Web APIs 第六天

正则&阶段案例

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

学习目标

Learning Objectives

1. 能够利用正则表达式校验输入信息的合法性
2. 具备利用正则表达式验证小兔鲜注册页面表单的能力

目录

Contents

- 正则表达式
- 综合案例
- 阶段案例

正则表达式

- 介绍
- 语法
- 元字符
- 修饰符

- 目标：学习正则表达式概念及语法，编写简单的正则表达式实现字符的查找或检测。

1.1 什么是正则表达式

- 正则表达式（Regular Expression）是用于匹配字符串中字符组合的模式。在 JavaScript中，正则表达式也是对象
- 通常用来查找、替换那些符合正则表达式的文本，许多语言都支持正则表达式。

- 请在上图中找出【戴帽子和眼镜的男人】
- 戴帽子、戴眼镜、男人都是描述信息，通过这些信息能够在人群中查找到确定的某个人，那么这些用于查找的描述信息编写一个模式，对应到计算机中就是所谓的正则表达式。

1.1 什么是正则表达式

- 正则表达式在 JavaScript中的使用场景：
  - 例如验证表单：用户名表单只能输入英文字母、数字或者下划线， 昵称输入框中可以输入中文(匹配)
    - 比如用户名： /^[a-z0-9_-]{3,16}$/
  - 过滤掉页面内容中的一些敏感词(替换)，或从字符串中获取我们想要的特定部分(提取)等 。

总结

1.正则表达式是什么？

- 是用于匹配字符串中字符组合的**模式**

2.正则表达式有什么作用？

- 表单验证（**匹配**）
- 过滤敏感词（**替换**）
- 字符串中提取我们想要的部分（**提取**）

正则表达式

- 介绍
- 语法
- 元字符
- 修饰符

- 目标：学习正则表达式概念及语法，编写简单的正则表达式实现字符的查找或检测。

1.2 语法

- 我们想要查找是否有戴眼镜的人，怎么做呢？

1. 定义规则： 戴眼镜的
2. 根据规则去查找：找到则返回

- 正则同样道理，我们分为两步：

1. 定义规则
2. 查找

- 比如：查找下面文本中是否包含字符串 '前端'

```
let str = 'IT培训, 前端开发培训,web前端培训,软件测试培训,产品经理培训'
```

1.2 语法

- JavaScript 中定义正则表达式的语法有两种，我们先学习其中比较简单的方法：
- **1. 定义正则表达式语法：**

```
const 变量名 = /表达式/
```

- 其中 / / 是正则表达式字面量
- 比如：

```
const reg = /前端/
```

1.2 语法

2.判断是否有符合规则的字符串：

test() 方法　用来查看正则表达式与指定的字符串是否匹配

- **语法：**

```
regObj.test(被检测的字符串)
```

- 比如：

```
// 要检测的字符串
const str = 'IT培训,前端开发培训,IT培训课程,web前端培训,Java培训,人工智能培训'
// 1.定义正则表达式，检测规则
const reg = /前端/
// 2.检测方法
console.log(reg.test(str)) // true
```

- 如果正则表达式与指定的字符串匹配 ，返回true，否则false

总结

1.正则表达式使用分为几步？

- 定义正则表达式
- 检测查找是否匹配

```
// 要检测的字符串
const str = 'IT培训,前端开发培训,IT培训课程,web前端培训,Java培训,人工智能培训'
// 1.定义正则表达式，检测规则
const reg = /前端/
// 2.检测方法
console.log(reg.test(str)) // true
```

