

时间戳转换date 格式的 js

```
1  return function(time){
2      var oDate = new Date(time),
3          oYear = oDate.getFullYear(),
4          oMonth = oDate.getMonth()+1,
5          oDay = oDate.getDate(),
6          oTime = oYear + '-' + getzf(oMonth) + '-' + getzf(oDay) ;//最后拼接时间
7      return oTime;
8  }(timestamp);
9
10 //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
11 function getzf(num){
12
13     if(parseInt(num) < 10){
14         num = '0'+num;
15     }
16     return num;
17 }
```

zzq 写的 封装的 zudp.js(很值得学习)

```
1  /**
2   * zudp
3   * 要求, 必须要有 JQuery
4   */
5  var zudp = {
6      util: {
7          assert: {
8              checkFunction: function (param) {
9                  if (!zudp.util.isFunction(param)) {
10                     throw '预期参数类型为 Function, 实际类型为 ' +
11 Object.prototype.toString.call(param);
12                 }
13             },
14             checkArray: function (param) {
15                 if (!zudp.util.isArray(param)) {
16                     throw '预期参数类型为 Array, 实际类型为 ' +
17 Object.prototype.toString.call(param);
18                 }
19             }
20         }
21     }
22 }
```

```
19     getjQueryObjById : function(jqueryId) {
20         return {
21             value: $('#' + jqueryId)
22         };
23     },
24     getjQuery: function(jqueryIdentifier) {
25         return $(jqueryIdentifier)
26     },
27     // 是否为开发模式
28     isDev: function(status) {
29         return zudp.global.profile === zudp.global.status.dev && status;
30     },
31     //是否字符串
32     isString: function(o) {
33         return Object.prototype.toString.call(o).slice(8, -1) === 'String'
34     },
35     //是否数字
36     isNumber: function(o) {
37         return Object.prototype.toString.call(o).slice(8, -1) === 'Number'
38     },
39     //是否对象
40     isObj: function(o) {
41         return Object.prototype.toString.call(o).slice(8, -1) === 'Object'
42     },
43     //是否数组
44     isArray: function(o) {
45         return Object.prototype.toString.call(o).slice(8, -1) === 'Array'
46     },
47     //是否时间
48     isDate: function(o) {
49         return Object.prototype.toString.call(o).slice(8, -1) === 'Date'
50     },
51     //是否boolean
52     isBoolean: function(o) {
53         return Object.prototype.toString.call(o).slice(8, -1) ===
54         'Boolean'
55     },
56     //是否函数
57     isFunction: function(o) {
58         return Object.prototype.toString.call(o).slice(8, -1) ===
59         'Function'
60     },
61     //是否为null
62     isNull: function(o) {
63         return Object.prototype.toString.call(o).slice(8, -1) === 'Null'
64     },
65 }
```

```

63      //是否undefined
64      isUndefined: function(o) {
65          return Object.prototype.toString.call(o).slice(8, -1) ===
'Undefined'
66      },
67      isFalse: function(o) {
68          if (!o || o === 'null' || o === 'undefined' || o === 'false' || o
=== 'NaN') return true
69          return false
70      },
71      isTrue: function(o) {
72          return !this.isFalse(o)
73      },
74      isIos: function() {
75          var u = navigator.userAgent;
76          if (u.indexOf('Android') > -1 || u.indexOf('Linux') > -1) {//安卓
手机
77              // return "Android";
78              return false
79          } else if (u.indexOf('iPhone') > -1) {//苹果手机
80              // return "iPhone";
81              return true
82          } else if (u.indexOf('iPad') > -1) {//iPad
83              // return "iPad";
84              return false
85          } else if (u.indexOf('Windows Phone') > -1) {//winphone手机
86              // return "Windows Phone";
87              return false
88          }else{
89              return false
90          }
91      },
92      //是否为PC端
93      isPC: function() {
94          var userAgentInfo = navigator.userAgent;
95          var Agents = ["Android", "iPhone", "SymbianOS", "Windows Phone",
"iPad", "iPod"];
96          var flag = true;
97          for (var v = 0; v < Agents.length; v++) {
98              if (userAgentInfo.indexOf(Agents[v]) > 0) {
99                  flag = false;
100                  break;
101              }
102          }
103          return flag;
104      },

