JQuery 1.1.4 with official plugins

Documentation generated automatically from source on Sun, 26 Aug 2007 11:45:03 GMT

1. Core

$(String,Element|jQuery)

`$(String expr, Element|jQuery context) returns jQuery`

This function accepts a string containing a CSS or basic XPath selector which is then used to match a set of elements.
The core functionality of jQuery centers around this function. Everything in jQuery is based upon this, or uses this in some way. The most basic use of this function is to pass in an expression (usually consisting of CSS or XPath), which then finds all matching elements.
By default, if no context is specified, $() looks for DOM elements within the context of the current HTML document. If you do specify a context, such as a DOM element or jQuery object, the expression will be matched against the contents of that context.
See [[DOM/Traversing/Selectors]] for the allowed CSS/XPath syntax for expressions.

Example:

Finds all p elements that are children of a div element.

```
$("div > p")
```

HTML:

```
<p>one</p> <div><p>two</p></div> <p>three</p>
```

Result:

```
[ <p>two</p> ]
```

Example:

Searches for all inputs of type radio within the first form in the document

```
$("input:radio", document.forms[0])
```

Example:

This finds all div elements within the specified XML document.

```
$("div", xml.responseXML)
```

$(String)

`$(String html) returns jQuery`

Create DOM elements on-the-fly from the provided String of raw HTML.

Example:

Creates a div element (and all of its contents) dynamically, and appends it to the body element. Internally, an element is created and its innerHTML property set to the given markup. It is therefore both quite flexible and limited.

```
$("<div><p>Hello</p></div>").appendTo("body")
```

$(Element|Array&lt;Element&gt;)

`$(Element|Array<Element> elems) returns jQuery`

Wrap jQuery functionality around a single or multiple DOM Element(s).
This function also accepts XML Documents and Window objects as valid arguments (even though they are not DOM Elements).

Example:

Sets the background color of the page to black.

```
$(document.body).css( "background", "black" );
```

Example:

Hides all the input elements within a form

```
$( myForm.elements ).hide()
```

$(Function)

`$(Function fn) returns jQuery`

A shorthand for $(document).ready(), allowing you to bind a function to be executed when the DOM document has finished loading. This function behaves just like $(document).ready(), in that it should be used to wrap other $() operations on your page that depend on the DOM being ready to be operated on. While this function is, technically, chainable - there really isn't much use for chaining against it.
You can have as many $(document).ready events on your page as you like.
See ready(Function) for details about the ready event.

Example:

Executes the function when the DOM is ready to be used.

```
$(function(){
 // Document is ready
});
```

Example:

Uses both the shortcut for $(document).ready() and the argument to write failsafe jQuery code using the $ alias, without relying on the global alias.

```
jQuery(function($) {
 // Your code using failsafe $ alias here...
});
```

length()

`length() returns Number`

The number of elements currently matched. The size function will return the same value.

Example:

```
$("img").length;
```

HTML:

```
<img src="test1.jpg"/> <img src="test2.jpg"/>
```

Result:

```
2
```

size()

`size() returns Number`

Get the number of elements currently matched. This returns the same number as the 'length' property of the jQuery object.

Example:

```
$("img").size();
```

HTML:

```
<img src="test1.jpg"/> <img src="test2.jpg"/>
```

Result:

```
2
```

get()

`get() returns Array<Element>`

Access all matched DOM elements. This serves as a backwards-compatible way of accessing all matched elements (other than the jQuery object itself, which is, in fact, an array of elements).
It is useful if you need to operate on the DOM elements themselves instead of using built-in jQuery functions.

Example:

Selects all images in the document and returns the DOM Elements as an Array

```
$("img").get();
```

HTML:

```
<img src="test1.jpg"/> <img src="test2.jpg"/>
```

Result:

```
[ <img src="test1.jpg"/> <img src="test2.jpg"/> ]
```

get(Number)

`get(Number num) returns Element`

Access a single matched DOM element at a specified index in the matched set. This allows you to extract the actual DOM element and operate on it directly without necessarily using jQuery functionality on it.

Example:

Selects all images in the document and returns the first one

```
$("img").get(0);
```

HTML:

```
<img src="test1.jpg"/> <img src="test2.jpg"/>
```

Result:

```
<img src="test1.jpg"/>
```

each(Function)

`each(Function fn) returns jQuery`

Execute a function within the context of every matched element. This means that every time the passed-in function is executed (which is once for every element matched) the 'this' keyword points to the specific DOM element.
Additionally, the function, when executed, is passed a single argument representing the position of the element in the matched set (integer, zero-index).

Example:

Iterates over two images and sets their src property

```
$("img").each(function(i){
 this.src = "test" + i + ".jpg";
});
```

HTML:

```
<img/><img/>
```

Result:

```
<img src="test0.jpg"/><img src="test1.jpg"/>
```

index(Element)

`index(Element subject) returns Number`

Searches every matched element for the object and returns the index of the element, if found, starting with zero. Returns -1 if the object wasn't found.

Example:

Returns the index for the element with ID foobar

```
$("*").index( $('#foobar')[0] )
```

HTML:

```
<div id="foobar"><b></b><span id="foo"></span></div>
```

Result:

```
0
```

Example:

Returns the index for the element with ID foo within another element

```
$("*").index( $('#foo')[0] )
```

HTML:

```
<div id="foobar"><b></b><span id="foo"></span></div>
```

Result:

```
2
```

Example:

Returns -1, as there is no element with ID bar

```
$("*").index( $('#bar')[0] )
```

HTML:

```
<div id="foobar"><b></b><span id="foo"></span></div>
```

Result:

```
-1
```

$.extend(Object)

`$.extend(Object prop) returns Object`

Extends the jQuery object itself. Can be used to add functions into the jQuery namespace and to [[Plugins/Authoring|add plugin methods]] (plugins).

Example:

Adds two plugin methods.

```
jQuery.fn.extend({
 check: function() {
   return this.each(function() { this.checked = true; });
 },
 uncheck: function() {
   return this.each(function() { this.checked = false; });
 }
});
$("input[@type=checkbox]").check();
$("input[@type=radio]").uncheck();
```

Example:

Adds two functions into the jQuery namespace

```
jQuery.extend({
 min: function(a, b) { return a < b ? a : b; },
 max: function(a, b) { return a > b ? a : b; }
});
```

$.noConflict()

`$.noConflict() returns undefined`

Run this function to give control of the $ variable back to whichever library first implemented it. This helps to make sure that jQuery doesn't conflict with the $ object of other libraries.
By using this function, you will only be able to access jQuery using the 'jQuery' variable. For example, where you used to do $("div p"), you now must do jQuery("div p").

Example:

Maps the original object that was referenced by $ back to $

```
jQuery.noConflict();
// Do something with jQuery
jQuery("div p").hide();
// Do something with another library's $()
$("content").style.display = 'none';
```

Example:

Reverts the $ alias and then creates and executes a function to provide the $ as a jQuery alias inside the functions scope. Inside the function the original $ object is not available. This works well for most plugins that don't rely on any other library.

```
jQuery.noConflict();
(function($) {
 $(function() {
   // more code using $ as alias to jQuery
 });
})(jQuery);
// other code using $ as an alias to the other library
```

eq(Number)

eq(Number pos) returns jQuery

Reduce the set of matched elements to a single element. The position of the element in the set of matched elements starts at 0 and goes to length - 1.

Example:

```
$("p").eq(1)
```

HTML:

```
<p>This is just a test.</p><p>So is this</p>
```

Result:

```
[ <p>So is this</p> ]
```

lt(Number)

`lt(Number pos) returns jQuery`

Reduce the set of matched elements to all elements before a given position. The position of the element in the set of matched elements starts at 0 and goes to length - 1.

Example:

```
$("p").lt(1)
```

HTML:

```
<p>This is just a test.</p><p>So is this</p>
```

Result:

```
[ <p>This is just a test.</p> ]
```

gt(Number)

`gt(Number pos) returns jQuery`

Reduce the set of matched elements to all elements after a given position. The position of the element in the set of matched elements starts at 0 and goes to length - 1.

Example:

```
$("p").gt(0)
```

HTML:

```
<p>This is just a test.</p><p>So is this</p>
```

Result:

```
[ <p>So is this</p> ]
```

2. DOM

Attributes

attr(String)

`attr(String name) returns Object`

Access a property on the first matched element. This method makes it easy to retrieve a property value from the first matched element.
If the element does not have an attribute with such a name, undefined is returned.

Example:

Returns the src attribute from the first image in the document.

```
$("img").attr("src");
```

HTML:

```
<img src="test.jpg"/>
```

Result:

```
test.jpg
```

attr(Map)

`attr(Map properties) returns jQuery`

Set a key/value object as properties to all matched elements.
This serves as the best way to set a large number of properties on all matched elements.

Example:

Sets src and alt attributes to all images.

```
$("img").attr({ src: "test.jpg", alt: "Test Image" });
```

HTML:

```
<img/>
```

Result:

```
<img src="test.jpg" alt="Test Image"/>
```

attr(String,Object)

`attr(String key, Object value) returns jQuery`

Set a single property to a value, on all matched elements.
Note that you can't set the name property of input elements in IE. Use $(html) or .append(html) or .html(html) to create elements on the fly including the name property.

Example:

Sets src attribute to all images.

```
$("img").attr("src","test.jpg");
```

HTML:

```
<img/>
```

Result:

```
<img src="test.jpg"/>
```

attr(String,Function)

`attr(String key, Function value) returns jQuery`

Set a single property to a computed value, on all matched elements.
Instead of supplying a string value as described [[DOM/Attributes#attr.28_key.2C_value_.29|above]], a function is provided that computes the value.

Example:

Sets title attribute from src attribute.

```
$("img").attr("title", function() { return this.src });
```

HTML:

```
<img src="test.jpg" />
```

Result:

```
<img src="test.jpg" title="test.jpg" />
```

Example:

Enumerate title attribute.

```
$("img").attr("title", function(index) { return this.title + (i + 1); });
```

HTML:

```
<img title="pic" /><img title="pic" /><img title="pic" />
```

Result:

```
<img title="pic1" /><img title="pic2" /><img title="pic3" />
```

text()

```
text() returns String
```

Get the text contents of all matched elements. The result is a string that contains the combined text contents of all matched elements. This method works on both HTML and XML documents.

Example:

Gets the concatenated text of all paragraphs

```
$("p").text();
```

HTML:

```
<p><b>Test</b> Paragraph.</p><p>Paraparagraph</p>
```

Result:

```
Test Paragraph.Paraparagraph
```

text(String)

`text(String val) returns String`

Set the text contents of all matched elements.
Similar to html(), but escapes HTML (replace "<" and ">" with their HTML entities).

Example:

Sets the text of all paragraphs.

```
$("p").text("<b>Some</b> new text.");
```

HTML:

```
<p>Test Paragraph.</p>
```

Result:

```
<p>&lt;b&gt;Some&lt;/b&gt; new text.</p>
```

Example:

Sets the text of all paragraphs.

```
$("p").text("<b>Some</b> new text.", true);
```

HTML:

```
<p>Test Paragraph.</p>
```

Result:

```
<p>Some new text.</p>
```

val()

`val() returns String`

Get the content of the value attribute of the first matched element.
Use caution when relying on this function to check the value of multiple-select elements and checkboxes in a form. While it will still work as intended, it may not accurately represent the value the server will receive because these elements may send an array of values. For more robust handling of field values, see the [http://www.malsup.com/jquery/form/#fields fieldValue function of the Form Plugin].

Example:

```
$("input").val();
```

HTML:

```
<input type="text" value="some text"/>
```

Result:

```
"some text"
```

val(String)

`val(String val) returns jQuery`

Set the value attribute of every matched element.

Example:

```
$("input").val("test");
```

HTML:

```
<input type="text" value="some text"/>
```

Result:

```
<input type="text" value="test"/>
```

html()

`html() returns String`

Get the html contents of the first matched element. This property is not available on XML documents.

Example:

```
$("div").html();
```

HTML:

```
<div><input/></div>
```

Result:

```
<input/>
```

html(String)

`html(String val) returns jQuery`

Set the html contents of every matched element. This property is not available on XML documents.

Example:

```
$("div").html("<b>new stuff</b>");
```

HTML:

```
<div><input/></div>
```

Result:

```
<div><b>new stuff</b></div>
```

removeAttr(String)

`removeAttr(String name) returns jQuery`

Remove an attribute from each of the matched elements.

