如果不知道问什么: 谁问的越多！谁培训费花的越值！

1. 笔记中或画图中那句话或者那个位置不明白，立刻问！

2. 案例中哪行代码或注释不明白，立刻问！

3. 课下自己做项目，功能不会做，立刻问！

4. 刷笔试题时遇到不会的笔试题，立刻问！——公众号: 前端大全 定期总结大厂笔试题

5. 看小程序问题清单时，遇到看不懂的问题，立刻问！

正课:

1. String类型提供的正则相关函数:

2. RegExp对象

一. String类型提供的正则相关函数:

1. 查找敏感词: 4种需求:

(1). 查找一个固定的敏感词出现的位置: var i=str.indexOf("敏感词",fromi)

找不到返回-1

(2). 用正则查找第一个敏感词出现的位置: var i=str.search(/正则/i)

找不到返回-1

问题: 只能返回位置！无法返回敏感词内容

(3). 返回敏感词的内容: 2种:

a. 只返回第一个敏感词的内容和位置:

1). var arr=str.match(/正则/i)

2). 意为: 在str中查找第一个符合正则要求的敏感词的内容和位置

3). 返回值:

i. 如果找到，返回一个数组:

arr=[

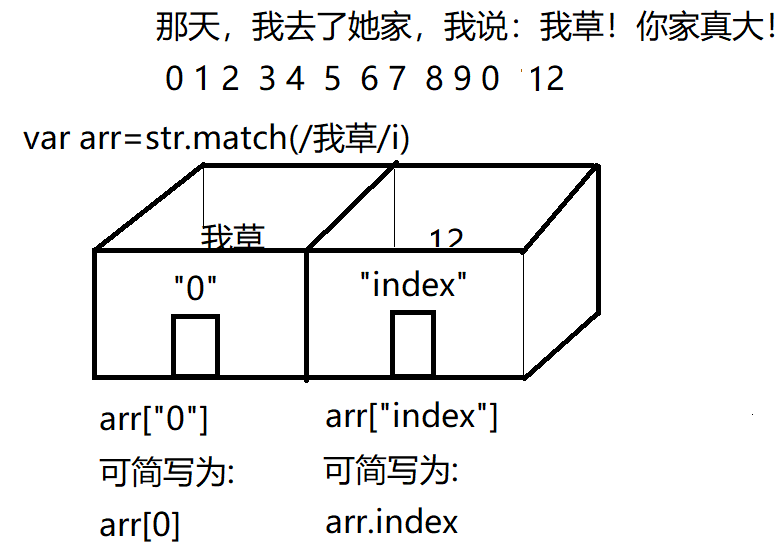
"0": "敏感词内容" ,

"index": 敏感词的位置i

]

如果获取本次找到的敏感词的内容: arr["0"] 可简写为 arr[0]

如果获取本次找到的敏感词的位置: arr["index"] 可简写为 arr.index



ii. 如果找不到: 返回null

4). 如果arr!=null，说明找到了！如果==null，说明没找到！

5). 示例: 使用match，既查找一个敏感词的内容，又查找一个敏感词的位置

|  |
| --- |
| //请用户输入一条消息内容  var input=prompt("请输入消息内容");  //查找消息中是否包含“我草”  var arr=input.match(/([我卧]|wo)\s\*([艹草槽]|cao)/i);  if(arr!=null){//如果找到    //就在网页上输出"包含敏感词，禁止发送"  //(向)网页(中)写(一行文本或一行HTML代码)    //document.write(`<h1 style="color:#555">在位置${arr["index"]}发现敏感词"${arr["0"]}",禁止发送</h1>`);    document.write(`<h1 style="color:#555">在位置${arr.index}发现敏感词"${arr[0]}",禁止发送</h1>`);  }else{//否则    //就在网页上输出"然哥说: xxxx"    document.write(`<h1 style="color:green">然哥说: ${input}</h1>`)  } |

6). 问题: 只能找出第一个敏感词，无法找所有敏感词

b. 查找所有敏感词的内容(不含位置):