```

```
105 // 获取浏览器类型
106 browserType: function(){
107     //取得浏览器的 userAgent 字符串
108     var userAgent = navigator.userAgent;
109     //判断是否Opera浏览器
110     var isOpera = userAgent.indexOf("Opera") > -1;
111     //判断是否IE浏览器
112     var isIE = userAgent.indexOf("compatible") > -1 &&
userAgent.indexOf("MSIE") > -1 && !isOpera;
113     //判断是否IE的Edge浏览器
114     var isEdge = userAgent.indexOf("Edge") > -1;
115     //判断是否Firefox浏览器
116     var isFF = userAgent.indexOf("Firefox") > -1;
117     //判断是否Safari浏览器
118     var isSafari = userAgent.indexOf("Safari") > -1 &&
userAgent.indexOf("Chrome") === -1;
119     //判断Chrome浏览器
120     var isChrome = userAgent.indexOf("Chrome") > -1 &&
userAgent.indexOf("Safari") > -1;
121     if (isIE) {
122         var reIE = new RegExp("MSIE (\\d+\\.\\d+)");
123         reIE.test(userAgent);
124         var fIEVersion = parseFloat(RegExp["$1"]);
125         if(fIEVersion === 7) return "IE7";
126         else if(fIEVersion === 8) return "IE8";
127         else if(fIEVersion === 9) return "IE9";
128         else if(fIEVersion === 10) return "IE10";
129         else if(fIEVersion === 11) return "IE11";
130         else return "IE7以下"//IE版本过低
131     }
132
133     if (isFF) return "FF";
134     if (isOpera) return "Opera";
135     if (isEdge) return "Edge";
136     if (isSafari) return "Safari";
137     if (isChrome) return "Chrome";
138 },
139 // 检查字符串是否符合某种类型
140 checkStr: function(str, type) {
141     switch (type) {
142         // 手机号码
143         case 'phone':
144             return /^1[3|4|5|7|8][0-9]{9}$/.test(str);
145         // 座机
146         case 'tel':
147             return /^(0\d{2,3}-\d{7,8})(-\d{1,4})?$/ .test(str);
```

```

148 // 身份证
149 case 'card':
150     return /^d{15}|\d{18}$/.test(str);
151 // 密码以字母开头，长度在6~18之间，只能包含字母、数字和下划线
152 case 'pwd':
153     return /^[a-zA-Z]\w{5,17}$/.test(str)
154 // 邮政编码
155 case 'postal':
156     return /[1-9]\d{5}(?!d)/.test(str);
157 // QQ号
158 case 'QQ':
159     return /^[1-9][0-9]{4,9}$/.test(str);
160 // 邮箱
161 case 'email':
162     return /^[w-]+(\.[w-]+)*@[w-]+(\.[w-]+)+$/.test(str);
163 // 金额(小数点2位)
164 case 'money':
165     return /^d*(?:\.\d{0,2})?$/ .test(str);
166 // 网址
167 case 'URL':
168     return /(http|ftp|https):\/\/[w\-\_]+(\.[w\-\_]+)([w\-\_.,@?^=%&:/~\+]*[w\-\_@?^=%&/~\+])?/.test(str)
169 // IP
170 case 'IP':
171     return (((?:25[0-5]|2[0-4]\d|[01]?\d?\d)\.){3}
172     (?:25[0-5]|2[0-4]\d|[01]?\d?\d))/ .test(str);
173 // 日期时间
174 case 'date':
175     return /^(d{4})-(d{2})-(d{2}) (d{2})(?:\:d{2}|:
176     (d{2}):d{2}))$/.test(str) || /^(d{4})-(d{2})-(d{2})$/.test(str)
177 // 数字
178 case 'number':
179     return /^[0-9]$/.test(str);
180 // 英文
181 case 'english':
182     return /^[a-zA-Z]+$/.test(str);
183 // 中文
184 case 'chinese':
185     return /^[\u4E00-\u9FA5]+$/.test(str);
186 // 小写
187 case 'lower':
188     return /^[a-z]+$/.test(str);
189 // 大写
190 case 'upper':
191     return /^[A-Z]+$/.test(str);
192 // HTML标记