Example:

```
$("input").removeAttr("disabled")
```

HTML:

```
<input disabled="disabled"/>
```

Result:

```
<input/>
```

addClass(String)

`addClass(String class) returns jQuery`

Adds the specified class(es) to each of the set of matched elements.

Example:

```
$("p").addClass("selected")
```

HTML:

```
<p>Hello</p>
```

Result:

```
[ <p class="selected">Hello</p> ]
```

Example:

```
$("p").addClass("selected highlight")
```

HTML:

```
<p>Hello</p>
```

Result:

```
[ <p class="selected highlight">Hello</p> ]
```

removeClass(String)

`removeClass(String class) returns jQuery`

Removes all or the specified class(es) from the set of matched elements.

Example:

```
$("p").removeClass()
```

HTML:

```
<p class="selected">Hello</p>
```

Result:

```
[ <p>Hello</p> ]
```

Example:

```
$("p").removeClass("selected")
```

HTML:

```
<p class="selected first">Hello</p>
```

Result:

```
[ <p class="first">Hello</p> ]
```

Example:

```
$("p").removeClass("selected highlight")
```

HTML:

```
<p class="highlight selected first">Hello</p>
```

Result:

```
[ <p class="first">Hello</p> ]
```

toggleClass(String)

`toggleClass(String class) returns jQuery`

Adds the specified class if it is not present, removes it if it is present.

Example:

```
$("p").toggleClass("selected")
```

HTML:

```
<p>Hello</p><p class="selected">Hello Again</p>
```

Result:

```
[ <p class="selected">Hello</p>, <p>Hello Again</p> ]
```

Manipulation

wrap(String)

`wrap(String html) returns jQuery`

Wrap all matched elements with a structure of other elements. This wrapping process is most useful for injecting additional stucture into a document, without ruining the original semantic qualities of a document.
This works by going through the first element provided (which is generated, on the fly, from the provided HTML) and finds the deepest ancestor element within its structure - it is that element that will en-wrap everything else.
This does not work with elements that contain text. Any necessary text must be added after the wrapping is done.

Example:

```
$("p").wrap("<div class='wrap'></div>");
```

HTML:

```
<p>Test Paragraph.</p>
```

Result:

```
<div class='wrap'><p>Test Paragraph.</p></div>
```

wrap(Element)

`wrap(Element elem) returns jQuery`

Wrap all matched elements with a structure of other elements. This wrapping process is most useful for injecting additional stucture into a document, without ruining the original semantic qualities of a document.
This works by going through the first element provided and finding the deepest ancestor element within its structure - it is that element that will en-wrap everything else.
This does not work with elements that contain text. Any necessary text must be added after the wrapping is done.

Example:

```
$("p").wrap( document.getElementById('content') );
```

HTML:

```
<p>Test Paragraph.</p><div id="content"></div>
```

Result:

```
<div id="content"><p>Test Paragraph.</p></div>
```

append(&lt;Content&gt;)

`append(<Content> content) returns jQuery`

Append content to the inside of every matched element.
This operation is similar to doing an appendChild to all the specified elements, adding them into the document.

Example:

Appends some HTML to all paragraphs.

```
$("p").append("<b>Hello</b>");
```

HTML:

```
<p>I would like to say: </p>
```

Result:

```
<p>I would like to say: <b>Hello</b></p>
```

Example:

Appends an Element to all paragraphs.

```
$("p").append( $("#foo")[0] );
```

HTML:

```
<p>I would like to say: </p><b id="foo">Hello</b>
```

Result:

```
<p>I would like to say: <b id="foo">Hello</b></p>
```

Example:

Appends a jQuery object (similar to an Array of DOM Elements) to all paragraphs.

```
$("p").append( $("b") );
```

HTML:

```
<p>I would like to say: </p><b>Hello</b>
```

Result:

```
<p>I would like to say: <b>Hello</b></p>
```

prepend(&lt;Content&gt;)

`prepend(<Content> content) returns jQuery`

Prepend content to the inside of every matched element.
This operation is the best way to insert elements inside, at the beginning, of all matched elements.

Example:

Prepends some HTML to all paragraphs.

```
$("p").prepend("<b>Hello</b>");
```

HTML:

```
<p>I would like to say: </p>
```

Result:

```
<p><b>Hello</b>I would like to say: </p>
```

Example:

Prepends an Element to all paragraphs.

```
$("p").prepend( $("#foo")[0] );
```

HTML:

```
<p>I would like to say: </p><b id="foo">Hello</b>
```

Result:

```
<p><b id="foo">Hello</b>I would like to say: </p>
```

Example:

Prepends a jQuery object (similar to an Array of DOM Elements) to all paragraphs.

```
$("p").prepend( $("b") );
```

HTML:

```
<p>I would like to say: </p><b>Hello</b>
```

Result:

```
<p><b>Hello</b>I would like to say: </p>
```

before(&lt;Content&gt;)

`before(<Content> content) returns jQuery`

Insert content before each of the matched elements.

Example:

Inserts some HTML before all paragraphs.

```
$("p").before("<b>Hello</b>");
```

HTML:

```
<p>I would like to say: </p>
```

Result:

```
<b>Hello</b><p>I would like to say: </p>
```

Example:

Inserts an Element before all paragraphs.

```
$("p").before( $("#foo")[0] );
```

HTML:

```
<p>I would like to say: </p><b id="foo">Hello</b>
```

Result:

```
<b id="foo">Hello</b><p>I would like to say: </p>
```

Example:

Inserts a jQuery object (similar to an Array of DOM Elements) before all paragraphs.

```
$("p").before( $("b") );
```

HTML:

```
<p>I would like to say: </p><b>Hello</b>
```

Result:

```
<b>Hello</b><p>I would like to say: </p>
```

after(&lt;Content&gt;)

after(<Content> content) returns jQuery

Insert content after each of the matched elements.

Example:

Inserts some HTML after all paragraphs.

```
$("p").after("<b>Hello</b>");
```

HTML:

```
<p>I would like to say: </p>
```

Result:

```
<p>I would like to say: </p><b>Hello</b>
```

Example:

Inserts an Element after all paragraphs.

```
$("p").after( $("#foo")[0] );
```

HTML:

```
<b id="foo">Hello</b><p>I would like to say: </p>
```

Result:

```
<p>I would like to say: </p><b id="foo">Hello</b>
```

Example:

Inserts a jQuery object (similar to an Array of DOM Elements) after all paragraphs.

```
$("p").after( $("b") );
```

HTML:

```
<b>Hello</b><p>I would like to say: </p>
```

Result:

```
<p>I would like to say: </p><b>Hello</b>
```

clone(Boolean)

`clone(Boolean deep) returns jQuery`

Clone matched DOM Elements and select the clones.
This is useful for moving copies of the elements to another location in the DOM.

Example:

Clones all b elements (and selects the clones) and prepends them to all paragraphs.

```
$("b").clone().prependTo("p");
```

HTML:

```
<b>Hello</b><p>, how are you?</p>
```

Result:

```
<b>Hello</b><p><b>Hello</b>, how are you?</p>
```

appendTo(&lt;Content&gt;)

```
appendTo(<Content> content) returns jQuery
```

Append all of the matched elements to another, specified, set of elements. This operation is, essentially, the reverse of doing a regular $(A).append(B), in that instead of appending B to A, you're appending A to B.

Example:

Appends all paragraphs to the element with the ID "foo"

```
$("p").appendTo("#foo");
```

HTML:

```
<p>I would like to say: </p><div id="foo"></div>
```

Result:

```
<div id="foo"><p>I would like to say: </p></div>
```

prependTo(&lt;Content&gt;)

`prependTo(<Content> content) returns jQuery`

Prepend all of the matched elements to another, specified, set of elements. This operation is, essentially, the reverse of doing a regular $(A).prepend(B), in that instead of prepending B to A, you're prepending A to B.

Example:

Prepends all paragraphs to the element with the ID "foo"

```
$("p").prependTo("#foo");
```

HTML:

```
<p>I would like to say: </p><div id="foo"><b>Hello</b></div>
```

Result:

```
<div id="foo"><p>I would like to say: </p><b>Hello</b></div>
```

insertBefore(&lt;Content&gt;)

`insertBefore(<Content> content) returns jQuery`

Insert all of the matched elements before another, specified, set of elements. This operation is, essentially, the reverse of doing a regular $(A).before(B), in that instead of inserting B before A, you're inserting A before B.

Example:

Same as $("#foo").before("p")

```
$("p").insertBefore("#foo");
```

HTML:

```
<div id="foo">Hello</div><p>I would like to say: </p>
```

Result:

```
<p>I would like to say: </p><div id="foo">Hello</div>
```

insertAfter(&lt;Content&gt;)

```
insertAfter(<Content> content) returns jQuery
```

Insert all of the matched elements after another, specified, set of elements. This operation is, essentially, the reverse of doing a regular $(A).after(B), in that instead of inserting B after A, you're inserting A after B.

Example:

Same as $("#foo").after("p")

```
$("p").insertAfter("#foo");
```

HTML:

```
<p>I would like to say: </p><div id="foo">Hello</div>
```

Result:

```
<div id="foo">Hello</div><p>I would like to say: </p>
```

remove(String)

```
remove(String expr) returns jQuery
```

Removes all matched elements from the DOM. This does NOT remove them from the jQuery object, allowing you to use the matched elements further.
Can be filtered with an optional expressions.

Example:

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

HTML:

```
<p>Hello</p> how are <p>you?</p>
```

Result:

```
how are
```

Example:

```
$("p").remove(".hello");
```

HTML:

```
<p class="hello">Hello</p> how are <p>you?</p>
```

Result:

```
how are <p>you?</p>
```

empty()

`empty() returns jQuery`

Removes all child nodes from the set of matched elements.

Example:

```
$("p").empty()
```

HTML:

```
<p>Hello, <span>Person</span> <a href="#">and person</a></p>
```

Result:

```
[ <p></p> ]
```

Traversing

end()

`end() returns jQuery`

Revert the most recent 'destructive' operation, changing the set of matched elements to its previous state (right before the destructive operation).
If there was no destructive operation before, an empty set is returned.
A 'destructive' operation is any operation that changes the set of matched jQuery elements. These functions are: <code>add</code>, <code>children</code>, <code>clone</code>, <code>filter</code>, <code>find</code>, <code>not</code>, <code>next</code>, <code>parent</code>, <code>parents</code>, <code>prev</code> and <code>siblings</code>.

Example:

Selects all paragraphs, finds span elements inside these, and reverts the selection back to the paragraphs.

```
$("p").find("span").end();
```

HTML:

```
<p><span>Hello</span>, how are you?</p>
```

Result:

```
[ <p>...</p> ]
```

find(String)

`find(String expr) returns jQuery`

Searches for all elements that match the specified expression.
This method is a good way to find additional descendant elements with which to process.
All searching is done using a jQuery expression. The expression can be written using CSS 1-3 Selector syntax, or basic XPath.

Example:

Starts with all paragraphs and searches for descendant span elements, same as $("p span")

```
$("p").find("span");
```

HTML:

```
<p><span>Hello</span>, how are you?</p>
```

Result:

```
[ <span>Hello</span> ]
```

filter(String)

`filter(String expression) returns jQuery`

Removes all elements from the set of matched elements that do not match the specified expression(s). This method is used to narrow down the results of a search.
Provide a comma-separated list of expressions to apply multiple filters at once.

Example:

Selects all paragraphs and removes those without a class "selected".

```
$("p").filter(".selected")
```

HTML:

```
<p class="selected">Hello</p><p>How are you?</p>
```

Result:

```
[ <p class="selected">Hello</p> ]
```

Example:

Selects all paragraphs and removes those without class "selected" and being the first one.

```
$("p").filter(".selected, :first")
```

HTML:

```
<p>Hello</p><p>Hello Again</p><p class="selected">And Again</p>
```

Result:

```
[ <p>Hello</p>, <p class="selected">And Again</p> ]
```

filter(Function)

`filter(Function filter) returns jQuery`

Removes all elements from the set of matched elements that do not pass the specified filter. This method is used to narrow down the results of a search.

Example:

Remove all elements that have a child ol element

```
$("p").filter(function(index) {
 return $("ol", this).length == 0;
})
```

HTML:

```
<p><ol><li>Hello</li></ol></p><p>How are you?</p>
```

Result:

```
[ <p>How are you?</p> ]
```

not(Element)

`not(Element el) returns jQuery`

Removes the specified Element from the set of matched elements. This method is used to remove a single Element from a jQuery object.

