# Js面试题总结

说明：面试并非笔试。

特别说明：本篇总结为个人心得总结，仅完美适用于作者本人。如有异议，请自行消化或删除文档。

作者：陶福荣

备注：格式如果错乱，请使用正版offic2013

版权所有，非作者本人同意，不得上传于网络

# 1.什么是闭包？举例说明其作用

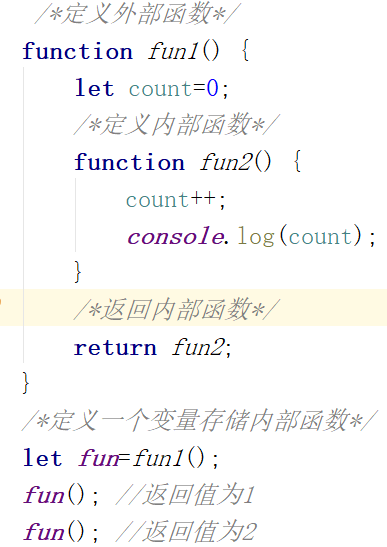
## 1）概念

我认为闭包是一个对象，这个对象里保存外部函数的局部变量，函数嵌套函数，内部函数引用外部函数的局部变量，外部函数返回内部函数；并在外部函数外面定义一个变量来存储这个函数。

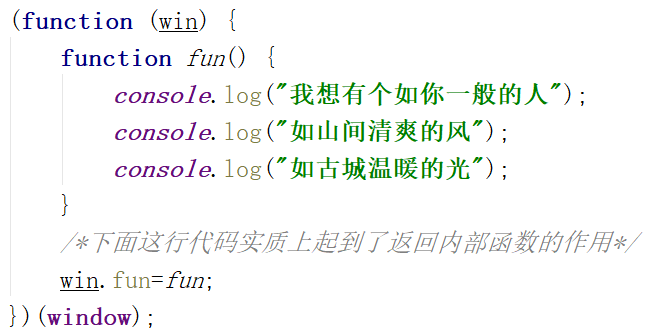
JavaScript的对象没有私有属性，也就是说对象的每一个属性都是曝露给外部的。这样可能会有安全隐患，譬如对象的使用者直接修改了某个属性，导致对象内部数据的一致性受到破坏等。这个时候我们就可以使用闭包来把一个对象封装起来，只返回一个访问器的对象。如下面图示中的代码，count只能通过fun来访问，除此之外，绝无其他方式可以找到count变量。

## 2）优缺点

闭包可以延长外部函数局部变量的生命周期，且保证这个局部变量不会污染全局变量 。但是闭包占用内存，容易造成内存溢出。所以使用完要及时清理闭包。以下为示例代码：



闭包最常见的就是在自执行匿名函数中封装的方法，示例代码如下：



通过上面图示中的代码可以看出，大部分自执行匿名都是闭包的一种应用和写法。当然，返回内部函数的代码在自执行匿名函数里用win.fun=fun;代替了。

# 2.js数组的方法大全

## 1）join

arr.join(“”);里面没有添加元素时，默认是使用空格，即将各元素不间隔的链接在一起，形成字符串。

arr.join(“|”)；里面添加符号元素时，表示将各数组以|隔开，形成字符串；

arr.join();没有参数的时候，默认字符串用,分隔.

返回的是一个对象。

## 2）push()和pop()

arr.push();给数组尾部添加一个元素；

arr.pop();删除一个尾部的元素

## 3）shift() 和 unshift()

arr.unshift(); 头部添加一个元素

arr.shift(); 删除头部的一个元素

## 4）sort()

arr.sort();按照升序排列数组，最小的排在前面（实际是参照unicode进行排列）；

## 5）reverse()

arr.reverse();将一个数组翻转排列，前面的放后面，后面的放前面；

## 6）concat()

1.数组.concat(元素);

直接在原数组后面增加一个元素，并返回新的数组。Ps：相当于组合一个新的数组。

2.数组.concat(数组);相当于将两个数组连接起来，返回一个新的数组；

3.数组.concat();相当于复制一个数组。

返回新的数组。

## 7）slice()

1、arr.slice(1,3); 从下标是1的位置开始到下标是3的位置结束，  
 获取从下标1到下标3之前所有数据

2、arr.slice(1,-1); 从下标是1开始到数组最后一个下标之前

3、arr.slice()；将原数组复制成为一个新数组，无引用关系，和arr.concat()相同  
4、arr.slice(1); 不写第二个结束下标，就是从下标1的位置到结尾  
5、arr.slice(-5,-1);从倒数第5位到倒数第1位之前的数据要求第一位必须比第二位小(前后都是正数或者负数)

