前端框架api使用说明文档

8. `isJSON(str)`
9. `RecursiveEliminationZero(value)`
10. `toThousands(num)`
11. `thousands2Num(str)`
12. `sestrictType(id)`

日期时间

1. `stampToDate(newstime, type)`
2. `timeStampNow()`
3. `timeStamp(time)`
4. `getEndTime(endTime)`
5. `getWeekFromDate(dateString)`
6. `addOneWeek(year, weeks)`
7. `subOneWeek(year, weeks)`
8. `getNowFormatDate()`
9. `getNewData(dateTemp, days)`
10. `dateDifference(sDate1, sDate2)`
11. `dateDiffFromDateString(sDate1, sDate2)`
12. `dateDiffFromDate(date1, date2)`
13. `getDatesFromWeek(year, weeks)`
14. `getNowDate()`
15. `addDate(date, days)`
16. `GetDateStr = function (date, AddDayCount)`
17. `datedifference(sDate1, sDate2)`
18. `timestampToDate(shijian)`
19. `getNMYear(time, n)`
20. `getNMCYear(time, n)`

数组方法

1. `uniqArr(array)`
2. `groupBy(array, f)`
3. groupBy2Arr(array, f)
4. groupByObj(array, f)
5. groupByFiled(array, f)
6. getNewArr(dataArr, repeat, minChar)
7. Array.prototype.uniquelize
8. Array.complement
9. Array.prototype.contains(obj)
10. Array.intersect(a, b)
11. Array.minus
12. Array.union = function (a, b)
13. isArray(obj)
14. isInArray(arr, value) ??
15. posInArray(arr, value) {
16. isInArray3(arr, value)
17 removeArray(val)
18. pushArrayBypos(dataArray, obj, pos)
19. updateJsonArrayByPos(dataArray, obj, pos)
20. function objsum(array, key)
21. sortObjectArray(objArr, keyArr, type)
22. updateJsonArrayById(dataArray, obj)
23. updateJsonArrayByField(dataArray, obj, field)
24. mergeTwoArray(dataArray, dataArray2, field)
25. mergeTwoNewArray(dataArray, dataArray2, field, deleteIds)
26. removeJsonArrayByPos(dataArray, pos)
27. removeJsonArrayByPosArr(dataArray, posArr)
28. getPosArrayById(dataArray, id)
29. getLastPosByField(dataArray, field, value)

30.getLastSonPosArrayById(dataArray, id)
31.getJsonArrayById(dataArray, id)
32.getObjFromArrayById(dataArray, id) {
33.getObjFromArrayByField(dataArray, field, value) {
34.getOneObjFromArrayByCondition(dataArray, conditions)
35.deleteJsonArrayById(dataArray, id)
36.deleteArrayByFiled(dataArray, value, filed)
37.deleteArrayByValue(dataArray, value) {
38.getDeleteArrayIdsByFiled(dataArray, value, filed) {
39.isEqualOfTwoObject(obj1, obj2, fieldArray)
40.isIncludeOfArray(array, obj, fieldArray) {
41.deleteObjectById(dataArray, id)
42.getSonArray(dataArray, id) {
43.isFromJsonByField(dataArray, field, value)

其他方法

1.randomRange(start, end)
2.browserInfo() {
3.getweb()
4.toUrl(url, self, clearRole=true, obj) {
5.uuid()
6.isIEBroswer()
7..$("textarea").autoHeight()
8.String.prototype.colorHex = function () {
9.String.prototype.colorRgb = function () {
10.GetUrlRelativePath()

核心框架API：

列表方法：

1.`getListBlockData(listId, fomId)`

- 根据配置自动从获取数据，渲染列表
- 表格可配置分页，系统会自动渲染分页，添加分页事件
- 可配置基础数据，系统自动渲染（下拉框）
- 可配置接口筛选，排序
- 渲染页面前后会调用berfor、after生命周期方法
- 如果没有配置列表模板，第一次运行会从页面获取模板并保存到配置

```
getListBlockData('qualifiedSupplierList');
```