1). var arr=str.match(/正则/ig) //g 意为global 全部

2). 意为: 在字符串str中查找所有符合正则要求的敏感词，放入数组中返回

3). 返回值:

i. 如果找到，返回包含本次找到的所有敏感词的内容，但是不再包含位置:

arr=[ "敏感词1", "敏感词2", ... ]

ii. 如果找不到: null

4). 如果arr!=null，说明找到了！如果==null，说明没找到！

5). 示例: 使用match加g，查找字符串中所有敏感词

|  |
| --- |
| var str="老师:请用 小红 我的 朋友 造句。小亮:小红是我的朋友。小然:朋友！小红是我的！";  //想找出所有以小字开头的人名  var arr1=str.match(/小[\u4e00-\u9fa5]/);  console.log(arr1);  var arr2=str.match(/小[\u4e00-\u9fa5]/g);  console.log(arr2); |

c. match的问题:

1). 不加g，只能找一个，但是既能获得内容，又能获得位置

2). 加g，可以找所有，但是不知道位置了！

(4). 既查找每个敏感词的内容，又查找每个敏感词的位置！

String类型没有提供该函数

只能用后边要学的reg.exec()函数

总结:

1. 凡是只返回一个下标i值的函数，如果找不到，返回-1

2. 凡是返回一个数组的函数，如果找不到，返回null

强调: 如果一个函数有可能返回null，则使用函数的返回值时必须加着12分的小心！！！

null是造成错误最多的根源之一。因为null后什么都不能加，加()报错，加.也报错。

解决: 凡是有可能返回null的函数的返回值，都要先判断不是null，才能正常使用，否则，必须为null的情况提供专门的解决办法. 用变量接住新值;

补: js中数组的本质:

|  |  |
| --- | --- |
| 第一阶段说: 数组分为索引数组和关联数组  索引数组: 下标都是数字的数组，比如: [ 1,2,3 ] => [ 0:1, 1: 2 , 2:3 ]  关联数组: 下标都是自定义的字符串名称的数组, 比如:  var ym=[];  ym["数学"]=89;  ym["语文"]=69;  ym["英语"]=91;  其实: 其实js中一切数组低层都是关联数组！  所有的下标，不管是数字，还是字符串，低层其实都是字符串  arr=[ 1,2,3 ] => [ "0":1, "1": 2 , "2":3 ]  var ym=[]; ym["数学"]=89; ym["语文"]=69; ym["英语"]=91;  [ "数学":89, "语文":69, "英语":91 ]  所有数组都可用 数组名["房间号"] 来取出某个位置上的值  比如: arr["0"] => 1 ym["数学"] => 89  但是因为""很麻烦，所以可以简写:  ym["数学"] 可简写为 ym.数学  arr["0"] 可简写为 arr[0] ~~arr.0 和程序中的小数点冲突了！~~  示例: 比较所谓的索引数组和关联数组的结构和用法   |  | | --- | | var arr=[ 1,2,3 ];  var ym=[];  ym["数学"]=89;  ym["语文"]=69;  ym["英语"]=91;  console.log(arr);  console.log(ym);  console.log(arr["0"], arr[0]);  //             1         1  console.log(ym["数学"], ym.数学);  //               89      89  console.log(arr["length"],arr.length);  //                3            3  console.log(ym["length"],ym.length);  //                0          0  //因为length只统计数字下标的个数 | |

2. 替换敏感词: 2种:

(1). 简单替换: 将所有找到的敏感词都替换为统一的新值！

a. str=str.replace(/正则/ig, "新值")

b. 取出原字符串str中的内容，替换其中所有符合正则要求的敏感词为统一的新值后，将替换后的新字符串，返回，并保存到str变量中。——才能接住替换后的新字符串

c. 示例: 替换字符串中所有敏感词为统一的新值

|  |
| --- |
| var str="老师:请用 小红 我的 朋友 造句。小亮:小红是我的朋友。小然:朋友！小红是我的！";  // var str="you can you up,no can no bibi";  //想找出所有以小字开头的人名  var arr=str.match(/小[\u4e00-\u9fa5]/g);  console.log(arr);  //想把所有小字开头的人名替换为\*\*  str=str.replace(/小[\u4e00-\u9fa5]/g,"\*\*");  console.log(str);  //一般同学:  // if(arr!=null){  //   console.log(`共替换${arr.length}处`);  // }else{  //   console.log(`共替换0处`);  // }  //基础好的同学:  console.log(`共替换${arr==null?0:arr.length}处`);  var arr=[12,2,123,32,23,1,3];  arr.sort(function(a,b){return a-b});  console.log(arr); |