返回新的数组；

## 8）splice()

1.arr.splice(0,3); 从第0位开始删除三个元素，返回被删除的数据集合数组

2.arr.splice(1,0,5);从第一位开始插入一个数字5，返回空数组；

3.arr.splice(2,2,7,9);从第二位开始，将两个元素替换为7和9。

返回从原数组中删除的项，如果没有删除，直接返回空数组。

## 9）indexOf()和 lastIndexOf() （ES5新增）

1.Var index=arr.index(1), 查找元素为1的下标。

2.Var index=arr.index(1，2), 查找元素为1的下标，从第2位开始，离第二位最近的。

如果没有查找到，返回为-1；

而lastIndexOf()表示从最后一位开始查找；

## 10）forEach() （ES5新增）

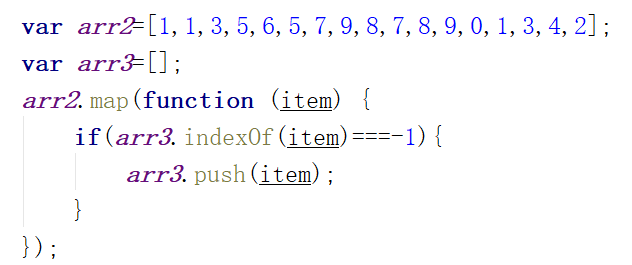
arr.forEach(function(item,index,arr){ }).遍历数组；

## 11）map() （ES5新增）

Arr.map(function(item,index,arr){

}) ，

返回新的数组，用来加工数组,如：return item\*2;则返回的数组是元数组值得两倍；forEach为正常遍历；



## 12）filter() （ES5新增）

Let arr=[1,2,3,4,5,6];

arr.filter(function(item,index,arr){

Return index>3;

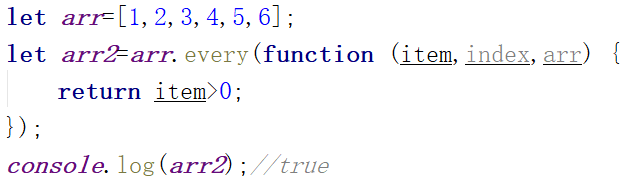
})

Console.log(arr) //[4,5,6]

遍历一个数组，返回满足条件的数组；

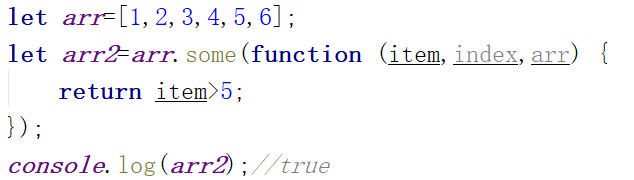
## 13）every() （ES5新增）

判断数组中每一项都是否满足条件，只有所有项都满足条件，才会返回true。



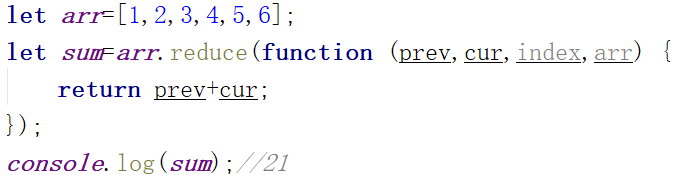
## 14）some() （ES5新增）

判断数组中是否存在满足条件的项，只要有一项满足条件，就会返回true。



## 15）reduce()和 reduceRight() （ES5新增）

这两个方法都会实现迭代数组的所有项，然后构建一个最终返回的值。reduce()方法从数组的第一项开始，逐个遍历到最后。而 reduceRight()则从数组的最后一项开始，向前遍历到第一项。



## 附加说明（十分重要）：

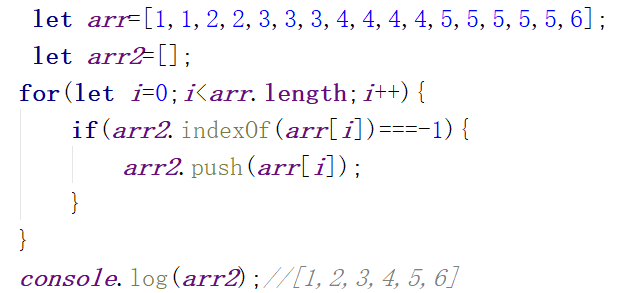
Js的迭代方法，就是指数组方法里面的**forEach,map,filter,every,some,**(reduce也是，不过大部分情况不包含她，若是问到，回答前面5个就ok);