2.`renderListByData(listId, data, formId)`

使用传入的数据渲染列表，其他同getListBlockData

```
//列表新增一条数据示例
var dataList = getJsonFromListForm('dataAttachmentId', 'contentForm');
dataList.push({
    title: title,
    id: uuid(),
    publicTime: getNowFormatDate(),
    fileId: fileId
})
renderListByData('dataAttachmentId', 'contentForm', dataList)
```

3. `getJsonFromListForm(tableListId, formId)`

获取listForm数据，返回对象数组【{}，{}】

4. `checkListFormMustInput(tableListId, formId)`

检查必填项，根据mustInput配置

5. `returnListTempByData(listId, listInfos)`

结合数据和配置模板，返回列表dom。注意，如果没有配置模板，该方法不会从页面上获取模板

form方法:

1. `serializeForm(formId)`

获取表单数据

- 如果formId存在，会把数据缓存在 `dataCenter['form'][formId]['down']['formInfo']`
- 对与id字段，如果配置 `dataCenter['form'][formId].infoId` 存在，会使用infoId，而不会使用页面上的

2. `getAllFormBlockData(formId)`

根据配置，从后台获取数据，渲染表单

3. `renderFormByData(formId,data)`

根据传入数据渲染页面

4. `renderForm（formId）`

根据配置渲染页面,一般是先赋值 `dataCenter['form'][formId]['down']['formInfo']` 之后调用该方法。

5. `submit(formId, errfun)`

提交form,调用前后后调用beforSubmit，afterSubmit生命周期方法

6. `submitListForm(listId,formId,failCallback,err)`

整体提交列表中的所有form

8. `checkMustInput(formId)`

提交时检验必填项,将所有的没有填写的必填项返回

9. `checkMaybeInput(formId)`

提交时检验可填项项,将所有的没有填写的可填项返回

接口方法:

1.`saveOrUpdate(classId, data, callback)`

保存或者更新，data.id存在时是更新

```
// 更新表单的示例
var data = serializeForm("WXJTransportationInformationForm");
data.id = dataCenter.form.mainForm.infoId
saveOrUpdate(sysJson.classid.component, data, function () {

});
```

2.`getInfoListByClassId(status, classid, callback)`

查询列表信息

```
//根据条件查询
var supplyArea = $("select[name=supplyArea]").val();
getInfoListByClassId({
    "searchField": "supplyArea,supplierStatus",
    "searchValue": supplyArea+",合格供应商",
    "searchCondition":"and$$$=,and$$$=",
},sysJson.classid.supplier,function(result){

})
```

3.`updateInfoByClassId(classid, InfoJson, callback)`

根据classid更新表单信息

```
//更新数据
var id = dataCenter.other.id;
updateInfoByClassId(sysJson.classid.fixedPoint, JSON.stringify({
    id: id,
    status: "已关闭"
}), function () {
    toUrl("./designatedListOfYCLList.html", true);
})
```

4.`updateListInfoByClassId(classid, InfoJsonList, callback)`

根据classid更新数组信息,不依赖页面，可以修改数据

```
//更新列表
var dataJson = getJsonFromListForm('dataList', 'contentForm');
for (var i = 0; i < dataJson.length; i++) {
    dataJson[i].id = dataJson[i].id;
    dataJson[i].companyName = dataJson[i].fixedPointSupplier;
    dataJson[i].fixedPointTime = getNowFormatDate();
    dataJson[i].area = dataCenter.form['mainForm'].down.formInfo.area;
    var array = [{
```

```
        year: new Date().getFullYear(),
        YPrice: dataJson[i].finalFixedPrice,
    }]
    dataJson[i].historyPriceJson = html2Escape(JSON.stringify(array));
}
updateListInfoByClassId(sysJson.classid.WXJPriceInventory,
JSON.stringify(dataJson),    function (result) {
    typeof callback != 'undefined' && callback();
})
```