Example:

Removes the element with the ID "selected" from the set of all paragraphs.

```
$("p").not( $("#selected")[0] )
```

HTML:

```
<p>Hello</p><p id="selected">Hello Again</p>
```

Result:

```
[ <p>Hello</p> ]
```

not(String)

`not(String expr) returns jQuery`

Removes elements matching the specified expression from the set of matched elements. This method is used to remove one or more elements from a jQuery object.

Example:

Removes the element with the ID "selected" from the set of all paragraphs.

```
$("p").not("#selected")
```

HTML:

```
<p>Hello</p><p id="selected">Hello Again</p>
```

Result:

```
[ <p>Hello</p> ]
```

not(jQuery)

`not(jQuery elems) returns jQuery`

Removes any elements inside the array of elements from the set of matched elements. This method is used to remove one or more elements from a jQuery object.
Please note: the expression cannot use a reference to the element name. See the two examples below.

Example:

Removes all elements that match "div p.selected" from the total set of all paragraphs.

```
$("p").not( $("div p.selected") )
```

HTML:

```
<div><p>Hello</p><p class="selected">Hello Again</p></div>
```

Result:

```
[ <p>Hello</p> ]
```

add(String)

add(String expr) returns jQuery

Adds more elements, matched by the given expression, to the set of matched elements.

Example:

Compare the above result to the result of <code>$('p')</code>, which would just result in <code><nowiki>[<p>Hello</p>]</nowiki></code>. Using add(), matched elements of <code>$('span')</code> are simply added to the returned jQuery-object.

```
$("p").add("span")
```

HTML:

```
(HTML) <p>Hello</p><span>Hello Again</span>
```

Result:

```
(jQuery object matching 2 elements) [ <p>Hello</p>, <span>Hello Again</span> ]
```

add(String html) returns jQuery

Adds more elements, created on the fly, to the set of matched elements.

Example:

```
$("p").add("<span>Again</span>")
```

HTML:

```
<p>Hello</p>
```

Result:

```
[ <p>Hello</p>, <span>Again</span> ]
```

add(Element|Array&lt;Element&gt;)

`add(Element|Array<Element> elements) returns jQuery`

Adds one or more Elements to the set of matched elements.

Example:

```
$("p").add( document.getElementById("a") )
```

HTML:

```
<p>Hello</p><p><span id="a">Hello Again</span></p>
```

Result:

```
[ <p>Hello</p>, <span id="a">Hello Again</span> ]
```

Example:

```
$("p").add( document.forms[0].elements )
```

HTML:

```
<p>Hello</p><p><form><input/><button/></form>
```

Result:

```
[ <p>Hello</p>, <input/>, <button/> ]
```

is(String)

`is(String expr) returns Boolean`

Checks the current selection against an expression and returns true, if at least one element of the selection fits the given expression.
Does return false, if no element fits or the expression is not valid.
filter(String) is used internally, therefore all rules that apply there apply here, too.

Example:

Returns true, because the parent of the input is a form element

```
$("input[@type='checkbox']").parent().is("form")
```

HTML:

```
<form><input type="checkbox" /></form>
```

Result:

```
true
```

Example:

Returns false, because the parent of the input is a p element

```
$("input[@type='checkbox']").parent().is("form")
```

HTML:

```
<form><p><input type="checkbox" /></p></form>
```

Result:

```
false
```

parent(String)

`parent(String expr) returns jQuery`

Get a set of elements containing the unique parents of the matched set of elements.
You may use an optional expression to filter the set of parent elements that will match.

Example:

Find the parent element of each paragraph.

```
$("p").parent()
```

HTML:

```
<div><p>Hello</p><p>Hello</p></div>
```

Result:

```
[ <div><p>Hello</p><p>Hello</p></div> ]
```

Example:

Find the parent element of each paragraph with a class "selected".

```
$("p").parent(".selected")
```

HTML:

```
<div><p>Hello</p></div><div class="selected"><p>Hello Again</p></div>
```

Result:

```
[ <div class="selected"><p>Hello Again</p></div> ]
```

parents(String)

`parents(String expr) returns jQuery`

Get a set of elements containing the unique ancestors of the matched set of elements (except for the root element).
The matched elements can be filtered with an optional expression.

Example:

Find all parent elements of each span.

```
$("span").parents()
```

HTML:

```
<html><body><div><p><span>Hello</span></p><span>Hello
Again</span></div></body></html>
```

Result:

```
[ <body>...</body>, <div>...</div>, <p><span>Hello</span></p> ]
```

Example:

Find all parent elements of each span that is a paragraph.

```
$("span").parents("p")
```

HTML:

```
<html><body><div><p><span>Hello</span></p><span>Hello
Again</span></div></body></html>
```

Result:

```
[ <p><span>Hello</span></p> ]
```

next(String)

`next(String expr) returns jQuery`

Get a set of elements containing the unique next siblings of each of the matched set of elements.
It only returns the very next sibling for each element, not all next siblings.
You may provide an optional expression to filter the match.

Example:

Find the very next sibling of each paragraph.

```
$("p").next()
```

HTML:

```
<p>Hello</p><p>Hello Again</p><div><span>And Again</span></div>
```

Result:

```
[ <p>Hello Again</p>, <div><span>And Again</span></div> ]
```

Example:

Find the very next sibling of each paragraph that has a class "selected".

```
$("p").next(".selected")
```

HTML:

```
<p>Hello</p><p class="selected">Hello Again</p><div><span>And Again</span></div>
```

Result:

```
[ <p class="selected">Hello Again</p> ]
```

prev(String)

`prev(String expr) returns jQuery`

Get a set of elements containing the unique previous siblings of each of the matched set of elements.
Use an optional expression to filter the matched set.
Only the immediately previous sibling is returned, not all previous siblings.

Example:

Find the very previous sibling of each paragraph.

```
$("p").prev()
```

HTML:

```
<p>Hello</p><div><span>Hello Again</span></div><p>And Again</p>
```

Result:

```
[ <div><span>Hello Again</span></div> ]
```

Example:

Find the very previous sibling of each paragraph that has a class "selected".

```
$("p").prev(".selected")
```

HTML:

```
<div><span>Hello</span></div><p class="selected">Hello Again</p><p>And Again</p>
```

Result:

```
[ <div><span>Hello</span></div> ]
```

siblings(String)

```
siblings(String expr) returns jQuery
```

Get a set of elements containing all of the unique siblings of each of the matched set of elements. Can be filtered with an optional expressions.

Example:

Find all siblings of each div.

```
$("div").siblings()
```

HTML:

```
<p>Hello</p><div><span>Hello Again</span></div><p>And Again</p>
```

Result:

```
[ <p>Hello</p>, <p>And Again</p> ]
```

Example:

Find all siblings with a class "selected" of each div.

```
$("div").siblings(".selected")
```

HTML:

```
<div><span>Hello</span></div><p class="selected">Hello Again</p><p>And Again</p>
```

Result:

```
[ <p class="selected">Hello Again</p> ]
```

children(String)

`children(String expr) returns jQuery`

Get a set of elements containing all of the unique children of each of the matched set of elements. This set can be filtered with an optional expression that will cause only elements matching the selector to be collected.

Example:

Find all children of each div.

```
$("div").children()
```

HTML:

```
<p>Hello</p><div><span>Hello Again</span></div><p>And Again</p>
```

Result:

```
[ <span>Hello Again</span> ]
```

Example:

Find all children with a class "selected" of each div.

```
$("div").children(".selected")
```

HTML:

```
<div><span>Hello</span><p class="selected">Hello Again</p><p>And Again</p></div>
```

Result:

```
[ <p class="selected">Hello Again</p> ]
```

contains(String)

`contains(String str) returns jQuery`

Filter the set of elements to those that contain the specified text.

Example:

```
$("p").contains("test")
```

HTML:

```
<p>This is just a test.</p><p>So is this</p>
```

Result:

```
[ <p>This is just a test.</p> ]
```

3. CSS

css(String)

`css(String name) returns String`

Access a style property on the first matched element. This method makes it easy to retrieve a style property value from the first matched element.

Example:

Retrieves the color style of the first paragraph

```
$("p").css("color");
```

HTML:

```
<p style="color:red;">Test Paragraph.</p>
```

Result:

```
"red"
```

Example:

Retrieves the font-weight style of the first paragraph.

```
$("p").css("font-weight");
```

HTML:

```
<p style="font-weight: bold;">Test Paragraph.</p>
```

Result:

```
"bold"
```

css(Map)

```
css(Map properties) returns jQuery
```

Set a key/value object as style properties to all matched elements.
This serves as the best way to set a large number of style properties on all matched elements.

Example:

Sets color and background styles to all p elements.

```
$("p").css({ color: "red", background: "blue" });
```

HTML:

```
<p>Test Paragraph.</p>
```

Result:

```
<p style="color:red; background:blue;">Test Paragraph.</p>
```

css(String,String|Number)

`css(String key, String|Number value) returns jQuery`

Set a single style property to a value, on all matched elements. If a number is provided, it is automatically converted into a pixel value.

Example:

Changes the color of all paragraphs to red

```
$("p").css("color","red");
```

HTML:

```
<p>Test Paragraph.</p>
```

Result:

```
<p style="color:red;">Test Paragraph.</p>
```

Example:

Changes the left of all paragraphs to "30px"

```
$("p").css("left",30);
```

HTML:

```
<p>Test Paragraph.</p>
```

Result:

```
<p style="left:30px;">Test Paragraph.</p>
```

width()

`width() returns String`

Get the current computed, pixel, width of the first matched element.

Example:

```
$("p").width();
```

HTML:

```
<p>This is just a test.</p>
```

Result:

```
300
```

width(String|Number)

`width(String|Number val) returns jQuery`

Set the CSS width of every matched element. If no explicit unit was specified (like 'em' or '%') then "px" is added to the width.

Example:

```
$("p").width(20);
```

HTML:

```
<p>This is just a test.</p>
```

Result:

```
<p style="width:20px;">This is just a test.</p>
```

Example:

```
$("p").width("20em");
```

HTML:

```
<p>This is just a test.</p>
```

Result:

```
<p style="width:20em;">This is just a test.</p>
```

height()

`height() returns String`

Get the current computed, pixel, height of the first matched element.

Example:

```
$("p").height();
```

HTML:

```
<p>This is just a test.</p>
```

Result:

```
300
```

height(String|Number)

`height(String|Number val) returns jQuery`

Set the CSS height of every matched element. If no explicit unit was specified (like 'em' or '%') then "px" is added to the width.

Example:

```
$("p").height(20);
```

HTML:

```
<p>This is just a test.</p>
```

Result:

```
<p style="height:20px;">This is just a test.</p>
```

Example:

```
$("p").height("20em");
```

HTML:

```
<p>This is just a test.</p>
```

Result:

```
<p style="height:20em;">This is just a test.</p>
```

4. JavaScript

$.extend(Object,Object,Object)

`$.extend(Object target, Object prop1, Object propN) returns Object`

Extend one object with one or more others, returning the original, modified, object. This is a great utility for simple inheritance.

Example:

Merge settings and options, modifying settings

```
var settings = { validate: false, limit: 5, name: "foo" };
var options = { validate: true, name: "bar" };
jQuery.extend(settings, options);
```

Result:

```
settings == { validate: true, limit: 5, name: "bar" }
```

Example:

Merge defaults and options, without modifying the defaults

```
var defaults = { validate: false, limit: 5, name: "foo" };
var options = { validate: true, name: "bar" };
var settings = jQuery.extend({}, defaults, options);
```

Result:

```
settings == { validate: true, limit: 5, name: "bar" }
```

$.each(Object,Function)

`$.each(Object obj, Function fn) returns Object`

A generic iterator function, which can be used to seamlessly iterate over both objects and arrays. This function is not the same as $().each() - which is used to iterate, exclusively, over a jQuery object. This function can be used to iterate over anything.
The callback has two arguments:the key (objects) or index (arrays) as first the first, and the value as the second.

Example:

This is an example of iterating over the items in an array, accessing both the current item and its index.

```
$.each( [0,1,2], function(i, n){
 alert( "Item #" + i + ": " + n );
});
```

Example:

This is an example of iterating over the properties in an Object, accessing both the current item and its key.

```
$.each( { name: "John", lang: "JS" }, function(i, n){
 alert( "Name: " + i + ", Value: " + n );
});
```

$.trim(String)

`$.trim(String str) returns String`

Remove the whitespace from the beginning and end of a string.

Example:

```
$.trim("  hello, how are you?  ");
```

Result:

```
"hello, how are you?"
```

$.merge(Array,Array)

`$.merge(Array first, Array second) returns Array`

Merge two arrays together by concatenating them.

