张东

zhangdong@tedu.cn

准备:

1. 犀牛书: JavaScript权威指南

2. 微信公众号: 前端大全

3. 上届笔记/思维导图

不允许记笔记！

正课:

1. 正则表达式:

2. String的正则API:

1. 正则表达式: Regular Expression

什么是: 规定字符串中字符出现规律的规则

比如: 手机号: 1 [345678] 9位数字

电子邮件:

任意字符任意个@任意字符任意个.com|.cn|.net,

何时: 2种:

1. 格式验证:

2. 模糊查找关键词:

如何:

1. 最简单的正则表达式就是关键词本身

2. 字符集:

什么是: 一位字符上，备选字符列表的集合

为什么: 有些字符可能发生变化！

何时: 只要一个字符有多种备选情况时

如何: [备选字列表]

强调: 一个[]只能匹配一位字符！

简写: 如果备选字符列表是连续的，可简写为[x-x]

比如: 一位数字: [0-9]

一位小写字母: [a-z]

一位大写字母: [A-Z]

一位字母:[A-Za-z]

一位字母或数字:[0-9A-Za-z]

一位汉字: [\u4e00-\u9fa5]

车牌号: 1位汉字 1位大写字母 · 五位大写字母或数字

3. 预定义字符集:

什么是: 对常用字符集的简化

包括: \d 一位数字

\w 一位字母数字或下划线 [0-9A-Za-z\_]

\s 一位空字符: 空格 tab ...

. 通配符，可匹配所有字符

何时: 如果规则完全匹配时，才能使用预定义字符集,

如果规则不完全匹配，则退一步用自定义[字符集]

问题: 一个字符集只能规定一位字符的变化

无法灵活定义出现的次数

4. 量词:

什么是: 专门规定一个字符集出现次数的规则

何时: 只要定义一位字符集出现的次数

如何:

必须放在一个字符集之后，修饰相邻的前一个字符集

包括2大类:

1. 有明确数量边界:

{n,m} 至少n次，最多m次

{n,} 至少n次，多了不限

{n} 必须n次

比如: qq号 \d{5,11}

2. 没有明确数量边界的

\* 可有可无，多了不限

? 可有可无，最多一次

+ 至少一次，多了不限

5. 选择和分组:

选择: 其实就是"或"

何时: 如果在多个规则之间选其一匹配即可

如何: 规则1|规则2

分组: 将多个规则合并为一组

为什么: 默认一个量词只能修饰相邻的前一个字符集

何时: 希望一个量词可以修饰之前多个字符集时

如何: (多个规则)

比如: 电子邮件地址:

任意字符任意个@任意字符任意个.com|cn|net .cn

.\*@.\*\.(com|cn|net)(\.cn)?

问题: .\* 或 .+ 容易发生贪婪模式

什么是: 正则表达式默认会匹配最长的符合条件的字符串

解决1: 懒惰模式: 仅匹配最短的符合条件的字符串

如何: (.\*?)

[^.@]+@[^.@]+\.(com|cn|net)(\.cn)?

解决2: 限制字符: [^xxx] ^称为"除了"

[^.@]+@[^.@]+\.(com|cn|net)(\.cn)?

比如: 手机号:

+86或0086 空格 1 [3-8] 9位数字

(\+86|0086)?\s\*1[3-8]\d{9}

比如: 我草 卧槽 我操 wo cao

我([艹草操槽]|cao)

比如: 身份证号:

15位数字 2位数字 1位数字或x

后三位整体可有可无，最多一次

\d{15}(\d{2}[0-9X])?

作业: 微 信 wei xin w x

6. 匹配特殊位置:

3种:

字符串开头: ^

字符串结尾: $

何时: 只要仅匹配开头的或结尾的字符时

比如: 开头的空字符 ^\s+

结尾的空字符 \s+$

同时匹配开头或结尾的空字符: ^\s+|\s+$

单词边界: \b

包括: ^ $ 空格 标点符号

比如: 每个单词首字母: \b[a-z]

选择一个单词no : \bno\b

2. String的正则API:

1. 查找关键词: 4种:

1. 查找一个固定的关键词的位置:

var i=str.indexOf("关键词"[,fromi])

var i=str.lastIndexOf("关键词")

问题: 不支持正则，只能查找一个固定的关键词

2. 用正则查找一个关键词的位置:

var i=str.search(/正则/) /xxx/是正则表达式的标识

返回关键词的位置i,

如果找不到，返回-1

问题: 默认正则表达式是区分大小的

解决: 在第二个/后加后缀i ignore忽略

问题: 只能返回位置，不能返回关键词内容

3. 查找关键词的内容:

2种:

1. 只查找一个关键词的内容和位置

var arr=str.match(/正则/i)

返回值: 返回一个数组: [ 0: "关键词", index: i ]

找不到: null

强调: 只查找第一个关键词

问题: 只能找一个关键词

2. 查找所有关键词的内容:

var arr=str.match(/正则/ig) g global 全部

返回值: [ "关键词1","关键词2",...]

arr.length 代表关键词个数！

找不到，返回null

小心: 先判断不是null，再使用！

强调: 只包含关键词内容，不包含下标

问题: 只能获得内容，无法获得每个关键词的位置

4. 查找每个关键词的内容和位置:

RegExp.exec(str)

2. 替换:

2种:

1. 简单替换: 将所有关键词都替换为一个统一的新值

str=str.replace(/正则/ig,"替换值")

2. 高级替换: 根据不同的关键词，选择替换不同的新值

str=str.replace(/正则/ig,function(kw){

//function(kw)每找到一个关键词，就自动调用一次

//每次调用时，kw: 自动获得本次找到的关键词

return 新值

})

衍生: 删除: 其实就是替换为空字符串

3. 切割:

2种:

1. 简单切割: 分隔符是固定的

var arr=str.split("分隔符")

固定用法: 将字符串打散为字符数组:

var chars=str.split("")

2. 高级切割: 分隔符是变化的

var arr=str.split(/正则/i)