5. `loopListGetData(dataA, callback, saveToField)`

根据条件同时获取多表的数据

```
//配置获取数据的列表
    var tableDatas = [
        {
            classId: sysJson.classid.MJInventory,
            search: { searchValue: projectId, searchField: 'projectId',
searchCondition: 'and$$$=', num: 1000 }
        },// 项目模具清单表20190329
        {
            classId: sysJson.classid.JJInventory,
            search: { searchValue: projectId, searchField: 'projectId',
searchCondition: 'and$$$=', num: 1000 }
        },// 项目检具清单表20190329
        .......
    ];

    loopListGetData(tableDatas, function (data, result, count) {
        if (count === tableDatas.length) {//此时所有数据已经获取完毕，获取到的信息在
infos字段上
            //数据特殊处理
            tableDatas.forEach(function (dt) {
                dt.infos ......
            })

        }
    })
```

6. `loopListInterface(dataA, callback)`

循环调接口更新数据列表(并发执行，多张表)

```
    var objArr = [{// 合同
        classId: sysJson.classid.contract,
        infos: contracts
    }, {// 模具清单
        classId: sysJson.classid.designatedListOfKHZD,
        infos: InventorList_Project
    }];
    loopListInterface(objArr, function (dataA, result, count) {
        if (count === 2) {

        }
    });
```

7. `loopListInterfaceOfDelete(dataA, callback)`

循环调接口删除数据列表(并发执行，多张表)

```
//多表批量删除的示例

// 根据条件批量获取多张表数据
    loopListGetData(tableDatas, function (data, result, count) {
        if (count === tableDatas.length) {//此时所有数据已经获取完毕，获取到的信息在
infos字段上
            // 要删除的数据
            var daleteData = tableDatas.filter(function (data) {
                return data.delInfos.length > 0;
            }).map(function (project) {
                var deleteIds = project.delInfos.map(function (info) {
                    var deleteIdsExcapt = project.deleteIdsExcapt;
                    if (deleteIdsExcapt && deleteIdsExcapt.length > 0) {
                        if (deleteIdsExcapt.indexOf(info.id) === -1) {
                            return info.id;
                        }
                    } else {
                        return info.id;
                    }
                }).filter(function (data) {
                    return data;
                })

                return {
                    classId: project.classId,
                    deleteIds: deleteIds
                }
            });
            // 删除数据
            loopListInterfaceOfDelete(daleteData, callback)
        }
    }, 'delInfos')
```

8. `saveList(listId, callback)`

保存数据列表数据，如果dataCenter.list[listId].deleteIds有数据，会自动删除

```
saveList(nowTabListId, function () {
    dataCenter.fromNode = dataCenter.fromNode.split('$$$')[0] + "$$$saved";
    reloadPage();
});
```

9.`delAllTableByCondition(tableDatas, callback)`

批量删除(多表同时)<先根据条件获取数据，然后再根据ID删除>

```
// 数据
var tableConfig = [{ infos: MJs, classId: sysJson.classid.MJInventory, search:
search
      },{//项目模具清单表
        infos: JJs, classId: sysJson.classid.JJInventory, search: search
      },{ // 项目检具清单表
          infos: WXJs, classId: sysJson.classid.WXJInventory, search: search
},{ // 项目外协件清单表
             infos: BZJs, classId: sysJson.classid.BZJInventory, search: search
         },
                  .......
    ];
// 删除配置
var tableDatasOfDelParma = tableConfig.map(function (dt) {
    //设置删除排除项
    dt.deleteIdsExcapt = dt.infos.map(function (info) {
        return info.id;
    })
    return dt;
})
delAllTableByCondition(tableDatasOfDelParma, function callback() {

})
```

10.`deleteInfoByIds(classid, infoids, success, err)`

根据classid和infoids批量删除信息

公共方法（常用的）：

字符串方法:

1.`trim(str, type)`

去除空格 type 1-所有空格 2-前后空格 3-前空格 4-后空格

```
trim('  12  23456 34 ', 1)
=> "122345634"
trim('  12  23456 34 ', 2)
=>"12  23456 34"
trim('  12  23456 34 ', 3)
=>"12  23456 34 "
trim('  12  23456 34 ', 4)
=>"  12  23456 34"
trim('  12  23456 34 ', 5)
=>"  12  23456 34 "
```