```

```

191         case 'HTML':
192             return /<("[^"]*"|'[^']*'|[">])*>/.test(str);
193         default:
194             return true;
195     }
196 },
197 timeformater: function (timestamp) {
198
199     if(!zudp.util.isNumber(timestamp)) {
200         throw "预期 timestamp 应该为 Number类型，传入参数类型
是: "+Object.prototype.toString.call(timestamp);
201     }
202
203     return function(time){
204
205         var oDate = new Date(time),
206             oYear = oDate.getFullYear(),
207             oMonth = oDate.getMonth()+1,
208             oDay = oDate.getDate(),
209             oTime = oYear + '-' + getzf(oMonth) + '-' + getzf(oDay) ;//最
后拼接时间
210
211         return oTime;
212     }(timestamp);
213
214     //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
215     function getzf(num){
216
217         if(parseInt(num) < 10){
218             num = '0'+num;
219         }
220         return num;
221     }
222 },
223 plugin: {
224 }
225 };
226 zudp.ajax=(function(){
227     var AjaxConfig = function(url) {
228         if (!url) {
229             throw "缺少 url 参数";
230         }
231
232         this.url = url;
233     };
234

```

```
235 AjaxConfig.prototype = {
236     config: function() {
237         return this;
238     },
239     // get 请求
240     get: function(data) {
241         this.type = "get";
242         if(data){
243             this.val = data;
244         }
245
246         return this;
247     },
248     // post 请求
249     post: function (data) {
250         this.type = "post";
251         if (data) {
252             this.val = data;
253         }
254
255         return this;
256     },
257     // put 请求
258     put: function (data) {
259         this.type = "put";
260         if (data) {
261             this.val = data;
262         }
263
264         return this;
265     },
266     // delete 请求
267     del: function (data) {
268         this.type = "delete";
269         if (data) {
270             this.val = data;
271         }
272
273         return this;
274     },
275     // 之后的操作
276     then: function (suc_method, err_method) {
277
278         if (!zudp.util.isFunction(err_method)) {
279             err_method = function (jqXHR, textStatus, errorThrown) {
280                 if (zudp.util.isDev()) {
```

```

281         console.log(jqXHR);
282         console.log(textStatus);
283         console.log(errorThrown);
284     }
285 }
286 }
287 $.ajax({
288     url: this.url,
289     type: this.type,
290     data: this.val,
291     cache: false,
292     dataType: 'json',
293     contentType: 'application/json',
294     success: function (data) {
295         suc_method(data.data);
296     },
297     error: function (jqXHR, textStatus, errorThrown) {
298         err_method(jqXHR, textStatus, errorThrown);
299     }
300 });
301
302 }
303 };
304
305 return function (url) {
306     return new AjaxConfig(url);
307 };
308 })();
309 /**
310  * radio 对应的操作, 包括
311  * 单选框
312  *
313  */
314 zudp.plugin.radio = (function() {
315
316     /**
317      * 生成一个 radio 对象
318      * @param {*string} jq jquery 定位的字符串, 如 #id。
319      */
320     function Radio(jq) {
321         var radio = {};
322
323         if (typeof jq === 'string') {
324             radio.jq = zudp.util.getJquery(jq);
325         } else {
326             throw "zudp.plugin.radio() 参数必须为字符串: " + jq;

