

```

1.  /**
2.   * AXISJ 유틸함수 라이브러리 axf 또는 AXUtil이라고 한다.
3.   * @namespace {Object} axf
4.   * @example
5.   * ```json
6.   * trace(axf.browser);
7.   * trace(AXUtil.browser);
8.   * ```
9.   */
10. var axf = AXUtil = {
11.   async: true,
12.   ajaxOkCode: "ok",
13.   ajaxResponseType: "",
14.   ajaxDataType: "",
15.   gridPassiveMode: false,
16.   gridPassiveRemoveHide: false,
17.   gridFitToWidthRightMargin: 10,
18.
19.   uniqueSeq: 0,
20.
21.   /**
22.    * 현재페이지에서 고유한 순번을 반환합니다.
23.    * @method axf.getUniqueId
24.    * @returns {Number} uniqueSeq
25.    * @example
26.    * ```
27.    * trace( axf.getUniqueId() );
28.    * ```
29.    */
30.
31.   getUniqueId: function(){ return (axf.uniqueSeq += 1); },
32.   /**
33.    * document.getElementById(id) 와 같습니다. 아이디가 같은 엘리먼트를 반환합니다.
34.    * @method axf.getId
35.    * @param {String} id
36.    * @returns {HtmlElement}
37.    * @example
38.    * ```
39.    * if(axf.getId("myele_id")){
40.    *   $("#myele_id").css({..});
41.    * }
42.    * ```
43.    */
44.   getId: function(id) { return document.getElementById(id); },
45.   /**
46.    * @method axf.each
47.    * @param {Array|Object} obj
48.    * @param {Function} callback
49.    * @description Array 또는 Object의 아이템만큼 callback 함수를 call합니다.
50.    * @example
51.    * ```
52.    * var new_array = [];
53.    * axf.each([0, 1, 2], function(){
54.    *   new_array.push(this*2);
55.    * });
56.    * var new_object = {};
57.    * axf.each({a:1, b:2, c:3}, function(k, v){
58.    *   new_object[k] = v*2;
59.    * });
60.    * ```
61.    */
62.   each: function(obj, callback){
63.     if(obj){
64.       var name, i = 0, length = obj.length,
65.           isObj = length === undefined || Object.isFunction( obj );
66.       if ( isObj ) {
67.         for ( name in obj ) {
68.           if ( callback.call( obj[ name ], name, obj[ name ] ) === false ) {
69.             break;
70.           }
71.         }
72.       } else {
73.         for ( ; i < length; i ) {
74.           if ( callback.call( obj[ i ], i, obj[ i ] ) === false ) {
75.             break;
76.           }
77.         }
78.       }
79.     }

```

```

80.     },
81.     /**
82.      * 브라우저의 이름과 버전 모바일여부
83.      * @member {Object} axf.browser
84.      * @example
85.      *~~~
86.      {
87.      name: {String} - browserName (ie/chrome/webkit/oprea),
88.      version: {Number} - browserVersion,
89.      mobile: {Boolean}
90.      }
91.      *~~~
92.      */
93.     browser: (function () {
94.         var ua = navigator.userAgent.toLowerCase();
95.         var mobile = (ua.search(/mobile/g) != -1);
96.         if (ua.search(/iphone/g) != -1) {
97.             return { name: "iphone", version: 0, mobile: true };
98.         } else if (ua.search(/ipad/g) != -1) {
99.             return { name: "ipad", version: 0, mobile: true };
100.        } else if (ua.search(/android/g) != -1) {
101.            var match = /(android)[ \\/]([^\w.]+)/.exec(ua) || [];
102.            var browserVersion = (match[2] || "0");
103.            return { name: "android", version: browserVersion, mobile: mobile };
104.        } else {
105.            var browserName = "";
106.            var match = /(chrome)[ \\/]([^\w.]+)/.exec(ua) ||
107.                /(webkit)[ \\/]([^\w.]+)/.exec(ua) ||
108.                /(opera)(?:.*version|)[ \\/]([^\w.]+)/.exec(ua) ||
109.                /(msie) ([^\w.]+)/.exec(ua) ||
110.                ua.indexOf("compatible") < 0 && /(mozilla)(?:.*? rv:([^\w.]+)/.exec(ua) ||
111.                []);
112.
113.            var browser = (match[1] || "");
114.            var browserVersion = (match[2] || "0");
115.
116.            if (browser == "msie") browser = "ie";
117.            return {
118.                name: browser,
119.                version: browserVersion,
120.                mobile: mobile
121.            }
122.        }
123.    })(),
124.    /**
125.     * 호환성보기 여부
126.     * @member {String} axf.docTD
127.     * @example
128.     *~~~
129.     axf.docTD = (Q/S)
130.     *~~~
131.     */
132.    docTD: (function () {
133.        if (!document.compatMode || document.compatMode != 'BackCompat') return "Q";
134.        else return "S";
135.    })(),
136.    /**
137.     * @method axf.timekey
138.     * @returns {String} timeKey
139.     * @description 밀리세컨드까지 조합한 문자열을 반환합니다.
140.     * @example
141.     *~~~js
142.     trace(axf.timeKey()); // A004222760
143.     *~~~
144.     */
145.    timekey: (function () {
146.        var d = new Date();
147.        return ("A" + d.getHours().setDigit(2) + d.getMinutes().setDigit(2) +
148.            d.getSeconds().setDigit(2) + d.getMilliseconds());
149.    }),
150.    overwriteObject: (function (tg, obj, rewrite) {
151.        if (rewrite == undefined) rewrite = true;
152.        //trace(tg[k]);
153.        if (obj) AXUtil.each(obj, function (k, v) {
154.            if (rewrite) { tg[k] = v; }
155.            else {
156.                //trace(tg[k]);
157.                if (tg[k] == undefined) tg[k] = v;
158.            }

