DOMAssistant

Version 2.8

DOMAssistant consists of 6 individual modules, where only the core DOMAssistant module is required. Any other module can be used without any relational dependency to each other. Within this documentation, you will find every available method documented together with some example code.

Table of Contents

DOMAssistant Core Module

The DOMAssistant Core module is required and it lays the groundwork for all DOMAssistant functionality. It consists of core functionality and a few important methods to work with.

`$(cssSelector/elementRef)`

The $ method is used to get a reference to one or several elements. It supports a CSS selector as a string, an already established element reference. It will return the matching element(s) with all the extra DOMAssistant methods applied. A call of any of those methods will fail silently, if the `$` method returned an empty array.

Parameters

Send in a CSS selector or an object reference.

Return value

Always return an array of matching elements for a CSS selector. If one single object is sent in, it return that same reference, and if several object references are sent in, it will return an array of those.

Example calls

```
$("#container input[type=text]");

$("#navigation a");

$("#news-list");

$("#container", "#navigation .important-item", "#content");

$(document.body);
```

`$$(elementId)`

The $$ method is used to get a quick reference to one element, just like

document.getElementById. It will return a direct reference to the found DOM element, with all the DOMAssistant methods applied to it.

Contrary to the <code>$</code> method, if the <code>$$</code> method didn't return any match, it will throw an error if you try to call any method on it.

Parameters

Send in the id of the element you're looking for.

Return value

Returns a DOM reference.

Example calls

```
$$("container");
```

```
$$("navigation");
```

cssSelect(cssSelector)

Intended to be used on existing object references, to use a CSS selector to find children element(s) of the current object.

Parameters

cssSelector

Used to select elements. Required.

Return value

All calls return an array of element references.

Example calls

```
$(document).cssSelect(".mandatory");
```

```
$$(DOMElementReference).cssSelect(".important[type=test]");
```

elmsByClass(className, tag)

For getting elements based on their `className`. The method has a required parameter which is the desired `className`, and one optional if you want to limit the search to a certain tag.

Parameters

className

Class name to search for. Required.

tag

Only search elements that have this tag name. Optional.

Return value

All calls return an array of element references.

Example calls

```
$("#container").elmsByClass("mandatory");

$$("container").elmsByClass("external-link", "a");
```

elmsByAttribute(attr, attrVal, tag)

For getting elements based on if they have a certain attribute. You can also specify if that attribute should have a speific value and if you want to limit the search to a certain tag. Only the first parameter specifying the attribute is required.

Parameters

attr

Attribute name to look for. Required.

attrVal

Value that the desired attribute has to have. Optional. Use wildcard character ("*") if you want any attribute value but still want to specify `tag`.

tag

Only search elements that have this tag name. Optional.

Return value

All calls return an array of element references.

Example calls

```
$("#container").elmsByAttribute("href");

$$("container").elmsByAttribute("name", "subscription");

$$("container").elmsByAttribute("type", "text", "input");
```

elmsByTag(tag)

For getting elements based on their `tag`, i.e. what element it is. The method has one required parameter which is the name of the desired `tag`.

Return value

All calls return an array of element references.

Parameters

tag

Tag name to search for. Required.

Example calls

```
$$("container").elmsByTag("input");
```

each(functionRef)

For running a function on each of the items in a returned array element reference collection.

Return value

All calls return either a single element reference or an array of element references.

Parameters

functionRef

Function which will be called for each item in the array of elements it's called on. Can be an anonymous function or a function reference. This function receives an index reference to the current item.

Example calls

```
$("#navigation a").each(function (idx) {
     alert("This is item " + idx);
});
```

indexOf(elementRef)

Returns the index of the first occurrence of the item within the collection of elements.

Return value

Zero-based index of the element if found, and -1 otherwise.

Parameters

elementRef

An element reference of the item to search for.

Example calls

```
var elm = $$("content");
var idx = $("div").indexOf(elm);
```

map(functionRef)

Runs a function on each item, and returns the results in an array.

Return value

An array of the mapped results.

Parameters

functionRef

Mapping function to be applied to each item. This function receives an index reference to the current item.

Example calls

```
// Creates an array of element ID
var arrayID = $("div").map( function(idx) {
     return this.id;
});
```

filter(functionRef)

Runs a function to filter the element collection, and returns all items where the function returned true.