2.`upDigit(n)`

现金额大写转换函数

```
upDigit(168752632) => "人民币壹亿陆仟捌佰柒拾伍万贰仟陆佰叁拾贰元整"
```

3.`formatText(str, size=3, delimiter='-')`

格式化处理字符串

```
formatText('1234asda567asd890') ==>"12,34a,sda,567,asd,890"
```

4.`html2Escape(html)`

html字符串转码,对 `< > & " ' \ \\ !` 转码

5.`escape2Html(str)`

字符串解码,对 `< > & " ' \ \\ !` 解码

6.`escapeObjToHtmlObj(obj)`

转码对象转成正常对象，即根据`sysJson['decode']`，对obj中各属性界面

7.`htmlObjToEscapeObj(htmlobj)`

正常对象转成转码对象，escapeObjToHtmlObj的反操作

8.`isNotEmpty(obj)`

字符串不为空判断。undefined，null，“”，0，false都被认为是空的

```
isNotEmpty()
=> false
isNotEmpty('')
=> false
isNotEmpty(0)
=> false
isNotEmpty(false)
=> false
isNotEmpty(undefined)
=> false
```

9.`isEmpty(obj)`

字符串为空判断，isNotEmpty的取反

10. `firstCase(str)`

字符串首字符大写

```
firstCase('key')
=> "Key"
```

11. `temEnginLite(listInfos, tem)`

字符模板渲染Lite

```
var listInfos = [{title:'首页',des:'第一页',color:'#ccc'},{title:'管理',des:'管理列表',color:'#fff'}];

var tem = '<div><p>{{title}}</p><span>{{des}}</span></div>';
temEnginLite(listInfos, tem)
=> "<div><p>首页</p><span>第一页</span></div><div><p>管理</p><span>管理列表</span></div>"
```

12. `temEngin(listInfos, tem)`

字符模板渲染，处理了值为json的情况，同时会自动解压

```
var listInfos = [{title:'首页',des:'第一页',chi:'{"name":"json内容1"}'},{title:'管理',des:'管理列表',chi:'{"name":"json内容2"}'}]
var tem = '<div><p>{{title}}</p><span>{{chi.name}}</span></div>';
temEngin(listInfos, tem)
=> "<div><p>首页</p><span>json内容1</span></div><div><p>管理</p><span>json内容1</span></div>"
var listInfos = [{title:'首页',des:'第一页',chi:'{"name":"json内容1"}'},{title:'管理',des:'管理列表',chi:'{"name":"json内容2"}'}]
```

数字金额

1. `checkType(str, type)`

检测字符串类型，email，phone，tel，number，english，text，chinese，lower，upper

```
checkType('12423563@gmail.com', 'email')
=> true
checkType('18366880000', 'phone')
=> true
checkType('abcDEF', 'english')
=> true
checkType('阿西吧', 'chinese')
=> true
checkType('abcd', 'lower')
=> true
checkType('GOOD', 'upper')
=> true
checkType('0564-5391987', 'tel')
=> true
```

2.`$(selector).numeral(b,t)`

限制数字输入。b=true只能输入浮点数，t=true输入自动添加千分位分隔符

```
//只能输入数字和小数点
$("input[name=liyong]").numeral(true);
$('input[name=danjia],input[name=shuliang],input[name=danjia]').numeral(true);
```

3.`numeralBySelector(select, bl, tl, pos)`

限制金额输入、兼容浏览器、屏蔽粘贴拖拽等

select页面元素逗号隔开（或者选择器），bl是否可以输入小数，tl是千位符，pos保留几位小数，默认2位

```
numeralBySelector($('input[name=nnnn]'), false, false);
```