```



```
327     }
328
329     radio.checked = function() {
330         // 设置选中
331         this.jq.prop("checked","true");
332         return this;
333     };
334     radio.unchecked = function() {
335         // 设置取消选中
336         this.jq.removeProp("checked");
337         return this;
338     };
339     radio.ischecked = function (){
340         // 获取单选框状态
341         var isChecked = this.jq.prop("checked");
342         return isChecked;
343     };
344     radio.disabled = function (){
345         //设置禁用
346         this.jq.prop("disabled","true");
347         return this;
348     };
349     radio.undisabled = function (){
350         //设置取消禁用
351         this.jq.removeProp("disabled");
352         return this;
353     };
354     radio.property = function (prop,val){
355         //设置属性
356         this.jq.prop(prop,val);
357         return this;
358     };
359     /**
360     * 设置 radio 的值
361     * @param {*string} val 设置当前 radio 的值
362     */
363     radio.val = function(val) {
364         // 获取值和设置值
365         if(val){
366             this.jq.attr("value",val);
367             return this;
368         }else{
369             return this.jq.attr("value");
370         }
371     }
372
```

```
373         return radio;
374     };
375     return Radio;
376 })()
377
378 /**
379  * checkbox 对应的操作，包括
380  * 复选框
381  *
382  */
383 zudp.plugin.checkbox = (function() {
384
385     /**
386      * 生成一个 checkbox 对象
387      * @param {*string} jq jquery 定位的字符串，如 #id。
388      */
389     function Checkbox(jq) {
390         var checkbox = {};
391
392         if (typeof jq === 'string') {
393             checkbox.jq = zudp.util.getJquery(jq);
394         } else {
395             throw "zudp.plugin.checkbox() 参数必须为字符串： " + jq;
396         }
397
398         checkbox.checked = function() {
399             // 设置选中
400             this.jq.prop("checked","true");
401             return this;
402         };
403         checkbox.unchecked = function() {
404             // 设置取消选中
405             this.jq.removeProp("checked");
406             return this;
407         };
408         checkbox.ischecked = function () {
409             // 获取复选框状态
410             var isCheck = this.jq.prop("checked");
411             return isCheck;
412         };
413         checkbox.disabled = function () {
414             //设置禁用
415             this.jq.prop("disabled","true");
416             return this;
417         };
418         checkbox.undisabled = function () {
```

```
419         //设置取消禁用
420         this.jq.removeProp("disabled");
421         return this;
422     };
423     checkbox.property = function (prop,val){
424         //设置禁用
425         this.jq.prop(prop,val);
426         return this;
427     };
428     /**
429     * 设置 checkbox 的值
430     * @param {*string} val 设置当前 checkbox 的值
431     */
432     checkbox.val = function(val) {
433         // 获取值和设置值
434         if(val){
435             this.jq.attr("value",val);
436             return this;
437         }else{
438             return this.jq.attr("value");
439         }
440     };
441     };
442     return checkbox;
443 };
444 return Checkbox;
445 })();
446 /**
447 * input[type="text"] 对应的操作，包括
448 * 文本框
449 *
450 */
451 zudp.plugin.text = (function(){
452     /**
453     * 生成一个 text 对象
454     * @param {*string} jq jquery 定位的字符串，如 #id。
455     */
456     function Text(jq){
457         var text = {};
458
459         if (typeof jq === 'string') {
460             text.jq = zudp.util.getJquery(jq);
461         } else {
462             throw "zudp.plugin.text() 参数必须为字符串：" + jq;
463         }
464         text.val = function(val){
```

```
465         // 获取值和设置值
466         if(val){
467             this.jq.attr("value",val);
468             return this;
469         }else{
470             return this.jq.attr("value");
471         };
472     }
473     text.property = function (prop,val){
474         //设置属性
475         this.jq.prop(prop,val);
476         return this;
477     };
478     text.disabled = function (){
479         //设置禁用
480         this.jq.prop("disabled","true");
481         return this;
482     };
483     text.undisabled = function (){
484         //设置取消禁用
485         this.jq.removeProp("disabled");
486         return this;
487     };
488     text.readonly = function (){
489         //设置只读
490         this.jq.prop("readonly","true");
491         return this;
492     };
493     text.unreadonly = function (){
494         //设置取消只读
495         this.jq.removeProp("readonly");
496         return this;
497     };
498     return text;
499 }
500 return Text;
501 })();
502 zudp.plugin.select = (function(){
503     /**
504     * 生成一个 select 对象
505     * @param {*string} jq jquery 定位的字符串, 如 #id。
506     */
507     function Select(jq){
508         var select = {};
509
510         if (typeof jq === 'string') {
```

```
511     select.jq = zudp.util.getJquery(jq);
512 } else {
513     // select.jq = jq;
514     throw "zudp.plugin.select() 参数必须为字符串：" + jq;
515 }
516 select.val = function(val){
517     // 获取值和设置值
518     if(val){
519         this.jq.find("option:selected").val(val);
520         return this;
521     }else{
522         return this.jq.find("option:selected").val();
523     };
524 }
525 select.init = function(val){
526     // 初始化
527     var selectVal=val
528     var inHtml="";
