合法性检验。

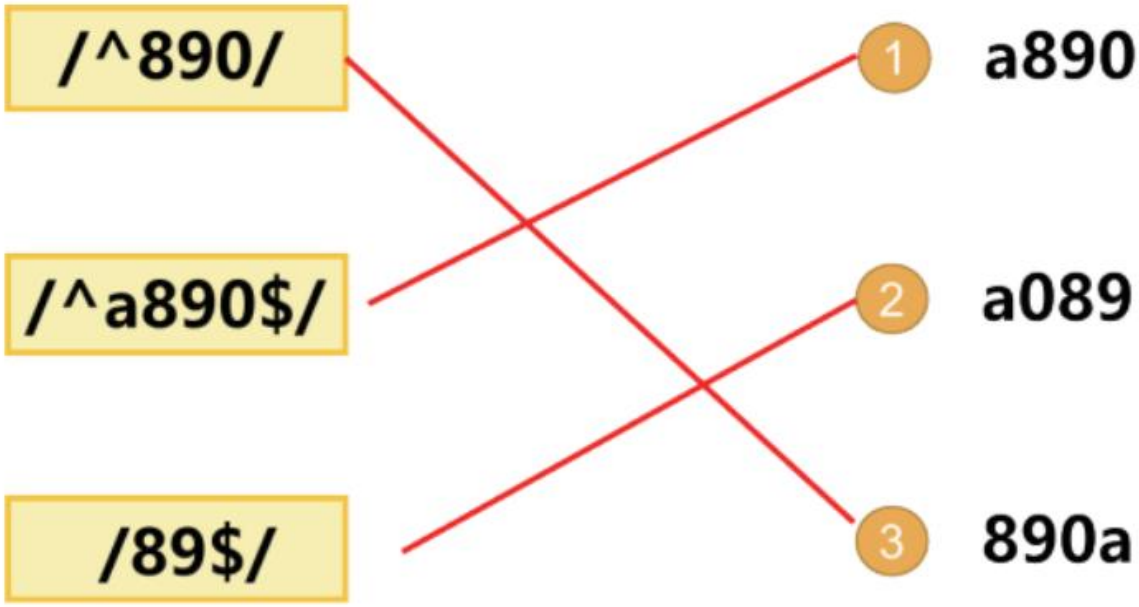
正则表达式由普通字符、元字符和量词组成。普通字符包括大小写字母与数字，元字符是具有特殊含义的字符，量词用来确定匹配字符的次数。

检测合法QQ号

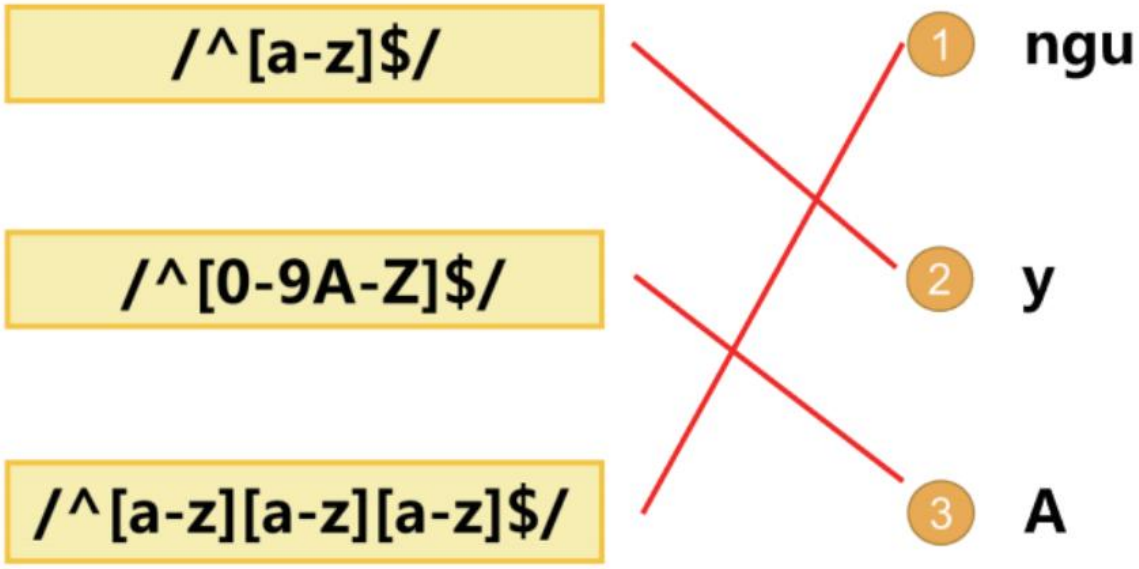
常见元字符:

字符	描述
^	匹配输入字符串的开始位置
$	匹配输入字符串的结束位置
[abc]	匹配所包含的任意一个字符
[a-z]或[A-Z]	匹配范围内的任意小写或大写字母字符
[0-9]	匹配从0到9的数字

练习1:

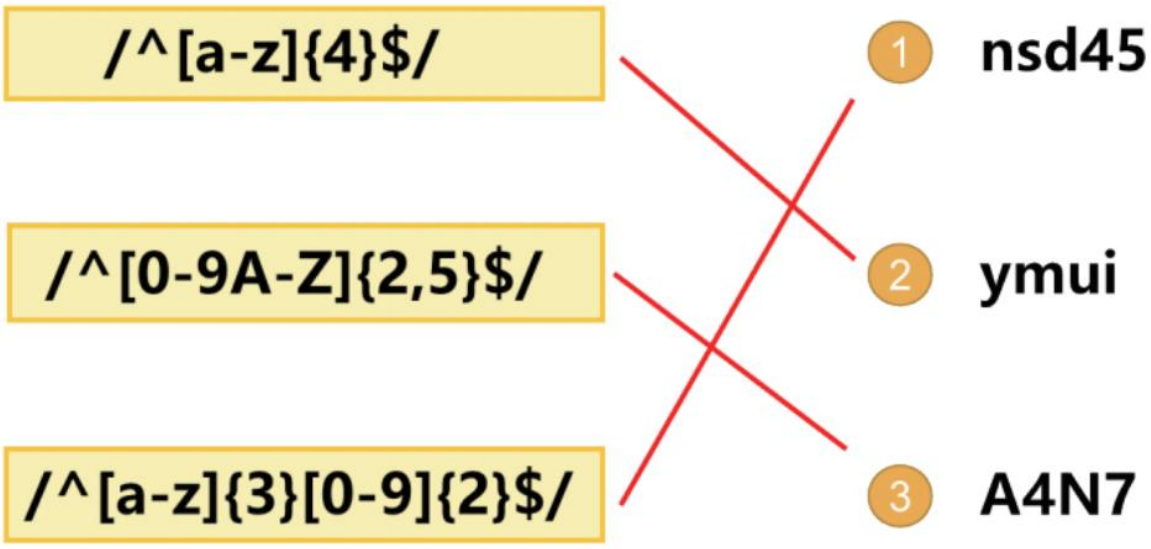


练习2：

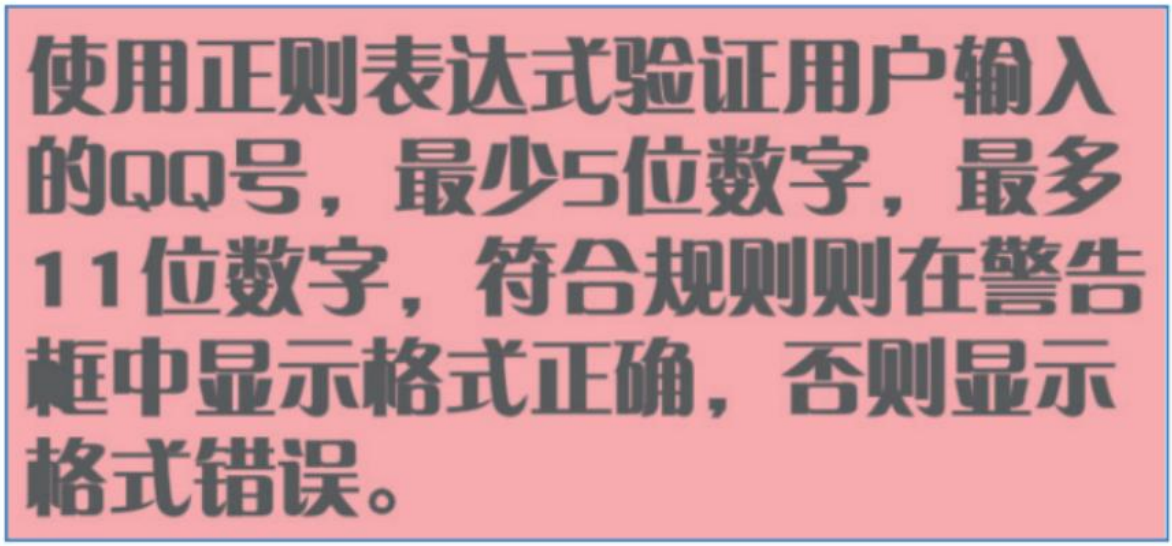


常见量词：

量词	描述
{n}	匹配的字符出现n次
{n,m}	匹配的字符出现最少n次，最多m次



test方法：