4.`upDigit(n)`

现金额大写转换函数

```
upDigit(168752632)
=> "人民币壹亿陆仟捌佰柒拾伍万贰仟陆佰叁拾贰元整"
upDigit(-1693)
=> "欠人民币壹仟陆佰玖拾叁元整"
```

5.`toNum(str)`

将千位符类型转成普通数字

```
toNum('12432,25345,46')
=> 124322534546
```

6.`numtoth(num)`

千分位分割，有小数位保留两位

```
numtoth(124322534546)
"124,322,534,546"
```

7.`checkNumber(theObj)`

验证字符串是否是数字

```
checkNumber(124322534546)
=> true
checkNumber('12432-25345-46')
=> false
```

8.`isJSON(str)`

判断是否是json字符串

```
isJSON(123)
=> undefined
isJSON("123")
=> false
isJSON(JSON.stringify({a:1}))
=> true
```

9. `RecursiveEliminationZero(value)`

消除数字前的0---value只能是字符串?

```
RecursiveEliminationZero('001234567')
=> "1234567"
```

10. `toThousands(num)`

将普通数字转成千位符类型的数字

```
toThousands(001234567)
=> "342,391.00"
toThousands('001234567')
=> "001,234,567.00"
toThousands('001abc4567')
=> "0,01a,bc4,567.00"
```

11. `thousands2Num(str)`

将千位符类型转成普通数字

```
thousands2Num(001234567)
=> 342391
thousands2Num('001234567')
=> 1234567
thousands2Num('1234567')
=> 1234567
thousands2Num("342,391.00")
=> 342391
thousands2Num("001,234,567.00")
=> 1234567
```

12. `sestrictType(id)`

初始化页面限制输入,系统自动调用,在元素上配置

```
<input sestrictType='integer'>
等同 numeralBySelector($(item), false, false);
<input sestrictType='decimal-1'>
等同 numeralBySelector($(item), true, true, 1);
<input sestrictType='decimal-2'>
等同 numeralBySelector($(item), true, true, 2);
```

日期时间

1. `stampToDate(newstime, type)`

时间戳转化

```
timeStampNow()
==> 1594780715
stampToDate(1594780715, 1)
==> "2020-07-15 "
stampToDate(1594780715, 2)
==> "2020-07-15 10:38:"
stampToDate(1594780715)
==> "2020-07-15 10:38:35"
```

2. `timeStampNow()`

当前时间戳（10位）

```
timeStampNow()
==> 1594780843
```

3. `timeStamp(time)`

时间转化为时间戳

```
timeStamp('2020-11-11')
==> 1605024000
timeStamp('2020/11/11')
==> 1605024000
timeStamp('2020-07-15 10:38:35')
==> 1594780715
```

4. `getEndTime(endTime)`

到某一个时间的倒计时

```
getEndTime('2027/7/22 16:0:0')
==> {d: 2563, h: 5, m: 14, s: 39}
```

5. `getWeekFromDate(dateString)`

根据日期获取周数。日期的格式是"2018-12-11"，返回值是数组，数组第一个值是年，第二个值是周

```
getWeekFromDate("2018-12-11")
==> [2018, 50]
getWeekFromDate("2027/7/22")
==>[2027, 26]
```

6. `addOneWeek(year, weeks)`

周数加一

```
addOneWeek(2020,11)
=> [2020, 12]
addOneWeek(2020,53)
=> [2021, 1]
```

7. `subOneWeek(year, weeks)`

周数减一

```
subOneWeek(2020,44)
=> [2020, 43]
subOneWeek(2020,1)
=> [2019, 53]
```

8. `getNowFormatDate()`

确认日期

```
getNowFormatDate()
=> "2020-07-15"
```

9. `getNewData(dateTemp, days)`

计算日期加上指定天数得到新的日期

```
getNewData("2020-07-15", 10)
=> "2020-07-25"
getNewData("2020-07-15", 100)
=> "2020-10-23"
getNewData("2020/07/15", 100)
=> "2020-10-23"
getNewData("2020-07-15", -100)
=> "2020-04-06"
```