1.2 语法

3.检索（查找）符合规则的字符串：

exec() 方法 在一个指定字符串中执行一个搜索匹配

- **语法：**

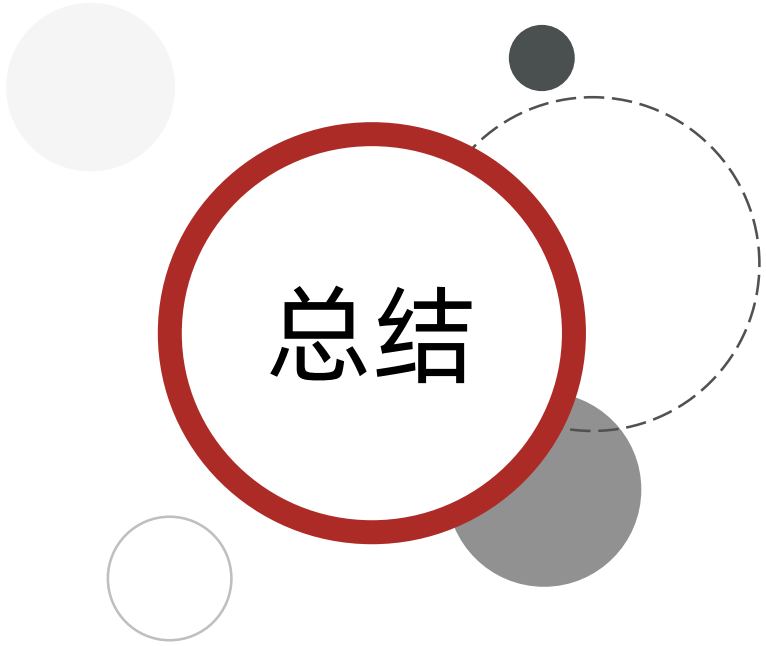
```
regObj.exec(被检测字符串)
```

- 比如：

```
// 要检测的字符串
const str = 'IT培训,前端开发培训,IT培训课程,web前端培训,Java培训,人工智能培训'
// 1.定义正则表达式，检测规则
const reg = /前端/
// 2.检测方法
console.log(reg.exec(str)) // 返回的是数组
```

```
▶ ['前端', index: 5, input: 'IT培训,前端开发培训,IT培训课程,web前端培训,Java培训,人工智能培训', groups: undefined]
```

- 如果匹配成功，exec() 方法返回一个数组，否则返回null



1.正则表达式检测查找 test方法和exec方法有什么区别？

- test方法 用于判断是否有符合规则的字符串，返回的是布尔值 找到返回true，否则false
- exec方法用于检索（查找）符合规则的字符串，找到返回数组，否则为 null

01

正则表达式

- 介绍
- 语法
- 元字符
- 修饰符

- 目标：学习正则表达式概念及语法，编写简单的正则表达式实现字符的查找或检测。

1.3 元字符

目标：能说出什么是元字符以及它的好处

- **普通字符：**

大多数的字符仅能够描述它们本身，这些字符称作普通字符，例如所有的字母和数字。

也就是说普通字符只能够匹配字符串中与它们相同的字符。

- **元字符(特殊字符)**

是一些具有特殊含义的字符，可以极大提高了灵活性和强大的匹配功能。

  - 比如，规定用户只能输入英文26个英文字母，普通字符的话 abcdefghijklm.....
  - 但是换成元字符写法： [a-z]

- **参考文档：**

  - MDN：https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Guide/Regular_Expressions
  - 正则测试工具: http://tool.oschina.net/regex

总结

1.什么是元字符以及它的好处是什么？

- 是一些具有特殊含义的字符，可以极大提高了灵活性和强大的匹配功能
- 比如英文26个英文字母，我们使用元字符［a-z］简介和灵活

1.3 元字符

为了方便记忆和学习，我们对众多的元字符进行了分类：

- 边界符（表示位置，开头和结尾，必须用什么开头，用什么结尾）
- 量词 （表示重复次数）
- 字符类 （比如 \d 表示 0~9）

1.3 元字符

1. 边界符

- 正则表达式中的边界符（位置符）用来提示字符所处的位置，主要有两个字符

边界符	说明
^	表示匹配行首的文本(以谁开始)
$	表示匹配行尾的文本(以谁结束)