```

```

159.     });
160.     return tg;
161. },
162. copyObject: function (obj) {
163.     //return Object.clone(obj);
164.     return Object.toJSON(obj).object();
165. },
166. consonantKR: function (cword) {
167.     var cons = [
168.         { c: "ㄱ", re: "[ㄱ-질]" }, { c: "ㄴ", re: "[ㄴ-질]" }, { c: "ㄷ", re: "[ㄷ-질]" }, { c:
"ㄹ", re: "[ㄹ-질]" }, { c: "ㄲ", re: "[ㄲ-질]" }, { c: "ㄴ", re: "[ㄴ-질]" },
169.         { c: "ㄷ", re: "[ㄷ-질]" }, { c: "ㄹ", re: "[ㄹ-질]" }, { c: "ㅁ", re: "[ㅁ-질]" }, { c:
"ㅂ", re: "[ㅂ-질]" }, { c: "ㅅ", re: "[ㅅ-질]" }, { c: "ㅇ", re: "[ㅇ-질]" }, { c: "ㅈ", re: "[ㅈ-질]" },
170.         { c: "ㅊ", re: "[ㅊ-질]" }, { c: "ㅊ", re: "[ㅊ-질]" }, { c: "ㅋ", re: "[ㅋ-질]" }, { c:
"ㅌ", re: "[ㅌ-질]" }, { c: "ㅍ", re: "[ㅍ-질]" }, { c: "ㅎ", re: "[ㅎ-질]" }
171.     ];
172.     var rword = "";
173.     var cwords = cword.split("");
174.     AXUtil.each(cwords, function (i, n) {
175.         var fos = cons.searchObject(function () {
176.             return this.item.c == n;
177.         });
178.         var fo = fos.first();
179.         if (fo) rword += fo.re;
180.         else rword += n;
181.     });
182.     return rword;
183. },
184. setCookie: function (name, value, expiredays) { if (expiredays) { var todayDate = new
Date(); todayDate.setDate(todayDate.getDate() + expiredays); document.cookie = name + '='
+ escape(value) + '; path=/; expires=' + todayDate.toGMTString() + ';'; } else {
document.cookie = name + '=' + escape(value) + '; path=/;'; } },
185. getCookie: function (name) { var nameOfCookie = name + "="; var x = 0; while (x <=
document.cookie.length) { var y = (x + nameOfCookie.length); if
(document.cookie.substring(x, y) == nameOfCookie) { if (endOfCookie =
document.cookie.indexOf(";", y)) == -1) endOfCookie = document.cookie.length; return
unescape(document.cookie.substring(y, endOfCookie)); } x = document.cookie.indexOf(" ", x)
+ 1; if (x == 0) break; } return ""; },
186. JSONFilter: /^\\\/+secure-([\s\S]*)\\\/\s*$/ ,
187. dayLen: function (y, m) { if ([3, 5, 8, 10].has(function () { return this.item == m; }))
{ return 30; } else if (m == 1) { return ((y % 4 == 0) && (y % 100 != 0)) || (y % 400 ==
0) ? 29 : 28; } else { return 31; } },
188. clientHeight: function () { return (AXUtil.docTD == "Q") ? document.body.clientHeight :
document.documentElement.clientHeight; },
189. scrollHeight: function () { return (AXUtil.docTD == "Q") ? document.body.scrollHeight :
document.documentElement.scrollHeight; },
190. clientWidth: function () { return (AXUtil.docTD == "Q") ? document.body.clientWidth :
document.documentElement.clientWidth; },
191. scrollWidth: function () { return (AXUtil.docTD == "Q") ? document.body.scrollWidth :
document.documentElement.scrollWidth; },
192. scrollTop: function(){
193.     return (document.documentElement && document.documentElement.scrollTop) ||
194.         document.body.scrollTop;
195. },
196. scrollLeft: function(){
197.     return (document.documentElement && document.documentElement.scrollLeft) ||
198.         document.body.scrollLeft;
199. },
200. Event: {
201.     KEY_BACKSPACE: 8,
202.     KEY_TAB: 9,
203.     KEY_RETURN: 13, KEY_ESC: 27, KEY_LEFT: 37, KEY_UP: 38, KEY_RIGHT: 39, KEY_DOWN: 40,
KEY_DELETE: 46,
204.     KEY_HOME: 36, KEY_END: 35, KEY_PAGEUP: 33, KEY_PAGEDOWN: 34, KEY_INSERT: 45, KEY_SPACE:
32, cache: {} },
205. console: function (obj) {
206.     var po = "";
207.     if (arguments.length > 1) {
208.         for (i = 0; i < arguments.length; i++) {
209.             var obji = arguments[i];
210.             var objStr = "";
211.             var type = (typeof obji).toLowerCase();
212.             if (type == "undefined" || type == "function") {
213.                 objStr = type;
214.             } else if (type == "boolean" || type == "number" || type == "string") {
215.                 objStr = obji;
216.             } else if (type == "object") {
217.                 objStr = Object.toJSON(obji);
218.             }

```

```

219.     if (po != "") po += ", ";
220.     po += "arg[" + i + "] : " + objStr;
221. }
222. } else {
223.     var type = (typeof obj).toLowerCase();
224.     if (type == "undefined" || type == "function") {
225.         po = type;
226.     } else if (type == "boolean" || type == "number" || type == "string") {
227.         po = obj;
228.     } else if (type == "object") {
229.         po = Object.toJSON(obj);
230.     }
231. }
232.
233. if(axf_mobileConsole){
234.     axf_mobileConsole.prepend("<div>" + po + "</div>");
235. }else{
236.     if (window.console == undefined) {
237.     } else {
238.         try {
239.             console.log( po );
240.             //+ ":" + axf.console.caller.name
241.         } catch (e) {
242.             alert(e);
243.         }
244.     }
245. }
246. },
247. alert: function (obj) {
248.     var po = "";
249.     if (arguments.length > 1) {
250.         for (i = 0; i < arguments.length; i++) {
251.             var obji = arguments[i];
252.             var objStr = "";
253.             var type = (typeof obji).toLowerCase();
254.             if (type == "undefined" || type == "function") {
255.                 objStr = type;
256.             } else if (type == "boolean" || type == "number" || type == "string") {
257.                 objStr = obji;
258.             } else if (type == "object") {
259.                 objStr = Object.toJSON(obji);
260.             }
261.             if (po != "") po += ", ";
262.             po += "arguments[" + i + "] : " + objStr;
263.         }
264.     } else {
265.         var type = (typeof obj).toLowerCase();
266.         if (type == "undefined" || type == "function") {
267.             po = type;
268.         } else if (type == "boolean" || type == "number" || type == "string") {
269.             po = obj;
270.         } else if (type == "object") {
271.             po = Object.toJSON(obj);
272.         }
273.     }
274.     alert(po);
275. },
276. confirm: function (obj) {
277.     var po = "";
278.     var type = (typeof obj).toLowerCase();
279.     if (type == "undefined" || type == "function") {
280.         po = type;
281.     } else if (type == "boolean" || type == "number" || type == "string") {
282.         po = obj;
283.     } else if (type == "object") {
284.         po = Object.toJSON(obj);
285.     }
286.     var result = confirm(po);
287.     return result;
288. },
289. importJS: function (src) {
290.     var scriptElement = document.createElement("script");
291.     scriptElement.setAttribute("src", src);
292.     scriptElement.setAttribute("type", "text/javascript");
293.     document.getElementsByTagName("head")[0].appendChild(scriptElement);
294. },
295. bindPlaceholder: function () {
296.
297. },
298. isEmpty: function (val) {