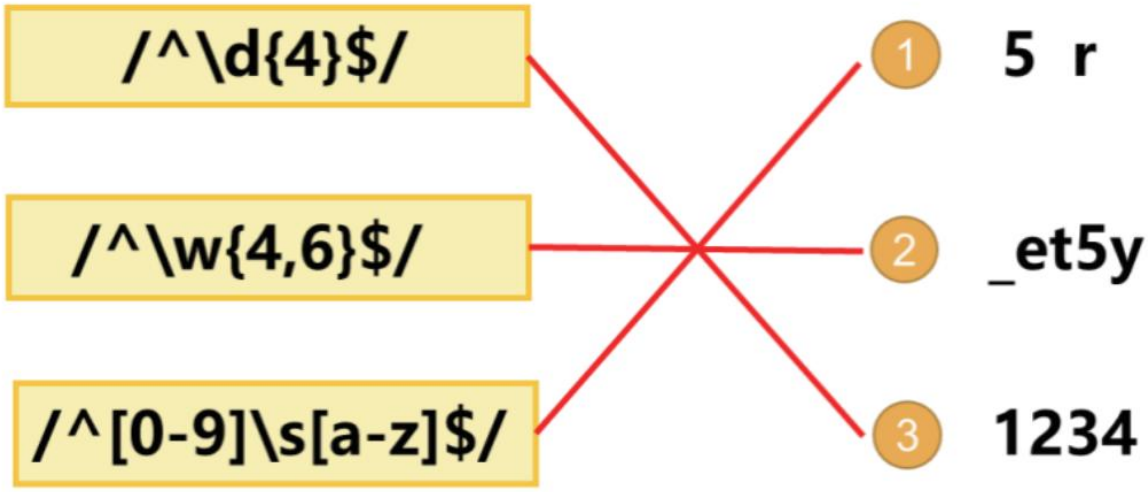



```
let reg = /^[0-9]{5,11}$/;
if(reg.test(prompt("请输入QQ号:")))
    alert("格式正确");
else
    alert("格式错误");
```

onblur事件

常见元字符：

字符	描述
\d	匹配一个数字字符,等价于 [0-9]
\D	匹配一个非数字字符
\w	匹配字母、数字、下划线,等价于[A-Za-z0-9_]
\W	匹配非字母、数字、下划线
\r	匹配一个回车符
\s	匹配任何空白字符，包括空格、制表符、换页符等等