Example:

Merges two arrays.

```
$.merge( [0,1,2], [2,3,4] )
```

Result:

```
[0,1,2,2,3,4]
```

$.unique(Array)

`$.unique(Array array) returns Array`

Reduce an array (of jQuery objects only) to its unique elements.

Example:

Reduces the arrays of jQuery objects to unique elements by removing the duplicates of x2 and x3

```
$.unique( [x1, x2, x3, x2, x3] )
```

Result:

```
[x1, x2, x3]
```

$.grep(Array,Function,Boolean)

`$.grep(Array array, Function fn, Boolean inv) returns Array`

Filter items out of an array, by using a filter function.
The specified function will be passed two arguments: The current array item and the index of the item in the array. The function must return 'true' to keep the item in the array, false to remove it.

Example:

```
$.grep( [0,1,2], function(i){
 return i > 0;
});
```

Result:

```
[1, 2]
```

$.map(Array,Function)

`$.map(Array array, Function fn) returns Array`

Translate all items in an array to another array of items.
The translation function that is provided to this method is called for each item in the array and is passed one argument: The item to be translated.
The function can then return the translated value, 'null' (to remove the item), or an array of values - which will be flattened into the full array.

Example:

Maps the original array to a new one and adds 4 to each value.

```
$.map( [0,1,2], function(i){
 return i + 4;
});
```

Result:

```
[4, 5, 6]
```

Example:

Maps the original array to a new one and adds 1 to each value if it is bigger then zero, otherwise it's removed-

```
$.map( [0,1,2], function(i){
 return i > 0 ? i + 1 : null;
});
```

Result:

```
[2, 3]
```

Example:

Maps the original array to a new one, each element is added with it's original value and the value plus one.

```
$.map( [0,1,2], function(i){
 return [ i, i + 1 ];
});
```

Result:

```
[0, 1, 1, 2, 2, 3]
```

$.browser()

`$.browser() returns Boolean`

Contains flags for the useragent, read from navigator.userAgent. Available flags are: safari, opera, msie, mozilla
This property is available before the DOM is ready, therefore you can use it to add ready events only for certain browsers.
There are situations where object detections is not reliable enough, in that cases it makes sense to use browser detection. Simply try to avoid both!
A combination of browser and object detection yields quite reliable results.

Example:

Returns true if the current useragent is some version of microsoft's internet explorer

```
$.browser.msie
```

Example:

Alerts "this is safari!" only for safari browsers

```
if($.browser.safari) { $( function() { alert("this is safari!"); } ); }
```

5. Events

bind(String,Object,Function)

`bind(String type, Object data, Function fn) returns jQuery`

Binds a handler to a particular event (like click) for each matched element. The event handler is passed an event object that you can use to prevent default behaviour. To stop both default action and event bubbling, your handler has to return false.
In most cases, you can define your event handlers as anonymous functions (see first example). In cases where that is not possible, you can pass additional data as the second parameter (and the handler function as the third), see second example.
Calling bind with an event type of "unload" will automatically use the one method instead of bind to prevent memory leaks.

Example:

```
$("p").bind("click", function(){
 alert( $(this).text() );
});
```

HTML:

```
<p>Hello</p>
```

Result:

```
alert("Hello")
```

Example:

Pass some additional data to the event handler.

```
function handler(event) {
 alert(event.data.foo);
}
$("p").bind("click", {foo: "bar"}, handler)
```

Result:

```
alert("bar")
```

Example:

Cancel a default action and prevent it from bubbling by returning false from your function.

```
$("form").bind("submit", function() { return false; })
```

Example:

Cancel only the default action by using the preventDefault method.

```
$("form").bind("submit", function(event){
 event.preventDefault();
});
```

Example:

Stop only an event from bubbling by using the stopPropagation method.

```
$("form").bind("submit", function(event){
 event.stopPropagation();
});
```

one(String,Object,Function)

`one(String type, Object data, Function fn) returns jQuery`

Binds a handler to a particular event (like click) for each matched element. The handler is executed only once for each element. Otherwise, the same rules as described in bind() apply. The event handler is passed an event object that you can use to prevent default behaviour. To stop both default action and event bubbling, your handler has to return false.
In most cases, you can define your event handlers as anonymous functions (see first example). In cases where that is not possible, you can pass additional data as the second paramter (and the handler function as the third), see second example.

Example:

```
$("p").one("click", function(){
 alert( $(this).text() );
});
```

HTML:

```
<p>Hello</p>
```

Result:

```
alert("Hello")
```

unbind(String,Function)

```
unbind(String type, Function fn) returns jQuery
```

The opposite of bind, removes a bound event from each of the matched elements.
Without any arguments, all bound events are removed.
If the type is provided, all bound events of that type are removed.
If the function that was passed to bind is provided as the second argument, only that specific event handler is removed.

Example:

```
$("p").unbind()
```

HTML:

```
<p onclick="alert('Hello');">Hello</p>
```

Result:

```
[ <p>Hello</p> ]
```

Example:

```
$("p").unbind( "click" )
```

HTML:

```
<p onclick="alert('Hello');">Hello</p>
```

Result:

```
[ <p>Hello</p> ]
```

Example:

```
$("p").unbind( "click", function() { alert("Hello"); } )
```

HTML:

```
<p onclick="alert('Hello');">Hello</p>
```

Result:

```
[ <p>Hello</p> ]
```

trigger(String,Array)

`trigger(String type, Array data) returns jQuery`

Trigger a type of event on every matched element. This will also cause the default action of the browser with the same name (if one exists) to be executed. For example, passing 'submit' to the trigger() function will also cause the browser to submit the form. This default action can be prevented by returning false from one of the functions bound to the event.
You can also trigger custom events registered with bind.

Example:

```
$("p").trigger("click")
```

HTML:

```
<p click="alert('hello')">Hello</p>
```

Result:

```
alert('hello')
```

Example:

Example of how to pass arbitrary data to an event

```
$("p").click(function(event, a, b) {
 // when a normal click fires, a and b are undefined
 // for a trigger like below a refers too "foo" and b refers to "bar"
}).trigger("click", ["foo", "bar"]);
```

Example:

```
$("p").bind("myEvent",function(event,message1,message2) {
 alert(message1 + ' ' + message2);
});
$("p").trigger("myEvent",["Hello","World"]);
```

Result:

```
alert('Hello World') // One for each paragraph
```

toggle(Function,Function)

`toggle(Function even, Function odd) returns jQuery`

Toggle between two function calls every other click. Whenever a matched element is clicked, the first specified function is fired, when clicked again, the second is fired. All subsequent clicks continue to rotate through the two functions.
Use unbind("click") to remove.

Example:

```
$("p").toggle(function(){
 $(this).addClass("selected");
},function(){
 $(this).removeClass("selected");
});
```

hover(Function,Function)

`hover(Function over, Function out) returns jQuery`

A method for simulating hovering (moving the mouse on, and off, an object). This is a custom method which provides an 'in' to a frequent task.
Whenever the mouse cursor is moved over a matched element, the first specified function is fired. Whenever the mouse moves off of the element, the second specified function fires. Additionally, checks are in place to see if the mouse is still within the specified element itself (for example, an image inside of a div), and if it is, it will continue to 'hover', and not move out (a common error in using a mouseout event handler).

Example:

```
$("p").hover(function(){
 $(this).addClass("hover");
},function(){
 $(this).removeClass("hover");
});
```

ready(Function)

`ready(Function fn) returns jQuery`

Bind a function to be executed whenever the DOM is ready to be traversed and manipulated. This is probably the most important function included in the event module, as it can greatly improve the response times of your web applications.
In a nutshell, this is a solid replacement for using window.onload, and attaching a function to that. By using this method, your bound function will be called the instant the DOM is ready to be read and manipulated, which is when what 99.99% of all JavaScript code needs to run.
There is one argument passed to the ready event handler: A reference to the jQuery function. You can name that argument whatever you like, and can therefore stick with the $ alias without risk of naming collisions.
Please ensure you have no code in your <body> onload event handler, otherwise $(document).ready() may not fire.
You can have as many $(document).ready events on your page as you like. The functions are then executed in the order they were added.

Example:

```
$(document).ready(function(){ Your code here... });
```

Example:

Uses both the [[Core#.24.28_fn_.29|shortcut]] for $(document).ready() and the argument to write failsafe jQuery code using the $ alias, without relying on the global alias.

```
jQuery(function($) {
 // Your code using failsafe $ alias here...
});
```

scroll(Function)

```
scroll(Function fn) returns jQuery
```

Bind a function to the scroll event of each matched element.

Example:

```
$("p").scroll( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onscroll="alert('Hello');">Hello</p>
```

submit(Function)

`submit(Function fn) returns jQuery`

Bind a function to the submit event of each matched element.

Example:

Prevents the form submission when the input has no value entered.

```
$("#myform").submit( function() {
 return $("input", this).val().length > 0;
} );
```

HTML:

```
<form id="myform"><input /></form>
```

submit()

`submit() returns jQuery`

Trigger the submit event of each matched element. This causes all of the functions that have been bound to that submit event to be executed, and calls the browser's default submit action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the submit event.
Note: This does not execute the submit method of the form element! If you need to submit the form via code, you have to use the DOM method, eg. $("form")[0].submit();

Example:

Triggers all submit events registered to the matched form(s), and submits them.

```
$("form").submit();
```

focus(Function)

`focus(Function fn) returns jQuery`

Bind a function to the focus event of each matched element.

Example:

```
$("p").focus( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onfocus="alert('Hello');">Hello</p>
```

focus()

`focus() returns jQuery`

Trigger the focus event of each matched element. This causes all of the functions that have been bound to thet focus event to be executed.
Note: This does not execute the focus method of the underlying elements! If you need to focus an element via code, you have to use the DOM method, eg. $("#myinput")[0].focus();

Example:

```
$("p").focus();
```

HTML:

```
<p onfocus="alert('Hello');">Hello</p>
```

Result:

```
alert('Hello');
```

keydown(Function)

`keydown(Function fn) returns jQuery`

Bind a function to the keydown event of each matched element.

Example:

```
$("p").keydown( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onkeydown="alert('Hello');">Hello</p>
```

dblclick(Function)

`dblclick(Function fn) returns jQuery`

Bind a function to the dblclick event of each matched element.

Example:

```
$("p").dblclick( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p ondblclick="alert('Hello');">Hello</p>
```

keypress(Function)

`keypress(Function fn) returns jQuery`

Bind a function to the keypress event of each matched element.

Example:

```
$("p").keypress( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onkeypress="alert('Hello');">Hello</p>
```

error(Function)

`error(Function fn) returns jQuery`

Bind a function to the error event of each matched element.

Example:

```
$("p").error( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onerror="alert('Hello');">Hello</p>
```

blur(Function)

`blur(Function fn) returns jQuery`

Bind a function to the blur event of each matched element.

Example:

```
$("p").blur( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onblur="alert('Hello');">Hello</p>
```

blur()

`blur() returns jQuery`

Trigger the blur event of each matched element. This causes all of the functions that have been bound to that blur event to be executed, and calls the browser's default blur action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the blur event.
Note: This does not execute the blur method of the underlying elements! If you need to blur an element via code, you have to use the DOM method, eg. $("#myinput")[0].blur();

Example:

```
$("p").blur();
```

HTML:

```
<p onblur="alert('Hello');">Hello</p>
```

Result:

```
alert('Hello');
```

load(Function)

`load(Function fn) returns jQuery`

Bind a function to the load event of each matched element.

Example:

```
$("p").load( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onload="alert('Hello');">Hello</p>
```

select(Function)

`select(Function fn) returns jQuery`

Bind a function to the select event of each matched element.

Example:

```
$("p").select( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onselect="alert('Hello');">Hello</p>
```

select()

`select() returns jQuery`

Trigger the select event of each matched element. This causes all of the functions that have been bound to that select event to be executed, and calls the browser's default select action on the matching element(s). This default action can be prevented by returning false from one of the functions bound to the select event.

Example:

```
$("p").select();
```

HTML:

```
<p onselect="alert('Hello');">Hello</p>
```

Result:

```
alert('Hello');
```

mouseup(Function)

`mouseup(Function fn) returns jQuery`

Bind a function to the mouseup event of each matched element.

Example:

```
$("p").mouseup( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onmouseup="alert('Hello');">Hello</p>
```

unload(Function)

`unload(Function fn) returns jQuery`

Bind a function to the unload event of each matched element.

Example:

```
$("p").unload( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onunload="alert('Hello');">Hello</p>
```

change(Function)

`change(Function fn) returns jQuery`

Bind a function to the change event of each matched element.