中午必须看的视频:

小程序->在线->JSCORE->day02 3. 回调函数 callback 亮哥 带饭

(2). 高级替换: 根据每次找到的敏感词的不同，动态选择不同的新值替换！

a. str=str.replace(/正则/ig, function(keyword){

return 根据keyword不同，返回不同的新值

})

b. 原理:

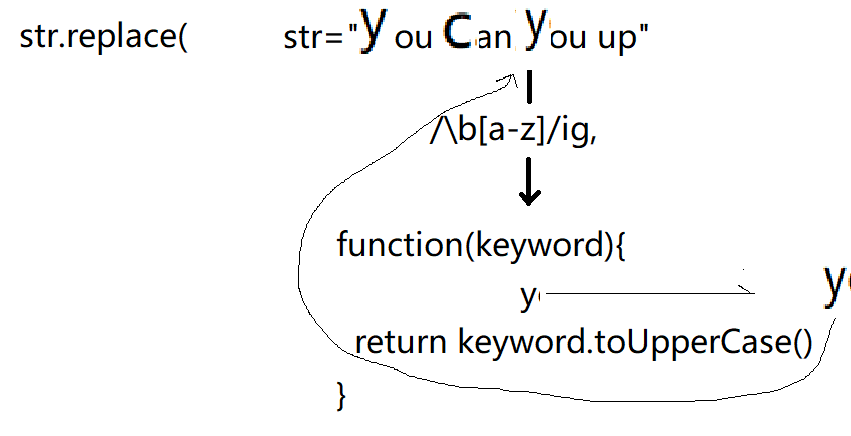
1). replace会用正则去str中查找每个符合要求的敏感词

2). 每找到一个关键词，就自动调用一次回调函数

3). 每次调用回调函数时，都会自动将本次找到的敏感词传给函数的形参变量keyword。

4). 回调函数内部，根据本次传入的keyword关键词不同，动态选择不同的新值返回给replace

5). replace会自动用回调函数返回的新值，替换到字符串中本次找到的关键词位置。



c. 回调函数如何学习：

1). 通过打桩(输出调用次数和变量的值)，让回调函数的执行过程现出原形！

2). 通过打桩输出的结果，推断出回调函数执行的过程。

d. 示例: 每个单词首字母转大写:

|  |
| --- |
| //想把每个单词首字母改为大写  var str="you can you up";  str=str.replace(/\b[a-z]/ig,function(keyword){    console.log(`调用了一次回调函数,本次获得关键词:keyword=${keyword},返回新值:${keyword.toUpperCase()}`)    //keyword依次接受: y c y u    return keyword.toUpperCase()  });  console.log(str); //You Can You Up |

补:

|  |
| --- |
| 大多数编程语言中: 字符串都是不可变类型！  什么是不可变类型: 一旦创建，值不可改变，只能整体替换  什么是可变类型: 创建后，值还可随时修改的数据类型。  正是因为字符串是不可变类型，所以所有字符串类型提供的函数，都无权修改原字符串！只能返回加工后的新字符串，再用变量接住！  但是因为数组是可变类型，所有几乎所有数组类型提供的函数，都可直接修改原数组！无需用变量接住！ |

(3). 删除关键词: 其实就是将找到的关键词替换为""