如果 ^ 和 $ 在一起，表示必须是精确匹配。

1.3 元字符

1. 边界符

- 正则表达式中的边界符（位置符）用来提示字符所处的位置，主要有两个字符

```
console.log(/哈/.test('哈'))  // true
console.log(/二哈/.test('二哈')) // true
console.log(/二哈/.test('很二哈哈'))  // true

// ^ 开头
console.log(/^二哈/.test('很二哈哈'))  // false
console.log(/^二哈/.test('二哈很傻'))  // true
// $ 结尾
console.log(/^二哈$/.test('二哈很傻'))  // false
console.log(/^二哈$/.test('二哈二哈'))  // false
console.log(/^二哈$/.test('二哈'))   // true
```

1.3 元字符

为了方便记忆和学习，我们对众多的元字符进行了分类：

- 边界符（表示位置，开头和结尾，必须用什么开头，用什么结尾）
- 量词 （表示重复次数）
- 字符类 （比如 \d 表示 0~9）

1.3 元字符

2. 量词

- 量词用来 设定某个模式出现的次数

量词	说明
*	重复零次或更多次
+	重复一次或更多次
?	重复零次或一次
{n}	重复n次
{n,}	重复n次或更多次
{n,m}	重复n到m次

注意： 逗号左右两侧千万不要出现空格

1.3 元字符

2. 量词

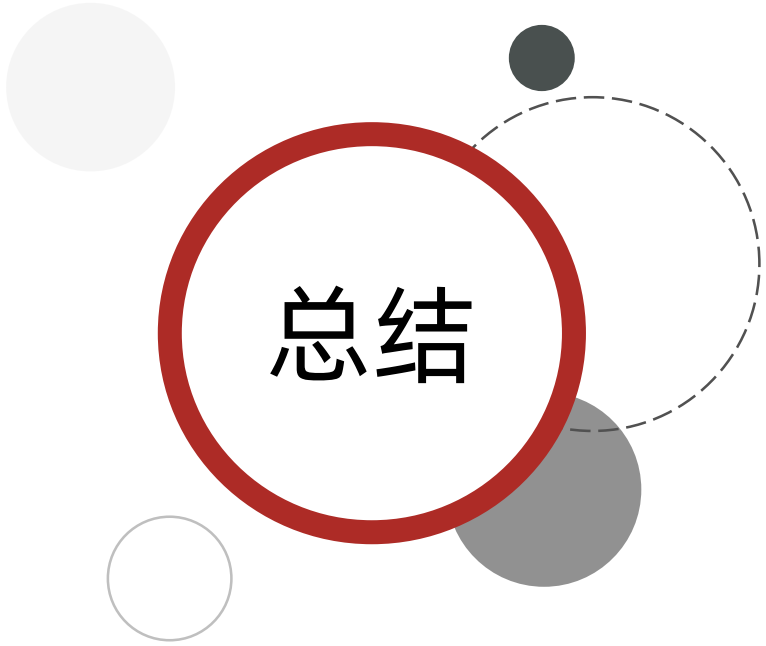
- 量词用来 设定某个模式出现的次数

```
// * 表示重复0次或者更多次
console.log(/^哈*$/.test(''))  // true
console.log(/^哈*$/.test('哈'))  // true
console.log(/^哈*$/.test('哈哈哈'))  // true
// + 表示重复1次或者更多次
console.log(/^哈+$/.test(''))  // false
console.log(/^哈+$/.test('哈'))  // true
console.log(/^哈+$/.test('哈哈哈'))  // true
// ？ 表示重复0次或者1次    0 || 1
console.log('------')
console.log(/^哈?$/.test(''))  // true
console.log(/^哈?$/.test('哈'))  // true
console.log(/^哈?$/.test('哈哈哈'))  // fasle
```

```
console.log(/^哈{2}$/.test(''))  // false
console.log(/^哈{2}$/.test('哈'))  // false
console.log(/^哈{2}$/.test('哈哈'))  // true
console.log(/^哈{2}$/.test('哈哈哈'))  // false
```

```
//  {n,}   是 >= n 的意思
console.log(/^哈{2,}$/.test(''))  // false
console.log(/^哈{2,}$/.test('哈'))  // false
console.log(/^哈{2,}$/.test('哈哈'))  // true
console.log(/^哈{2,}$/.test('哈哈哈'))  // true
```

```
//  {n,m}   是 次数 >= n  <= m 的意思 注意，逗号左右千万不要有空格
console.log(/^哈{2,4}$/.test(''))  // false
console.log(/^哈{2,4}$/.test('哈'))  // false
console.log(/^哈{2,4}$/.test('哈哈'))  // true
console.log(/^哈{2,4}$/.test('哈哈哈'))  // true
console.log(/^哈{2,4}$/.test('哈哈哈哈'))  // true
console.log(/^哈{2,4}$/.test('哈哈哈哈哈'))  // false
```