```

```

299.     return (val == "" || val == null || val == undefined ? true : false);
300. },
301. getUrlInfo: function () {
302.     var url, url_param, param, referUrl, pathName, AXparam, pageProtocol, pageHostName;
303.     url_param = window.location.href;
304.     param = window.location.search;
305.     referUrl = document.referrer;
306.     pathName = window.location.pathname;
307.     url = url_param.replace(param, '');
308.     param = param.replace(/^\?/, '');
309.     pageProtocol = window.location.protocol;
310.     pageHostName = window.location.hostname;
311.     AXparam = url_param.replace(pageProtocol + "://" + "", "");
312.     AXparam = (param ? AXparam.replace(pageHostName + pathName + "?", param, "") :
AXparam.replace(pageHostName + pathName, ""));
313.     return {
314.         url : url,
315.         param : param,
316.         anchorData : AXparam,
317.         urlParam : url_param,
318.         referUrl : referUrl,
319.         pathName : pathName,
320.         protocol : pageProtocol,
321.         hostName : pageHostName
322.     };
323. },
324. encParam: function (str) {
325.     var re = new RegExp("[^&]*?=[^&]*?", "ig");
326.     var pars = [];
327.     var arr;
328.     while ((arr = re.exec(str)) != null) {
329.         var strContent = arr.toString();
330.         var dotIndex = strContent.indexOf("=");
331.         pars.push(strContent.substring(0, dotIndex) + "=" + strContent.substring(dotIndex +
1).enc());
332.     }
333.     return pars.join("&");
334. },
335. readyMobileConsole: function(){
336.     AXUtil.mobileConsole = axdom("<div class='AXMobileConsole'></div>");
337.     axdom(document.body).append(AXUtil.mobileConsole);
338. },
339. parsingTable: function (elemObj, returnType){
340.     var head = {}, body = [];
341.     elemObj.find("thead tr td").each(function(){
342.         var elem = axdom( this );
343.         var attrs = {
344.             key: elem.attr("name"),
345.             label: (elem.html() || ""),
346.             width: (elem.attr("width") || "*"),
347.             align: (elem.attr("align") || "")
348.         };
349.         head[attrs.key] = attrs;
350.     });
351.
352.     elemObj.find("tbody tr").each(function(){
353.         var item = {};
354.         axdom( this ).find("td").each(function ){
355.             var elem = axdom( this );
356.             item[elem.attr("name")] = elem.html();
357.         };
358.         body.push(item);
359.     });
360.     return {
361.         head: head, body: body
362.     };
363. },
364. mousewheelEvt: (/Firefox/i.test(navigator.userAgent) ? "DOMMouseScroll" : "mousewheel")
365. },
366. var axdom;
367. if(window jQuery) axdom = jQuery;
368. if(window axdomConverter) axdom = axdomConverter;
369.
370. // extend implement block
371. var Class = (function () {
372.     function subclass() { }
373.     function create() { var parent = null; properties = AX_A(arguments); if
Object.isFunction(properties[0]) parent = properties.shift(); function klass() {
this.initialize.apply(this, arguments); } Object.extend(klass, Class Methods);
klass.superclass = parent, klass.subclasses = []; if (parent) { subclass.prototype =

```

```

parent.prototype; class.prototype = new subclass; parent.subclasses.push(class); } for
var i = 0; i < properties.length; i++) class.addMethods(properties[i]); if
(!class.prototype.initialize) class.prototype.initialize = Prototype.emptyFunction;
class.prototype.constructor = class; return class; }
374. function addMethods(source) { var ancestor = this.superclass &&
this.superclass.prototype; var properties = Object.keys(source); if (!Object.keys({
toString: true }).length) { if (source.toString != Object.prototype.toString)
properties.push("toString"); if (source.valueOf != Object.prototype.valueOf)
properties.push("valueOf"); } for (var i = 0, length = properties.length; i < length; i++)
{ var property = properties[i], value = source[property]; if (ancestor &&
Object.isFunction(value) && value.argumentNames().first() == "AXJ_super") { var method =
value; value = (function(m) { return function() { return ancestor[m].apply(this,
arguments); }})(property).wrap(method); value.valueOf = method.valueOf.bind(method);
value.toString = method.toString.bind(method); } this.prototype[property] = value; }
return this; }
375. return { create: create, Methods: { addMethods: addMethods } };
376. }();
377.
378. // Object extend
379. function () {
380. var _toString = Object.prototype.toString;
381. //function extend(destination, source) { for (var property in source)
destination[property] = source[property]; return destination; }
382.
383.
384. function extend() {
385. var target = arguments[0] || {}, items = arguments[1], overwrite = arguments[2] || false;
386. if (typeof target != "object" && typeof target != "function") {
387. target = {};
388. }
389. if (typeof items == "string") {
390. target = items;
391. }
392. else {
393. if (overwrite == true) {
394. for (var k in items) target[k] = items[k];
395. }
396. else
397. if (overwrite == false) {
398. for (var k in items) {
399. if (typeof target[k] == "undefined") target[k] = items[k];
400. }
401. }
402. }
403. return target;
404. }
405.
406.
407. function inspect(obj) { try { if (isUndefined(obj)) return 'undefined'; if (obj == null)
return 'null'; return obj.inspect ? obj.inspect() : String(obj); } catch (e) { if (e
instanceof RangeError) return '...'; throw e; } }
408. function toJSON(object, quote) {
409. var type = typeof object;
410. var isquote = quote;
411. if (isquote == undefined) isquote = true;
412. switch (type) {
413. case 'undefined': return "undefined";
414. //case 'function': return "\"" + object.toString().replace(/\n/g, "\\n") + "\"";
415. case 'function': return;
416. case 'unknown': return "unknown";
417. case 'boolean': return object.toString();
418. case 'number': return object.toString();
419. case 'string': return object.axtoJSON(true);
420. }
421. if (object == null) return 'null';
422. if (object.axtoJSON) return object.axtoJSON(isquote);
423. if (isElement(object)) return;
424. var results = [];
425. for (var property in object) {
426. if (object.hasOwnProperty(property)) {
427. var value = toJSON(object[property], isquote);
428. if (!isUndefined(value)) results.push(property.axtoJSON(isquote) + ':' + value);
429. }
430. }
431. return '{' + results.join(', ') + '}';
432. }
433. function toJSONfn(object, quote) {
434. var type = typeof object;
435. var isquote = quote;
436. if (isquote == undefined) isquote = true;
437. switch (type) {

