

JavaScript进阶

---RegExp正则表达式



河北师范大学软件学院
Software College of Hebei Normal University

内容提纲

- **正则表达式简介及正则对象**
- **RegExp及String相关的正则方法**
- **正则表达式应用案例**



正则表达式 (Regular Expression) 简介

• 什么是正则表达式

- 正则表达式是对字符串和特殊字符操作的一种逻辑公式，是对字符串执行模式匹配的工具
- 正则表达式通常被用来检索、替换那些符合某个模式(规则)的文本
- JS中正则表达式是一个描述字符模式的对象，可以通过字面量、RegExp构造器来生成

//正则对象的创建方式一 通过字面量直接创建

```
var reg1 = /[bcf]at/gi;
```

//正则对象的创建方式二 通过RegExp构造函数来实例化正则对象

```
var reg2 = new RegExp(/[bcf]at/,"gi");
```

```
var reg3 = new RegExp("[bcf]at","gi");
```

```
console.log("a fAt bat ,a faT cat".match(reg1));//同reg2 reg3
```

```
► (4) ["fAt", "bat", "faT", "cat"]
```

正则表达式 (regular expression) 简介

- 正则的语法概述和修饰符 (**g**全局,**i**忽略大小写,**m**包含换行)

```
var regExp = /a?b/gi;  
var matchResult = "xxabcaabbbxyz".match(regExp);  
console.log(matchResult);
```

► (4) ["ab", "ab", "b", "b"]

正则表达式在线分析工具 <https://regexper.com/>
<https://regex101.com/> 或 <https://jex.im/regex>

- 用正则对象来匹配字符串，也可以调用字符串方法来匹配

```
console.log("a2b3c4d".replace(/[2-3]/, "X")); //aXb3c4d  
console.log("a2b3c4d".replace(/[2-3]/g, "X")); //aXbXc4d
```

正则对象使用**方式一**
作为字符串方法中的
参数

```
var regExp = /a/gi;  
console.log(regExp.test("ab")); //true  
console.log(regExp.test("ab")); //false 为什么?  
console.log(regExp.test("ab")); //true  
console.log(regExp.test("ab")); //false 为什么?
```

正则对象使用**方式二**
调用正则对象的方法

正则对象相关字符

• 正则表达式的元字符 及 \ 相关字符

元字符: 1 3 5 a b c 等

转义字符: \t、\v、\n、\r、\0、\f、\cX

与\相关的预定义特殊字符: \d、\D、\w、\W、\s、\S、\b、\B (注意大小写的含义)

匹配将依照下列规则：

在非特殊字符之前的反斜杠表示下一个字符是特殊的，不能从字面上解释。例如，没有前面'\的'b'通常匹配小写'b'，无论它们出现在哪里。如果加了'\,这个字符变成了一个特殊意义的字符，意思是匹配一个字符边界。

反斜杠也可以将其后的特殊字符，转义为字面量。例如，模式 /a*/ 代表会匹配 0 个或者多个 a。相反，模式 /a*/ 将 '*' 的特殊性移除，从而可以匹配像 "a*" 这样的字符串。

使用 new RegExp("pattern") 的时候不要忘记将 \ 进行转义，因为 \ 在字符串里面也是一个转义字符。

https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Guide/Regular_Expressions#note

正则表达式定义及作用

• 正则表达式特殊字符 一 （字符类）

[]代表字符类，如[abc]代表abc中的任意一个字符，可以配合范围符号-如[a-c3-9]

[^]代表字符类取反，如[^abc]代表非abc中的任意一个字符

一个 - 代表范围，如[a-z]、[a-zA-Z]

一个 . 代表一个除了回车和换行符之外的所有字符 等效于[^\r\n]，（注意与*的区别和含义）

• 正则表达式特殊字符 二 （边界相关）

边界字符 ^ \$ \b \B （注意^代表的意义与在[]中代表的意义不同）

• 正则表达式特殊字符 三 （量词）

? 出现0次或1次（最多1次） + 出现1次或多次（至少1次） * 出现0次或多次（任意次）

{n} 出现n次 {n,m} 出现n到m次 {n,} 出现至少n次

正则表达式定义及作用

- 正则表达式贪婪模式与非贪婪模式

思考: "12345678" .replace(/\d{3,6}/,'X');返回多少?

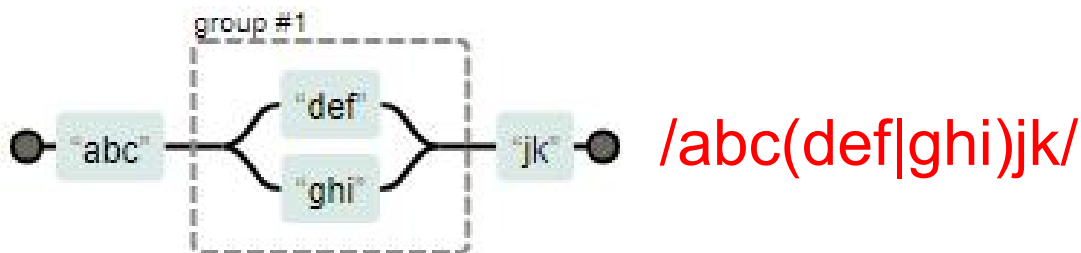
默认为贪婪模式 (即尽可能多的匹配), 在量词后加? 可设置为非贪婪模式

- 正则表达式的分组

思考: 匹配Name连续出现3次的正则, /Name{3}/, 这样可以么?

