## 时间戳转换date 格式的 js

```
return function(time){
    var oDate = new Date(time),
        oYear = oDate.getFullYear(),
        oMonth = oDate.getMonth()+1,
        oDay = oDate.getDate(),
        oTime = oYear +'-'+ getzf(oMonth) +'-'+ getzf(oDay) ;//最后拼接时间
    return oTime;
}(timestamp);

//补0操作,当时间数据小于10的时候，给该数据前面加一个0
function getzf(num){

    if(parseInt(num) < 10){
        num = '0'+num;
    }
    return num;
}
```

## zzq 写的 封装的 zudp.js(很值得学习)

```
/**
 * zudp
 * 要求，必须要有 JQuery
 */
var zudp = {
    util: {
        assert: {
            checkFunction: function (param) {
                if (!zudp.util.isFunction(param)) {
                    throw '预期参数类型为 Function，实际类型为 ' +
Object.prototype.toString.call(param);
                }
            },
            checkArray: function (param) {
                if (!zudp.util.isArray(param)) {
                    throw '预期参数类型为 Array，实际类型为 ' +
Object.prototype.toString.call(param);
                }
            }
        },
```

```
        getJqueryObjById : function(jqueryId) {
            return {
                value: $('#' + jqueryId)
            };
        },
        getJquery: function(jqueryIdentifier) {
            return $(jqueryIdentifier)
        },
        // 是否为开发模式
        isDev: function(status) {
            return zudp.global.profile === zudp.global.status.dev && status;
        },
        //是否字符串
        isString: function(o) {
            return Object.prototype.toString.call(o).slice(8, -1) === 'String'
        },
        //是否数字
        isNumber: function(o) {
            return Object.prototype.toString.call(o).slice(8, -1) === 'Number'
        },
        //是否对象
        isObj: function(o) {
            return Object.prototype.toString.call(o).slice(8, -1) === 'Object'
        },
        //是否数组
        isArray: function(o) {
            return Object.prototype.toString.call(o).slice(8, -1) === 'Array'
        },
        //是否时间
        isDate: function(o) {
            return Object.prototype.toString.call(o).slice(8, -1) === 'Date'
        },
        //是否boolean
        isBoolean: function(o) {
            return Object.prototype.toString.call(o).slice(8, -1) ===
'Boolean'
        },
        //是否函数
        isFunction: function(o) {
            return Object.prototype.toString.call(o).slice(8, -1) ===
'Function'
        },
        //是否为null
        isNull: function(o) {
            return Object.prototype.toString.call(o).slice(8, -1) === 'Null'
        },
```

```
        //是否undefined
        isUndefined: function(o) {
            return Object.prototype.toString.call(o).slice(8, -1) === 'Undefined'
        },
        isFalse: function(o) {
            if (!o || o === 'null' || o === 'undefined' || o === 'false' || o === 'NaN') return true
            return false
        },
        isTrue: function(o) {
            return !this.isFalse(o)
        },
        isIos: function() {
            var u = navigator.userAgent;
            if (u.indexOf('Android') > -1 || u.indexOf('Linux') > -1) {//安卓手机
                // return "Android";
                return false
            } else if (u.indexOf('iPhone') > -1) {//苹果手机
                // return "iPhone";
                return true
            } else if (u.indexOf('iPad') > -1) {//iPad
                // return "iPad";
                return false
            } else if (u.indexOf('Windows Phone') > -1) {//winphone手机
                // return "Windows Phone";
                return false
            }else{
                return false
            }
        },
        //是否为PC端
        isPC: function() {
            var userAgentInfo = navigator.userAgent;
            var Agents = ["Android", "iPhone", "SymbianOS", "Windows Phone", "iPad", "iPod"];
            var flag = true;
            for (var v = 0; v < Agents.length; v++) {
                if (userAgentInfo.indexOf(Agents[v]) > 0) {
                    flag = false;
                    break;
                }
            }
            return flag;
        },
```

```
        // 获取浏览器类型
        browserType: function(){
            //取得浏览器的 userAgent 字符串
            var userAgent = navigator.userAgent;
            //判断是否Opera浏览器
            var isOpera = userAgent.indexOf("Opera") > -1;
            //判断是否IE浏览器
            var isIE = userAgent.indexOf("compatible") > -1 &&
userAgent.indexOf("MSIE") > -1 && !isOpera;
            //判断是否IE的Edge浏览器
            var isEdge = userAgent.indexOf("Edge") > -1;
            //判断是否Firefox浏览器
            var isFF = userAgent.indexOf("Firefox") > -1;
            //判断是否Safari浏览器