+ 表示重复至少 1 次

? 表示重复 0 次或1次

* 表示重复 0 次或多次

{m, n} 表示复 m 到 n 次

1.3 元字符

为了方便记忆和学习，我们对众多的元字符进行了分类：

- 边界符（表示位置，开头和结尾，必须用什么开头，用什么结尾）
- 量词 （表示重复次数）
- 字符类 （比如 \d 表示 0~9）

1.3 元字符

3. 字符类：

(1) [ ] 匹配字符集合

- 后面的字符串只要包含 abc 中任意**一个字符**，都返回 true 。

```
// 只要中括号里面的任意字符出现都返回为true
console.log(/[abc]/.test('andy'))   // true
console.log(/[abc]/.test('baby'))   // true
console.log(/[abc]/.test('cry'))   // true
console.log(/[abc]/.test('die'))   // false
```

1.3 元字符

3. 字符类：

(1) [ ] 里面加上 - 连字符

- 使用连字符 - 表示一个范围

```
console.log(/^[a-z]$/.test('c'))  // true
```

- 比如：
  - [a-z] 表示 a 到 z 26个英文字母都可以
  - [a-zA-Z] 表示大小写都可以
  - [0-9] 表示 0~9 的数字都可以
- 认识下：

```
腾讯QQ号：  ^[1-9][0-9]{4,}$ (腾讯QQ号从10000开始)
```