Return value

The filtered element collection.

Parameters

functionRef

Filtering function to be applied to each item. This function receives an index reference to the current item.

Example calls

```
var oddItems = $("div").filter( function(idx) {
     return (idx % 2 === 1);
});
```

every(functionRef)

Runs a function on each item in the element collection, while it returns true.

Return value

Boolean value, true if all items return true on the function, false otherwise.

Parameters

functionRef

Function to be applied to each item. Returns either true or false. This function receives an index reference to the current item.

Example calls

```
var allHaveChildren = $("div").every( function(idx) {
     return (this.childNodes.length > 0);
});
```

some(functionRef)

Runs a function on each item in the element collection, while it returns false.

Return value

Boolean value, true if some items return true on the function, false otherwise.

Parameters

functionRef

Function to be applied to each item. Returns either true or false. This function receives an index reference to the current item.

Example calls

```
var someHaveChildren = $("div").some( function(idx) {
     return (this.childNodes.length > 0);
});
```

`first()`

To get a reference to the first match in a collection.

Return value

Returns a reference to the first element match in a collection, with the DOMAssistant methods applied.

Parameters

None

Example calls

```
$("#navigation a").first();
```

`end()`

To step one step up in the currect chaining context.

Return value

Returns a value to the previous element set in the chain.

Parameters

None

Example calls

```
// Returns a reference to the anchor elements
$("#navigation a").create("span", null, true).end();
```

DOMAssistantAJAX Module

The DOMAssistantAJAX module offers basic AJAX interaction for retrieving data from a URL and then passing on the returned content to any function, or load the content into an element.

get(url, callBack)

Makes a GET request to the specified URL and calls the set callBack function. The first parameter of the then called callBack function will be the `responseText` of the AJAX call. If this method is called on an element, the callBack function context will be the element, i.e. the keyword `this` will refer to the element which called the `get` method in the first place.

Parameters

url

URL to make an AJAX call to. Required.

callBack

Function name or anonymous function to call when the AJAX request is complete. Optional.

Return value

Element which called the method.

Example calls

```
$("news").get("news.php", insertNews);

DOMAssistant.AJAX.get("my-url.aspx", callbackFunctionName);
```

post(url, callBack)

Makes a POST request to the specified URL and calls the set callBack function. The first parameter of the then called callBack function will be the `responseText` of the AJAX call. If this method is called on an element, the callBack function context will be the element, i.e. the keyword `this` will refer to the element which called the `post` method in the first place.

Parameters

url

URL to make an AJAX call to. Required.

callBack

Function name or anonymous function to call when the AJAX request is complete. Optional.

Return value

Element which called the method.

Example calls

```
$("news").post("news.php?value=true", insertNews);
```

```
DOMAssistant.AJAX.post("my-url.aspx?number=10",
callbackFunctionName);
```

load(url, add)

Makes a request to the specified URL and loads the returned content into the element which the method is called on.

Parameters

url

URL to make an AJAX call to. Required.

add

A boolean, if the retrieved content shall be appended to the already existing content, as opposed to replacing it. Optional.

Return value

Element which called the method.

Example calls

```
$("directions").load("maps.php");
```

```
$("contacts").load("staff.aspx", true);
```

ajax(option)

A general-purpose method of doing more advanced AJAX calls where parameters are set manually.

Parameters

option

A JavaScript object with different paramets set. Available parameters are:

- url
- method ("GET" or "POST")
- params
- callback (function reference)
- headers
- responseType ("text", "xml" or "json")
- timeout
- exception (function reference)

The object is required, but out of the parameters, only url is mandatory. If not filled in, it defaults to a GET request with responseType `"text"`.

Return value

Element which called the method.

Example calls

```
$("#container").ajax({
     url : "ajax.php",
     method : "POST",
     params : "name=DOMAssistant",
     callback : functionReference,
     headers : {
```

```
          "Content-type" : "application/x-www-form-urlencoded"
     },
     responseType : "text",
     timeout : 5000,
     exception : function(e) {
          alert("Ajax error: " + (e.message || e.description));
     }
});
```

`getReadyState()`

Helper method to check the current `readyState` of the XMLHttpRequest.