529     $.each(selectVal,function(name,val){
530         inHtml += "<option value='"+ val +"' >" + name + "</option>"
531     })
532     this.jq.html(inHtml);
533 }
534 select.disabled = function (){
535     //设置禁用
536     this.jq.prop("disabled","true");
537     return this;
538 };
539 select.undisabled = function (){
540     //设置取消禁用
541     this.jq.removeProp("disabled");
542     return this;
543 };
544 select.required = function (){
545     //设置禁用
546     this.jq.prop("required","true");
547     return this;
548 };
549 select.unrequired = function (){
550     //设置禁用
551     this.jq.prop("required","false");
552     return this;
553 };
554 return select;
555 }
556 return Select;
```

```
557 })();
558 zudp.plugin.select2 = (function(){
559     /**
560      * 生成一个 select 对象
561      * @param {*string} jq jquery 定位的字符串, 如 #id。
562      */
563     function Select2(jq){
564         var select2 = {};
565
566         if (typeof jq === 'string') {
567             select2.jq = zudp.util.getJquery(jq);
568         } else {
569             // select2.jq = jq;
570             throw "zudp.plugin.select2() 参数必须为字符串: " + jq;
571         }
572         select2.val = function(val){
573             // 获取值和设置值
574             if(val){
575                 this.jq.val(val).trigger('change');
576                 return this;
577             }else{
578                 return this.jq.select2("data");
579                 // return list;
580             };
581
582         }
583         select2.init = function(obj){
584             // 初始化
585             // obj.multiple = true;
586             if(obj){
587                 this.jq.select2(obj);
588             }else{
589                 this.jq.select2();
590             }
591             return this;
592         };
593         select2.clear = function(){
594             this.jq.val(null).trigger("change");
595             return this;
596         }
597         select2.disabled = function (){
598             //设置禁用
599             this.jq.prop("disabled", true);
600             return this;
601         };
602         select2.undisabled = function (){
```

```

603         //设置取消禁用
604         this.jq.removeProp("disabled");
605         return this;
606     };
607     select2.ajax = function(url){
608         if(url){
609             this.jq.select2({
610                 ajax:{
611                     url: url,
612                     dataType: 'json',
613                     // delay: 250,
614                     data: function (params) {
615                         return {
616                             q: params.term,
617                         };
618                     },
619                     processResults: function (data) {
620                         return {
621                             results: data
622                         };
623                     },
624                     cache: true
625                 }
626             })
627         }
628     }
629     return select2;
630 }
631 return Select2;
632 })();
633 zudp.plugin.textarea = (function(){
634     /**
635     * 生成一个 text 对象
636     * @param {*}string} jq jquery 定位的字符串, 如 #id。
637     */
638     function Textarea(jq){
639         var textarea = {};
640
641         if (typeof jq === 'string') {
642             textarea.jq = zudp.util.getJquery(jq);
643         } else {
644             throw "zudp.plugin.textarea() 参数必须为字符串: " + jq;
645         }
646         textarea.text = function(txt){
647             // 获取文本和设置文本
648             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 {
```



```

695 //          throw "zudp.plugin.menu() 参数必须为字符串：" + jq;
696 //      }
697 //      menu.init = function(json){
698 //          // 初始化菜单
699 //          var roots=[];
700 //          var parent=[];
701 //          var childs=[];
702 //      }
703 //      $.each(json, function(admin, el) {
704 //          if(el.parentId==""){
705 //              roots.push(el);
706 //          }else{
707 //              // parent[admin]=el;
708 //              if(el.parentIds!="" && el.parentId==el.parentIds){
709 //                  parent.push(el);
710 //              }else{
711 //                  childs.push(el);
712 //              }
713 //          }
714 //      }
715 //      });
716 //  }
717 //  return Menu;
718 // }})();
719 // jqueryId、url、搜索条件、成功所执行的方法
720 zudp.plugin.table = (function () {
721     var TableConfig = function (jqueryIdentifier) {
722         this.config = {
723             jqueryIdentifier: jqueryIdentifier
724         };
725         this.settings = {
726             jQueryUI: false,    // JQuery UI 风格样式，已被列入弃用
727             pagingType: "full_numbers",    // 分页按钮显示选项
728             dom: '<"top"<"clear">><"toolbar">rt<"bottom"<"table-fi"i><"table-
729 fl"l>p<"clear">>',
730             processing: true,    // 是否显示处理状态（排序的时候，数据很多耗费时间长
731 的话，也会显示这个）
732             serverSide: true,    // 打开后台分页，是否开启服务器模式
733             lengthChange: true,    // 改变每页显示数据数量
734             stateSave: false,    // 保存状态 - 在页面重新加载的时候恢复状态（页码等
735 内容）
736             retrieve: true,    // 检索已经存在的 Datatables 实例
737             ordering: true,    // 是否允许 Datatables 开启排序
738             info: true,    // 页脚信息
739             autoWidth: true,    // 控制 Datatables 是否自适应宽度
740             scrollY: 400,    // 设置表格高度