10. `dateDifference(sDate1, sDate2)`

两个时间相差天数 兼容firefox chrome。sDate1和sDate2是2006-12-18格式

```
getNewData("2020/07/15", 100)
=> "2020-10-23"
dateDifference("2020/07/15", "2020-10-23")
=> 100
```

11. `dateDiffFromDateString(sDate1, sDate2)`

计算两个日期之间的天数 。sDate1和sDate2是2017-9-25格式

```
dateDiffFromDateString("2020/07/15", "2020-10-23")
=> -100
```

12. `dateDiffFromDate(date1, date2)`

计算两个日期之间的天数，sDate1和sDate2是13位时间戳格式

```
var t1 = timeStamp('2020/11/11')*1000
var t2 = timeStamp('2020-07-15 10:38:35')*1000

dateDiffFromDate(t1, t2)
=> 118
dateDiffFromDate(t2, t1)
=>118
```

13. `getDatesFromWeek(year, weeks)`

根据年和周数获取当周的天数区间

```
getDatesFromWeek(2020, 20)
=> ["2020-05-10", "2020-05-11", "2020-05-12", "2020-05-13", "2020-05-14", "2020-05-15", "2020-05-16"]
```

14. `getNowDate()`

获取当前时间字符串

```
getNowDate()
"2020-07-15 14:01:28"
```

15. `addDate(date, days)`

日期，在原有日期基础上，增加days天数，默认增加1天

```
addDate('2020-11-1', 22)
=> "2020-11-23"
addDate('2020-11-1', -2)
=> "2020-10-30"
```

16. `GetDateStr = function (date, AddDayCount)`

获取AddDayCount天后的日期

```
Date()
=> "Wed Jul 15 2020 11:22:57 GMT+0800 (中国标准时间)"
GetDateStr(Date(),10)
=> "2020-07-25"
GetDateStr("2020-07-25",10)
=> "2020-08-04"
```

17. `datedifference(sDate1, sDate2)`

计算两个时间相差天数

```
datedifference("2020-08-04", "2020-08-06")
=> 2
```

18. `timestampToDate(shijian)`

时间戳(10位)转时间

```
timestampToDate(Date.parse("2020-08-04")/1000)
"2020-08-04 08:00:00"
```

19.`getNMYear(time, n)`

获取当月后n个月的年份,time只能是 YYYY-MM-DD

```
getNMYear("2020-08-04", 2)
=>2020
getNMYear("2020-08-04", 12)
=>2021
getNMYear("2020-08-04", 22)
=>2022
```

20.`getNMCYear(time, n)`

获取当月后n个月的完整年月日,time只能是 YYYY-MM-DD

```
getNMCYear("2020-08-04", 2)
=>"2020-10-04"
getNMCYear("2020-08-04", 12)
=>"2021-08-04"
getNMCYear("2020-08-04", -2)
=>"2020-06-04"
```

数组方法

1.`uniqArr(array)`

数组去重

```
uniqArr([1,2,3,'3',3,4])
=> [1, 2, 3, "3", 4]
```

2.`groupBy(array, f)`

将数组按条件分组,返回数组

```
const a = [{class:'1',name:'zhangsan',age:8},{class:'1',name:'lishi',age:7},
{class:'2',name:'lebulong',age:7},{class:'2',name:'mayu',age:8}]
//安class分组
groupBy(a,data=>data.class)
=>
[[{"class":"1","name":"zhangsan","age":8},{"class":"1","name":"lishi","age":7}],
[{"class":"2","name":"lebulong","age":7},{"class":"2","name":"mayu","age":8}]]
```

3.groupBy2Arr(array, f)

同groupBy,返回数组

4.groupByObj(array, f)

分组,返回对象

```
groupByObj(a,data=>data.class)
=>
{'1':[{"class":"1","name":"zhangsan","age":8},
{"class":"1","name":"lishi","age":7}],
'2':[{"class":"2","name":"lebulong","age":7},
{"class":"2","name":"mayu","age":8}]}
```

5.groupByFiled(array, f)