还有一个数组方法，Array.from(arr);可以将伪数组转换为数组，使用数组的一般方法。

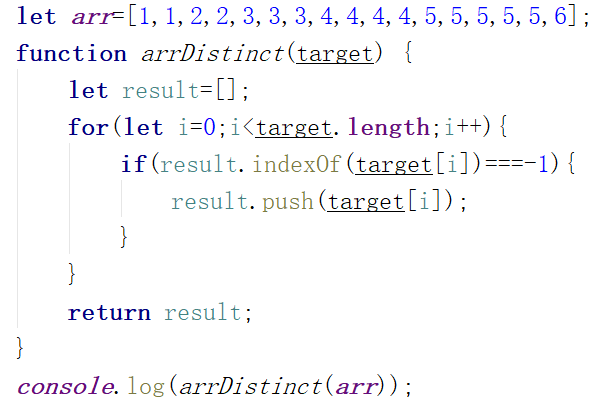
# 3.数组去重和排序

数组去重有6大方法。这里值介绍普通的一种和ES6的一种。

## 普通



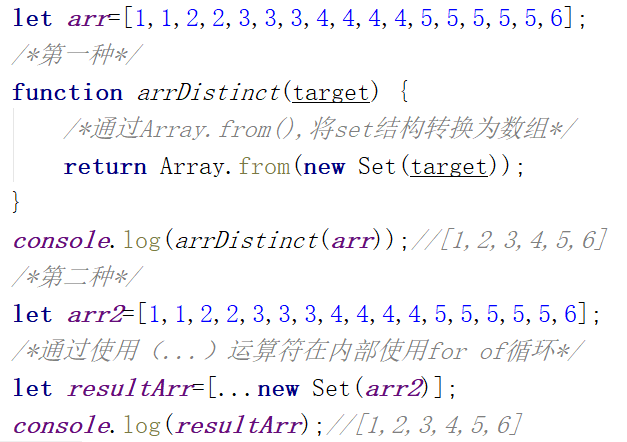
上图中的示例代码十分的简单，通过for循环遍历，比较arr2里面是否有arr[i]项。如果没有就添加，如果有就不添加。当然，为了引起注意（装大拿），我们可以封装成函数。代码如下：



这就显得代码结构十合理，而且减少了代码耦合，提高了复用性；

## 2）ES6方法

ES6方法十分简单，但是低版本的浏览器不支持。作为加分项，不会不要画蛇添足；



以上两种方法可完美应对去重。For of见后面《ES6常见问题》。

# 4.字符串方法

## 1）获取字符的位置和Unicode编码转换

str.charAt(0) 返回字符串第几位的字符元素，与str[0]相同；

str.charCodeAt();将字符转换为unicode编码

str.fromCharCode();将Unicode编码转换为字符

65-91:A-Z; 97-123:a-z; 48-58:0-9;

## 2）concat，indexof

用法与数组的相同。

## 3）其他方法

### 1.replace

str1=str.replace("c","e"); 替换字符

原字符不改变，返回一个替换后的新字符;

### 2.转换大小写

toUpperCase() 返回大写字符 原字符不改变

toLowerCase() 返回小写字符 原字符不改变

### 3.split

str.split("|"); 按照查找到分割字符，分割为数组

### 4.截取字符串

1） str.slice 和数组用法相同  
2）substr(开始位置,截取长度);开始位置可以是正数或者负数，截取长度只能是正数.

3）str.substring(开始位置，结束位置)  
 只能取正值不能取负值,负值都是0，下标可以前小后大，也可以前大后小，

不包含最后一位；

# 5.正则表达式

正则表达式面试不会问的很难，但是基础的东西必须都懂，才能从容应对。

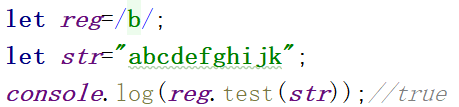
Ps:主要问题集中在去除空格；

基础表达式：let reg=/正则表达式/;

## 1）基本方法

### （1）test方法

判断内容中是否有符合正则对象中的验证内容；返回true或false；



上述示例中，str中包含“b”，故返回值为true；

### （2）exec

判断字符串中是否有符合正则表达对象中的验证内容，并返回数组；数组中有查找的正则值item，索引index，以及input（实质就是完整的str），如果有多项返回多个数组，如果没有返回null；