Example:

```
$("p").change( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onchange="alert('Hello');">Hello</p>
```

mouseout(Function)

`mouseout(Function fn) returns jQuery`

Bind a function to the mouseout event of each matched element.

Example:

```
$("p").mouseout( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onmouseout="alert('Hello');">Hello</p>
```

keyup(Function)

`keyup(Function fn) returns jQuery`

Bind a function to the keyup event of each matched element.

Example:

```
$("p").keyup( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onkeyup="alert('Hello');">Hello</p>
```

click(Function)

`click(Function fn) returns jQuery`

Bind a function to the click event of each matched element.

Example:

```
$("p").click( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onclick="alert('Hello');">Hello</p>
```

click()

`click() returns jQuery`

Trigger the click event of each matched element. This causes all of the functions that have been bound to thet click event to be executed.

Example:

```
$("p").click();
```

HTML:

```
<p onclick="alert('Hello');">Hello</p>
```

Result:

```
alert('Hello');
```

resize(Function)

`resize(Function fn) returns jQuery`

Bind a function to the resize event of each matched element.

Example:

```
$("p").resize( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onresize="alert('Hello');">Hello</p>
```

mousemove(Function)

`mousemove(Function fn) returns jQuery`

Bind a function to the mousemove event of each matched element.

Example:

```
$("p").mousemove( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onmousemove="alert('Hello');">Hello</p>
```

mousedown(Function)

`mousedown(Function fn) returns jQuery`

Bind a function to the mousedown event of each matched element.

Example:

```
$("p").mousedown( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onmousedown="alert('Hello');">Hello</p>
```

mouseover(Function)

`mouseover(Function fn) returns jQuery`

Bind a function to the mouseover event of each matched element.

Example:

```
$("p").mouseover( function() { alert("Hello"); } );
```

HTML:

```
<p>Hello</p>
```

Result:

```
<p onmouseover="alert('Hello');">Hello</p>
```

6. Effects

show()

`show() returns jQuery`

Displays each of the set of matched elements if they are hidden.

Example:

```
$("p").show()
```

HTML:

```
<p style="display: none">Hello</p>
```

Result:

```
[ <p style="display: block">Hello</p> ]
```

show(String|Number,Function)

`show(String|Number speed, Function callback) returns jQuery`

Show all matched elements using a graceful animation and firing an optional callback after completion.
The height, width, and opacity of each of the matched elements are changed dynamically according to the specified speed.

Example:

```
$("p").show("slow");
```

Example:

```
$("p").show("slow",function(){
 alert("Animation Done.");
});
```

hide()

`hide() returns jQuery`

Hides each of the set of matched elements if they are shown.

Example:

```
$("p").hide()
```

HTML:

```
<p>Hello</p>
```

Result:

```
[ <p style="display: none">Hello</p> ]
```

hide(String|Number,Function)

`hide(String|Number speed, Function callback) returns jQuery`

Hide all matched elements using a graceful animation and firing an optional callback after completion.
The height, width, and opacity of each of the matched elements are changed dynamically according to the specified speed.

Example:

```
$("p").hide("slow");
```

Example:

```
$("p").hide("slow",function(){
 alert("Animation Done.");
});
```

toggle()

`toggle() returns jQuery`

Toggles each of the set of matched elements. If they are shown, toggle makes them hidden. If they are hidden, toggle makes them shown.

Example:

```
$("p").toggle()
```

HTML:

```
<p>Hello</p><p style="display: none">Hello Again</p>
```

Result:

```
[ <p style="display: none">Hello</p>, <p style="display: block">Hello Again</p> ]
```

slideDown(String|Number,Function)

`slideDown(String|Number speed, Function callback) returns jQuery`

Reveal all matched elements by adjusting their height and firing an optional callback after completion.
Only the height is adjusted for this animation, causing all matched elements to be revealed in a "sliding" manner.

Example:

```
$("p").slideDown("slow");
```

Example:

```
$("p").slideDown("slow",function(){
 alert("Animation Done.");
});
```

slideUp(String|Number,Function)

`slideUp(String|Number speed, Function callback) returns jQuery`

Hide all matched elements by adjusting their height and firing an optional callback after completion.
Only the height is adjusted for this animation, causing all matched elements to be hidden in a "sliding" manner.

Example:

```
$("p").slideUp("slow");
```

Example:

```
$("p").slideUp("slow",function(){
 alert("Animation Done.");
});
```

slideToggle(String|Number,Function)

`slideToggle(String|Number speed, Function callback) returns jQuery`

Toggle the visibility of all matched elements by adjusting their height and firing an optional callback after completion.
Only the height is adjusted for this animation, causing all matched elements to be hidden in a "sliding" manner.

Example:

```
$("p").slideToggle("slow");
```

Example:

```
$("p").slideToggle("slow",function(){
 alert("Animation Done.");
});
```

fadeIn(String|Number,Function)

`fadeIn(String|Number speed, Function callback) returns jQuery`

Fade in all matched elements by adjusting their opacity and firing an optional callback after completion.
Only the opacity is adjusted for this animation, meaning that all of the matched elements should already have some form of height and width associated with them.

Example:

```
$("p").fadeIn("slow");
```

Example:

```
$("p").fadeIn("slow",function(){
 alert("Animation Done.");
});
```

fadeOut(String|Number,Function)

`fadeOut(String|Number speed, Function callback) returns jQuery`

Fade out all matched elements by adjusting their opacity and firing an optional callback after completion.
Only the opacity is adjusted for this animation, meaning that all of the matched elements should already have some form of height and width associated with them.

Example:

```
$("p").fadeOut("slow");
```

Example:

```
$("p").fadeOut("slow",function(){
 alert("Animation Done.");
});
```

fadeTo(String|Number,Number,Function)

`fadeTo(String|Number speed, Number opacity, Function callback) returns jQuery`

Fade the opacity of all matched elements to a specified opacity and firing an optional callback after completion.
Only the opacity is adjusted for this animation, meaning that all of the matched elements should already have some form of height and width associated with them.

Example:

```
$("p").fadeTo("slow", 0.5);
```

Example:

```
$("p").fadeTo("slow", 0.5, function(){
 alert("Animation Done.");
});
```

animate(Hash,String|Number,String,Function)

`animate(Hash params, String|Number speed, String easing, Function callback) returns jQuery`

A function for making your own, custom animations. The key aspect of this function is the object of style properties that will be animated, and to what end. Each key within the object represents a style property that will also be animated (for example: "height", "top", or "opacity").
Note that properties should be specified using camel case eg. marginLeft instead of margin-left.
The value associated with the key represents to what end the property will be animated. If a number is provided as the value, then the style property will be transitioned from its current state to that new number. Otherwise if the string "hide", "show", or "toggle" is provided, a default animation will be constructed for that property.

Example:

```
$("p").animate({
 height: 'toggle', opacity: 'toggle'
}, "slow");
```

Example:

```
$("p").animate({
 left: 50, opacity: 'show'
}, 500);
```

Example:

An example of using an 'easing' function to provide a different style of animation. This will only work if you have a plugin that provides this easing function (Only "swing" and "linear" are provided by default, with jQuery).

```
$("p").animate({
 opacity: 'show'
}, "slow", "easein");
```

7. Ajax

loadIfModified(String,Map,Function)

```
loadIfModified(String url, Map params, Function callback)
returns jQuery
```

Load HTML from a remote file and inject it into the DOM, only if it's been modified by the server.

Example:

```
$("#feeds").loadIfModified("feeds.html");
```

HTML:

```
<div id="feeds"></div>
```

Result:

```
<div id="feeds"><b>45</b> feeds found.</div>
```

load(String,Object,Function)

`load(String url, Object params, Function callback) returns jQuery`

Load HTML from a remote file and inject it into the DOM.
Note: Avoid to use this to load scripts, instead use $.getScript. IE strips script tags when there aren't any other characters in front of it.

Example:

```
$("#feeds").load("feeds.html");
```

HTML:

```
<div id="feeds"></div>
```

Result:

```
<div id="feeds"><b>45</b> feeds found.</div>
```

Example:

Same as above, but with an additional parameter and a callback that is executed when the data was loaded.

```
$("#feeds").load("feeds.html",
 {limit: 25},
 function() { alert("The last 25 entries in the feed have been loaded"); }
);
```

serialize()

`serialize() returns String`

Serializes a set of input elements into a string of data. This will serialize all given elements.
A serialization similar to the form submit of a browser is provided by the [http://www.malsup.com/jquery/form/ Form Plugin]. It also takes multiple-selects into account, while this method recognizes only a single option.

Example:

Serialize a selection of input elements to a string

```
$("input[@type=text]").serialize();
```

HTML:

```
<input type='text' name='name' value='John'/>
<input type='text' name='location' value='Boston'/>
```

ajaxStart(Function)

`ajaxStart(Function callback) returns jQuery`

Attach a function to be executed whenever an AJAX request begins and there is none already active.

Example:

Show a loading message whenever an AJAX request starts (and none is already active).

```
$("#loading").ajaxStart(function(){
 $(this).show();
});
```

ajaxStop(Function)

`ajaxStop(Function callback) returns jQuery`

Attach a function to be executed whenever all AJAX requests have ended.

Example:

Hide a loading message after all the AJAX requests have stopped.

```
$("#loading").ajaxStop(function(){
 $(this).hide();
});
```

ajaxComplete(Function)

`ajaxComplete(Function callback) returns jQuery`

Attach a function to be executed whenever an AJAX request completes.
The XMLHttpRequest and settings used for that request are passed as arguments to the callback.

Example:

Show a message when an AJAX request completes.

```
$("#msg").ajaxComplete(function(request, settings){
 $(this).append("<li>Request Complete.</li>");
});
```

ajaxSuccess(Function)

`ajaxSuccess(Function callback) returns jQuery`

Attach a function to be executed whenever an AJAX request completes successfully.
The XMLHttpRequest and settings used for that request are passed as arguments to the callback.

Example:

Show a message when an AJAX request completes successfully.

```
$("#msg").ajaxSuccess(function(request, settings){
 $(this).append("<li>Successful Request!</li>");
});
```

ajaxError(Function)

`ajaxError(Function callback) returns jQuery`

Attach a function to be executed whenever an AJAX request fails.
The XMLHttpRequest and settings used for that request are passed as arguments to the callback.
A third argument, an exception object, is passed if an exception occured while processing the request.

Example:

Show a message when an AJAX request fails.

```
$("#msg").ajaxError(function(request, settings){
 $(this).append("<li>Error requesting page " + settings.url + "</li>");
});
```

ajaxSend(Function)

`ajaxSend(Function callback) returns jQuery`

Attach a function to be executed before an AJAX request is sent.
The XMLHttpRequest and settings used for that request are passed as arguments to the callback.

Example:

Show a message before an AJAX request is sent.

```
$("#msg").ajaxSend(function(request, settings){
 $(this).append("<li>Starting request at " + settings.url + "</li>");
});
```

$.get(String,Map,Function)

`$.get(String url, Map params, Function callback) returns XMLHttpRequest`

Load a remote page using an HTTP GET request.
This is an easy way to send a simple GET request to a server without having to use the more complex $.ajax function. It allows a single callback function to be specified that will be executed when the request is complete (and only if the response has a successful response code). If you need to have both error and success callbacks, you may want to use $.ajax.

Example:

```
$.get("test.cgi");
```

Example:

```
$.get("test.cgi", { name: "John", time: "2pm" } );
```

Example:

```
$.get("test.cgi", function(data){
 alert("Data Loaded: " + data);
});
```

Example:

```
$.get("test.cgi",
 { name: "John", time: "2pm" },
 function(data){
   alert("Data Loaded: " + data);
 }
);
```

$.getIfModified(String,Map,Function)

`$.getIfModified(String url, Map params, Function callback) returns XMLHttpRequest`

Load a remote page using an HTTP GET request, only if it hasn't been modified since it was last retrieved.

Example:

```
$.getIfModified("test.html");
```

Example:

```
$.getIfModified("test.html", { name: "John", time: "2pm" } );
```

Example:

```
$.getIfModified("test.cgi", function(data){
 alert("Data Loaded: " + data);
});
```

Example:

```
$.getifModified("test.cgi",
 { name: "John", time: "2pm" },
 function(data){
   alert("Data Loaded: " + data);
 }
);
```

$.getScript(String,Function)

`$.getScript(String url, Function callback) returns XMLHttpRequest`

Loads, and executes, a remote JavaScript file using an HTTP GET request.
Warning: Safari <= 2.0.x is unable to evaluate scripts in a global context synchronously. If you load functions via getScript, make sure to call them after a delay.