Parameters

None

Return value

Integer.

0 = uninitialized
1 = loading
2 = loaded
3 = interactive
4 = complete

Example calls

```
DOMAssistant.AJAX.getReadyState();
```

`getStatus()`

Helper method to check the current `status` of the XMLHttpRequest.

Parameters

None

Return value

Integer. Examples are 200 for OK and 404 for Not Found.

Example calls

```
DOMAssistant.AJAX.getStatus();
```

`getStatusText()`

Helper method to check the current `status text` of the XMLHttpRequest.

Parameters

None

Return value

String. The text accompanying the status code.

Example calls

```
DOMAssistant.AJAX.getStatusText();
```

DOMAssistantContent Module

The DOMAssistantContent module offers various methods for adding and removing content and elements to the page.

`prev()`

Gets a reference to the previous HTML element, automatically bypassing any text nodes that might in between.

Parameters

None.

Return value

Element's previous sibling element.

Example calls

```
$("container").prev();
```

next()

Gets a reference to the next HTML element, automatically bypassing any text nodes that might in between.

Parameters

None.

Return value

Element's next sibling element.

Example calls

```
$("container").next();
```

create(name, attr, append, content)

Creates an element, and optionally sets attributes on it, appends it to the current element and adds content to it.

Parameters

name

Tag name for the new element. Required.

attr

An object containing attributes and their values. Optional.

append

Boolean if the new element should be appended right away. Optional.

content

A string or HTML object that will become the content of the newly created element. Optional.

Return value

Element created by the method.

Example calls

```
$("container").create("div");
```

```
$("container").create("div", {id : "my-div",className : "my-class" });
```

```
$("container").create("div", null, false, "Hello!");
```

```
$("container").create("div", {id : "my-div",className : "my-class" }, true, "Hi there!");
```

setAttributes(attr)

Sets attributes on the current element.

Parameters

attr

An object containing attributes and their values. Required.

Return value

Element which called the method.

Example calls

```
$("container").setAttributes({id : "my-div",className : "my-class" });
```

`addContent(content)`

Adds content to the current element.

Parameters

content

Can either be a string, which will then be applied using `innerHTML`, or an HTML object that will be applied using `appendChild`.

Return value

Element which called the method.

Example calls

```
$("container").addContent("<p>A new paragraph</p>");

$("container").addContent(document.createElement("p"));
```

`replaceContent(newContent)`

Replaces the content of the current element with new content.

Parameters

newContent

Can either be a string, which will then be applied using `innerHTML`, or a HTML object that will be applied using `appendChild`.

Return value

Element which called the method.

Example calls

```
$("container").replaceContent("<p>A new paragraph</p>");

$("container").replaceContent(document.createElement("p"));
```

`replace(content, returnNew)`

Replaces the current element with a new one.

Parameters

content

New content that can be specified either through an element, string or number reference.

returnNew

If set to true, the new element is returned; otherwise the replaced element is returned. Optional.

Return value

Either the replaced element or the new element.

Example calls

```
$("container").replace("<div><em>Way</em> cooler content!</div>");

var newElem = $$("placeholder").replace(elementRef, true);
```

`remove()`

Removes the current element.

Parameters

None.

Return value

Null.

Example calls

```
$("container").remove();
```

DOMAssistantCSS Module

The DOMAssistantCSS module offers various methods for adding and removing CSS classes, checking if an element has a certain class and getting the rendered style for a specific property on an element.

`addClass(className)`

Adds a class name to the current element, unless it already exists.

Parameters

className

Desired class name to be added. Required.

Return value

Element which called the method.

Example calls

```
$("container").addClass("selected");
```

`removeClass(className)`

Removes a class name from the current element.

Parameters

className

Desired class name to remove. Required.

Return value

Element which called the method.

Example calls

```
$("container").removeClass("selected");
```

hasClass(className)

Checks whether the current element has a certain class name.

Parameters

className

Class name to check if it exists on the current element. Required.

Return value

Boolean. Whether the element has the requested class or not.

Example calls

```
$("container").hasClass("selected");
```

setStyle(style, value)

Sets one or several styles inline of an element. Note: it uses CSS syntax instead of JavaScript property syntax. i.e. `background-color` instead of `backgroundColor` etc.