```

```

438.     case 'undefined': return "undefined";
439.     case 'function':
440.         try {
441.             return toJSONfn(object(), isquote);
442.         } catch (e) {
443.             return;
444.         }
445.     case 'unknown': return "unknown";
446.     case 'boolean': return object.toString();
447.     case 'number': return object.toString();
448.     case 'string': return object.axtoJSON(true);
449. }
450. if (object === null) return 'null';
451. if (object.axtoJSON) return object.axtoJSON(isquote);
452. if (isElement(object)) return;
453. var results = [];
454. for (var property in object) {
455.     if (object.hasOwnProperty(property)) {
456.         var value = toJSONfn(object[property], isquote);
457.         if (!isUndefined(value)) results.push(property.axtoJSON(isquote) + ':' + value);
458.     }
459. }
460. return '{' + results.join(', ') + '}';
461. }
462. function toJSONforMobile(object) {
463.     var type = typeof object;
464.     switch (type) {
465.         case 'undefined':
466.         case 'function': return;
467.         case 'unknown': return;
468.         case 'boolean': return "\"" + object.toString() + "\"";
469.         case 'number': return "\"" + object.toString() + "\"";
470.         case 'string': return object.axtoJSON(true);
471.     }
472.     if (object === null) return 'null';
473.     if (object.toJSONforMobile) return object.toJSONforMobile(true);
474.     if (isElement(object)) return;
475.     var results = [];
476.     for (var property in object) {
477.         if (object.hasOwnProperty(property)) {
478.             var value = axtoJSON(object[property]);
479.             if (!isUndefined(value)) results.push(property.axtoJSON(true) + ':' + value);
480.         }
481.     }
482.     return '{' + results.join(', ') + '}';
483. }
484. function keys(obj) { var results = []; for (var property in obj) results.push(property);
return results; }
485. function values(obj) { var results = []; for (var property in obj)
results.push(obj[property]); return results; }
486. function clone obj { return extend({}, obj); }
487. function isElement(obj) { return !!(obj && obj.nodeType == 1); }
488. function isObject(obj) { return _toString.call(obj) == "[object Object]"; }
489. function isArray(obj) { return _toString.call(obj) == "[object Array]"; }
490. function isHash(obj) { return obj instanceof Hash; }
491. function isFunction(obj) { return typeof obj == "function"; }
492. function isString(obj) { return _toString.call(obj) == "[object String]"; }
493. function isNumber(obj) { return _toString.call(obj) == "[object Number]"; }
494. function isUndefined(obj) { return typeof obj == "undefined"; }
495. extend(Object, { extend: extend, inspect: inspect, toJSON: toJSON, toJSONfn: toJSONfn,
toJSONforMobile: toJSONforMobile, keys: keys, values: values, clone: clone, isElement:
isElement, isObject: isObject, isArray: isArray, isHash: isHash, isFunction: isFunction,
isString: isString, isNumber: isNumber, isUndefined: isUndefined });
496. }();
497.
498. Object.extend(Function.prototype, { function () {
499.     var slice = Array.prototype.slice;
500.     function update(array, args) { var arrayLength = array.length, length = args.length;
while (length--) array[arrayLength + length] = args[length]; return array; }
501.     function merge(array, args) { array = slice.call(array, 0); return update(array, args); }
502.     function argumentNames() { var names = this.toString().match(/^[^\\s\\(\\)]*function\\([^\\s\\(\\)]*\\)/)
[1].replace(/\\/\\/.*?[\r\n][\\\/\\n](?:.?[\\r\n])?*/g, '').replace(/\\s+/g,
'').split(','); return names.length == 1 && !names[0] ? [] : names; }
503.     function bind(context) { if (arguments.length < 2 && Object.isUndefined(arguments[0]))
return this; var __method = this, args = slice.call(arguments, 1); return function () {
var a = merge(args, arguments); return __method.apply(context, a); } }
504.     function curry() { if (arguments.length) return this; var __method = this, args =
slice.call(arguments, 0); return function () { var a = merge(args, arguments); return
__method.apply(this, a); } }
505.     function delay(timeout) { var __method = this, args = slice.call(arguments, 1); timeout =

```

```

        timeout = 1000; return window.setTimeout(function () { return __method.apply(__method,
        args); }, timeout); }
506.     function defer() { var args = update([0,0], arguments); return this.delay.apply(this,
        args); }
507.     function wrap(wrapper) { var __method = this; return function () { var a =
        update([__method.bind(this)], arguments); return wrapper.apply(this, a); } }
508.     function methodize() { if (this._methodized, return this._methodized; var __method =
        this; return this._methodized = function () { var a = update([this], arguments); return
        __method.apply(null, a); }; } }
509.     function addPrototype(fns) { var name, i = 0, length = fns.length, isObj = length ===
        undefined || Object.isFunction(fns); if (isObj) { for (name in fns) {
        this.prototype[name] = fns[name]; } } }
510.     return { argumentNames: argumentNames, bind: bind, curry: curry, delay: delay, defer:
        defer, wrap: wrap, methodize: methodize, addPrototype: addPrototype }
511. })());
512.
513. Object.extend(String.prototype, {function () {
514.     function password() { return Math.tan(45).toString().substr(7);
515.     function left(strLen) { return this.toString().substr(0, strLen); }
516.     function right(strLen) { return this.substring(this.length - strLen, this.length); }
517.     function dec() {
518.         var decodeURI;
519.         try { decodeURI = decodeURIComponent(this.replace(/%+/g, " ")); } catch (e) { var decodeURI =
        this; }
520.         return (this) ? (decodeURI) : this;
521.     }
522.     function enc() { return (this) ? encodeURIComponent(this) : this; }
523.     function object() { try { var res = this.evalJSON(); } catch (e) { res = { error:
        "syntaxerr", result: "syntaxerr", msg: "to object error, " + e.print() + ", " + this };
        try { mask.close(); } catch (e) { } } return res; }
524.     function array() { try { var res = this.split(/,/g); } catch (e) { res = { error:
        "syntaxerr", result: "syntaxerr", msg: "to object error, " + e.print() + ", " + this };
        return res; }
525.     function toDate(separator, defaultDate) {
526.         if (this.length == 14) {
527.             try {
528.                 var va = this.replace(/\\D/g, "");
529.                 return new Date(va.substr(0, 4), va.substr(4, 2).number()-1, va.substr(6, 2),
        va.substr(8, 2), va.substr(10, 2), va.substr(12, 2));
530.             } catch (e) {
531.                 return (defaultDate || new Date());
532.             }
533.         } else if (this.length == 10) {
534.             try {
535.                 var aDate = this.split(separator || "-");
536.                 return new Date(aDate[0], ((aDate[1]) - 1).number(), (aDate[2]).number(), 12);
537.             } catch (e) {
538.                 return (defaultDate || new Date());
539.             }
540.         } else if (this.length == 8) {
541.             var va = this.replace(/\\D/g, "");
542.             return new Date(va.substr(0, 4), (va.substr(4, 2).number()-1), va.substr(6,
        2).number(), 12);
543.         } else if (this.length < 10) {
544.             return (defaultDate || new Date());
545.         } else if (this.length > 15) {
546.             try {
547.                 var aDateTime = this.split(/ /g);
548.                 var aDate = aDateTime[0].split(separator || "-");
549.                 if (aDateTime[1]) {
550.                     var aTime = aDateTime[1];
551.                 } else {
552.                     var aTime = "09:00";
553.                 }
554.                 var is24 = true;
555.                 if (aTime.right(2) == "AM" || aTime.right(2) == "PM") {
556.                     is24 = false;
557.                 }
558.                 var aTimes = aTime.left(5).split(":");
559.                 var hh = aTimes[0];
560.                 var mm = aTimes[1];
561.                 if (!is24) hh += 12;
562.                 return new Date(aDate[0], (parseFloat(aDate[1]) - 1), parseFloat(aDate[2]),
        parseFloat(hh), parseFloat(mm));
563.             } catch (e) {
564.                 var now = new Date();
565.                 return (defaultDate || new Date(now.getFullYear(), now.getMonth(), now.getDate(),
        12));
566.             }
567.         } else { // > 10
568.             var now = new Date();