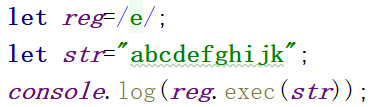
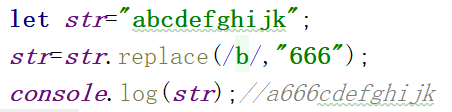
 

图1 示例代码 图2 返回结果

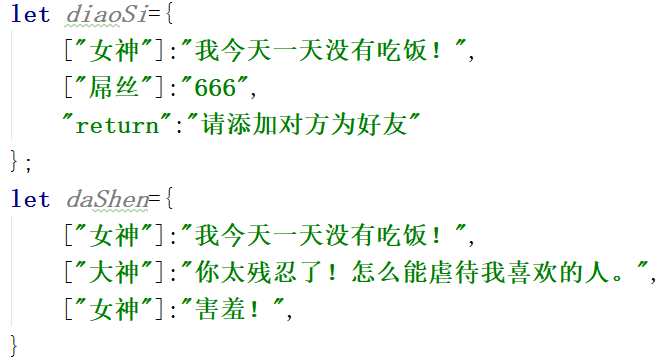
## 2）字符串的方法

### （1）repalce



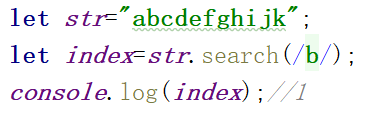
可以明显看到，repalce将b替换成了666；

**轻松一刻：**



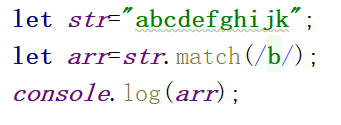
### （2）search

返回正则的索引值。



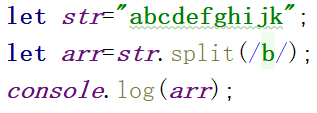
### （3）match

返回查找的结果，格式同正则的exec方法；



### （4）split

按照正则切割成数组。

## 3）字符与元字符

字符就是普通的字符数字，元字符就是代表特殊含义的字母；元字符要使用的话需要使用“\”来进行转义。

“.”点可以代表任何符号，它在正则里面的名称叫做通配符。如果要查找“.”而不是使用通配符，那我们就要使用“\”来对“.”进行转义。

## 4）[ ]中括号及相关用法

### （1）基础用法

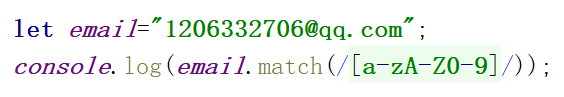
[qmn] 一个中括号内的字符只表示一个字符的范围，这个字符可以是q或m或n；

[a-z] 表示所有的小写字母；

[A-Z] 表示所有的大写字母；

[0-9] 表示0-9所有的数字；

例子：



例子中的正则格式可以匹配任何字母或数字；

### （2）深入用法

[ ^a ] 表示出a以外的字符，如果要单纯的使用“^”则需要转义；

[ . ] 表示的是字符的“.”，不是通配符

[ [ ] ] 匹配中括号需要使用转义 [ \[ \] ];