同groupByObj

removeRepeatArray(arr)

```
removeRepeatArray([13,333,33,33])
=> [13, 333, 33]
a = {name:'xixi'}
removeRepeatArray([1,2,3,a,a])
=> [1, 2, 3, {…}]
```

6.getNewArr(dataArr, repeat, minChar)

数组根据某个字段或某几个字段去重

```
var b = [{class:'1',name:'zhangsan',age:8},{class:'1',name:'lishi',age:7},
{class:'2',name:'lebulong',age:8},{class:'2',name:'mayu',age:8}]

getNewArr(b,['class','age'])
=> [ {class: "1", name: "zhangsan", age: 8}, {class: "1", name: "lishi", age:
7}, {class: "2", name: "lebulong", age: 8}]
```

7.Array.prototype.uniquelize

数组中元素根据某一字段的值排序（根据up升降排序,默认升序）

```
[1,2,4,5,6,2,4].uniquelize ()
=> [1, 2, 4, 5, 6]
```

8.Array.complement

两个集合的补集

```
var a = [1,2,3,4];
var b = [3,4,5,6];
Array.complement(a,b)
```

9.Array.prototype.contains(obj)

判断数组是否包含某一元素

```
[3,4,5,6].contains(3) =>true
```

10.Array.intersect(a, b)

两个集合的交集

```
var a = [1,2,3,4];
var b = [3,4,5,6];
Array.intersect(a,b)
```

11.Array.minus

两个集合的差集

```
var a = [1,2,3,4];
var b = [3,4,5,6];
alert(Array.minus(a,b));
```

12.Array.union = function (a, b)

求两个集合的并集

```
var a = [1,2,3,4]
var b = [3,4,5,6];
Array.union(a,b);
```

13.isArray(obj)

是否是数组

14.isInArray(arr, value) ??

判断一个元素是否存在于一个数组中

15.posInArray(arr, value) {

判断一个元素是否存在于一个数组中

16.isInArray3(arr, value)

判断元素是否存在于数组中

17 removeArray(val)

删除数组指定的某个元素

18.pushArrayBypos(dataArray, obj, pos)

在数组指定的位置添加一条数据,pos为位置

19.updateJsonArrayByPos(dataArray, obj, pos)

替换数组指定位置的元素

20.function objsum(array, key)

对数组对象的某一字段求和

```
a = [ {class: "1", name: "zhangsan", age: 8}, {class: "1", name: "lishi", age:
7}, {class: "2", name: "lebulong", age: 8}];
objsum(a,'age')
=> 23
```

21.sortObjectArray(objArr, keyArr, type)

对JSON对象字符串数组进行多字段（多列）排序

- objArr: 目标数组
- keyArr: 排序字段，以数组形式传递
- type：排序方式，undefined以及asc都是按照升序排序，desc按照降序排序

22.updateJsonArrayById(dataArray, obj)

根据位id添修改JSON数据(其实是替换)

```
updateJsonArrayById([{id:1,title:'name'},{id:2,title:'hh'}], {id:1,title:'age'})
=> [{id:1,title:'age'},{id:2,title:'hh'}]
```

23.updateJsonArrayByField(dataArray, obj, field)

根据字段添修改JSON数据 ,如果没有匹配的,就添加在末尾

```
updateJsonArrayByField([{id:1,title:'name'},{id:2,title:'hh'}],
{id:1,title:'age'},'title')
=> [{id:1,title:'age'},{id:2,title:'hh'}]
```

24.mergeTwoArray(dataArray, dataArray2, field)

两个数组合并,此方法会修改第一个参数,并已经 field 字段去重

25.mergeTwoNewArray(dataArray, dataArray2, field, deleteIds)

同上,删除id值在deleteIds的元素

26.removeJsonArrayByPos(dataArray, pos)

根据位置删除JSON数据,此方法会修改原数组,返回修改后的数组

27.removeJsonArrayByPosArr(dataArray, posArr)

根据传入的posArr,返回新的数组

28.getPosArrayById(dataArray, id)

