

# 20190724课堂笔记

## 数组去重

```
1 let ary = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1, 2, 3];
2 // 1.创建一个空对象
3 let obj = {};
4 // 2.循环数组中的每一项，把每一项向对象中进行存储 => item:item
5 for (let i = 0; i < ary.length; i++) {
6     let item = ary[i];
7     // 3.每一次存储之前进行判断:验证obj中是否存在这一项
8     if (obj[item] !== undefined) {
9         // 已经存在这一项
10        ary.splice(i, 1);
11        i--;
12        continue;
13    }
14    obj[item] = item;
15 }
16 console.log(ary);
```

基于splice实现删除性能不好：当前项被删后，后面每一项的索引都要向前提一位，如果后面内容过多，一定影响性能

```
1  /*
2   * unique: 实现数组去重的方法
3   * @params
4   *     ary [Array] 要去重的数组
5   * @return
6   *     [Array] 去重后的数组
7   * by zhouxiaotian on 20190724
8   */
9  function unique(ary) {
10      let obj = {};
11      for (let i = 0; i < ary.length; i++)
12      {
13          let item = ary[i];
14          if (obj[item] !== undefined) {
15              ary[i] = ary[ary.length -
16  1];
17              ary.length--;
18              i--;
19              continue;
20          }
21          obj[item] = item;
22      }
23      return ary;
24  }
25  let aa = [12, 23, 12, 15, 25, 23, 25,
26  14, 16];
27  aa = unique(aa);
28  aa.sort((a, b) => a - b);
29  console.log(aa); //=>[12, 14, 15, 16,
30  23, 25]
```

# 字符串中常用的方法

所有用 单引号、双引号、反引号 包起来的都是字符串

```
1 let str = 'zhufengpeixunyangfanqihang';
2 // 每一个字符串都是由零到多个字符组成的
3 str.length //=>字符串长度
4 str[0] //=>获取索引为零（第一个）字符
5 str[str.length-1] //=>获取最后一个字符
   str.length-1最后一项索引
6 str[10000] //=>undefined 不存在这个索引
7
8 //循环输出字符串中的每一个字符
9 for (let i = 0; i < str.length; i++) {
10     let char = str[i];
11     console.log(char);
12 }
```

charAt / charCodeAt

```
1 /*
2  * charAt: 根据索引获取指定位置的字符
3  * charCodeAt: 获取指定字符的ASCII码值
   (Unicode编码值)
4  * @params
5  *     n [number] 获取字符指定的索引
6  * @return
7  *     返回查找到的字符
8  *     找不到返回的是空字符串不是undefined,
   或者对应的编码值
```

```

9  */
10 let str = 'zhufengpeixunyangfanqihang';
11 console.log(str.charAt(0)); //=>'z'
12 console.log(str[0]); //=>'z'
13 console.log(str.charAt(10000)); //=>' '
14 console.log(str[10000]); //=>undefined
15 console.log(str.charCodeAt(0)); //=>122
16 console.log(String.fromCharCode(122));
   //=>'z'

```

### substr / substring / slice

```

1  /*
2   * 都是为了实现字符串的截取（在原来字符串中查找
   到自己想要的）
3   *    substr(n,m): 从索引n开始截取m个字符，m
   不写截取到末尾（后面方法也是）
4   *    substring(n,m): 从索引n开始找到索引为m
   处(不含m)
5   *    slice(n,m): 和substring一样，都是找到
   索引为m处，但是slice可以支持负数作为索引，其余
   两个方法是不可以的
6  */
7  let str = 'zhufengpeixunyangfanqihang';
8  console.log(str.substr(3, 7));
   //=>'fengpei'
9  console.log(str.substring(3, 7));
   //=>'feng'
10 console.log(str.substr(3));
   //=>'fengpeixunyangfanqihang' 截取到末尾

```

```

11 console.log(str.substring(3, 10000));
    //=>'fengpeixunyangfanqihang' 截取到末尾
    (超过索引的也只截取到末尾)
12
13 console.log(str.substring(3, 7));
    //=>'feng'
14 console.log(str.slice(3, 7)); //=>'feng'
15 console.log(str.substring(-7, -3));
    //=>' ' substring不支持负数索引
16 console.log(str.slice(-7, -3));
    //=>'nqih' slice支持负数索引 =>快捷查找: 负
    数索引, 我们可以按照 STR.LENGTH+负索引 的方式
    找    =>slice(26-7,26-3)    =>slice(19,23)