```



```

569.     return {defaultDate :| new Date(now.getFullYear(), now.getMonth(), now.getDate(),
12)};
570. }
571. }
572. function toNum() {
573.     var pair = this.replace(/,/g, "").split(".");
574.     var isMinus = false
575.     if (parseFloat(pair[0]) < 0) isMinus = true;
576.     if (pair[0] == "-0") isMinus = true;
577.     var returnValue = 0.0; pair[0] = pair[0].replace(/[-|+]?[0-9]/gi, "");
578.     if (pair[1]) {
579.         pair[1] = pair[1].replace(/\D/gi, "");
580.         returnValue = parseFloat(pair[0] + "." + pair[1]) || 0;
581.     } else {
582.         returnValue = parseFloat(pair[0]) || 0;
583.     }
584.     return (isMinus) ? -returnValue : returnValue;
585. }
586. function parseF() { return parseFloat(this); }
587. function strip() { return this.replace(/^\s+/, '').replace(/\s+$/, ''); }
588. function stripTags() { return this.replace(/<\w+(\s+("[^"]*"|'[^']*'|>))+?>|
<\s*>/gi, ''); }
589. function stripScript() {
590.     //스크립트 제거
591.     var cStr;
592.     var RegExpJS = new RegExp("<[ ]*script[>]*>[<]*<[/ ]*script[>]*>", "gi");
593.     cStr = this.replace(RegExpJS, "");
594.
595.     cStr = cStr.replace(/[\s]*onclick[=]*=/gi, " xonclick=");
596.     cStr = cStr.replace(/[\s]*onmouseover[=]*=/gi, " xonmouseover=");
597.     cStr = cStr.replace(/[\s]*onmouseout[=]*=/gi, " xonmouseout=");
598.     cStr = cStr.replace(/[\s]*onchange[=]*=/gi, " xonchange=");
599.     cStr = cStr.replace(/[\s]*onblur[=]*=/gi, " xonblur=");
600.     cStr = cStr.replace(/[\s]*onerror[=]*=/gi, " xonerror=");
601.     cStr = cStr.replace(/[\s]*onload[=]*=/gi, " xonload=");
602.     cStr = cStr.replace(/[\s]*href[=]*=[\s]*["']?javascript/gi, " href=\"xjavascript");
603.
604.     return cStr;
605. }
606. function times(count) { return count < 1 ? '' : new Array(count + 1).join(this); }
607. function inspect(useDoubleQuotes) {
608.     var escapedString = this.replace(
609.         /[\x00-\x1f\\]/g,
610.         function (character) {
611.             try {
612.                 if (character in String.specialChar) return String.specialChar[character];
613.             } catch (e) {}
614.             return '\\x00' + character.charCodeAt(1)
615.         }
616.     );
617.     if (useDoubleQuotes) return '"' + escapedString.replace(/"/g, '\\\"') + '"';
618.     return "<\" + escapedString.replace(/'/g, '\\\'') + "<\";
619. }
620. function axtoJSON(TF) {
621.     return this.inspect(TF || false);
622. }
623. function blank() { return /\s*$/< test(this); }
624. function isJSON() { var str = this; if (str.isBlank()) return false; str =
this.replace(/\\./g, '<')< replace(/["\\n\r]*"/g, '<'); return (/^[:{}\\]\d-9.\-
+Eaeflnr-u \n\r\t]*$/<).test(str); } //<
625. function unfilterJSON(filter) { return this.replace(filter || AXutil.JSONFilter, '$1'); }
626. function evalJSON< sanitize() {
627.     var json = this.unfilterJSON();
628.     try {
629.         var _evl = eval<
630.         if (sanitize || json.isJSON()) return _evl("< + json + "<");
631.         else return { error: "syntaxerr", result: "syntaxerr", msg: "JSON syntax error. fail to
convert Object\n" + this };
632.         _evl = null;
633.     } catch (e) {
634.         return {
635.             error: e,
636.             result: "syntaxerr",
637.             msg: e,
638.             body: this
639.         };
640.     }
641. }
642. function queryToObject< separator) { var match = this.trim().match(/([?#]*)(#.*)?$/<); if
(!match) return {}; var rs = match[1].split(separator || '<'); var returnObj = {}; var i =

```

```

0; while (i < rs.length) { var pair = rs[i].split("="); var k = pair[0], v = pair[1]; if
returnObj[k] := undefined) { if (!Object.isArray(returnObj[k])) returnObj[k] =
returnObj[k]; returnObj[k].push(v); } else { returnObj[k] = v; } i++; } return
returnObj; }
643. function queryToObjectDec(separator) { var match = this.trim().match(/([?#]*)(#.*)?$/);
if (!match) return {}; var rs = match[1].split(separator || '&'); var returnObj = {}; var
i = 0; while (i < rs.length) { var pair = rs[i].split("="); var k = pair[0], v = pair[1];
if (returnObj[k] := undefined) { if (!Object.isArray(returnObj[k])) returnObj[k] =
returnObj[k]; returnObj[k].push(v.dec()); } else { returnObj[k] = v.dec(); } i++; }
return returnObj; }
644. function crlf(replaceTarget, replacer) { return this.replace(replaceTarget || /\n/g,
replacer || "<br/>"); }
645. function ecrlf(replaceTarget, replacer) { return this.replace(replaceTarget || /\r/g,
replacer || "<br/>"); }
646. function formatDigit(length, padder) { var string = this; return (padder ||
'0').times(length - string.length) + string; }
647. function getFileName() { var sToMatch = this; var reAt = /[\\\/]?([\\\/]?\.?
[\\\/\]+)$/; var reArr = sToMatch.match(reAt); return RegExp.$1; }
648. function toColor(sharp) { var colorValue = ""; if (this.left(3) == "rgb") { var val =
this, var reAt = /rgb\((.+)\)/; val.match(reAt); var vals = RegExp.$1.split(","); for
var a = 0; a < vals.length; a++ { vals[a] = vals[a].number().setDigit(2, '0', 16); }
colorValue = vals.join(""); } else { colorValue = this.replace("#", ""); } var preFix =
sharp ? "#" : ""; return preFix + colorValue; }
649. function toMoney() { return this.number().money(); }
650. function toByte() { return this.number().byte(); }
651. function lcase() { return this.toLowerCase(); }
652. function ucase() { return this.toUpperCase(); }
653. function getByte() {
654.   var valueByte = this.length;
655.   for (i = 0, l = this.length; i < l; i++) if (this.charCodeAt(i) > 128) valueByte++;
656.   return valueByte;
657. }
658. function toPhoneString() {
659.   if (this == "") return this
660.   var _this = this.replace(/\D+/g, "");
661.   var myLocalNums = "";
662.   var num1 = "", num2 = "";
663.   var localNum =
"031/032/033/041/042/043/051/052/053/054/055/061/062/063/064/010/011/016/017/019/070/080/060";
664.   if (_this.left(2) == "02") {
665.     myLocalNums = "02";
666.   } else {
667.     var localNums = localNum.split(/\//g);
668.     var tempNum = _this.left(3);
669.     AXUtil.each(localNums, function () {
670.       if (this == tempNum) {
671.         myLocalNums = this;
672.         return false;
673.       }
674.     });
675.   }
676.
677.   if (myLocalNums == "") {
678.     myLocalNums = "02";
679.     if (_this.length > 7) {
680.       num1 = _this.substr(0, 4);
681.       num2 = _this.substr(4);
682.     } else {
683.       num1 = _this.substr(0, 3);
684.       num2 = _this.substr(3);
685.     }
686.   } else {
687.     try {
688.       var snum = myLocalNums.length;
689.       if (_this.length - snum > 7) {
690.         num1 = _this.substr(snum, 4);
691.         num2 = _this.substr(snum + 4);
692.       } else {
693.         num1 = _this.substr(snum, 3);
694.         num2 = _this.substr(snum + 3);
695.       }
696.     } catch (e) {
697.       //trace(e);
698.     }
699.   }
700.
701.   var returnString = myLocalNums;
702.   if (num1 != "") returnString += "-" + num1;
703.   if (num2 != "") returnString += "-" + num2;
704.
705.   return returnString

