[ScrollMagic](http://docs.google.com/index.html)

* [Classes](http://docs.google.com/classes.list.html)
  + [Controller](http://docs.google.com/ScrollMagic.Controller.html)
  + [Scene](http://docs.google.com/ScrollMagic.Scene.html)
* [Events](http://docs.google.com/events.list.html)
  + [add](http://docs.google.com/ScrollMagic.Scene.html#event:add)
  + [change](http://docs.google.com/ScrollMagic.Scene.html#event:change)
  + [destroy](http://docs.google.com/ScrollMagic.Scene.html#event:destroy)
  + [end](http://docs.google.com/ScrollMagic.Scene.html#event:end)
  + [enter](http://docs.google.com/ScrollMagic.Scene.html#event:enter)
  + [leave](http://docs.google.com/ScrollMagic.Scene.html#event:leave)
  + [progress](http://docs.google.com/ScrollMagic.Scene.html#event:progress)
  + [remove](http://docs.google.com/ScrollMagic.Scene.html#event:remove)
  + [shift](http://docs.google.com/ScrollMagic.Scene.html#event:shift)
  + [start](http://docs.google.com/ScrollMagic.Scene.html#event:start)
  + [update](http://docs.google.com/ScrollMagic.Scene.html#event:update)
* [Plugins](http://docs.google.com/mixins.list.html)
  + [GSAP](http://docs.google.com/animation.GSAP.html)
  + [Velocity](http://docs.google.com/animation.Velocity.html)
  + [addIndicators](http://docs.google.com/debug.addIndicators.html)
  + [jQuery](http://docs.google.com/framework.jQuery.html)

Source: ScrollMagic/\_util.js

/\*  
 \* TODO: DOCS (private for dev)  
 \*/  
  
var \_util = ScrollMagic.\_util = (function (window) {  
 var U = {}, i;  
   
 /\*\*  
 \* ------------------------------  
 \* internal helpers  
 \* ------------------------------  
 \*/  
  
 // parse float and fall back to 0.  
 var floatval = function (number) {  
 return parseFloat(number) || 0;  
 };  
 // get current style IE safe (otherwise IE would return calculated values for 'auto')  
 var \_getComputedStyle = function (elem) {  
 return elem.currentStyle ? elem.currentStyle : window.getComputedStyle(elem);  
 };  
  
 // get element dimension (width or height)  
 var \_dimension = function (which, elem, outer, includeMargin) {  
 elem = (elem === document) ? window : elem;  
 if (elem === window) {  
 includeMargin = false;  
 } else if (!\_type.DomElement(elem)) {  
 return 0;  
 }  
 which = which.charAt(0).toUpperCase() + which.substr(1).toLowerCase();  
 var dimension = (outer ? elem['offset' + which] || elem['outer' + which] : elem['client' + which] || elem['inner' + which]) || 0;  
 if (outer && includeMargin) {  
 var style = \_getComputedStyle(elem);  
 dimension += which === 'Height' ? floatval(style.marginTop) + floatval(style.marginBottom) : floatval(style.marginLeft) + floatval(style.marginRight);  
 }  
 return dimension;  
 };  
 // converts 'margin-top' into 'marginTop'  
 var \_camelCase = function (str) {  
 return str.replace(/^[^a-z]+([a-z])/g, '$1').replace(/-([a-z])/g, function (g) { return g[1].toUpperCase(); });  
 };  
  
 /\*\*  
 \* ------------------------------  
 \* external helpers  
 \* ------------------------------  
 \*/  
  
 // extend obj – same as jQuery.extend({}, objA, objB)  
 U.extend = function (obj) {  
 obj = obj || {};  
 for (i = 1; i < arguments.length; i++) {  
 if (!arguments[i]) {  
 continue;  
 }  
 for (var key in arguments[i]) {  
 if (arguments[i].hasOwnProperty(key)) {  
 obj[key] = arguments[i][key];  
 }  
 }  
 }  
 return obj;  
 };  
  