Example:

```
$.getScript("test.js");
```

Example:

```
$.getScript("test.js", function(){
 alert("Script loaded and executed.");
});
```

$.getJSON(String,Map,Function)

`$.getJSON(String url, Map params, Function callback) returns XMLHttpRequest`

Load JSON data using an HTTP GET request.

Example:

```
 $.getJSON("test.js", function(json){
  alert("JSON Data: " + json.users[3].name);
 });
```

Example:

```
 $.getJSON("test.js",
  { name: "John", time: "2pm" },
  function(json){
    alert("JSON Data: " + json.users[3].name);
  }
 );
```

$.post(String,Map,Function)

`$.post(String url, Map params, Function callback) returns XMLHttpRequest`

Load a remote page using an HTTP POST request.

Example:

```
$.post("test.cgi");
```

Example:

```
$.post("test.cgi", { name: "John", time: "2pm" } );
```

Example:

```
$.post("test.cgi", function(data){
 alert("Data Loaded: " + data);
});
```

Example:

```
$.post("test.cgi",
 { name: "John", time: "2pm" },
 function(data){
   alert("Data Loaded: " + data);
 }
);
```

$.ajaxTimeout(Number)

`$.ajaxTimeout(Number time) returns undefined`

Set the timeout in milliseconds of all AJAX requests to a specific amount of time. This will make all future AJAX requests timeout after a specified amount of time.
Set to null or 0 to disable timeouts (default).
You can manually abort requests with the XMLHttpRequest's (returned by all ajax functions) abort() method.
Deprecated. Use $.ajaxSetup instead.

Example:

Make all AJAX requests timeout after 5 seconds.

```
 $.ajaxTimeout( 5000 );
```

$.ajaxSetup(Map)

`$.ajaxSetup(Map settings) returns undefined`

Setup global settings for AJAX requests.
See $.ajax for a description of all available options.

Example:

Sets the defaults for AJAX requests to the url "/xmlhttp/", disables global handlers and uses POST instead of GET. The following AJAX requests then sends some data without having to set anything else.

```
$.ajaxSetup( {
 url: "/xmlhttp/",
 global: false,
 type: "POST"
} );
$.ajax({ data: myData });
```

$.ajax(Map)

`$.ajax(Map properties) returns XMLHttpRequest`

Load a remote page using an HTTP request.
This is jQuery's low-level AJAX implementation. See $.get, $.post etc. for higher-level abstractions that are often easier to understand and use, but don't offer as much functionality (such as error callbacks).
$.ajax() returns the XMLHttpRequest that it creates. In most cases you won't need that object to manipulate directly, but it is available if you need to abort the request manually.
'''Note:''' If you specify the dataType option described below, make sure the server sends the correct MIME type in the response (eg. xml as "text/xml"). Sending the wrong MIME type can lead to unexpected problems in your script. See [[Specifying the Data Type for AJAX Requests]] for more information.
Supported datatypes are (see dataType option):
"xml": Returns a XML document that can be processed via jQuery.
"html": Returns HTML as plain text, included script tags are evaluated.
"script": Evaluates the response as Javascript and returns it as plain text.
"json": Evaluates the response as JSON and returns a Javascript Object
$.ajax() takes one argument, an object of key/value pairs, that are used to initalize and handle the request. These are all the key/values that can be used:
(String) url - The URL to request.
(String) type - The type of request to make ("POST" or "GET"), default is "GET".
(String) dataType - The type of data that you're expecting back from the server. No default: If the server sends xml, the responseXML, otherwise the responseText is passed to the success callback.
(Boolean) ifModified - Allow the request to be successful only if the response has changed since the last request. This is done by checking the Last-Modified header. Default value is false, ignoring the header.
(Number) timeout - Local timeout in milliseconds to override global timeout, eg. to give a single request a longer timeout while all others timeout after 1 second. See $.ajaxTimeout() for global timeouts.
(Boolean) global - Whether to trigger global AJAX event handlers for this request, default is true. Set to false to prevent that global handlers like ajaxStart or ajaxStop are triggered.
(Function) error - A function to be called if the request fails. The function gets passed tree arguments: The XMLHttpRequest object, a string describing the type of error that occurred and an optional exception object, if one occured.
(Function) success - A function to be called if the request succeeds. The function gets passed one argument: The data returned from the server, formatted according to the 'dataType' parameter.
(Function) complete - A function to be called when the request finishes. The function gets passed two arguments: The XMLHttpRequest object and a string describing the type of success of the request.
(Object|String) data - Data to be sent to the server. Converted to a query string, if not already a

string. Is appended to the url for GET-requests. See processData option to prevent this automatic processing.
(String) contentType - When sending data to the server, use this content-type. Default is "application/x-www-form-urlencoded", which is fine for most cases.
(Boolean) processData - By default, data passed in to the data option as an object other as string will be processed and transformed into a query string, fitting to the default content-type "application/x-www-form-urlencoded". If you want to send DOMDocuments, set this option to false.
(Boolean) async - By default, all requests are sent asynchronous (set to true). If you need synchronous requests, set this option to false.
(Function) beforeSend - A pre-callback to set custom headers etc., the XMLHttpRequest is passed as the only argument.

Example:

Load and execute a JavaScript file.

```
$.ajax({
 type: "GET",
 url: "test.js",
 dataType: "script"
})
```

Example:

Save some data to the server and notify the user once its complete.

```
$.ajax({
 type: "POST",
 url: "some.php",
 data: "name=John&location=Boston",
 success: function(msg){
   alert( "Data Saved: " + msg );
 }
});
```

Example:

Loads data synchronously. Blocks the browser while the requests is active. It is better to block user interaction by other means when synchronization is necessary.

```
var html = $.ajax({
url: "some.php",
async: false
}).responseText;
```

Example:

Sends an xml document as data to the server. By setting the processData option to false, the automatic conversion of data to strings is prevented.

```
var xmlDocument = [create xml document];
$.ajax({
 url: "page.php",
 processData: false,
 data: xmlDocument,
 success: handleResponse
});
```

8. Plugins

Accordion

Accordion(Map)

`Accordion(Map options) returns jQuery`

Make the selected elements Accordion widgets.
Semantic requirements:
If the structure of your container is flat with unique tags for header and content elements, eg. a definition list (dl > dt + dd), you don't have to specify any options at all.
If your structure uses the same elements for header and content or uses some kind of nested structure, you have to specify the header elements, eg. via class, see the second example.
Use activate(Number) to change the active content programmatically.
A change event is triggered everytime the accordion changes. Apart from the event object, all arguments are jQuery objects. Arguments: event, newHeader, oldHeader, newContent, oldContent

Example:

Creates an Accordion from the given definition list

```
jQuery('#nav').Accordion();
```

HTML:

```
<dl id="nav">
 <dt>Header 1</dt>
 <dd>Content 1</dd>
 <dt>Header 2</dt>
 <dd>Content 2</dd>
</dl>
```

Example:

Creates an Accordion from the given div structure

```
jQuery('#nav').Accordion({
 header: '.title'
});
```

HTML:

```
<div id="nav">
<div>
  <div class="title">Header 1</div>
  <div>Content 1</div>
</div>
<div>
  <div class="title">Header 2</div>
  <div>Content 2</div>
</div>
</div>
```

Example:

Creates an Accordion from the given navigation list, activating those accordion parts that match the current location.href. Assuming the user clicked on "Fantasy" in the "Movies" section, the accordion displayed after loading the page with the "Movies" section open and the "Fantasy" link highlighted with a class "selected".

```
jQuery('#nav').Accordion({
 header: '.head',
  navigation: true
});
```

HTML:

```
<ul id="nav">
 <li>
   <a class="head" href="books/">Books</a>
   <ul>
     <li><a href="books/fantasy/">Fantasy</a></li>
     <li><a href="books/programming/">Programming</a></li>
   </ul>
 </li>
 <li>
   <a class="head" href="movies/">Movies</a>
   <ul>
     <li><a href="movies/fantasy/">Fantasy</a></li>
     <li><a href="movies/programming/">Programming</a></li>
   </ul>
 </li>
</ul>
```

Example:

Updates the element with id status with the text of the selected header every time the accordion changes

```
jQuery('#accordion').Accordion().change(function(event, newHeader, oldHeader,
newContent, oldContent) {
 jQuery('#status').html(newHeader.text());
});
```

activate(String|Element|jQuery|Boolean|Number)

`activate(String|Element|jQuery|Boolean|Number index) returns jQuery`

Activate a content part of the Accordion programmatically.
The index can be a zero-indexed number to match the position of the header to close or a string expression matching an element. Pass -1 to close all (only possible with alwaysOpen:false).

Example:

Activate the second content of the Accordion contained in <div id="accordion">.

```
jQuery('#accordion').activate(1);
```

Example:

Activate the first element matching the given expression.

```
jQuery('#accordion').activate("a:first");
```

Example:

Close all content parts of the accordion.

```
jQuery('#nav').activate(false);
```

Button

button(hOptions)

`button(hOptions hash) returns jQuery`

Creates a button from an image element.
This function attempts to mimic the functionality of the "button" found in modern day GUIs. There are two different buttons you can create using this plugin; Normal buttons, and Toggle buttons.

Center

center()

`center() returns jQuery`

Takes all matched elements and centers them, absolutely, within the context of their parent element. Great for doing slideshows.

Example:

```
$("div img").center();
```

Cookie

$.cookie(String,String,Object)

`$.cookie(String name, String value, Object options) returns undefined`

Create a cookie with the given name and value and other optional parameters.

Example:

Set the value of a cookie.

```
$.cookie('the_cookie', 'the_value');
```

Example:

Create a cookie with all available options.

```
$.cookie('the_cookie', 'the_value', {expires: 7, path: '/', domain: 'jquery.com',
secure: true});
```

Example:

Create a session cookie.

```
$.cookie('the_cookie', 'the_value');
```

Example:

Delete a cookie by passing null as value.

```
$.cookie('the_cookie', null);
```

$.cookie(String)

`$.cookie(String name) returns String`

Get the value of a cookie with the given name.

Example:

Get the value of a cookie.

```
 $.cookie('the_cookie');
```

Form

ajaxSubmit(options)

`ajaxSubmit(options object) returns jQuery`

ajaxSubmit() provides a mechanism for submitting an HTML form using AJAX.
ajaxSubmit accepts a single argument which can be either a success callback function or an options Object. If a function is provided it will be invoked upon successful completion of the submit and will be passed the response from the server. If an options Object is provided, the following attributes are supported:

target: Identifies the element(s) in the page to be updated with the server response.
  This value may be specified as a jQuery selection string, a jQuery object,
  or a DOM element.
  default value: null

url: URL to which the form data will be submitted.
  default value: value of form's 'action' attribute

type: The method in which the form data should be submitted, 'GET' or 'POST'.
  default value: value of form's 'method' attribute (or 'GET' if none found)

beforeSubmit: Callback method to be invoked before the form is submitted.
  default value: null

success: Callback method to be invoked after the form has been successfully submitted
  and the response has been returned from the server
  default value: null

dataType: Expected dataType of the response. One of: null, 'xml', 'script', or 'json'
  default value: null

semantic: Boolean flag indicating whether data must be submitted in semantic order (slower).
  default value: false

resetForm: Boolean flag indicating whether the form should be reset if the submit is successful

clearForm: Boolean flag indicating whether the form should be cleared if the submit is successful

The 'beforeSubmit' callback can be provided as a hook for running pre-submit logic or for validating the form data. If the 'beforeSubmit' callback returns false then the form will not be submitted. The 'beforeSubmit' callback is invoked with three arguments: the form data in array format, the jQuery object, and the options object passed into ajaxSubmit. The form data array takes the following form:

[{ name: 'username', value: 'jresig' }, { name: 'password', value: 'secret' }]

If a 'success' callback method is provided it is invoked after the response has been returned from the server. It is passed the responseText or responseXML value (depending on dataType). See jQuery.ajax for further details.

The dataType option provides a means for specifying how the server response should be handled. This maps directly to the jQuery.httpData method. The following values are supported:

'xml': if dataType == 'xml' the server response is treated as XML and the 'success' callback method, if specified, will be passed the responseXML value

'json': if dataType == 'json' the server response will be evaluted and passed to the 'success' callback, if specified

'script': if dataType == 'script' the server response is evaluated in the global context

Note that it does not make sense to use both the 'target' and 'dataType' options. If both are provided the target will be ignored.