```

```

706.
707. }
708. function getAnchorData() {
709.     var idx = this.indexOf("#", 0);
710.     if (idx < 0) return "";
711.     var cnt = this.length;
712.     var str = this.substring(idx + 1, cnt);
713.     return str;
714. }
715. function print() {
716.     return this;
717. }
718. return {
719.     ppassword: password,
720.     left: left,
721.     right: right,
722.     dec: dec,
723.     decode: dec,
724.     enc: enc,
725.     object: object,
726.     array: array,
727.     date: toDate,
728.     number: toNum,
729.     num: parseF,
730.     money: toMoney,
731.     byte: toByte,
732.     trim: strip,
733.     delHtml: stripTags,
734.     delScript: stripScript,
735.     removeScript: stripScript,
736.     times: times,
737.     inspect: inspect,
738.     axtoJSON: axtoJSON,
739.     isBlank: blank,
740.     isJSON: isJSON,
741.     unfilterJSON: unfilterJSON,
742.     evalJSON: evalJSON,
743.     queryToObject: queryToObject,
744.     queryToObjectDec: queryToObjectDec,
745.     crlf: crlf,
746.     ecrlf: ecrlf,
747.     setDigit: formatDigit,
748.     getFileName: getFileName,
749.     toColor: toColor,
750.     lcase: lcase,
751.     ucase: ucase,
752.     getByte: getByte,
753.     phone: toPhoneString,
754.     getAnchorData: getAnchorData,
755.     print: print
756. }
757.})();
758.
759. Object.extend(Number.prototype, (function () {
760.
761.     function left(strLen) { return this.toString().substr(0, strLen); }
762.     function right(strLen) { return this.toString().substring(this.toString().length -
763. strLen, this.toString().length); }
764.     function toMoney() {
765.         var txtNumber = '' + this;
766.         if (isNaN(txtNumber) || txtNumber == "") { return ""; }
767.         else {
768.             var rxSplit = new RegExp('([0-9])([0-9]{0-9}[0-9]{0-9}[0-9]{0-9}[0-9]{0-9})');
769.             var arrNumber = txtNumber.split('.');
770.             arrNumber[0] += '.';
771.             do {
772.                 arrNumber[0] = arrNumber[0].replace(rxSplit, '$1,$2');
773.             } while (rxSplit.test(arrNumber[0]));
774.             if (arrNumber.length > 1) {
775.                 return arrNumber.join('.');
776.             } else {
777.                 return arrNumber[0].split('.')[0];
778.             }
779.         }
780.     }
781.     function toByte() { var n_unit = "KB"; var myByte = this / 1024; if (myByte / 1024 > 1) {
782. n_unit = "MB"; myByte = myByte / 1024; } if (myByte / 1024 > 1) { n_unit = "GB"; myByte =
783. myByte / 1024; } return myByte.round(1) + n_unit; }
784.     function toNum() { return this; }
785.     function formatDigit(length, padder, radix) { var string = this.toString(radix || 10);

```

```

return (padder || "0").times(length - string.length) + string; }
783. function range(start) { var ra = []; for (var a = (start || 0); a < this + 1; a++)
ra.push(a); return ra; }
784. function axtoJSON() { return this; }
785. function abs() { return Math.abs(this); }
786. function round(digit) {
787.   return (typeof digit == "undefined") ? Math.round(this) : +
Math.round(this+"e-"+digit)+"e-"+digit;
788. }
789. function ceil() { return Math.ceil(this); }
790. function floor() { return Math.floor(this); }
791. function date() { return new Date(this); }
792. function div(divisor) { if (divisor != 0) { return this / divisor; } else { return 0; } }
793. function none() { return this; }
794. function times(count) { return count < 1 ? '' : new Array(count +
1).join(this.toString()); }
795. function phone() {
796.   var txtNumber = '' + this;
797.   return txtNumber.phone();
798. }
799. return {
800.   left: left,
801.   right: right,
802.   abs: abs,
803.   round: round,
804.   ceil: ceil,
805.   floor: floor,
806.   money: toMoney,
807.   byte: toByte,
808.   num: toNum,
809.   number: toNum,
810.   setDigit: formatDigit,
811.   date: date,
812.   div: div,
813.   dec: none,
814.   enc: none,
815.   rangeFrom: range,
816.   axtoJSON: axtoJSON,
817.   times: times,
818.   phone: phone
819. }
820. }());
821.
822. Object.extend(Date.prototype, (function () {
823.   function dateAdd(daynum, interval) {
824.     interval = interval || "d";
825.     var interval = interval.toLowerCase();
826.     var DyMilli = ((1000 * 60 * 60) * 24;
827.     var aDate = new Date(this.getUTCFullYear(), this.getMonth(), this.getDate(), 12);
828.
829.     if (interval == "d") {
830.       //trace(aDate.getTime(), (daynum), (DyMilli));
831.       aDate.setTime(aDate.getTime() + (daynum * DyMilli));
832.     } else if (interval == "m") {
833.       var yy = aDate.getFullYear();
834.       var mm = aDate.getMonth();
835.       var dd = aDate.getDate();
836.       /*if (mm == 0 && dd == 1) yy += 1;*/
837.       yy = yy + parseInt(daynum / 12);
838.       mm += daynum % 12;
839.       var mxdd = AXUtil.dayLen(yy, mm);
840.       if (mxdd < dd) dd = mxdd;
841.       aDate = new Date(yy, mm, dd, 12);
842.     } else if (interval == "y") {
843.       aDate.setTime(aDate.getTime() + ((daynum * 365) * DyMilli));
844.     } else {
845.       aDate.setTime(aDate.getTime() + (daynum * DyMilli));
846.     }
847.     return aDate;
848.   }
849.   function dayDiff(edDate, tp) {
850.     var DyMilli = ((1000 * 60 * 60) * 24;
851.     //trace(this.print() + "/" + edDate.print() + "/" + ((edDate.date() - this) / DyMilli) +
"/" + ((edDate.date() - this) / DyMilli).floor());
852.     var y1 = this.getFullYear();
853.     var m1 = this.getMonth();
854.     var d1 = this.getDate();
855.     var hh1 = this.getHours();
856.     var mm1 = this.getMinutes();
857.     var dd1 = new Date(y1, m1, d1, hh1, mm1, this.getSeconds());
858.