```

## indexOf / lastIndexOf / includes

```

1  /*
2   * 验证字符是否存在
3   *    indexOf(x,y): 获取x第一次出现位置的索引, y是控制查找的起始位置索引
4   *    lastIndexOf(x): 最后一次出现位置的索引
5   *    =>没有这个字符, 返回的结果是-1
6   */
7  let str = 'zhufengpeixunyangfanqihang';
8  console.log(str.indexOf('n')); //=>5
9  console.log(str.lastIndexOf('n'));
    //=>24
10
11 console.log(str.indexOf('@')); //=>-1 不存在返回-1
12 if (str.indexOf('@') === -1) {

```

```

13      // 字符串中不包含@这个字符
14  }
15
16  console.log(str.indexOf('feng')); //=>3
    验证整体第一次出现的位置，返回的索引是第一个字符
    所在位置的索引值
17  console.log(str.indexOf('peiy')); //=>-1
18
19  console.log(str.indexOf('n', 7)); //=>12
    查找字符串索引7及之后的字符串中，n第一次出现的位
    置索引
20
21  if (!str.includes('@')) {
22      console.log('当前字符串不包含@');
23  }

```

## toUpperCase / toLowerCase

```

1  /*
2   * 字符串中字母的大小写转换
3   *   toUpperCase(): 转大写
4   *   toLowerCase(): 转小写
5   */
6  let str = 'ZhuFengPeiXunYangFanQiHang';
7  str = str.toUpperCase();
8  console.log(str);
    //=>'ZHUFENGPEIXUNYANGFANQIHANG'
9
10 str = str.toLowerCase();
11 console.log(str);
    //=>'zhufengpeixunyangfanqihang'

```

```
12
13 // 实现首字母大写
14 str = str.substr(0, 1).toUpperCase() +
    str.substr(1);
15 console.log(str);
    //=>'Zhufengpeixunyangfanqihang'
```

## split

```
1  /*
2   * split([分隔符]):把字符串按照指定的分隔符拆
   分成数组（和数组中join对应）
3   *
4   * split支持传递正则表达式
5   */
6  // 需求：把|分隔符变为,分隔符
7  let str = 'music|movie|eat|sport';
8  let ary = str.split('|'); //=>["music",
   "movie", "eat", "sport"]
9  str = ary.join(',');
10 console.log(str);
    //=>"music,movie,eat,sport"
```

## replace

```

1  /*
2   * replace(老字符,新字符): 实现字符串的替换 (经常伴随着正则而用)
3   */
4  let str = '珠峰@培训@扬帆@起航';
5  // str = str.replace('@', '-');
6  // console.log(str); //=>"珠峰-培训@扬帆@起航" 在不使用正则表达式的情况下, 执行一次REPLACE只能替换一次字符
7
8  str = str.replace(/@/g, '-');
9  console.log(str); //=>珠峰-培训-扬帆-起航

```

match

localCompare

trim / trimLeft / trimRight

...

控制台输出 String.prototype 查看所有字符串中提供的方法

## 实现一些常用的需求

时间字符串的处理

```

1  let time = '2019-7-24 12:6:23';
2  //=> 变为自己需要呈现的格式, 例如:

```



```

3 // "2019年07月24日 12时06分23秒"
4 // "2019年07月24日"
5 // "07/24 12:06"
6 // ...
7
8 //不足十位补零的方法
9 let addZero = val => val.length < 2 ?
  '0' + val : val;
10
11 //处理方式
12 let ary = time.split(/(?: |-|:)/g);
13 //=>["2019", "7", "24", "12", "6", "23"]
14 time = ary[0] + '年' + addZero(ary[1]) +
  '月' + addZero(ary[2]) + '日';

```

实现一个方法 queryURLParameter 获取一个URL地址问号后面传递的参数信息

```

1 /*
2  * queryURLParams: 获取URL地址中间号传参的信息和哈希值
3  *   @params
4  *     url [string] 要解析的URL字符串
5  *   @return
6  *     [object] 包含参数和哈希值信息的对象
7  * by zhouxiaotian on 2019/07/24
8  * 16:29:00
9  */
10 function queryURLParams(url) {
  //1. 获取?和#后面的信息