1.3 元字符

3. 字符类：

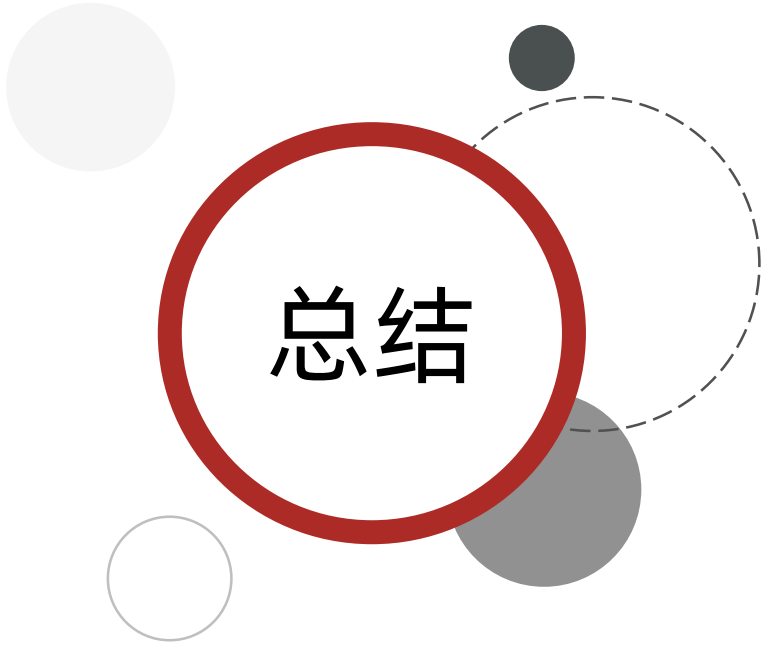
(1) [ ] 里面加上 ^ 取反符号

- 比如：
- [^a-z] 匹配除了小写字母以外的字符
- 注意要写到中括号里面

1.3 元字符

3. 字符类：

- （2）. 匹配除换行符之外的任何单个字符



1. 字符类 . (点)代表什么意思？
   - 匹配除换行符之外的任何单个字符
2. 字符类 [] 有若干代表什么意思？
   - [abc] 匹配abc其中的任何单个字符
   - [a-z] 匹配26个小写英文字母其中的任何单个字符
   - [^a-z] 匹配除了26个小写英文字母之外的其他任何单个字符

案例

用户名验证案例

需求：用户名要求用户英文字母,数字,下划线或者短横线组成，并且用户名长度为 6~16位

分析：

①：首先准备好这种正则表达式模式 /^[a-zA-Z0-9-_]{6,16}$/

②：当表单失去焦点就开始验证.

③：如果符合正则规范, 则让后面的span标签添加 right 类.

④：如果不符合正则规范, 则让后面的span标签添加 wrong 类.

练习

- 昵称案例

需求：要求用户只能输入中文

分析：

①：首先准备好这种正则表达式模式 /^[\u4e00-\u9fa5]{2,8}$/

②：当表单失去焦点就开始验证.

③：如果符合正则规范, 则让后面的span标签添加 right 类.

④：如果不符合正则规范, 则让后面的span标签添加 wrong 类.

1.3 元字符

3. 字符类：

- (3) 预定义：指的是 某些常见模式的简写方式。

预定类	说明
\d	匹配0-9之间的任一数字，相当于[0-9]
\D	匹配所有0-9以外的字符，相当于 [^0-9]
\w	匹配任意的字母、数字和下划线，相当于[A-Za-z0-9_]
\W	除所有字母、数字和下划线以外的字符，相当于 [^A-Za-z0-9_]
\s	匹配空格（包括换行符、制表符、空格符等），相等于[\t\r\n\v\f]
\S	匹配非空格的字符，相当于 [^ \t\r\n\v\f]

```
日期格式：^\d{4}-\d{1,2}-\d{1,2}
```

01

正则表达式

- 介绍
- 语法
- 元字符
- 修饰符

- 目标：学习正则表达式概念及语法，编写简单的正则表达式实现字符的查找或检测。

1.4 修饰符

- 修饰符约束正则执行的某些细节行为，如是否区分大小写、是否支持多行匹配等
- 语法：

```
/表达式/修饰符
```

- i 是单词 ignore 的缩写，正则匹配时字母不区分大小写
- g 是单词 global 的缩写，匹配所有满足正则表达式的结果

```
console.log(/a/i.test('a'))  // true
console.log(/a/i.test('A'))  // true
```

1.4 修饰符

- **替换 replace 替换**
- **语法：**

```
字符串.replace(/正则表达式/, '替换的文本')
```

案例

过滤敏感字

需求：要求用户不能输入敏感字

比如，pink老师上课很有**

分析：

①：用户输入内容

②：内容进行正则替换查找，找到敏感词，进行**

③：要全局替换使用修饰符 g

目录

Contents

- 正则表达式
- 综合案例
- 阶段案例

案例 **小兔鲜页面注册**


设置用户名称
昵称长度为6到10个字符
输入手机号码
请输入正确的手机号
短信验证码
发送验证码
请输入正确的验证码
设置6至20位字母、数字和符号组合
设置6至20位字母、数字和符号组合
请再次输入上面密码
已阅读并同意《用户服务协议》
下一步