```

```

859.     var day2 = edDate.date();
860.     var y2 = day2.getFullYear();
861.     var m2 = day2.getMonth();
862.     var d2 = day2.getDate();
863.     var hh2 = day2.getHours();
864.     var mm2 = day2.getMinutes();
865.     var dd2 = new Date(y2, m2, d2, hh2, mm2, this.getSeconds());
866.
867.     if (tp != undefined) {
868.         if (tp == "D") {
869.             DyMilli = ((1000 * 60) * 60) * 24;
870.             dd2 = new Date(y2, m2, d2, hh1, mm1, this.getSeconds());
871.         } else if (tp == "H") {
872.             DyMilli = ((1000 * 60) * 60);
873.         } else if (tp == "mm") {
874.             DyMilli = (1000 * 60);
875.         } else {
876.             DyMilli = ((1000 * 60) * 60) * 24;
877.             dd2 = new Date(y2, m2, d2, hh1, mm1, this.getSeconds());
878.         }
879.     }
880.
881.     return ((dd2.getTime() - dd1.getTime()) / DyMilli).floor();
882.
883. }
884. function toString(format) {
885.     if (format == undefined) {
886.         var sSeper = "-";
887.         return this.getUTCFullYear() + sSeper + (this.getMonth() + 1).setDigit(2) + sSeper +
this.getDate().setDigit(2);
888.     } else {
889.         var fStr = format;
890.         var nY, nM, nD, nH, nMM, nS, nDW;
891.         nY = this.getUTCFullYear();
892.         nM = (this.getMonth() + 1).setDigit(2);
893.         nD = this.getDate().setDigit(2);
894.         nH = this.getHours().setDigit(2);
895.         nMM = this.getMinutes().setDigit(2);
896.         nS = this.getSeconds().setDigit(2);
897.         nDW = this.getDay();
898.
899.         var yre = /[^\y]*(yyyy)[^\y]*/gi; yre.exec(fStr); var regY = RegExp.$1;
900.         var mre = /[^\m]*(mm)[^\m]*/gi; mre.exec(fStr); var regM = RegExp.$1;
901.         var dre = /[^\d]*(dd)[^\d]*/gi; dre.exec(fStr); var regD = RegExp.$1;
902.         var hre = /[^\h]*(hh)[^\h]*/gi; hre.exec(fStr); var regH = RegExp.$1;
903.         var mire = /[^\m]*(mi)[^\i]*/gi; mire.exec(fStr); var regMI = RegExp.$1;
904.         var sre = /[^\s]*(ss)[^\s]*/gi; sre.exec(fStr); var regS = RegExp.$1;
905.         var dwre = /[^\d]*(dw)[^\w]*/gi; dwre.exec(fStr); var regDW = RegExp.$1;
906.
907.         if (regY == "yyyy") {
908.             fStr = fStr.replace(regY, nY.substr(regY.length));
909.         }
910.         if (regM == "mm") {
911.             if (regM.length == 1) nM = (this.getMonth() + 1);
912.             fStr = fStr.replace(regM, nM);
913.         }
914.         if (regD == "dd") {
915.             if (regD.length == 1) nD = this.getDate();
916.             fStr = fStr.replace(regD, nD);
917.         }
918.         if (regH == "hh") {
919.             fStr = fStr.replace(regH, nH);
920.         }
921.         if (regMI == "mi") {
922.             fStr = fStr.replace(regMI, nMM);
923.         }
924.         if (regS == "ss") {
925.             fStr = fStr.replace(regS, nS);
926.         }
927.         if (regDW == "dw") {
928.             fStr = fStr.replace(regDW, AXConfig.weekDays[nDW].label);
929.         }
930.         return fStr;
931.     }
932. }
933. function getTimeAgo() {
934.
935.     var rtnStr = "";
936.     var nMinute = Math.abs(new Date().diff(this, "mm"));
937.

```

```

938.     var wknames = [];
939.     wknames.push("일", "월", "화", "수", "목", "금", "토");
940.
941.     if (isNaN(nMinute)) {
942.         rtnStr = "알수없음";
943.     } else {
944.         if (parseInt(nMinute / 60 / 24) >= 1) {
945.             rtnStr = this.print("yyyy년 mm월 dd일") + " " + wknames[this.getDay()];
946.         } else {
947.             rtnStr = nMinute;
948.
949.             if ((nMinute / 60) > 1) {
950.                 rtnStr = parseInt(nMinute / 60) + "시간" + (nMinute % 60) + "분 전";
951.             } else {
952.                 rtnStr = nMinute + "분 전";
953.             }
954.         }
955.     }
956.     return rtnStr;
957. }
958. function date() { return this; }
959. function axtoJSON() { return "" + this.getUTCFullYear() + '-' + (this.getUTCMonth() +
1).setDigit(2) + '-' + this.getUTCDate().setDigit(2) + 'T' +
this.getUTCHours().setDigit(2) + ':' + this.getUTCMinutes().setDigit(2) + ':' +
this.getUTCSeconds().setDigit(2) + 'Z'; }
960. function axGetDay(dayOfStart){
961.     if(dayOfStart == undefined) dayOfStart = 0;
962.     var myDay = this.getDay() - dayOfStart;
963.     if(myDay < 0) myDay = 7 + myDay;
964.     return myDay;
965. }
966. return {
967.     add: dateAdd,
968.     diff: dayDiff,
969.     print: toString,
970.     date: date,
971.     axtoJSON: axtoJSON,
972.     getTimeAgo: getTimeAgo,
973.     axGetDay: axGetDay
974. }
975.})();
976.
977. Object.extend(Error.prototype, {function () {
978.     function print() {
979.         return (this.number & 0xFFFF) + " : " + this;
980.     }
981.     return {
982.         print: print
983.     }
984.})();
985.
986. Object.extend(Array.prototype, {function () {
987.     function clear() {
988.         this.length = 0;
989.         return this;
990.     }
991.     function first() {
992.         return this[0];
993.     }
994.     function last() {
995.         return this[this.length - 1];
996.     }
997.     function getToSeq(seq) {
998.         if (seq > (this.length - 1)) {
999.             return null;
1000.         } else {
1001.             return this[seq];
1002.         }
1003.     }
1004.     function axtoJSON(qoute) {
1005.         var results = [];
1006.         for (var i = 0; i < this.length; i++) results.push(Object.toJSON(this[i], qoute));
1007.         return '[' + results.join(', ') + ']';
1008.     }
1009.     function toJSONforMobile() {
1010.         var results = [];
1011.         for (var i = 0; i < this.length; i++) results.push(Object.toJSONforMobile this[i]);
1012.         return '[' + results.join(', ') + ']';
1013.     }
1014.     function remove(callback) {