The semantic argument can be used to force form serialization in semantic order. This is normally true anyway, unless the form contains input elements of type='image'. If your form must be submitted with name/value pairs in semantic order and your form contains an input of type='image" then pass true for this arg, otherwise pass false (or nothing) to avoid the overhead for this logic.

When used on its own, ajaxSubmit() is typically bound to a form's submit event like this:

$("#form-id").submit(function() {
  $(this).ajaxSubmit(options);
  return false; // cancel conventional submit });

When using ajaxForm(), however, this is done for you.

Example:

Submit form and alert server response

```
$('#myForm').ajaxSubmit(function(data) {
   alert('Form submit succeeded! Server returned: ' + data);
});
```

Example:

Submit form and update page element with server response

```
var options = {
   target: '#myTargetDiv'
};
$('#myForm').ajaxSubmit(options);
```

Example:

Submit form and alert the server response

```
var options = {
   success: function(responseText) {
       alert(responseText);
   }
};
$('#myForm').ajaxSubmit(options);
```

Example:

Pre-submit validation which aborts the submit operation if form data is empty

```
var options = {
   beforeSubmit: function(formArray, jqForm) {
       if (formArray.length == 0) {
           alert('Please enter data.');
           return false;
       }
   }
};
$('#myForm').ajaxSubmit(options);
```

Example:

json data returned and evaluated

```
var options = {
   url: myJsonUrl.php,
   dataType: 'json',
   success: function(data) {
      // 'data' is an object representing the the evaluated json data
   }
};
$('#myForm').ajaxSubmit(options);
```

Example:

XML data returned from server

```
var options = {
   url: myXmlUrl.php,
   dataType: 'xml',
   success: function(responseXML) {
      // responseXML is XML document object
      var data = $('myElement', responseXML).text();
```

```
    }
};
$('#myForm').ajaxSubmit(options);
```

Example:

submit form and reset it if successful

```
var options = {
   resetForm: true
};
$('#myForm').ajaxSubmit(options);
```

Example:

Bind form's submit event to use ajaxSubmit

```
$('#myForm).submit(function() {
  $(this).ajaxSubmit();
  return false;
});
```

ajaxForm(options)

```
ajaxForm(options object) returns jQuery
```

ajaxForm() provides a mechanism for fully automating form submission.
The advantages of using this method instead of ajaxSubmit() are:
1: This method will include coordinates for <input type="image" /> elements (if the element
 is used to submit the form). 2. This method will include the submit element's name/value data (for the element that was
 used to submit the form). 3. This method binds the submit() method to the form for you.
Note that for accurate x/y coordinates of image submit elements in all browsers you need to also use the "dimensions" plugin (this method will auto-detect its presence).
The options argument for ajaxForm works exactly as it does for ajaxSubmit. ajaxForm merely passes the options argument along after properly binding events for submit elements and the form itself. See ajaxSubmit for a full description of the options argument.

Example:

Bind form's submit event so that 'myTargetDiv' is updated with the server response when the form is submitted.

```
var options = {
   target: '#myTargetDiv'
};
$('#myForm').ajaxSForm(options);
```

Example:

Bind form's submit event so that server response is alerted after the form is submitted.

```
var options = {
   success: function(responseText) {
       alert(responseText);
   }
};
$('#myForm').ajaxSubmit(options);
```

Example:

Bind form's submit event so that pre-submit callback is invoked before the form is submitted.

```
var options = {
   beforeSubmit: function(formArray, jqForm) {
       if (formArray.length == 0) {
           alert('Please enter data.');
           return false;
```

```
        }
    }
};
$('#myForm').ajaxSubmit(options);
```

ajaxFormUnbind()

`ajaxFormUnbind() returns jQuery`

ajaxFormUnbind unbinds the event handlers that were bound by ajaxForm

formToArray(semantic)

`formToArray(semantic true) returns Array<Object>`

formToArray() gathers form element data into an array of objects that can be passed to any of the following ajax functions: $.get, $.post, or load. Each object in the array has both a 'name' and 'value' property. An example of an array for a simple login form might be:
[{ name: 'username', value: 'jresig' }, { name: 'password', value: 'secret' }]
It is this array that is passed to pre-submit callback functions provided to the ajaxSubmit() and ajaxForm() methods.
The semantic argument can be used to force form serialization in semantic order. This is normally true anyway, unless the form contains input elements of type='image'. If your form must be submitted with name/value pairs in semantic order and your form contains an input of type='image" then pass true for this arg, otherwise pass false (or nothing) to avoid the overhead for this logic.

Example:

Collect all the data from a form and submit it to the server.

```
var data = $("#myForm").formToArray();
$.post( "myscript.cgi", data );
```

formSerialize(semantic)

`formSerialize(semantic true) returns String`

Serializes form data into a 'submittable' string. This method will return a string in the format: name1=value1&name2=value2
The semantic argument can be used to force form serialization in semantic order. If your form must be submitted with name/value pairs in semantic order then pass true for this arg, otherwise pass false (or nothing) to avoid the overhead for this logic (which can be significant for very large forms).

Example:

Collect all the data from a form into a single string

```
var data = $("#myForm").formSerialize();
$.ajax('POST', "myscript.cgi", data);
```

fieldSerialize(successful)

`fieldSerialize(successful true) returns String`

Serializes all field elements in the jQuery object into a query string. This method will return a string in the format: name1=value1&name2=value2
The successful argument controls whether or not serialization is limited to 'successful' controls (per http://www.w3.org/TR/html4/interact/forms.html#successful-controls). The default value of the successful argument is true.

Example:

Collect the data from all successful input elements into a query string

```
var data = $("input").formSerialize();
```

Example:

Collect the data from all successful radio input elements into a query string

```
var data = $(":radio").formSerialize();
```

Example:

Collect the data from all successful checkbox input elements in myForm into a query string

```
var data = $("#myForm :checkbox").formSerialize();
```

Example:

Collect the data from all checkbox elements in myForm (even the unchecked ones) into a query string

```
var data = $("#myForm :checkbox").formSerialize(false);
```

Example:

Collect the data from all successful input, select, textarea and button elements into a query string

```
var data = $(":input").formSerialize();
```

fieldValue(Boolean)

`fieldValue(Boolean successful) returns Array<String>`

Returns the value(s) of the element in the matched set. For example, consider the following form:
<form><fieldset>
    <input name="A" type="text" />
    <input name="A" type="text" />
    <input name="B" type="checkbox" value="B1" />
    <input name="B" type="checkbox" value="B2"/>
    <input name="C" type="radio" value="C1" />
    <input name="C" type="radio" value="C2" /> </fieldset></form>
var v = $(':text').fieldValue(); // if no values are entered into the text inputs v == ['',''] // if values entered into the text inputs are 'foo' and 'bar' v == ['foo','bar']
var v = $(':checkbox').fieldValue(); // if neither checkbox is checked v === undefined // if both checkboxes are checked v == ['B1', 'B2']
var v = $(':radio').fieldValue(); // if neither radio is checked v === undefined // if first radio is checked v == ['C1']
The successful argument controls whether or not the field element must be 'successful' (per http://www.w3.org/TR/html4/interact/forms.html#successful-controls). The default value of the successful argument is true. If this value is false the value(s) for each element is returned.
Note: This method *always* returns an array. If no valid value can be determined the
    array will be empty, otherwise it will contain one or more values.

Example:

Alerts the current value of the myPasswordElement element

```
var data = $("#myPasswordElement").fieldValue();
alert(data[0]);
```

Example:

Get the value(s) of the form elements in myForm

```
var data = $("#myForm :input").fieldValue();
```

Example:

Get the value(s) for the successful checkbox element(s) in the jQuery object.

```
var data = $("#myForm :checkbox").fieldValue();
```

Example:

Get the value(s) of the select control

```
  var data = $("#mySingleSelect").fieldValue();
```

Example:

Get the value(s) of the text input or textarea elements

```
  var data = $(':text').fieldValue();
```

Example:

Get the values for the select-multiple control

```
  var data = $("#myMultiSelect").fieldValue();
```

fieldValue(Element,Boolean)

`fieldValue(Element el, Boolean successful) returns String or Array<String> or null or undefined`

Returns the value of the field element.
The successful argument controls whether or not the field element must be 'successful' (per http://www.w3.org/TR/html4/interact/forms.html#successful-controls). The default value of the successful argument is true. If the given element is not successful and the successful arg is not false then the returned value will be null.
Note: If the successful flag is true (default) but the element is not successful, the return will be null
Note: The value returned for a successful select-multiple element will always be an array. Note: If the element has no value the return value will be undefined.

Example:

Gets the current value of the myPasswordElement element

```
  var data = jQuery.fieldValue($("#myPasswordElement")[0]);
```

clearForm()

`clearForm() returns jQuery`

Clears the form data. Takes the following actions on the form's input fields: - input text fields will have their 'value' property set to the empty string - select elements will have their 'selectedIndex' property set to -1 - checkbox and radio inputs will have their 'checked' property set to false - inputs of type submit, button, reset, and hidden will *not* be effected - button elements will *not* be effected

Example:

Clears all forms on the page.

```
  $('form').clearForm();
```

clearFields()

`clearFields() returns jQuery`

Clears the selected form elements. Takes the following actions on the matched elements: - input text fields will have their 'value' property set to the empty string - select elements will have their 'selectedIndex' property set to -1 - checkbox and radio inputs will have their 'checked' property set to false - inputs of type submit, button, reset, and hidden will *not* be effected - button elements will *not* be effected

Example:

Clears all inputs with class myInputs

```
  $('.myInputs').clearFields();
```

resetForm()

`resetForm() returns jQuery`

Resets the form data. Causes all form elements to be reset to their original value.

Example:

Resets all forms on the page.

```
  $('form').resetForm();
```

Interface

Accordion(Hash)

`Accordion(Hash hash) returns jQuery`

Create an accordion from a HTML structure

Example:

Converts definition list with id 'myAccordion' into an accordion width dt tags as headers and dd tags as panels

```
  $('#myAccordion').Accordion(
      {
       headerSelector : 'dt',
       panelSelector : 'dd',
       activeClass  : 'myAccordionActive',
       hoverClass  : 'myAccordionHover',
       panelHeight  : 200,
       speed   : 300
      }
     );
```

animateClass(Mixed,Mixed,Function,String)

```
animateClass(Mixed classToAnimate, Mixed speed, Function
callback, String easing) returns jQuery
```

animateStyle(String,Mixed,Function,String)

```
animateStyle(String styleToAnimate, Mixed speed, Function
callback, String easing) returns jQuery
```

3D Carousel(Hash)

`3D Carousel(Hash hash) returns jQuery`

Created a 3D Carousel from a list of images, with reflections and animated by mouse position

Example:

Creates a 3D carousel from all images inside div tag with id 'carousel'

```
window.onload =
   function()
   {
    $('#carousel').Carousel(
     {
      itemWidth: 110,
      itemHeight: 62,
      itemMinWidth: 50,
      items: 'a',
      reflections: .5,
      rotationSpeed: 1.8
     }
    );
   }
HTML
   <div id="carousel">
    <a href="" title=""><img src="" width="100%" /></a>
    <a href="" title=""><img src="" width="100%" /></a>
    <a href="" title=""><img src="" width="100%" /></a>
    <a href="" title=""><img src="" width="100%" /></a>
    <a href="" title=""><img src="" width="100%" /></a>
   </div>
CSS
   #carousel
   {
    width: 700px;
    height: 150px;
    background-color: #111;
    position: absolute;
    top: 200px;
    left: 100px;
   }
   #carousel a
   {
    position: absolute;
    width: 110px;
   }
```

Fisheye(Hash)

`Fisheye(Hash hash) returns jQuery`

Build a Fisheye menu from a list of links

Autocomplete(Hash)

`Autocomplete(Hash hash) returns jQuery`

Attach AJAX driven autocomplete/sugestion box to text input fields.

Draggable(Hash)

`Draggable(Hash hash) returns jQuery`

Create a draggable element with a number of advanced options including callback, Google Maps type draggables, reversion, ghosting, and grid dragging.

DraggableDestroy()

`DraggableDestroy() returns jQuery`

Destroy an existing draggable on a collection of elements

Example:

```
$('#drag2').DraggableDestroy();
```

Droppable(Hash)

`Droppable(Hash options) returns`

With the Draggables plugin, Droppable allows you to create drop zones for draggable elements.