Parameters

style

CSS property to set. Can be a string, and is then used in conjunction with the `value`, *or as an object with several style values*. Required.

value

The value of the desired style, if the `style` parameter is a string. Optional.

Return value

Element which called the method.

Example calls

```
$("#container").setStyle("border", "10px solid red");
```

```
$("#container").setStyle({
     background : "#ffffa2",
     color : "#f00",
     opacity : 0.5
});
```

`getStyle(cssRule)`

Gets the rendered style for a certain CSS property on the current element, no matter if it was applied inline or from an external stylesheet. Note: make sure to look for the specific property instead of general ones. I.e. `background-color` instead of `background` etc.

Parameters

cssRule

CSS property to check value of. Use CSS syntax for the property, i.e. `background-color` instead of `backgroundColor`. Required.

Return value

String. The value of the requested style.

Example calls

```
$("container").getStyle("background-color");
```

DOMAssistantEvents Module

The DOMAssistantEvents module offers various methods for adding and removing handlers for one or several events on an element, using either traditional event handling or event delegation. It also contains functionality for stopping default actions and bubbling of events.

`addEvent(evt, func)`

Adds an event handler to the current element. Multiple event handlers are supported, and

the receiving funtion will have an event object reference and a `this` reference to the element it occurred on, no matter what web browser. For accessibility reasons, please make sure to only apply click events to elements that can handle them without JavaScript enabled.

Parameters

evt

Event to apply, specified as a string, without the "on" prefix. Custom events are acceptable.

func

Function to handle the event, specified as a function reference (without parentheses) or an anonymous function.

Return value

Element which called the method.

Example calls

```
$("container").addEvent("click", getListing);
```

```
$("container").addEvent("click", function (){alert("Hello darling!");});
```

removeEvent(evt, func)

Removes either a specific event handler or all of them from the current element.

Parameters

evt

Event to remove, specified as a string, without the "on" prefix.

func

Function to stop from handling the event, specified as a function reference (without parentheses). Optional. If unspecified, all handlers (including inline event handlers) for the

event will be removed.

Return value

Element which called the method.

Example calls

```
$("container").removeEvent("click", getListing);
```

```
$("tr:nth-child(odd)")
      .removeEvent("mouseover")
      .removeEvent("mouseout");
```

relayEvent(evt, selector, func)

Adds a centralized event handler to the current element. Events that occur on elements matching the selector bubble up and get handled in the current element. Most bubbling events are supported, including focus and blur.

Parameters

evt

Event to apply, specified as a string, without the "on" prefix. Custom events are acceptable.

selector

Actual targets that the event should occur on, specified as a CSS selectors.

func

Function to handle the event, specified as a function reference (without parentheses) or an anonymous function.

Return value

Element which called the method.

Example calls

```
$("ul").relayEvent("click", "li", jumpToPage);

$("form").relayEvent("focus", "input[type=text]", function() {
     this.addClass("yellow");
});
```

unrelayEvent(evt)

Removes all relayed events of the specific type from the current element.

Parameters

evt

Event to remove, specified as a string, without the "on" prefix.

Return value

Element which called the method.

Example calls

```
$("ul").unrelayEvent("click");
```

triggerEvent(evt, target)

Triggers an event on the current element, and optionally sets the target to which the event is dispatched to. Note that the actual event does not happen - this method merely triggers the event handlers.

Parameters

evt

Event to trigger, specified as a string, without the "on" prefix. Custom events are acceptable.

target

The target element to which the event is dispatched. Optional.

Return value

Element which called the method.

Example calls

```
$("news").triggerEvent("click");

$("home-link").triggerEvent("customevent", elementRef);
```

`preventDefault(evt)`

Prevents the default action of an event. Can be called from any function, and is not a method of any element.

Parameters

evt

Event to prevent the default action of.

Return value

None.

Example calls

```
DOMAssistant.preventDefault(eventReference);
```

`cancelBubble(evt)`

Cancels the bubbling of an event. Can be called from any function, and is not a method of any element.

Parameters

evt

Event to cancel the bubbling of.

Return value

None.