步骤

- 小兔鲜页面注册

分析业务模块：

①： 发送验证码模块

②： 各个表单验证模块

③： 勾选已经阅读同意模块

④： 下一步验证全部模块

只要上面有一个input验证不通过就不同意提交

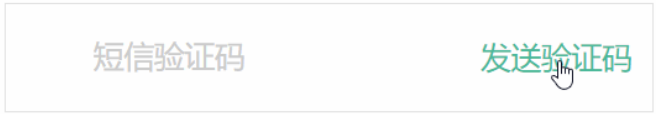
步骤

- 小兔鲜页面注册

需求①： 发送验证码

用户点击之后，显示 05秒后重新获取

时间到了，自动改为 重新获取



需求②： 用户名验证（注意封装函数 verifyxxx），失去焦点触发这个函数

正则 /^[a-zA-Z0-9-_]{6,10}$/

如果不符合要求，则出现提示信息 并 return false 中断程序

否则 则返回return true

之所以返回 布尔值，是为了 最后的提交按钮做准备

侦听使用change事件，当鼠标离开了表单，并且表单值发生了变化时触发（类似京东效果）

步骤

- 小兔鲜页面注册

需求③： 手机号验证

正则: /^1(3\d|4[5-9]|5[0-35-9]|6[567]|7[0-8]|8\d|9[0-35-9])\d{8}$/

其余同上

需求④： 验证码验证

正则 /^\d{6}$/

其余同上

需求⑤： 密码验证

正则 /^[a-zA-Z0-9-_]{6,20}$/

其余同上

步骤

- 小兔鲜页面注册

需求⑥： 再次密码验证

如果本次密码不等于上面输入的密码则返回错误信息

其余同上

需求⑦： 我同意模块

添加类 .icon-queren2 则是默认选中样式 可以使用 toggle切换类

需求⑧： 表单提交模块

使用 submit 提交事件

如果没有勾选同意协议，则提示 需要勾选

classList.contains() 看看有没有包含某个类，如果有则返回true，么有则返回false

如果上面input表单 只要有模块，返回的是 false 则 阻止提交

目录

Contents

- 正则表达式
- 综合案例
- 阶段案例

案例 小兔鲜登录页面

需求：

①： tab切换

案例 小兔鲜登录页面

需求②： 点击登录可以跳转页面

- 先阻止默认行为
- 如果没有勾选同意，则提示要勾选
- required 属性不能为空
- 假设登录成功

  把用户名记录到本地存储中

  同时跳转到首页 location.href

案例 小兔鲜首页页面

需求：

1. 从登录页面跳转过来之后，自动显示用户名
2. 如果点击退出，则不显示用户名

案例 小兔鲜首页页面

```
<ul class="xtx_navs">
  <li>
    <a href="javascript:;">
      <i class="iconfont icon-user">…</i> == $0
    </a>
  </li>
```

步骤：

最好写个渲染函数，因为一会的退出还需要用到

①：如果本地存储有记录的用户名， 读取本地存储数据

需要把用户名写到 第一个li里面

格式： pink老师

因为登录了，所以第二个 里面的文字变为，退出登录

格式： 退出登录

②：如果本地存储没有数据，则复原为默认的结构

案例 小兔鲜首页页面

步骤：

④： 点击 退出登录

删除本地存储对应的用户名数据

重新调用渲染函数即可

案例

小兔鲜首页页面

需求：

如果是移动端打开，则跳转到移动端页面

如果本地存储有数据，则 显示 你好 xxxx

否则 显示 请跳转到注册页面

今日作业

1. 整理今天笔记
2. 直接复习综合案例
3. 检测题： PC端地址： https://ks.wjx.top/vj/OCUF8u5.aspx
4. 作业- 今日案例和阶段案例，明天上午复习整个api阶段，下午开始准备实战，可以开始先做

今天多一份拼搏，明日多一份欢笑

传智教育旗下高端IT教育品牌