onblur事件：当对象失去焦点时发生。

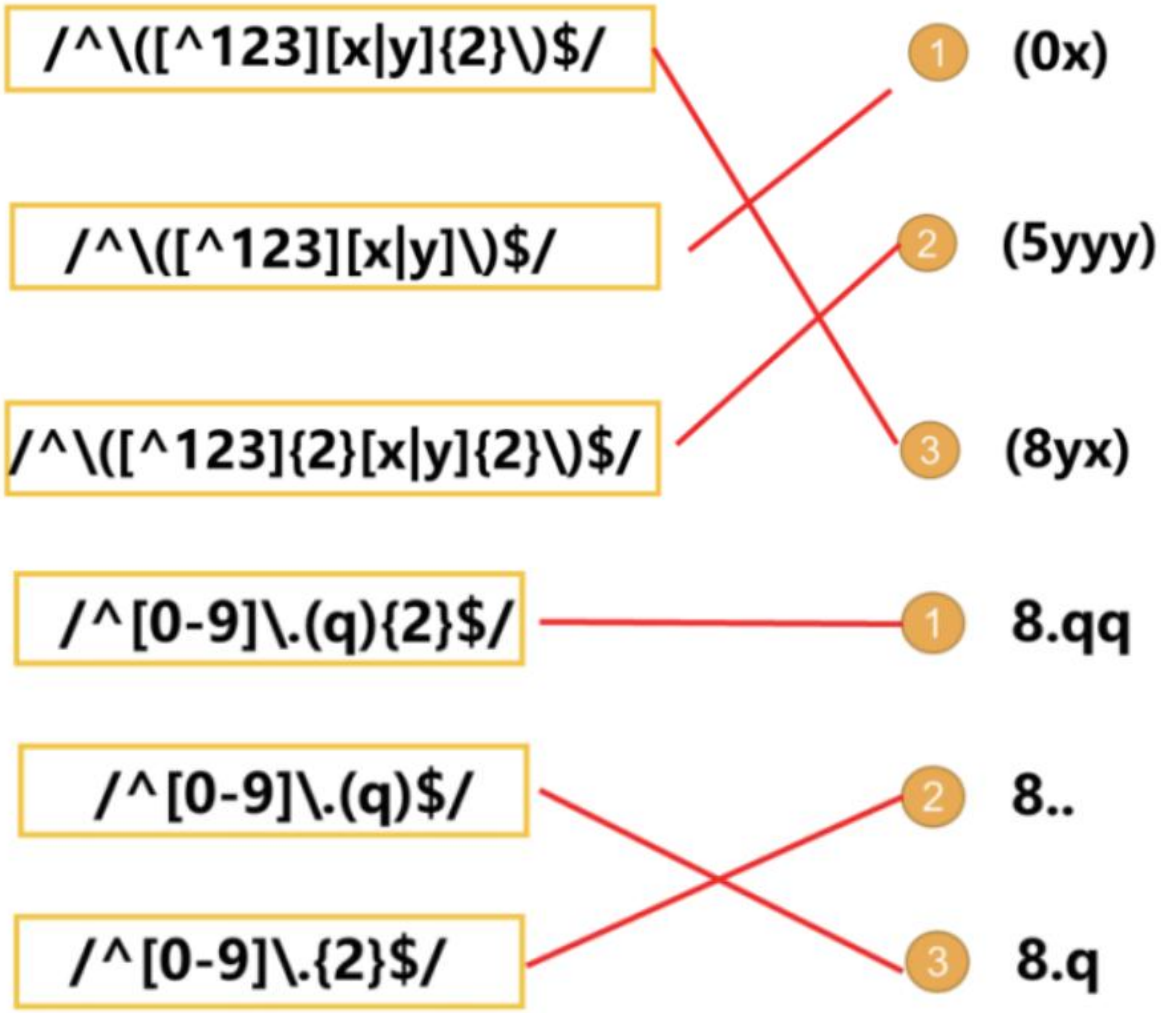
例如，在用户名和密码输入框输入内容，光标离开输入框自动检测用户名或密码是否合法。

```
<label for="nickname">昵称</label>
<input type="text" name="nickname" id="nickname">
<script>
    let nickname = document.getElementById("nickname");
    nickname.onblur = function(){
        let reg = /^\w{2,5}$/;
        if(reg.test(nickname.value))
            alert("当前昵称可用");
        else
            alert("昵称不符合命名规范");
    }
</script>
```

search和match

常见元字符：

字符	描述
\	将下一个字符标记为一个特殊字符,例如"\("匹配"("
x|y	匹配 x 或 y
[^abc]	匹配未包含的任意字符(除了abc)
(pattern)	匹配pattern并获取这一匹配



常见量词:

量词	描述
*	匹配前面的子表达式零次或多次
+	匹配前面的子表达式一次或多次
?	匹配前面的子表达式零次或一次

search()方法：用于检索字符串中指定的子串，或检索与正则表达式匹配的子串。结果返回第一个匹配到子串的起始位置，如果没有找到返回-1。标志i忽略大小写。

```
let str = "abchello world.defhillo World.";
// let reg = /h[i|e]llo/;
let reg = /World/i;
let res = str.search(reg);
console.log(res);
```

match()方法：可以检索到多个匹配结果。g全局匹配，gi全局忽略大小写匹配。

```
let str = "abchello world.defhillo World.";
let reg = /h[i|e]llo/g;
// let reg = /World/gi;
let res = str.match(reg);
console.log(res);
```