            var isSafari = userAgent.indexOf("Safari") > -1 &&
userAgent.indexOf("Chrome") === -1;
            //判断Chrome浏览器
            var isChrome = userAgent.indexOf("Chrome") > -1 &&
userAgent.indexOf("Safari") > -1;
            if (isIE) {
                var reIE = new RegExp("MSIE (\\d+\\.\\d+);");
                reIE.test(userAgent);
                var fIEVersion = parseFloat(RegExp["$1"]);
                if(fIEVersion === 7) return "IE7";
                else if(fIEVersion === 8) return "IE8";
                else if(fIEVersion === 9) return "IE9";
                else if(fIEVersion === 10) return "IE10";
                else if(fIEVersion === 11) return "IE11";
                else return "IE7以下"//IE版本过低
            }

            if (isFF) return "FF";
            if (isOpera) return "Opera";
            if (isEdge) return "Edge";
            if (isSafari) return "Safari";
            if (isChrome) return "Chrome";
        },
        // 检查字符串是否符合某种类型
        checkStr: function(str, type) {
            switch (type) {
                // 手机号码
                case 'phone':
                    return /^1[3|4|5|7|8][0-9]{9}$/.test(str);
                // 座机
                case 'tel':
                    return /^(0\d{2,3}-\d{7,8})(-\d{1,4})?$/.test(str);
```

```
                // 身份证
                case 'card':
                    return /^\d{15}|\d{18}$/.test(str);
                // 密码以字母开头，长度在6~18之间，只能包含字母、数字和下划线
                case 'pwd':
                    return /^[a-zA-Z]\w{5,17}$/.test(str)
                // 邮政编码
                case 'postal':
                    return /[1-9]\d{5}(?!\d)/.test(str);
                // QQ号
                case 'QQ':
                    return /^[1-9][0-9]{4,9}$/.test(str);
                // 邮箱
                case 'email':
                    return /^[\w-]+(\.[\w-]+)*@[\w-]+(\.[\w-]+)+$/.test(str);
                // 金额(小数点2位)
                case 'money':
                    return /^\d*(?:\.\d{0,2})?$/.test(str);
                // 网址
                case 'URL':
                    return /(http|ftp|https):\/\/[\w\-_]+(\.[\w\-_]+)+([\w\-\.,@?^=%&:/~\+#]*[\w\-\@?^=%&/~\+#])?/.test(str)
                // IP
                case 'IP':
                    return /((?:(?:25[0-5]|2[0-4]\\d|[01]?\\d?\\d)\\.){3}(?:25[0-5]|2[0-4]\\d|[01]?\\d?\\d))/.test(str);
                // 日期时间
                case 'date':
                    return /^(\d{4})\-(\d{2})\-(\d{2}) (\d{2})(?:\:\d{2}|:(\d{2}):(\d{2}))$/.test(str) || /^(\d{4})\-(\d{2})\-(\d{2})$/.test(str)
                // 数字
                case 'number':
                    return /^[0-9]$/.test(str);
                // 英文
                case 'english':
                    return /^[a-zA-Z]+$/.test(str);
                // 中文
                case 'chinese':
                    return /^[\u4E00-\u9FA5]+$/.test(str);
                // 小写
                case 'lower':
                    return /^[a-z]+$/.test(str);
                // 大写
                case 'upper':
                    return /^[A-Z]+$/.test(str);
                // HTML标记
```

```
                case 'HTML':
                    return /<("[^"]*"|'[^']*'|[^'">])*>/.test(str);
                default:
                    return true;
            }
        },
        timeformater: function (timestamp) {

            if(!zudp.util.isNumber(timestamp)) {
                throw "预期 timestamp 应该为 Number类型，传入参数类型是："+Object.prototype.toString.call(timestamp);
            }

            return function(time){

                var oDate = new Date(time),
                    oYear = oDate.getFullYear(),
                    oMonth = oDate.getMonth()+1,
                    oDay = oDate.getDate(),
                    oTime = oYear +'-'+ getzf(oMonth) +'-'+ getzf(oDay) ;//最后拼接时间
                return oTime;
            }(timestamp);

            //补0操作,当时间数据小于10的时候，给该数据前面加一个0
            function getzf(num){

                if(parseInt(num) < 10){
                    num = '0'+num;
                }
                return num;
            }
        }
    },
    plugin: {
    }
};
zudp.ajax=(function(){
    var AjaxConfig = function(url) {
        if (!url) {
            throw "缺少 url 参数";
        }

        this.url = url;
    };

```

```
    AjaxConfig.prototype = {
        config: function() {
            return this;
        },
        // get 请求
        get: function(data) {
            this.type = "get";
            if(data){
                this.val = data;
            }