示例: 删除字符串开头和结尾的空字符

|  |
| --- |
| //希望定义三个函数:  //第一个函数可以只去掉字符串开头的空字符  function ltrim(str){    console.log("调用自己的ltrim");    return str.replace(/^\s+/,"");  }  //第二个函数可以只去掉字符串结尾的空字符  function rtrim(str){    console.log("调用自己的rtrim");    return str.replace(/\s+$/,"")  }  //第三个函数可以同时去掉字符串开头和结尾的空字符  function trim(str){    console.log("调用自己的trim");    return str.replace(/^\s+|\s+$/g,"");  }  var str="   zhang  dong   ";  console.log(ltrim(str));  console.log(rtrim(str));  console.log(trim(str));  //其实新ES6/ES7标准已经提供了trim等一系列去空格的函数  console.log(str.trimLeft());  console.log(str.trimRight());  console.log(str.trim()); |

3. 切割: 将一个字符串按指定的分割字符，分割为多段子字符串:

(1). 简单切割: 分割符是一个固定不变的字符

a. var arr=str.split("分割符");

b. 将字符串str，按字符串中指定的分割符，分割为多段子字符串，保存在一个数组中返回.

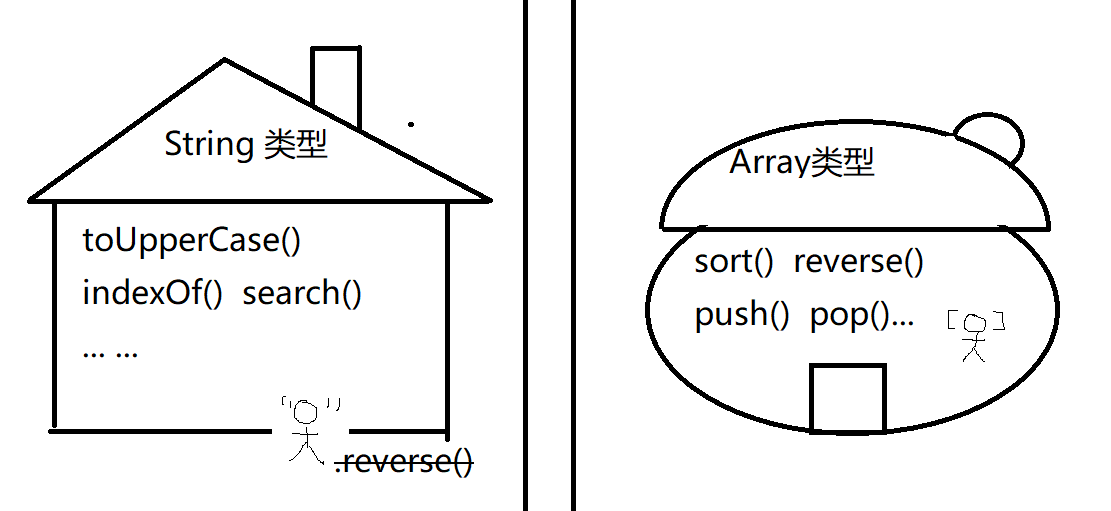
强调: 在切割的结果中，不包含切割符了。

c. 示例: 将电子邮件切割为用户名和域名，将英文的句子切割为单词数组

|  |
| --- |
| var email="zhangdong@tedu.cn";  //                  |  //      ["zhangdong","tedu.cn"]  //           [0]        [1]  //想获得电子邮件中的用户名和域名  //将email按@切割  var arr=email.split("@");  console.log(arr);  var uname=arr[0];  var domain=arr[1];  console.log(`用户名:${uname},域名:${domain}`);  var str="you can you up";  //想把一句英文的句子切割为单词数组  var arr2=str.split(" ");  console.log(arr2); |

d. 固定套路: 打散一个字符串为字符数组

1). 问题: 字符串虽然长得像数组，但是因为不是数组家孩子！无权使用数组家的函数！



2). 解决: 先将字符串打散为字符的数组，就可以使用数组家的函数了。但是处理完之后，还要再拼接回字符串。

3). 如何: var arr=str.split("")

4). 示例:翻转一个字符串

|  |
| --- |
| var arr=[1,2,3];  arr.reverse();  console.log(arr);  var str="helloworld";  //鄙视题: 翻转字符串:  //先将字符串打散为字符数组:  var arr=str.split("");  //[h,e,l,l,o,w,o,r,l,d];  arr.reverse();  console.log(arr);  //[d,l,r,o,w,o,l,l,e,h]  str=arr.join("");//"dlrowolleh"  console.log(str);//"dlrowolleh" |