Example calls

```
DOMAssistant.cancelBubble(eventReference);
```

DOMAssistantLoad Module

The DOMAssistantLoad module offers a way to call a number of functions as soon as the DOM has loaded, as opposed to waiting for all images and other external files to completely load. It was inspired and influenced by Dean Edwards, Matthias Miller, and John Resig: window.onload (again).

`DOMReady()`

From any file, just call the DOMReady method with desired functions and they will be executed as soon as the DOM has loaded.

Parameters

Send in any number of function references, anonymous functions or strings with function names and parentheses.

Return value

None.

Example calls

```
DOMAssistant.DOMReady(myFunc);

DOMAssistant.DOMReady("myFunc('Some text')");

DOMAssistant.DOMReady(myFunc, "anotherFunction()");

DOMAssistant.DOMReady(myFunc, function(){// Perform some magic});
```

`setErrorHandling()`

Offers a possibility to handle errors when the `DOMReady` method is being called.

Parameters

Send in a function reference.

Return value

None.

Example calls

```
DOMAssistant.DOMLoad.setErrorHandling(function (e) {
     // e is the error object passed
});
```

Supported CSS Selectors (CSS 1, CSS 2, CSS 3)

With the DOMAssistant Core module and its $ method comes support for using CSS selectors to get a reference to one or multiple elements. There is support for any major CSS element selector in CSS 1, CSS 2 and CSS 3, with the exact same syntax as you would use in a CSS file.

For an in-depth explanation of the CSS selectors and examples of how us them, please read CSS 2.1 selectors and CSS 3 selectors explained.

Implemented CSS selectors

#container

Get an element with the id "container".

.item

Get all elements with the class name "item".

#container.item

Get an element with the id "container", if it also has the class name "item".

p.info.error

Get all P elements with both a class name consisting of both "info" and "error".

div p

Get all P elements which are descendants of any DIV element.

div > p

Get all P elements which are direct descendants of any DIV element.

div + p

Get all P elements where their immediate previous sibling element is a DIV element.

```
div ~ p
```

Get all P elements which are following siblings to a DIV element (not necessarily immediate siblings).

```
div[id]
```

Get any DIV element which has an ID attribute.

```
div[id=container]
```

Get any DIV element which has an ID attribute with the value of "container".

```
input[type=text][value=Yes]
```

Get any INPUT element which has a TYPE attribute with the value of "text" and a VALUE attribute which is ="Yes".

```
div[id^=empt]
```

Get any DIV element where the ID attribute value begins with "empt".

```
div[id$=parent]
```

Get any DIV element where the ID attribute value ends with "parent".

```
div[id*=mpt]
```

Get any DIV element where the ID attribute value contains the text "mpt".

```
div[foo~=bar]
```

Get any DIV element where the foo attribute is a list of space-separated values, one of which is exactly equal to "bar".

```
div[lang|=en]
```

Get any DIV element where the lang attribute has a hyphen-separated list of values beginning (from the left) with "en".

div:first-child

Get any DIV element which is the first child of its parent element.

div:last-child

Get any DIV element which is the last child of its parent.

div:only-child

Get any DIV element which is the only child of its parent.

div#container p:first-of-type

Get the P element which is the first P element child of a DIV element with an ID attribute with the value "container".

p:last-of-type

Get the P element which is the last P element child of its parent element.

p:only-of-type

Get any P element which is the only P element child of its parent element.

p:nth-of-type(7)

Get the P element which is the 7th P element child of its parent element.

div:empty

Get any DIV element which is completely empty (including text nodes).

div:not([id=container])

Get any DIV element where its ID attribute is not "container".

div:nth-child(3)

Get any DIV element which is the third child element of its parent element.

div:nth-child(odd)

Get every odd DIV child element of its parent element.

div:nth-child(even)

Get every even DIV child element of its parent element.

div:nth-child(5n+3)

Get every 5th DIV child element of its parent element, starting at 3, then 8, 13, 18 etc.

tr:nth-last-of-type(-n+3)

Get the last three rows of any tables.

td:nth-last-child(n+1)

Get all but the last column of any tables.

input:enabled

Get any INPUT element which is enabled.

input:disabled

Get any INPUT element which is disabled.

input:checked

Get any INPUT element which is checked.

div:lang(zh)

Get any DIV element which is in Chinese.

p:target

Get the P element which is the target of the referring URL.

p, a

Get all P elements and all A elements.