            return this;
        },
        // post 请求
        post: function (data) {
            this.type = "post";
            if (data) {
                this.val = data;
            }

            return this;
        },
        // put 请求
        put: function (data) {
            this.type = "put";
            if (data) {
                this.val = data;
            }

            return this;
        },
        // delete 请求
        del: function (data) {
            this.type = "delete";
            if (data) {
                this.val = data;
            }

            return this;
        },
        // 之后的操作
        then: function (suc_method, err_method) {

            if (!zudp.util.isFunction(err_method)) {
                err_method = function (jqXHR, textStatus, errorThrown) {
                    if (zudp.util.isDev()) {
```

```
                        console.log(jqXHR);
                        console.log(textStatus);
                        console.log(errorThrown);
                    }
                }
            }
            $.ajax({
                        url: this.url,
                        type: this.type,
                        data: this.val,
                        cache: false,
                        dataType: 'json',
                        contentType: 'application/json',
                        success: function (data) {
                            suc_method(data.data);
                        },
                        error: function (jqXHR, textStatus, errorThrown) {
                            err_method(jqXHR, textStatus, errorThrown);
                        }
                    });

        }
    };

    return function (url) {
        return new AjaxConfig(url);
    };
})();
/**
 * radio 对应的操作，包括
 * 单选框
 *
 */
zudp.plugin.radio = (function() {

    /**
     * 生成一个 radio 对象
     * @param {*string} jq jquery 定位的字符串，如 #id。
     */
    function Radio(jq) {
        var radio = {};

        if (typeof jq === 'string') {
            radio.jq = zudp.util.getJquery(jq);
        } else {
            throw "zudp.plugin.radio() 参数必须为字符串: " + jq;
```

```
        }

        radio.checked = function() {
             // 设置选中
            this.jq.prop("checked","true");
            return this;
        };
        radio.unchecked = function() {
            // 设置取消选中
            this.jq.removeProp("checked");
            return this;
        };
        radio.ischecked = function (){
            // 获取单选框状态
            var isCheck = this.jq.prop("checked");
            return isCheck;
        };
        radio.disabled = function (){
            //设置禁用
            this.jq.prop("disabled","true");
            return this;
        };
        radio.undisabled = function (){
            //设置取消禁用
            this.jq.removeProp("disabled");
            return this;
        };
        radio.property = function (prop,val){
            //设置属性
            this.jq.prop(prop,val);
            return this;
        };
        /**
        * 设置 radio 的值
        * @param {*string} val 设置当前 radio 的值
        */
        radio.val = function(val) {
           // 获取值和设置值
           if(val){
                this.jq.attr("value",val);
                return this;
           }else{
                return this.jq.attr("value");
           }
        }

```

```
        return radio;
    };
    return Radio;
})()

/**
 * checkbox 对应的操作，包括
 * 复选框
 *
 */
zudp.plugin.checkbox = (function() {

    /**
     * 生成一个 checkbox 对象
     * @param {*string} jq jquery 定位的字符串，如 #id。
     */
    function Checkbox(jq) {
        var checkbox = {};

        if (typeof jq === 'string') {
            checkbox.jq = zudp.util.getJquery(jq);
        } else {
            throw "zudp.plugin.checkbox() 参数必须为字符串: " + jq;
        }

        checkbox.checked = function() {
             // 设置选中
            this.jq.prop("checked","true");
            return this;
        };
        checkbox.unchecked = function() {
            // 设置取消选中
            this.jq.removeProp("checked");
            return this;
        };
        checkbox.ischecked = function (){
            // 获取复选框状态
            var isCheck = this.jq.prop("checked");
            return isCheck;
        };
        checkbox.disabled = function (){
            //设置禁用
            this.jq.prop("disabled","true");
            return this;
        };
        checkbox.undisabled = function (){
```

```
            //设置取消禁用
            this.jq.removeProp("disabled");
            return this;
        };
        checkbox.property = function (prop,val){
