正课:

1. String

内置对象

包装类型

字符串API

2. \*\*\*\*正则表达式

1. String

什么是: 由多个字符组成的\*只读\*字符数组

vs 数组: 相同: 1. 下标; 2. .length; 3. slice

不同: 类型不同, API不通用！

内置对象: ES标准中规定的，浏览器厂商已经实现的对象

包括: 11个:

String Number Boolean ——包装类型

Array Date Math RegExp

Error

Function Object

Global (在浏览器中被window代替)

包装类型:

什么是: 专门包装一个原始类型的值，并提供操作原始类型值的API

为什么: 原始类型的值本身不具有任何功能

何时: 一般不会主动使用

只要试图用原始类型的值调用函数时，会自动创建包装类型的对象，调用对象的函数执行操作。

比如: n.toFixed(2) => typeof n=>number

=>new Number(n).toFixed(2)

str.charCodeAt()=>typeof str=>string

=>new String(str).charCodeAt()

String API: 强调: 所有String API都无权修改原字符串，只能返回新字符串

大小写转换:

将字符串中所有字母，统一转为大写或小写

str.toLowerCase() 转小写

str.toUpperCase() 转大写

何时: 不区分大小时:

验证码，用户名，电子邮件

获得指定位置的字符:

str.charAt(i) <=> str[i]

获得指定位置字符的unicode号

str.charCodeAt(i) 获得i位置的字符的unicode号

将unicode号，反向转回原文

String.fromCharCode(unicode)

强调: 一次只能转一个字

获取子字符串:

str.slice(starti,endi+1)

str.substring(starti,endi+1) 用法同slice，但不支持负数

str.substr(starti,n) => str.slice(starti,starti+n);

查找关键词: 4种:

1. 查找一个固定关键词的位置:

var i=str.indexOf("关键词",fromi)

从fromi位置开始，找str中下一个"关键词"的位置

返回值: 找到的关键词的下标位置i

如果没找到，返回-1

简写: 省略fromi，默认从0开始

查找最后一个关键词的位置:

var lasti=str.lastIndexOf("关键词");

问题: 只能查找一个固定关键词的位置

解决: 正则表达式

2. 判断字符串中\*是否包含\*符合规则的关键词

var i=str.search(/reg/i)

返回值: 返回关键词的位置

如果没找到返回-1

问题: 正则表达式默认都是大小写敏感的

解决: 在第二个/后加i,表示忽略(ignore)

何时: 只要仅判断有没有关键词时，首选search

问题: 1. 只能查找一个关键词

2. 只能返回位置，无法返回关键词内容

3. 查找所有关键词的内容:

var kwords=str.match(/reg/ig);

返回值: 返回所有关键词组成的数组

如果没找到，返回null

问题: 正则表达式默认都只匹配第一个关键词

解决: 在第二个/后加g，表示全部(global)

何时: 仅希望获得关键词内容时

问题: 仅能获得内容，无法获得每个关键词的位置

解决:

4. 即查找关键词内容，又查找位置: ?

替换: 将字符串中找到的关键词，替换为指定新内容

1. 简单替换: 将所有关键词都替换为统一的内容

str=str.replace(/reg/ig,"新值");

2. 高级替换: 根据每个关键词的不同，动态选择替换不同的新值

str=str.replace(/reg/ig,function(kw){

kw//可自动获得本次找到的关键词

return 根据kw的不同，动态返回不同的值

})

衍生: 删除: 将关键词替换为""

str=str.replace(/reg/ig,"")

切割: 按指定字符，将一个字符串切割为多段子字符串

var subs=str.split(/reg/i);

说明: 在切割后的结果数组中，不包含切割符了

固定套路: 将字符串打散为字符数组:

2. \*\*\*\*正则表达式: Regular Expression

什么是: 规定一个字符串中字符出现规律的规则

何时: 2种:

1. 模糊匹配多种关键词

2. 密码格式

如何:

1. 最简单的正则表达式，是关键词原文本身

2. 字符集:

什么是: 规定一位字符可用的备选字符列表的集合

何时: 只要一位字符，有多种可能备选字时

如何: [备选字符列表]

强调: 一个字符集只能规定一位字符的备选字符列表

简写: 如果字符列表中部分字符是连续的，可用-省略中间字符

比如: [0-9] 一位数字

[a-z] 一位小写字母

[A-Z] 一位大写字母

[A-Za-z] 一位字母

[0-9A-Za-z] 一位字母或数字

[\u4e00-\u9fa5] 一位汉字

特殊: 除了...

[^字符列表]

3. 预定义字符集:

什么是: 对常用字符集的简写: 4个:

\d 一位数字 =>[0-9]

\w 一位字母,数字或\_ =>[0-9A-Za-z\_]

\s 一位空字符: 空格, Tab...

. 通配符

4. 量词:

什么是: 专门规定字符集出现次数的规则

何时: 只要规定字符集出现的位数/次数

如何: 量词必须紧跟在字符集之后，修饰相邻的前一个字符集。

包括:2大类:

1. 有明确数量边界的:

字符集{n,m} 至少n个，最多m个

字符集{n,} 至少n个，多了不限

字符集{n} 必须n个

2. 没有明确数量边界的:

字符集? 可有可无，最多1个

字符集\* 可有可无，多了不限

字符集+ 至少一个，多了不限

5. 选择和分组:

选择: 或 多个规则，只要匹配其一即可！

规则1|规则2|...

|在正则中，优先级最低

分组: 用()将多个规则分为一组

何时: 如果希望量词同时修饰多个字符集时，就要先将多个字符集分为一组，再用量词修饰分组

为什么: 默认字符集仅修饰相邻的前一个字符集

身份证号: 15位数字 2位数字 1位数字或Xx

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

后两部分，可有可无，最多1次

手机号: +86或0086 可有可无，最多一次

空字符 可有可无，多了不限

1

3,4,5,7,8 选一个

9位数字

(\+86|0086)?\s\*1[34578]\d{9}

微 信 wei xin w x

(微|w(ei)?)\s\*(信|x(in)?)

6. 指定匹配位置: 3种:

^ 字符串开头 比如: 字符串开头的空字符: ^\s+

$ 字符串结尾 比如: 字符串结尾的空字符: \s+$

开头或结尾的空字符:

\b 单词边界: ^ $ 空格 标点符号

比如: 查找单词no，不包含单词内的no

\bno\b