 // check if a css display type results in margin-collapse or not  
 U.isMarginCollapseType = function (str) {  
 return ["block", "flex", "list-item", "table", "-webkit-box"].indexOf(str) > -1;  
 };  
  
 // implementation of requestAnimationFrame  
 // based on https://gist.github.com/paulirish/1579671  
 var  
 lastTime = 0,  
 vendors = ['ms', 'moz', 'webkit', 'o'];  
 var \_requestAnimationFrame = window.requestAnimationFrame;  
 var \_cancelAnimationFrame = window.cancelAnimationFrame;  
 // try vendor prefixes if the above doesn't work  
 for (i = 0; !\_requestAnimationFrame && i < vendors.length; ++i) {  
 \_requestAnimationFrame = window[vendors[i] + 'RequestAnimationFrame'];  
 \_cancelAnimationFrame = window[vendors[i] + 'CancelAnimationFrame'] || window[vendors[i] + 'CancelRequestAnimationFrame'];  
 }  
  
 // fallbacks  
 if (!\_requestAnimationFrame) {  
 \_requestAnimationFrame = function (callback) {  
 var  
 currTime = new Date().getTime(),  
 timeToCall = Math.max(0, 16 - (currTime - lastTime)),  
 id = window.setTimeout(function () { callback(currTime + timeToCall); }, timeToCall);  
 lastTime = currTime + timeToCall;  
 return id;  
 };  
 }  
 if (!\_cancelAnimationFrame) {  
 \_cancelAnimationFrame = function (id) {  
 window.clearTimeout(id);  
 };  
 }  
 U.rAF = \_requestAnimationFrame.bind(window);  
 U.cAF = \_cancelAnimationFrame.bind(window);  
  
 // (BUILD) - REMOVE IN MINIFY - START  
 var  
 loglevels = ["error", "warn", "log"],  
 console = window.console || {};  
  
 console.log = console.log || function(){}; // no console log, well - do nothing then...  
 // make sure methods for all levels exist.  
 for(i = 0; i<loglevels.length; i++) {  
 var method = loglevels[i];  
 if (!console[method]) {  
 console[method] = console.log; // prefer .log over nothing  
 }  
 }  
 U.log = function (loglevel) {  
 if (loglevel > loglevels.length || loglevel <= 0) loglevel = loglevels.length;  
 var now = new Date(),  
 time = ("0"+now.getHours()).slice(-2) + ":" + ("0"+now.getMinutes()).slice(-2) + ":" + ("0"+now.getSeconds()).slice(-2) + ":" + ("00"+now.getMilliseconds()).slice(-3),  
 method = loglevels[loglevel-1],  
 args = Array.prototype.splice.call(arguments, 1),  
 func = Function.prototype.bind.call(console[method], console);  
 args.unshift(time);  
 func.apply(console, args);  
 };  
 // (BUILD) - REMOVE IN MINIFY - END  
  
 /\*\*  
 \* ------------------------------  
 \* type testing  
 \* ------------------------------  
 \*/  
  
 var \_type = U.type = function (v) {  
 return Object.prototype.toString.call(v).replace(/^\[object (.+)\]$/, "$1").toLowerCase();  
 };  
 \_type.String = function (v) {  
 return \_type(v) === 'string';  
 };  
 \_type.Function = function (v) {  
 return \_type(v) === 'function';  
 };  
 \_type.Array = function (v) {  
 return Array.isArray(v);  
 };  
 \_type.Number = function (v) {  
 return !\_type.Array(v) && (v - parseFloat(v) + 1) >= 0;  
 };  
 \_type.DomElement = function (o){  
 return (  
 typeof HTMLElement === "object" || typeof HTMLElement === "function"? o instanceof HTMLElement || o instanceof SVGElement : //DOM2  
 o && typeof o === "object" && o !== null && o.nodeType === 1 && typeof o.nodeName==="string"  
 );  
 };  
  