            //设置禁用
            this.jq.prop(prop,val);
            return this;
        };
        /**
        * 设置 checkbox 的值
        * @param {*string} val 设置当前 checkbox 的值
        */
        checkbox.val = function(val) {
           // 获取值和设置值
           if(val){
                this.jq.attr("value",val);
                return this;
           }else{
                return this.jq.attr("value");
           }

        };
        return checkbox;
    };
    return Checkbox;
})();
/**
 * input[type="text"] 对应的操作，包括
 * 文本框
 *
 */
zudp.plugin.text = (function(){
    /**
     * 生成一个 text 对象
     * @param {*string} jq jquery 定位的字符串，如 #id。
     */
    function Text(jq){
        var text = {};

        if (typeof jq === 'string') {
            text.jq = zudp.util.getJquery(jq);
        } else {
            throw "zudp.plugin.text() 参数必须为字符串: " + jq;
        }
        text.val = function(val){
```

```
            // 获取值和设置值
            if(val){
                this.jq.attr("value",val);
                return this;
            }else{
                return this.jq.attr("value");
            };
        }
        text.property = function (prop,val){
            //设置属性
            this.jq.prop(prop,val);
            return this;
        };
        text.disabled = function (){
            //设置禁用
            this.jq.prop("disabled","true");
            return this;
        };
        text.undisabled = function (){
            //设置取消禁用
            this.jq.removeProp("disabled");
            return this;
        };
        text.readonly = function (){
            //设置只读
            this.jq.prop("readonly","true");
            return this;
        };
        text.unreadonly = function (){
            //设置取消只读
            this.jq.removeProp("readonly");
            return this;
        };
        return text;
    }
    return Text;
})();
zudp.plugin.select = (function(){
    /**
     * 生成一个 select 对象
     * @param {*string} jq jquery 定位的字符串，如 #id。
     */
    function Select(jq){
        var select = {};

        if (typeof jq === 'string') {
```

```
            select.jq = zudp.util.getJquery(jq);
        } else {
            // select.jq = jq;
            throw "zudp.plugin.select() 参数必须为字符串: " + jq;
        }
        select.val = function(val){
            // 获取值和设置值
            if(val){
                this.jq.find("option:selected").val(val);
                return this;
            }else{
                return this.jq.find("option:selected").val();
            };
        }
        select.init = function(val){
            // 初始化
            var selectVal=val
            var inHtml="";
            $.each(selectVal,function(name,val){
                inHtml += "<option value='"+ val +"' >"+ name +"</option>"
            })
            this.jq.html(inHtml);
        }
        select.disabled = function (){
            //设置禁用
            this.jq.prop("disabled","true");
            return this;
        };
        select.undisabled = function (){
            //设置取消禁用
            this.jq.removeProp("disabled");
            return this;
        };
        select.required = function (){
            //设置禁用
            this.jq.prop("required","true");
            return this;
        };
        select.unrequired = function (){
            //设置禁用
            this.jq.prop("required","false");
            return this;
        };
        return select;
    }
    return Select;
```

```
})();
zudp.plugin.select2 = (function(){
    /**
     * 生成一个 select 对象
     * @param {*string} jq jquery 定位的字符串，如 #id。
     */
    function Select2(jq){
        var select2 = {};

        if (typeof jq === 'string') {
            select2.jq = zudp.util.getJquery(jq);
        } else {
            // select2.jq = jq;
            throw "zudp.plugin.select2() 参数必须为字符串: " + jq;
        }
        select2.val = function(val){
            // 获取值和设置值
            if(val){
                this.jq.val(val).trigger('change');
                return this;
            }else{
                return this.jq.select2("data");
                // return list;
            };

        }
        select2.init = function(obj){
            // 初始化
            // obj.multiple = true;
            if(obj){
                this.jq.select2(obj);
            }else{
                this.jq.select2();
            }
            return this;
        };
        select2.clear = function(){
            this.jq.val(null).trigger("change");
            return this;
        }
        select2.disabled = function (){
            //设置禁用