使用小括号来进行分组, 如: /(Name){3}/g

或 |、分组中的或 |



正则表达式定义及作用

- 正则表达式的分组的反向引用

思考：如何将2017-10-23转成10/23/2017

```
"2017-10-23".replace(/(\d{4})-(\d{2})-(\d{2})/, "$2/$3/$1");  
"10/23/2017"
```

- 正则表达式的分组的忽略分组

在分组内加上？：即可

```
"2017-10-23".replace(/(?:\d{4})-(\d{2})-(\d{2})/, "$2/$3/$1");  
"23/$3/10"
```


内容提纲

- 正则表达式简介及正则对象
- **RegExp及String相关的正则方法**
- 正则表达式应用案例



RegExp及String相关的正则属性和方法

• 正则表达式对象的属性（源自RegExp.prototype）

- global 默认为false
- ignore case 默认为false
- multiline 默认为false
- lastIndex 表示当前匹配内容的最后一个字符的下一个位置
- source 正则表达式文本字符串

```
var reg1 = /\w/;  
var reg2 = /\w/gi;  
console.log(reg1.global,reg1.ignoreCase,reg1.lastIndex,reg1.source);  
console.log(reg2.global,reg2.ignoreCase,reg2.lastIndex,reg2.source);  
  
false false 0 "\w"  
true true 0 "\w"
```

RegExp及String相关的正则属性和方法

- 正则表达式RegExp原型方法 (**test**)

```
var regExp = /a/gi;  
console.log(regExp.test("ab")); //true  
console.log(regExp.test("ab")); //false 为什么?
```

- 正则表达式RegExp原型方法 (**exec**) , 可获得详细信息

```
var execExp2 = /\d{1,2}(\d)(\d)/g;  
var ts = "12s342dsfsf233s";  
console.log(execExp2.exec(ts), execExp2.lastIndex); //lastIndex为 6  
console.log(execExp2.exec(ts), execExp2.lastIndex); //lastIndex为 14
```

```
▶ (3) ["342", "4", "2", index: 3, input: "12s342dsfsf233s"] 6
```

```
▶ (3) ["233", "3", "3", index: 11, input: "12s342dsfsf233s"] 14
```

RegExp及String相关的正则属性和方法

- 字符串与正则相关的原型方法 (**String.prototype.search**)

```
console.log("a1b2c3d4".search(/1/)); // 返回index 1  
console.log("a1b2c3d4".search(/f/)); // 返回index -1 没找到  
console.log("a1b2c3d4".search(/\d/g)); // 返回index 1 忽略全局  
console.log("a1b2c3d4".search(/\d\w/g)); // 返回index 1 忽略全局
```

- 字符串与正则相关的原型方法 (**String.prototype.match**)

```
console.log("a1b2c".match(/1/)); // [ '1', index: 1, input: 'a1b2c' ]  
console.log("a1b2c".match(/f/)); // null  
console.log("a1b2c".match(/\d/)); // [ '1', index: 1, input: 'a1b2c' ]  
console.log("a1b2c".match(/\d/g)); // [ '1', '2' ]
```


RegExp及String相关的正则属性和方法

- 字符串与正则相关的原型方法 (**String.prototype.replace**)

```
console.log("a,b,c,d".replace(",","X")); // aXb,c,d  
console.log("a,b,c,d".replace(/,/g,"X")); // aXbXcXd  
console.log("a2b3c4d".replace(/[2-3]/,"X")); // aXb3c4d  
console.log("a2b3c4d".replace(/[2-3]/g,"X")); // aXbXc4d
```

- 字符串与正则相关的原型方法 (**String.prototype.split**)

```
console.log("a,b,c,d".split(","));  
console.log("a2b3c4d".split(/\d/));
```

▶ (4) ["a", "b", "c", "d"]

▶ (4) ["a", "b", "c", "d"]

内容提纲

- 正则表达式简介及正则对象
- RegExp及String相关的正则方法
- 正则表达式应用案例



RegExp正则案例

• 常用正则表达式

`/\w+((-\\w+)|(\\.\\w+))*\\@[A-Za-z0-9]+((\\.|-)[A-Za-z0-9]+)*\\. [A-Za-z0-9]+/` (邮箱)

`/^[A-Za-z0-9_-]+$` (密码)

`/((?:25[0-5]|2[0-4]\\d|[01]?\\d?\\d)\\.){3}(?:25[0-5]|2[0-4]\\d|[01]?\\d?\\d)/` (IP地址)

`/.(*)\\.(rar|zip|7zip|tgz)$` (压缩格式)

`/.(*)\\.(jpg|bmp|gif|ico|pcx|jpeg|tif|png|raw|tga)$` (图片判断)

`/^#[a-fA-F0-9]{6}$` (颜色值)

`/^[A-Za-z0-9_\\-\\u4e00-\\u9fa5]+$` (用户名)

`/0?(13|14|15|18)[0-9]{9}/` (手机号)

`/^[A-Za-z0-9_()\\-\\u4e00-\\u9fa5]+$` (公司名称)

• 集成开发环境中的应用



参考链接:

<http://www1.qdfuns.com/tools.php?mod=regex>

参见实例Demo10 字符串与正则相关的原型方法

The background of the slide is decorated with various abstract shapes in shades of green and yellow. These shapes, which include circles, ovals, and teardrop-like forms, are scattered across the top and right sides of the slide, creating a modern and organic feel.

Thank You!

作业

- 阅读《深入理解JavaScript》第19章 正则表达式