 /\*\*  
 \* ------------------------------  
 \* DOM Element info  
 \* ------------------------------  
 \*/  
 // always returns a list of matching DOM elements, from a selector, a DOM element or an list of elements or even an array of selectors  
 var \_get = U.get = {};  
 \_get.elements = function (selector) {  
 var arr = [];  
 if (\_type.String(selector)) {  
 try {  
 selector = document.querySelectorAll(selector);  
 } catch (e) { // invalid selector  
 return arr;  
 }  
 }  
 if (\_type(selector) === 'nodelist' || \_type.Array(selector) || selector instanceof NodeList) {  
 for (var i = 0, ref = arr.length = selector.length; i < ref; i++) { // list of elements  
 var elem = selector[i];  
 arr[i] = \_type.DomElement(elem) ? elem : \_get.elements(elem); // if not an element, try to resolve recursively  
 }  
 } else if (\_type.DomElement(selector) || selector === document || selector === window){  
 arr = [selector]; // only the element  
 }  
 return arr;  
 };  
 // get scroll top value  
 \_get.scrollTop = function (elem) {  
 return (elem && typeof elem.scrollTop === 'number') ? elem.scrollTop : window.pageYOffset || 0;  
 };  
 // get scroll left value  
 \_get.scrollLeft = function (elem) {  
 return (elem && typeof elem.scrollLeft === 'number') ? elem.scrollLeft : window.pageXOffset || 0;  
 };  
 // get element height  
 \_get.width = function (elem, outer, includeMargin) {  
 return \_dimension('width', elem, outer, includeMargin);  
 };  
 // get element width  
 \_get.height = function (elem, outer, includeMargin) {  
 return \_dimension('height', elem, outer, includeMargin);  
 };  
  
 // get element position (optionally relative to viewport)  
 \_get.offset = function (elem, relativeToViewport) {  
 var offset = {top: 0, left: 0};  
 if (elem && elem.getBoundingClientRect) { // check if available  
 var rect = elem.getBoundingClientRect();  
 offset.top = rect.top;  
 offset.left = rect.left;  
 if (!relativeToViewport) { // clientRect is by default relative to viewport...  
 offset.top += \_get.scrollTop();  
 offset.left += \_get.scrollLeft();  
 }  
 }  
 return offset;  
 };  
  
 /\*\*  
 \* ------------------------------  
 \* DOM Element manipulation  
 \* ------------------------------  
 \*/  
  
 U.addClass = function(elem, classname) {  
 if (classname) {  
 if (elem.classList)  
 elem.classList.add(classname);  
 else  
 elem.className += ' ' + classname;  
 }  
 };  
 U.removeClass = function(elem, classname) {  
 if (classname) {  
 if (elem.classList)  
 elem.classList.remove(classname);  
 else  
 elem.className = elem.className.replace(new RegExp('(^|\\b)' + classname.split(' ').join('|') + '(\\b|$)', 'gi'), ' ');  
 }  
 };  
 // if options is string -> returns css value  
 // if options is array -> returns object with css value pairs  
 // if options is object -> set new css values  
 U.css = function (elem, options) {  
 if (\_type.String(options)) {  
 return \_getComputedStyle(elem)[\_camelCase(options)];  
 } else if (\_type.Array(options)) {  
 var  
 obj = {},  
 style = \_getComputedStyle(elem);  
 options.forEach(function(option, key) {  
 obj[option] = style[\_camelCase(option)];  
 });  
 return obj;  
 } else {  
 for (var option in options) {  
 var val = options[option];  
 if (val == parseFloat(val)) { // assume pixel for seemingly numerical values  
 val += 'px';  
 }  
 elem.style[\_camelCase(option)] = val;  
 }  
 }  
 };  
  
 return U;  
}(window || {}));

© Jan Paepke 2015

Documentation generated by [JSDoc 3.5.5](https://github.com/jsdoc3/jsdoc) using a customized version of the [DocStrap template](https://github.com/terryweiss/docstrap).