```

```

1015.     var _self = this;
1016.     var collect = [];
1017.     AXUtil.each(this, function (index, O) {
1018.         if (!callBack.call({ index: index, item: O }, index, O)) collect.push(O);
1019.     });
1020.     return collect;
1021. }
1022. function search(callBack) {
1023.     var _self = this;
1024.     var collect = [];
1025.     AXUtil.each(this, function (index, O) {
1026.         if (callBack.call({ index: index, item: O }, index, O)) collect.push(O);
1027.     });
1028.     return collect.length;
1029. }
1030. function getObject(callBack) {
1031.     var _self = this;
1032.     var collect = [];
1033.     AXUtil.each(this, function (index, O) {
1034.         if (callBack.call({ index: index, item: O }, index, O)) collect.push(O);
1035.     });
1036.     return collect;
1037. }
1038. function hasObject(callBack) {
1039.     var _self = this;
1040.     var collect = null;
1041.     AXUtil.each(this, function (index, O) {
1042.         if (callBack.call({ index: index, item: O }, index, O)) {
1043.             collect = O;
1044.             return false;
1045.         }
1046.     });
1047.     return collect;
1048. }
1049. /* 13-06-13 테스트 확장 */
1050. function getMinObject(key) {
1051.     var tempArray = this.concat();
1052.     tempArray = tempArray.sort(function (pItem, nItem) {
1053.         var v1 = pItem[key];
1054.         var v2 = nItem[key];
1055.         if (v1 < v2) return -1;
1056.         else if (v1 > v2) return 1;
1057.         else if (v1 == v2) return 0;
1058.     });
1059.     return (tempArray.first() || {});
1060. }
1061. function getMaxObject(key) {
1062.     var tempArray = this.concat();
1063.     tempArray = tempArray.sort(function (pItem, nItem) {
1064.         var v1 = pItem[key];
1065.         var v2 = nItem[key];
1066.         if (v1 < v2) return 1;
1067.         else if (v1 > v2) return -1;
1068.         else if (v1 == v2) return 0;
1069.     });
1070.     return (tempArray.first() || {});
1071. }
1072.
1073. function m_notall(context) {
1074.     context = context || function (x) { return x; };
1075.     var result = true;
1076.     var i = 0;
1077.     while (i < this.length) {
1078.         result = !Boolean(context(this[i]));
1079.         if (!result) break;
1080.         i++;
1081.     }
1082.     return result;
1083. }
1084. function m_any(context) {
1085.     context = context || function (x) { return x; };
1086.     var result = false;
1087.     var i = 0;
1088.     while (i < this.length) {
1089.         result = Boolean(context(this[i], i));
1090.         if (result) break;
1091.         i++;
1092.     }
1093.     return result;
1094. }

```

```

1095. function m_find(context) {
1096.   context = context || function (x) { return false; };
1097.   var myselect;
1098.   var i = 0;
1099.   while (i < this.length) {
1100.     if (context(this[i], i)) {
1101.       myselect = this[i];
1102.       break;
1103.     }
1104.     i++;
1105.   }
1106.   return myselect;
1107. }
1108. function m_find2(context) {
1109.   if (!Object.isFunction(context)) {
1110.     findObj = context;
1111.     context = function (x) { return (x == findObj); };
1112.   }
1113.   var myselect, myindex;
1114.   var i = 0;
1115.   while (i < this.length) {
1116.     if (context(this[i], i)) {
1117.       myselect = this[i];
1118.       myindex = i;
1119.       break;
1120.     }
1121.     i++;
1122.   }
1123.   return { obj: myselect, index: myindex };
1124. }
1125. function m_findAll(context) {
1126.   context = context || function (x) { return false; };
1127.   var myselect = [];
1128.   var i = 0;
1129.   while (i < this.length) {
1130.     if (context(this[i], i)) myselect.push(this[i]);
1131.     i++;
1132.   }
1133.   return myselect;
1134. }
1135. function convertTree(parentKey, childKey, hashDigit) {
1136.   var tree = {};
1137.   var pointer = {};
1138.   var seq = 0;
1139.   var hashDigit = hashDigit || 3;
1140.   for (var idx = 0; idx < this.length; idx++) {
1141.     var L = this[idx];
1142.     if (!L.isRoot) {
1143.       pointer[L[childKey]] = idx;
1144.
1145.       if (L[parentKey].number() == 0) {
1146.         L["subTree"] = [];
1147.         L.__subTreeLength = 0;
1148.         L["pHash"] = "0".setDigit(hashDigit);
1149.         L["hash"] = "0".setDigit(hashDigit) + "_" + seq.setDigit(hashDigit);
1150.         tree.push(AxUtil.copyObject(L));
1151.         seq++;
1152.       } else {
1153.         L.__subTreeLength = 0;
1154.       }
1155.     }
1156.   }
1157.
1158.   for (var idx = 0; idx < this.length; idx++) {
1159.     var L = this[idx];
1160.     if (L["pHash"] == undefined && !L.isRoot) {
1161.       var pItem = this[pointer[L[parentKey]]];
1162.       var pHash = pItem["hash"];
1163.       var pHashs = pHash.split(/_/g);
1164.       var pTree = tree;
1165.       var pTreeItem;
1166.       axf.each(pHashs, function (idx, T) {
1167.         if (idx > 0) {
1168.           pTreeItem = pTree[T.number()];
1169.           pTree = pTreeItem.subTree;
1170.         }
1171.       });
1172.       L["subTree"] = [];
1173.       var __subTreeLength = pItem.__subTreeLength
1174.

```



```

1175.     L["pHash"] = pHash;
1176.     L["hash"] = pHash + "_" + __subTreeLength.setDigit(hashDigit);
1177.     pTree.push(AXutil.copyObject(L));
1178.     pItem.__subTreeLength++;
1179.     pTreeItem.__subTreeLength = pItem.__subTreeLength;
1180. }
1181. }
1182. return tree;
1183. }
1184. function getIndex(context) {
1185.     if (!Object.isFunction(context)) {
1186.         findObj = context;
1187.         context = function (x) { return (x == findObj); };
1188.     }
1189.     var findObject, findIndex;
1190.     var i = 0;
1191.     while (i < this.length) {
1192.         var sobj = {
1193.             index: i,
1194.             item: this[i]
1195.         };
1196.         if (context.call(sobj, sobj)) {
1197.             findObject = this[i];
1198.             findIndex = i;
1199.             break;
1200.         }
1201.         i++;
1202.     }
1203.     return { item: findObject, index: findIndex };
1204. }
1205.
1206. return {
1207.     clear: clear,
1208.     first: first,
1209.     last: last
1210.     getToSeq: getToSeq,
1211.     axtoJSON: axtoJSON,
1212.     toJSONforMobile: toJSONforMobile,
1213.     remove: remove,
1214.     search: search,
1215.     has: hasObject,
1216.     searchObject: getObject,
1217.     getMinObject: getMinObject,
1218.     getMaxObject: getMaxObject,
1219.
1220.     not: m_notall,
1221.     or: m_any,
1222.     get: m_find,
1223.     gets: m_findAll,
1224.     getObj: m_find2,
1225.     getIndex: getIndex,
1226.     convertTree: convertTree
1227. };
1228.})();
1229.
1230. //JSON.stringify = Object.toJSON;
1231. function AXgetId(id) { return document.getElementById(id); }
1232. function AX_A(iterable) { if (!iterable) return []; if ('toArray' in Object(iterable))
    return iterable.toArray(); var length = iterable.length || 0, results = new Array(length);
    while (length--) results[length] = iterable[length]; return results; }
1233.
1234. var trace = axf.console;
1235. var getUrlInfo = axf.getUrlInfo;

```