* Jump To … +  
    
  [<<< back to documentation](http://docs.google.com/index.html)  [base.js](http://docs.google.com/base.html)   [constraint.js](http://docs.google.com/constraint.html)   [defaults.js](http://docs.google.com/defaults.html)   [factory.js](http://docs.google.com/factory.html)   [field.js](http://docs.google.com/field.html)   [form.js](http://docs.google.com/form.html)   [main.js](http://docs.google.com/main.html)   [multiple.js](http://docs.google.com/multiple.html)   [pubsub.js](http://docs.google.com/pubsub.html)   [remote.js](http://docs.google.com/remote.html)   [ui.js](http://docs.google.com/ui.html)   [utils.js](http://docs.google.com/utils.html)   [validator.js](http://docs.google.com/validator.html)   [validator\_registry.js](http://docs.google.com/validator_registry.html)

# utils.js

* [¶](#gjdgxs)  
  import $ from 'jquery';  
    
  var globalID = 1;  
  var pastWarnings = {};  
    
  var Utils = {
* [¶](#30j0zll)  
  Parsley DOM-API returns object from dom attributes and values  
   attr: function (element, namespace, obj) {  
   var i;  
   var attribute;  
   var attributes;  
   var regex = new RegExp('^' + namespace, 'i');  
    
   if ('undefined' === typeof obj)  
   obj = {};  
   else {
* [¶](#1fob9te)  
  Clear all own properties. This won’t affect prototype’s values  
   for (i in obj) {  
   if (obj.hasOwnProperty(i))  
   delete obj[i];  
   }  
   }  
    
   if (!element)  
   return obj;  
    
   attributes = element.attributes;  
   for (i = attributes.length; i--; ) {  
   attribute = attributes[i];  
    
   if (attribute && attribute.specified && regex.test(attribute.name)) {  
   obj[this.camelize(attribute.name.slice(namespace.length))] = this.deserializeValue(attribute.value);  
   }  
   }  
    
   return obj;  
   },  
    
   checkAttr: function (element, namespace, checkAttr) {  
   return element.hasAttribute(namespace + checkAttr);  
   },  
    
   setAttr: function (element, namespace, attr, value) {  
   element.setAttribute(this.dasherize(namespace + attr), String(value));  
   },  
    
   getType: function(element) {  
   return element.getAttribute('type') || 'text';  
   },  
    
   generateID: function () {  
   return '' + globalID++;  
   },  
    
   /\*\* Third party functions \*\*/  
   deserializeValue: function (value) {  
   var num;  
    
   try {  
   return value ?  
   value == "true" ||  
   (value == "false" ? false :  
   value == "null" ? null :  
   !isNaN(num = Number(value)) ? num :  
   /^[\[\{]/.test(value) ? JSON.parse(value) :  
   value)  
   : value;  
   } catch (e) { return value; }  
   },
* [¶](#3znysh7)  
  Zepto camelize function  
   camelize: function (str) {  
   return str.replace(/-+(.)?/g, function (match, chr) {  
   return chr ? chr.toUpperCase() : '';  
   });  
   },
* [¶](#2et92p0)  
  Zepto dasherize function  
   dasherize: function (str) {  
   return str.replace(/::/g, '/')  
   .replace(/([A-Z]+)([A-Z][a-z])/g, '$1\_$2')  
   .replace(/([a-z\d])([A-Z])/g, '$1\_$2')  
   .replace(/\_/g, '-')  
   .toLowerCase();  
   },  
    
   warn: function () {  
   if (window.console && 'function' === typeof window.console.warn)  
   window.console.warn(...arguments);  
   },  
    
   warnOnce: function(msg) {  
   if (!pastWarnings[msg]) {  
   pastWarnings[msg] = true;  
   this.warn(...arguments);  
   }  
   },  
    
   \_resetWarnings: function () {  
   pastWarnings = {};  
   },  
    
   trimString: function(string) {  
   return string.replace(/^\s+|\s+$/g, '');  
   },  
    
   parse: {  
   date: function(string) {  
   let parsed = string.match(/^(\d{4,})-(\d\d)-(\d\d)$/);  
   if (!parsed)  
   return null;  
   let [\_, year, month, day] = parsed.map(x => parseInt(x, 10));  
   let date = new Date(year, month - 1, day);  
   if (date.getFullYear() !== year || date.getMonth() + 1 !== month || date.getDate() !== day)  
   return null;  
   return date;  
   },  
   string: function(string) {  
   return string;  
   },  
   integer: function(string) {  
   if (isNaN(string))  
   return null;  
   return parseInt(string, 10);  
   },  
   number: function(string) {  
   if (isNaN(string))  
   throw null;  
   return parseFloat(string);  
   },  
   'boolean': function \_boolean(string) {  
   return !(/^\s\*false\s\*$/i.test(string));  
   },  
   object: function(string) {  
   return Utils.deserializeValue(string);  
   },  
   regexp: function(regexp) {  
   var flags = '';
* [¶](#tyjcwt)  
  Test if RegExp is literal, if not, nothing to be done, otherwise, we need to isolate flags and pattern  
   if (/^\/.\*\/(?:[gimy]\*)$/.test(regexp)) {
* [¶](#3dy6vkm)  
  Replace the regexp literal string with the first match group: ([gimy]\*) If no flag is present, this will be a blank string  
   flags = regexp.replace(/.\*\/([gimy]\*)$/, '$1');
* [¶](#1t3h5sf)  
  Again, replace the regexp literal string with the first match group: everything excluding the opening and closing slashes and the flags  
   regexp = regexp.replace(new RegExp('^/(.\*?)/' + flags + '$'), '$1');  
   } else {
* [¶](#4d34og8)  
  Anchor regexp:  
   regexp = '^' + regexp + '$';  
   }  
   return new RegExp(regexp, flags);  
   }  
   },  
    
   parseRequirement: function(requirementType, string) {  
   var converter = this.parse[requirementType || 'string'];  
   if (!converter)  
   throw 'Unknown requirement specification: "' + requirementType + '"';  
   let converted = converter(string);  
   if (converted === null)  
   throw `Requirement is not a ${requirementType}: "${string}"`;  
   return converted;  
   },  
    
   namespaceEvents: function(events, namespace) {  
   events = this.trimString(events || '').split(/\s+/);  
   if (!events[0])  
   return '';  
   return $.map(events, evt => `${evt}.${namespace}`).join(' ');  
   },  
    
   difference: function(array, remove) {
* [¶](#2s8eyo1)  
  This is O(N^2), should be optimized  
   let result = [];  
   $.each(array, (\_, elem) => {  
   if (remove.indexOf(elem) == -1)  
   result.push(elem);  
   });  
   return result;  
   },
* [¶](#17dp8vu)  
  Alter-ego to native Promise.all, but for jQuery  
   all: function(promises) {
* [¶](#3rdcrjn)  
  jQuery treats $.when() and $.when(singlePromise) differently; let’s avoid that and add spurious elements  
   return $.when(...promises, 42, 42);  
   },
* [¶](#26in1rg)  
  Object.create polyfill, see <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/create#Polyfill>  
   objectCreate: Object.create || (function () {  
   var Object = function () {};  
   return function (prototype) {  
   if (arguments.length > 1) {  
   throw Error('Second argument not supported');  
   }  
   if (typeof prototype != 'object') {  
   throw TypeError('Argument must be an object');  
   }  
   Object.prototype = prototype;  
   var result = new Object();  
   Object.prototype = null;  
   return result;  
   };  
   })(),  
    
   \_SubmitSelector: 'input[type="submit"], button:submit'  
  };  
    
  export default Utils;