```

```
11     let askIn = url.indexOf('?'),
12         wellIn = url.indexOf('#'),
13         askText = '',
14         wellText = '';
15     // #不存在
16     wellIn === -1 ? wellIn = url.length
17 : null;
18     // ?存在
19     askIn >= 0 ? askText =
url.substring(askIn + 1, wellIn) : null;
20     wellText = url.substring(wellIn +
21 1);
22     //2. 获取每一部分信息
23     let result = {};
24     wellText !== '' ? result['HASH'] =
wellText : null;
25     if (askText !== '') {
26         let ary = askText.split('&');
27         ary.forEach(item => {
28             let itemAry =
item.split('=');
29             result[itemAry[0]] =
itemAry[1];
30         });
31     }
32     return result;
33 }
34 /*
```

```

35 //基于正则封装的才是最完美的
36 function queryURLParams(url) {
37     let result = {},
38         reg1 = /(?:^?=&#)+=(?:^?=&#)+)/g,
39         reg2 = /#(?:^?=&#)+)/g;
40     url.replace(reg1, (n, x, y) =>
41         result[x] = y);
42     url.replace(reg2, (n, x) =>
43         result['HASH'] = x);
44     return result;
45 }
46 */
47
48 let aa =
49     'http://www.zhufengpeixun.cn/index.html?
50     lx=1&name=zhufeng&teacher=aaa#box';
51 let paramsObj = queryURLParams(aa);
52 console.log(paramsObj);

```

实现一个最LOW的验证码：数字+字母共四位

验证码目的：防止外挂程序恶意批量注入的

```

1 <body>
2     <input type="text" id="codeInp">
3     <br>
4     <span id="codeBox">AAAA</span>
5     <button id="changeCode">看不清换一张
6 </button>

```

```
6
7      <!-- IMPORT JS -->
8      <script>
9          let codeInp =
document.getElementById('codeInp'),
10              codeBox =
document.getElementById('codeBox'),
11              changeCode =
document.getElementById('changeCode');
12
13          /*
14              * queryCode: 获取到四位随机验证码,
然后放到指定的盒子中
15              * @params
16              * @return
17              * by Team on 2019/07/24
18              */
19          function queryCode() {
20              // 准备获取范围的字符串 0~61
21              let area =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
22              let result = "";
23              for (let i = 0; i < 4; i++)
{
24                  // 每一次循环都获取一个随机
的数字索引
25                  let ran =
Math.round(Math.random() * 61);
```

```
26          // 再根据获取的索引从范围字
字符串中找到对应的字符，把找到的字符拼接到最后的结
果中
27          result +=
area.charAt(ran);
28      }
29      // 放到盒子里面
30      codeBox.innerHTML = result;
31  }
32
33      // 第一次加载页面需要执行方法，让其显
示在页面中
34      queryCode();
35
36      // 点击看不清按钮，需要重新执行方法生
成新的验证码
37      changeCode.onclick = queryCode;
38
39      // 文本框失去焦点的时候：验证用户输入
的内容和验证码是否相同，给予相关的提示，如果不一
样需要重新生成验证码
40      // onBlur: 文本框失去焦点事件
41      codeInp.onblur = function () {
42          // 获取用户和验证码内容（表单元
素.value / 非标单元素.innerHTML 获取内容）
43          let val = codeInp.value,
44              code =
codeBox.innerHTML;
45          // 不区分大小写的验证（都转成小
写）
```

```
46         if (val.toLowerCase() ===  
code.toLowerCase()) {  
47             alert('温馨提示：验证码输入  
成功！');  
48         } else {  
49             alert('温馨提示：验证码输入  
有误，请重试！');  
50             codeInp.value = '';  
51             // 重新生成验证码  
52             queryCode();  
53         }  
54     }  
55     </script>  
56 </body>
```

思考题：

1. slice(n,m) 探索n/m各种情况下的结果都是啥
2. 简单了解一下数组深度克隆
3. 案例扩展
  - 数组去重可以尝试研究更多方式
  - 时间字符串格式化这个案例，也可以尝试多种方式
  - 验证码案例实现四位不重复的验证码（单词A和a算重复）
4. 赶紧复习吧宝宝们