            this.jq.prop("disabled", true);
            return this;
        };
        select2.undisabled = function (){
```

```
            //设置取消禁用
            this.jq.removeProp("disabled");
            return this;
        };
        select2.ajax = function(url){
            if(url){
               this.jq.select2({
                ajax:{
                    url: url,
                    dataType: 'json',
                    // delay: 250,
                    data: function (params) {
                      return {
                        q: params.term,
                      };
                    },
                    processResults: function (data) {
                      return {
                        results: data
                      };
                    },
                    cache: true
                  }
               })
            }
        }
        return select2;
    }
    return Select2;
})();
zudp.plugin.textarea = (function(){
    /**
     * 生成一个 text 对象
     * @param {*string} jq jquery 定位的字符串，如 #id。
     */
    function Textarea(jq){
        var textarea = {};

        if (typeof jq === 'string') {
            textarea.jq = zudp.util.getJquery(jq);
        } else {
            throw "zudp.plugin.textarea() 参数必须为字符串: " + jq;
        }
        textarea.text = function(txt){
            // 获取文本和设置文本
           if(txt){
```

```
649                 this.jq.text(txt);
650                 return this;
651             }else{
652                 return this.jq.text();
653             };
654         }
655         textarea.disabled = function (){
656             //设置禁用
657             this.jq.prop("disabled","true");
658             return this;
659         };
660         textarea.undisabled = function (){
661             //设置取消禁用
662             this.jq.removeProp("disabled");
663             return this;
664         };
665         textarea.readonly = function (){
666             //设置禁用
667             this.jq.prop("readonly","true");
668             return this;
669         };
670         textarea.unreadonly = function (){
671             //设置取消禁用
672             this.jq.removeProp("readonly");
673             return this;
674         };
675         textarea.property = function (prop,val){
676             //设置属性
677             this.jq.prop(prop,val);
678             return this;
679         };
680         return textarea;
681     }
682     return Textarea;
683 })();
684 // zudp.plugin.threeLevelMenu = (function(){
685 //     /**
686 //      * 生成一个 threeLevelMenu 对象
687 //      * @param {*string} jq jquery 定位的字符串，如 #id。
688 //      */
689 //     function Menu(jq){
690 //         var menu = {};
691 
692 //         if (typeof jq === 'string') {
693 //             menu.jq = zudp.util.getJquery(jq);
694 //         } else {
```

```
//             throw "zudp.plugin.menu() 参数必须为字符串: " + jq;
//         }
//         menu.init = function(json){
//             // 初始化菜单
//             var roots=[];
//             var parent=[];
//             var childs=[];
//         }
//         $.each(json, function(admin, el) {
//             if(el.parentId==""){
//                 roots.push(el);
//             }else{
//                 // parent[admin]=el;
//                 if(el.parentIds!="" && el.parentId==el.parentIds){
//                     parent.push(el);
//                 }else{
//                     childs.push(el);
//                 }

//             }
//         });
//     }
//     return Menu;
// })();
// jqueryId、url、搜索条件、成功所执行的方法
zudp.plugin.table = (function () {
    var TableConfig = function (jqueryIdentifier) {
        this.config = {
            jqueryIdentifier: jqueryIdentifier
        };
        this.settings = {
            jQueryUI: false,    // JQuery UI 风格样式，已被列入弃用
            pagingType: "full_numbers",    // 分页按钮显示选项
            dom: '<"top"<"clear">><"toolbar">rt<"bottom"<"table-fi"i><"table-fl"l>p<"clear">>',
            processing: true,  // 是否显示处理状态 (排序的时候，数据很多耗费时间长的话，也会显示这个)
            serverSide: true,  // 打开后台分页，是否开启服务器模式
            lengthChange: true,  // 改变每页显示数据数量
            stateSave: false,  // 保存状态 - 在页面重新加载的时候恢复状态（页码等内容）
            retrieve: true,   // 检索已经存在的 Datatables 实例
            ordering: true,   // 是否允许 Datatables 开启排序
            info: true,      // 页脚信息
            autoWidth: true,  // 控制 Datatables 是否自适应宽度
            scrollY: 400,       // 设置表格高度
```

```
            scrollCollapse: true,  // 表格高度自适应
            language: {
                "sSearch": "搜索",
                "sLengthMenu": "每页显示 _MENU_ 条记录",