```

```
738     scrollCollapse: true, // 表格高度自适应
739     language: {
740         "sSearch": "搜索",
741         "sLengthMenu": "每页显示 _MENU_ 条记录",
742         "sZeroRecords": "没有检索到数据",
743         "sInfo": "显示 _START_ 至 _END_ 条 &nbsp;&nbsp;&nbsp;共 _TOTAL_ 条",
744         "sInfoFiltered": "(筛选自 _MAX_ 条数据)",
745         "sInfoEmpty": "没有找到数据",
746         "sProcessing": "正在加载数据...",
747
748         "oPaginate":
749             {
750                 "sFirst": "首页",
751                 "sPrevious": "前一页",
752                 "sNext": "后一页",
753                 "sLast": "末页"
754             }
755     }
756 };
757
758
759 TableConfig.prototype = {
760     url: function (url) {
761         this.config.url = url;
762         return this;
763     },
764     search: function (fn) {
765         zudp.util.assert.checkFunction(fn);
766
767         this.config.search = fn;
768         return this;
769     },
770     columns: function (columnsParam) {
771         zudp.util.assert.checkArray(columnsParam);
772
773         var column;
774         var configColumns = [];
775         for (var i = 0; i < columnsParam.length; i++) {
776             if (!zudp.util.isBoolean(columnsParam[i].visible)) {
777                 columnsParam[i].visible = true;
778             }
779
780             if (!zudp.util.isBoolean(columnsParam[i].searchable)) {
781                 columnsParam[i].searchable = true;
782             }
783         }
784     }
785 }
```

```

784         column = {
785             data: columnsParam[i].data || null,
786             visible: columnsParam[i].visible,
787             searchable: columnsParam[i].searchable,
788             width: columnsParam[i].width || undefined,
789             render: columnsParam[i].render || undefined
790         };
791
792         if (column.data === null) {
793             zudp.util.assert.checkFunction(column.render);
794         }
795
796         if (!column.render) {
797             zudp.util.data = null;
798         }
799
800         configColumns[i] = column;
801     }
802
803     this.config.columns = configColumns;
804     return this;
805 },
806 then: function (successMethod, errorMethod) {
807     var config = this.config;
808     this.settings.ajax = function(data, callback, settings) {
809         var searchCondition = config.search();
810         for (var condition in searchCondition) {
811             if (searchCondition.hasOwnProperty(condition)) {
812                 data[condition] = searchCondition[condition];
813             }
814         }
815
816         data.columns = undefined;
817         data.page_size = data.length;
818         data.page_no = data.start / data.page_size + 1;
819
820         var succMethod;
821         if (!zudp.util.isFunction(successMethod)) {
822             succMethod = function(res) {
823                 setTimeout(
824                     function () {
825                         var returnData = {
826                             recordsFiltered: res.total,
827                             data: res.result,
828                             recordsTotal: res.total,
829                             length: res.size

```

```

830         };
831         callback(returnData);
832     },
833     200);
834     };
835     } else {
836         if
(zudp.util.isDev(!zudp.util.isUndefined(successMethod))) {
837             console.warn("预期的参数为 函数 类型，实际上它的类型为： "
+ (typeof succMethod));
838         }
839         succMethod = successMethod;
840     }
841
842     zudp.ajax(config.url).get(data).then(succMethod, errorMethod);
843 };
844
845     this.settings.columns = this.config.columns;
846
847     return
zudp.util.getJquery(this.config.jqueryIdentifier).DataTable(this.settings);
848     }
849 };
850
851     return function($identifier) {
852         return new TableConfig($identifier);
853     }
854 })();

```

表单序列化 转json

```

1 $.fn.serializeObject = function()
2 {
3     var o = {};
4     var a = this.serializeArray();
5     $.each(a, function() {
6         if (o[this.name]) {
7             if (!o[this.name].push) {
8                 o[this.name] = [o[this.name]];
9             }
10            o[this.name].push(this.value || '');
11        } else {
12            o[this.name] = this.value || '';
13        }

```

```
14     });
15     return o;
16 };
17
18     $(function() {
19         var data = $("#form").serializeObject();
20         console.log(JSON.stringify(data));
21     });
```