(2). 复杂切割: 切割符不是固定的，但是却有规律:

a. var arr=str.split(/正则/)

b. 意为: 按字符串str中所有符合正则要求的分隔符切割字符串为多段子字符串

c. 强调: 不用加g，split也能找到所有的切割符切割字符串。

d. 示例: 提取HTML中ul下多个li中的分数

|  |
| --- |
| <body>  <ul id="ulScores">  <li>85</li>  <li>95</li>  <li>66</li>  <li>72</li>  </ul>  <script>  //想获得HTML中ul的内容中所有的成绩，放在一个数组中！  //1. 查找id为ulScores的ul对象  var ul=document.getElementById("ulScores");  //2. 获得ul的HTML内容:  var html=ul.innerHTML;  //3. 将HTML去掉开头的"空字符<li>"和结尾的"</li>空字符"  html=html.replace(/^\s\*<li>|<\/li>\s\*$/ig, "");  console.log(html);  //4. 按"</li>空字符<li>"切割整个字符串  var arr=html.split(/<\/li>\s\*<li>/);  console.log(arr);//[85,95,66,72]  </script>  </body> |

二. RegExp对象:

1. 什么是RegExp对象: 专门保存一条正则表达式，并提供了用正则执行验证和查找操作的函数 的对象

2. 为什么: 因为正则表达式是一套独立于任何语言之外的特殊的语法。Js语言默认不认识正则表达式。所以需要一个专门的对象来解析正则表达式，然后再使用正则表达式。

3. 何时: 今后只要在js中用正则表达式，都要先创建正则表达式对象来保存正则表达式，然后才能使用正则表达式。

4. 如何:

(1). 创建正则表达式对象: 2种:

a. 标准写法: var reg=new RegExp("正则","ig")

何时: 既适合于固定的写死的正则，又可以根据其条件动态生成正则表达式

因为new RegExp()的参数需要的是一个字符串，而js中有无数种办法可以拼接处想要的任意字符串！

冲突: 如果正则中包含\d,\s,\w，为了避免js解析\，都要改为\\d, \\s, \\w

b. 简化写法: var reg=/正则/ig (广泛)

何时: 只适合于正则表达式是固定的写死的。无法再程序运行时，根据其他条件动态生成正则表达式。

因为//之中是正则表达式的地盘，不认识js语句，所以无法执行任何拼接等js操作。

冲突: 如果正则中包含/，则必须改为\/

c. 示例: 动态生成正则表达式

|  |
| --- |
| <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>    <script>      //假设从服务器端请求过来一个敏感词数组      var kws=["明月","青天","白鹭","紫烟"];      //希望正则是明月|青天|白鹭      //var reg=kws.join("|");      //console.log(reg);    </script>  </head>  <body>  <script>  //请用户输入一条消息内容  var input=prompt("请输入消息内容");  //查找消息中是否包含服务器要求我们防住的几个词  var arr=input.match(new RegExp(kws.join("|")));  if(arr!=null){//如果找到    //就在网页上输出"包含敏感词，禁止发送"    document.write(`<h1 style="color:#555">在位置${arr.index}发现敏感词"${arr[0]}",禁止发送</h1>`);  }else{//否则    //就在网页上输出"然哥说: xxxx"    document.write(`<h1 style="color:green">然哥说: ${input}</h1>`)  }  </script>  </body> |

(2). 验证字符串格式:

a. var bool=reg.test(str);

b. 用正则表达式对象reg，检查字符串str是否符合正则的格式要求！

c. 返回值:

1). 如果字符串str符合reg正则表达式的要求，就返回true

2). 否则如果字符串str不符合reg正则表达式要求，就返回false

d. 坑: 默认情况下，直到能在字符串str中找到部分符合reg要求的子字符串，就返回true！

e. 解决: 今后凡是验证，都必须前加^，同时后加$，表示从头到尾必须完全匹配的意思！

f. 示例: 验证手机号格式:

|  |
| --- |
| var input=prompt("请用户输入手机号");  //定义验证手机号的正则表达式  var reg=/^1[3-9]\d{9}$/;  //如果用reg验证input格式正确  if(reg.test(input)){    //就在页面显示: 手机号格式正确    document.write(`<h1 style="color:green">手机号格式正确</h1>`)  }else{//否则    //就在页面显示: 手机号格式不正确    document.write(`<h1 style="color:red">手机号格式不正确</h1>`)  } |

(3). 高级查找: 既查找每个敏感词的内容，又查找每个敏感词的位置

a. var arr=reg.exec(str);

b. 意为: 在字符串str中查找下一个符合reg正则要求的敏感词的内容和位置

c. 返回值:

1).如果本次找到了一个敏感词，也返回一个数组，结构和match不加g时一样！

arr=[ 0: 敏感词内容, index: 敏感词位置 ]

2). 如果找不到返回: null

d. 示例: 查找每个关键词的内容和位置

|  |
| --- |
| var str="老师:请用 小红 我的 朋友 造句。小亮:小红是我的朋友。小然:朋友！小红是我的！";  //先用match反复调用，找小字开头的人名  var reg=/小[\u4e00-\u9fa5]/g;  // var arr=str.match(reg);  // console.log(arr);  // arr=str.match(reg);  // console.log(arr);  // arr=str.match(reg);  // console.log(arr);  do{    var arr=reg.exec(str);    if(arr!=null){      //console.log(arr);      console.log(`在位置${arr.index}发现敏感词${arr[0]}`)    }  }while(arr!=null);//何时才有必要继续找下一个？ |

总结:

1. 正则:

(1). 一个敏感词正文就是一个固定的写死的规则

(2). 如果规则中一位字符有多种备选字时: 字符集 []

(3). 如果字符集中部分备选字符是连续的: -

(4). 常用字符集:

a. 一位数字: [0-9] 或 \d

b. 一位小写字母: [a-z]

c. 一位大写字母: [A-Z]

d. 一位字母(大小写都行): [A-Za-z]

e. 一位字母或数字都行: [0-9A-Za-z] ~~不能用\w~~

f. 一位字母数字或\_都行: [0-9A-Za-z\_] 或 \w

g. 一位汉字: [\u4e00-\u9fa5]

h. 一位空字符: \s

(5). 数量词:

a. 带有明确数量边界的量词:

(1). 必须出现n次，不能多也不能少: {n}

(2). 至少出现n次，最多m次: {n,m}

(3). 至少出现n次，多了不限: {n,}

b. 没有明确数量边界的量词:

(1). 可有可无，多了不限: \*

(2). 可有可无，最多出现一次: ?

(3). 至少出现一次，多了不限: +

(6). 选择和分组:

a. 两个子规则或多个子规则，二选一或多选一匹配即可: | 或者

b. 正则中局部子规则先匹配，再和整个正则联合匹配: () 分组

(7). 特殊位置:

a. 字符串开头: ^

b. 字符串结尾: $

c. 单词边界: \b

2. String类型提供的正则函数:

(1). 查找敏感词:

a. 查找一个固定的敏感词出现的位置: var i=str.indexOf("敏感词",fromi)

找不到返回-1

b. 用正则查找第一个敏感词出现的位置: var i=str.search(/正则/i)

找不到返回-1

作业:

1. (必须) 复习三目运算:

思维导图: javascript->分支结构

2. (必须) 看小程序视频学习回调函数:

小程序->在线->JSCORE->day02 3. 回调函数 callback 亮哥 带饭

3. (必须) 看小程序视频复习声明提前:

小程序->在线->JSCORE->day03

1. 声明提前

2. 声明提前完整鄙视题

4. (必须)看小程序视频深入理解reg.exec()函数执行原理:

小程序->在线->JSCORE->day02 10. 查找敏感词 既查找每个敏感词的内容。。。

5. (扩展)看小程序视频学习高频鄙视题:

小程序->在线->JSCORE->day02 扩展: 鄙视题1.1~1.4 使用xxx解析url