                "sZeroRecords": "没有检索到数据",
                "sInfo": "显示 _START_ 至 _END_ 条   共 _TOTAL_ 条",
                "sInfoFiltered": "(筛选自 _MAX_ 条数据)",
                "sInfoEmtpy": "没有找到数据",
                "sProcessing": "正在加载数据...",

                "oPaginate":
                    {
                        "sFirst": "首页",
                        "sPrevious": "前一页",
                        "sNext": "后一页",
                        "sLast": "末页"
                    }
            }
        };
    };

    TableConfig.prototype = {
        url: function (url) {
            this.config.url = url;
            return this;
        },
        search: function (fn) {
            zudp.util.assert.checkFunction(fn);

            this.config.search = fn;
            return this;
        },
        columns: function (columnsParam) {
            zudp.util.assert.checkArray(columnsParam);

            var column;
            var configColumns = [];
            for (var i = 0; i < columnsParam.length; i++) {
                if (!zudp.util.isBoolean(columnsParam[i].visible)) {
                    columnsParam[i].visible = true;
                }

                if (!zudp.util.isBoolean(columnsParam[i].searchable)) {
                    columnsParam[i].searchable = true;
                }

```

```
                column = {
                    data: columnsParam[i].data || null,
                    visible: columnsParam[i].visible,
                    searchable: columnsParam[i].searchable,
                    width: columnsParam[i].width || undefined,
                    render: columnsParam[i].render || undefined
                };

                if (column.data === null) {
                    zudp.util.assert.checkFunction(column.render);
                }

                if (!column.render) {
                    zudp.util.data = null;
                }

                configColumns[i] = column;
            }

            this.config.columns = configColumns;
            return this;
        },
        then: function (successMethod, errorMethod) {
            var config = this.config;
            this.settings.ajax = function(data, callback, settings) {
                var searchCondition = config.search();
                for (var condition in searchCondition) {
                    if (searchCondition.hasOwnProperty(condition)) {
                        data[condition] = searchCondition[condition];
                    }
                }

                data.columns = undefined;
                data.page_size = data.length;
                data.page_no = data.start / data.page_size + 1;

                var succMethod;
                if (!zudp.util.isFunction(successMethod)) {
                    succMethod = function(res) {
                        setTimeout(
                            function () {
                                var returnData = {
                                    recordsFiltered: res.total,
                                    data: res.result,
                                    recordsTotal: res.total,
                                    length: res.size
```

```
                                };
                                callback(returnData);
                            },
                            200);
                    };
                } else {
                    if (zudp.util.isDev(!zudp.util.isUndefined(successMethod))) {
                        console.warn("预期的参数为 函数 类型，实际上它的类型为：" + (typeof succMethod));
                    }
                    succMethod = successMethod;
                }

                zudp.ajax(config.url).get(data).then(succMethod, errorMethod);
            };

            this.settings.columns = this.config.columns;

            return zudp.util.getJquery(this.config.jqueryIdentifier).DataTable(this.settings);
        }
    };

    return function($identifier) {
        return new TableConfig($identifier);
    }
})();
```

表单序列化 转json

```
$.fn.serializeObject = function()
    {
       var o = {};
       var a = this.serializeArray();
       $.each(a, function() {
           if (o[this.name]) {
               if (!o[this.name].push) {
                   o[this.name] = [o[this.name]];
               }
               o[this.name].push(this.value || '');
           } else {
               o[this.name] = this.value || '';
           }
```

```
        });
        return o;
    };

     $(function() {
        var data = $("form").serializeObject();
        console.log(JSON.stringify(data));
    });
```