\w a-zA-Z0-9

\W 除a-zA-Z0-9以外的

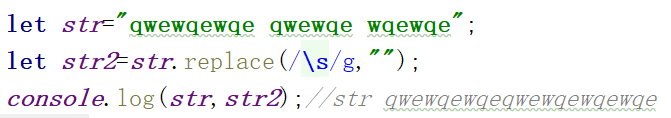
\d 0-9

\D 除了0-9以外的

\s 空字符串

\S 除了空字符串

例子：

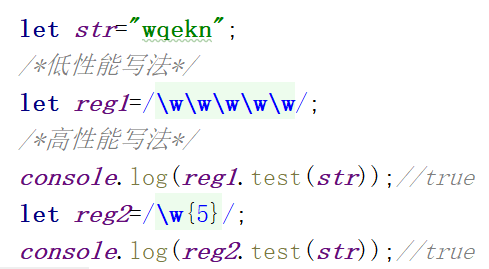


实例代码中可以看到，我们将空格替换成“”空串来去掉空格；其中g代表全局匹配；i代表不区分大小写匹配；m代表可以进行多行匹配；

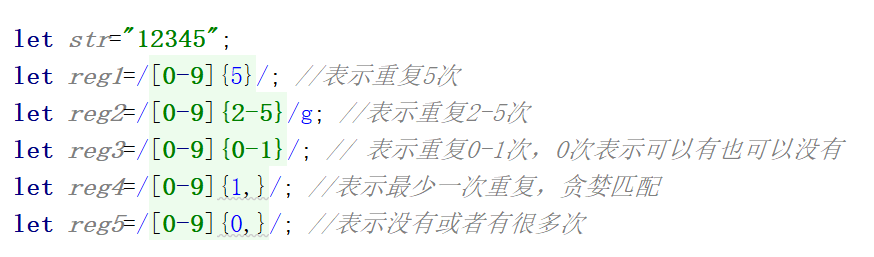
## 5）重复、贪婪、非贪婪

### （1）重复

当匹配的代码有多个时，使用{个数} 来实现重复的次数，减少代码量。



### （2）贪婪、非贪婪

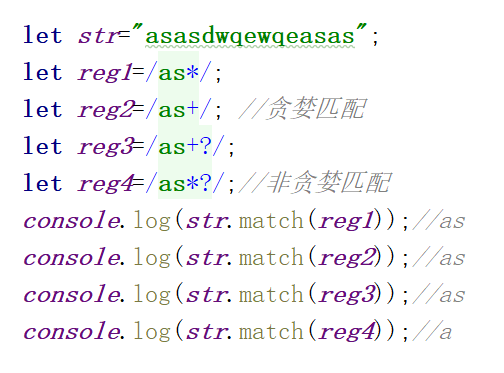
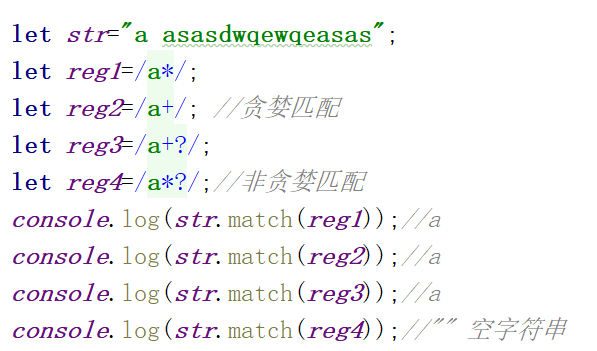


？ {0-1}

+ {1，}

\* {0，}

给最后加？表示非贪婪匹配；



通过上面两个例子可看出，贪婪匹配就是要匹配到最大的，能匹配多少就匹配多少。而非贪婪匹配就是只要匹配到就好，可以不匹配就不匹配；

## 6）或者“|”

“|”代表或者，如果在中括号中，它就只代表单纯的字符“|”；

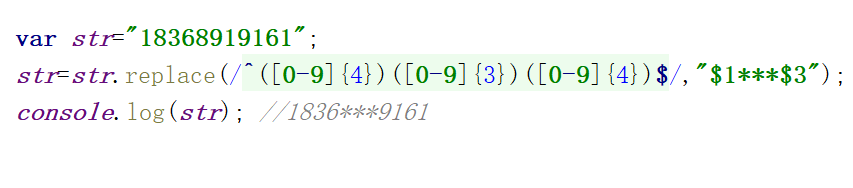
Let reg=/a|b/;表示a或b；

## 7）分组

^ 起始

$ 结束

$1,第一组



# 附加1：面试邮箱验证

正则一般出现的笔试中，面试较少，其中邮箱基本为考题。

^[a-zA-Z0-9\_.-][+@[a-zA-Z0-9-]+(\.[a-zA-Z0-9-]+)\*\.[a-zA-Z0-9]{2,6}$](mailto:+@[a-zA-Z0-9-]+(\.%5ba-zA-Z0-9-%5d+)*\.%5ba-zA-Z0-9%5d%7b2,6%7d$)

解释：开头的结尾的^$分别为开始符和结束符，第一个中括号内的表示所有的数字，小写字母，大写字母，下划线，点，减号。后面的加号表示{1，}，即最少重复一次；后面的\.为转义符。将通配符点转义为普通的符号点。@后面的中括号表示所有的大小写字母，数字。后面中括号的，如果邮箱有两个后缀名，那么可以协商，其中\*表示可以有也可以没有。

# 附加2：将字符串”<tr><td>{$id}</td><td>{$name}</td></tr>”中的{$id}替换成10，{$name}替换成Tony （使用正则表达式）

"<tr><td>{$id}</td><td>{$id}\_{$name}</td></tr>".replace(/{\$id}/g,'10').replace(/{\$name}/g, 'Tony');

# 附加3：写一个function，清除字符串前后的空格。

# 