根据id得到对象在数组中的位置,找不到返回-1

29.getLastPosByField(dataArray, field, value)

根据field的值给出最后一个符合条件的对象的位置,找不到返回-1

30.getLastSonPosArrayById(dataArray, id)

根据id得到对象的最后一个儿子在数组中的位置

31.getJsonArrayById(dataArray, id)

根据id获取数据

32.getObjFromArrayById(dataArray, id) {

根据id获取数据

33.getObjFromArrayByField(dataArray, field, value) {

根据field获取数据,取不到返回空对象{}

34.getOneObjFromArrayByCondition(dataArray, conditions)

根据多个字段获取一个数据,取不到返回false

```
a = [ {class: "1", name: "zhangsan", age: 8}, {class: "1", name: "lishi", age:
7}, {class: "2", name: "lebulong", age: 8}];
objsum(a,{class: "1", name: "zhangsan"})
=> {class: "1", name: "zhangsan", age: 8}
```

35.deleteJsonArrayById(dataArray, id)

根据id属性删除数据,返回删除后的数组,同时会修改原数组

36.deleteArrayByFiled(dataArray, value, filed)

根据filed的值来删除数据,返回删除后的数组,同时会修改原数组

37.deleteArrayByValue(dataArray, value) {

根据value的值来删除数据,返回删除后的数组,同时会修改原数组

38.getDeleteArrayIdsByFiled(dataArray, value, filed) {

根据filed的值获取删除数据的id

```
//获取一年级学生的id
a = [ {class: "1", name: "zhangsan", age: 8,id:1}, {class: "1", name: "lishi",
age: 7,id:2}, {class: "2", name: "lebulong", age: 8,id:4}];
getDeleteArrayIdsByFiled(a,{class: "1", name: "zhangsan"})
=> [1,2]
```

39.isEqualOfTwoObject(obj1, obj2, fieldArray)

比较两个对象的多个字段是否相等

40.isIncludeOfArray(array, obj, fieldArray) {

判断数组对象中是否有对应项

```
//获取一年级学生的id
a = [ {class: "1", name: "zhangsan", age: 8,id:1}, {class: "1", name: "lishi",
age: 7,id:2}, {class: "2", name: "lebulong", age: 8,id:4}];
isIncludeOfArray(a,{class: "1", name: "zhangsan",age:'9'},['class','name'])
=> {class: "1", name: "zhangsan", age: 8,id:1}
```

41.deleteObjectById(dataArray, id)

删除自己后自己的后代元素(通过pid关联后代)

42.getSonArray(dataArray, id) {

根据id获取所有的子对象数组，这里只是儿子，不是子孙

43.isFromJsonByField(dataArray, field, value)

判断dataArray中是否存在 指定字段filed和字段对应的值 的对象

其他方法

1.randomRange(start, end)

范围随机数

2.browserInfo() {

浏览器信息

```
browserInfo() =>
{
    isMobi,// 是否是手机浏览器
    isWeiXin,//是否为微信浏览器
    isAppleMobileDevice,//判断是否苹果移动设备访问
    isAndroidMobileDevice,//判断是否安卓移动设备访问
    isMobileUserAgent,//判断是否移动设备访问
    type//浏览器类型. "IE"|"Firefox"|"Chrome"|"Safari"
}
```

3.getweb()

浏览器信息,返回"IE"|"Firefox"|"Chrome"|"Safari"

4.toUrl(url, self, clearRole=true, obj) {

跳转链接,url:下一个页面的url,self:是否是在本页面跳转,clearRole:清除角色信息,obj:url参数

5.uuid()

随机码,可以用做id

```
uuid()
"f38dd965-0b6c-4242-abe4-c085bac42581"
```

6.isIEBroswer()

判断是否是IE

7..$("textarea").autoHeight()

textarea高度自适应

8.String.prototype.colorHex = function () {

RGB颜色转换为16进制

9.String.prototype.colorRgb = function () {

16进制颜色转为RGB格式

10.GetUrlRelativePath()

获取当前相对路径的方法