Example:

```
$('#dropzone1').Droppable(
                           {
                             accept : 'dropaccept',
                             activeclass: 'dropzoneactive',
                             hoverclass: 'dropzonehover',
                             ondrop: function (drag) {
                                            alert(this); //the droppable
                                            alert(drag); //the draggable
                                      },
                             fit: true
                           }
                         )
```

DroppableDestroy()

`DroppableDestroy() returns jQuery`

Destroy an existing droppable on a collection of elements

Example:

```
  $('#drag2').DroppableDestroy();
```

$.recallDroppables()

```
$.recallDroppables() returns jQuery
```

Recalculate all Droppables

Example:

```
  $.recallDroppable();
```

Expander(Mixed)

`Expander(Mixed limit) returns jQuery`

Expands text and textarea elements while new characters are typed to the a miximum width

BlindUp(Mixed,Function,String)

```
BlindUp(Mixed speed, Function callback, String easing) returns
jQuery
```

BlindDown(Mixed,Function,String)

```
BlindDown(Mixed speed, Function callback, String easing)
returns jQuery
```

BlindToggleVertically(Mixed,Function,String)

```
BlindToggleVertically(Mixed speed, Function callback, String
easing) returns jQuery
```

BlindLeft(Mixed,Function,String)

```
BlindLeft(Mixed speed, Function callback, String easing)
returns jQuery
```

BlindRight(Mixed,Function,String)

```
BlindRight(Mixed speed, Function callback, String easing)
returns jQuery
```

BlindToggleHorizontally(Mixed,Function,String)

```
BlindToggleHorizontally(Mixed speed, Function callback, String
easing) returns jQuery
```

Bounce(Integer,Function)

```
Bounce(Integer hight, Function callback) returns jQuery
```

DropOutDown(Mixed,Function,String)

```
DropOutDown(Mixed speed, Function callback, String easing)
returns jQuery
```

DropInDown(Mixed,Function,String)

```
DropInDown(Mixed speed, Function callback, String easing)
returns jQuery
```

DropToggleDown(Mixed,Function,String)

```
DropToggleDown(Mixed speed, Function callback, String easing)
returns jQuery
```

DropOutUp(Mixed,Function,String)

```
DropOutUp(Mixed speed, Function callback, String easing)
returns jQuery
```

DropInUp(Mixed,Function,String)

```
DropInUp(Mixed speed, Function callback, String easing)
returns jQuery
```

DropToggleUp(Mixed,Function,String)

```
DropToggleUp(Mixed speed, Function callback, String easing)
returns jQuery
```

DropOutLeft(Mixed,Function,String)

```
DropOutLeft(Mixed speed, Function callback, String easing)
returns jQuery
```

DropInLeft(Mixed,Function,String)

```
DropInLeft(Mixed speed, Function callback, String easing)
returns jQuery
```

DropToggleLeft(Mixed,Function,String)

```
DropToggleLeft(Mixed speed, Function callback, String easing)
returns jQuery
```

DropOutRight(Mixed,Function,String)

```
DropOutRight(Mixed speed, Function callback, String easing)
returns jQuery
```

DropInRight(Mixed,Function,String)

```
DropInRight(Mixed speed, Function callback, String easing)
returns jQuery
```

DropToggleRight(Mixed,Function,String)

```
DropToggleRight(Mixed speed, Function callback, String easing)
returns jQuery
```

Fold(Mixed,Integer,Function,String)

```
Fold(Mixed speed, Integer height, Function callback, String
easing) returns jQuery
```

UnFold(Mixed,Integer,Function,String)

```
UnFold(Mixed speed, Integer height, Function callback, String
easing) returns jQuery
```

FoldToggle(Mixed,Integer,Function,String)

```
FoldToggle(Mixed speed, Integer height, Function callback,
String easing) returns jQuery
```

Highlight(Mixed,String,Function,String)

`Highlight(Mixed speed, String color, Function callback, String easing) returns jQuery`

CloseVertically(Mixed,Function,String)

```
CloseVertically(Mixed speed, Function callback, String easing)
returns jQuery
```

CloseHorizontally(Mixed,Function,String)

```
CloseHorizontally(Mixed speed, Function callback, String
easing) returns jQuery
```

SwitchHorizontally(Mixed,Function,String)

```
SwitchHorizontally(Mixed speed, Function callback, String
easing) returns jQuery
```

SwitchVertically(Mixed,Function,String)

```
SwitchVertically(Mixed speed, Function callback, String
easing) returns jQuery
```

OpenVertically(Mixed,Function,String)

```
OpenVertically(Mixed speed, Function callback, String easing)
returns jQuery
```

OpenHorizontally(Mixed,Function,String)

```
OpenHorizontally(Mixed speed, Function callback, String
easing) returns jQuery
```

Bounce(Mixed,Integer,Function)

```
Bounce(Mixed speed, Integer times, Function callback) returns
jQuery
```

Grow(Mixed,Function,String)

```
Grow(Mixed speed, Function callback, String easing) returns
jQuery
```

Shrink(Mixed,Function,String)

```
Shrink(Mixed speed, Function callback, String easing) returns
jQuery
```

Puff(Mixed,Function,String)

```
Puff(Mixed speed, Function callback, String easing) returns
jQuery
```

Scale(Mixed,Integer,Integer,Boolean,Function,String)

```
Scale(Mixed speed, Integer from, Integer to, Boolean reastore,
Function callback, String easing) returns jQuery
```

ScrollTo(Mixed,String,String)

```
ScrollTo(Mixed speed, String axis, String easing) returns
jQuery
```

ScrollToAnchors(Mixed,String,String)

```
ScrollToAnchors(Mixed speed, String axis, String easing)
returns jQuery
```

Shake(Integer,Function)

```
Shake(Integer times, Function callback) returns jQuery
```

SlideInUp(Mixed,Function,String)

```
SlideInUp(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideOutUp(Mixed,Function,String)

```
SlideOutUp(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideToggleUp(Mixed,Function,String)

```
SlideToggleUp(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideInDown(Mixed,Function,String)

```
SlideInDown(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideOutDown(Mixed,Function,String)

```
SlideOutDown(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideToggleDown(Mixed,Function,String)

```
SlideToggleDown(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideInLeft(Mixed,Function,String)

```
SlideInLeft(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideOutLeft(Mixed,Function,String)

```
SlideOutLeft(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideToggleLeft(Mixed,Function,String)

```
SlideToggleLeft(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideInRight(Mixed,Function,String)

```
SlideInRight(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideOutRight(Mixed,Function,String)

```
SlideOutRight(Mixed speed, Function callback, String easing)
returns jQuery
```

SlideToggleRight(Mixed,Function,String)

```
SlideToggleRight(Mixed speed, Function callback, String
easing) returns jQuery
```

TransferTo(Hash)

```
TransferTo(Hash hash) returns jQuery
```

Imagebox(Hash)

`Imagebox(Hash hash) returns jQuery`

This a jQuery equivalent for Lightbox2. Alternative to image popups that will display images in an overlay. All links that have attribute 'rel' starting with 'imagebox' and link to an image will display the image inside the page. Galleries can by build buy giving the value 'imagebox-galname' to attribute 'rel'. Attribute 'title' will be used as caption. Keyboard navigation: - next image: arrow right, page down, 'n' key, space - previous image: arrow left, page up, 'p' key, backspace - close: escape

CSS

```
 #ImageBoxOverlay
 {
  background-color: #000;
 }
 #ImageBoxCaption
 {
  background-color: #F4F4EC;
 }
 #ImageBoxContainer
 {
  width: 250px;
  height: 250px;
  background-color: #F4F4EC;
 }
 #ImageBoxCaptionText
 {
  font-weight: bold;
  padding-bottom: 5px;
  font-size: 13px;
  color: #000;
 }
 #ImageBoxCaptionImages
 {
  margin: 0;
 }
 #ImageBoxNextImage
 {
  background-image: url(images/imagebox/spacer.gif);
  background-color: transparent;
 }
 #ImageBoxPrevImage
 {
```

```
 background-image: url(images/imagebox/spacer.gif);
 background-color: transparent;
}
#ImageBoxNextImage:hover
{
 background-image: url(images/imagebox/next_image.jpg);
 background-repeat: no-repeat;
 background-position: right top;
}
#ImageBoxPrevImage:hover
{
 background-image: url(images/imagebox/prev_image.jpg);
 background-repeat: no-repeat;
 background-position: left bottom;
}
```

Resizable(Hash)

`Resizable(Hash hash) returns jQuery`

Create a resizable element with a number of advanced options including callback, dragging

ResizableDestroy()

`ResizableDestroy() returns jQuery`

Destroy a resizable

Slider(Hash)

`Slider(Hash hash) returns jQuery`

Create a slider width options

SliderSetValues(Array)

```
SliderSetValues(Array values) returns jQuery
```

Set value/position for slider indicators

SliderSetValues()

`SliderSetValues() returns jQuery`

Get value/position for slider indicators

Slideshow(Hash)

`Slideshow(Hash hash) returns jQuery`

Creates an image slideshow. The slideshow can autoplay slides, each image can have caption, navigation links: next, prev, each slide. A page may have more then one slideshow, eachone working independently. Each slide can be bookmarked. The source images can be defined by JavaScript in slideshow options or by HTML placing images inside container.

Sortable(Hash)

`Sortable(Hash options) returns`

Allows you to resort elements within a container by dragging and dropping. Requires the Draggables and Droppables plugins. The container and each item inside the container must have an ID. Sortables are especially useful for lists.

Example:

```
$('ul').Sortable(
                                   {
                                    accept : 'sortableitem',
                                    activeclass : 'sortableactive',
                                     hoverclass : 'sortablehover',
                                     helperclass : 'sorthelper',
                                     opacity:  0.5,
                                     fit : false
                                    }
                                   )
```

SortableAddItem(DOMElement)

`SortableAddItem(DOMElement elem) returns jQuery`

A new item can be added to a sortable by adding it to the DOM and then adding it via SortableAddItem.

Example:

```
$('#sortable1').append('<li id="newitem">new item</li>')
                       .SortableAddItem($("#new_item")[0])
```

SortableDestroy()

`SortableDestroy() returns jQuery`

Destroy a sortable

Example:

```
$('#sortable1').SortableDestroy();
```

$.SortSerialize()

`$.SortSerialize( ) returns String`

This function returns the hash and an object (can be used as arguments for $.post) for every sortable in the page or specific sortables. The hash is based on the 'id' attributes of container and items.

ToolTip(Hash)

`ToolTip(Hash hash) returns jQuery`

Creates tooltips using title attribute

EnableTabs()

```
EnableTabs() returns jQuery
```

Enable tabs in textareas

DisableTabs()

`DisableTabs() returns jQuery`

Disable tabs in textareas

Tabs

tabs(Number,Object)

`tabs(Number initial, Object settings) returns jQuery`

Create an accessible, unobtrusive tab interface based on a particular HTML structure.
The underlying HTML has to look like this:

```
<div id="container">
  <ul>
     <li><a href="#fragment-1">Section 1</a></li>
     <li><a href="#fragment-2">Section 2</a></li>
     <li><a href="#fragment-3">Section 3</a></li>
  </ul>
  <div id="fragment-1">
  </div>
  <div id="fragment-2">
  </div>
  <div id="fragment-3">
  </div> </div>
```

Each anchor in the unordered list points directly to a section below represented by one of the divs (the URI in the anchor's href attribute refers to the fragment with the corresponding id). Because such HTML structure is fully functional on its own, e.g. without JavaScript, the tab interface is accessible and unobtrusive.
A tab is also bookmarkable via hash in the URL. Use the History/Remote plugin (Tabs will auto-detect its presence) to fix the back (and forward) button.

Example:

Create a basic tab interface.

```
$('#container').tabs();
```

Example:

Create a basic tab interface with the second tab initially activated.

```
$('#container').tabs(2);
```

Example:

Create a tab interface with the third and fourth tab being disabled.

```
$('#container').tabs({disabled: [3, 4]});
```

Example:

Create a tab interface that uses slide down/up animations for showing/hiding tab content upon tab switching.

```
$('#container').tabs({fxSlide: true});
```

triggerTab(String|Number)

`triggerTab(String|Number tab) returns jQuery`

Activate a tab programmatically with the given position (no zero-based index) or its id, e.g. the URL's fragment identifier/hash representing a tab, as if the tab itself were clicked.

Example:

Activate the second tab of the tab interface contained in <div id="container">.

```
$('#container').triggerTab(2);
```

Example:

Activate the first tab of the tab interface contained in <div id="container">.

```
$('#container').triggerTab(1);
```

Example:

Activate the first tab of the tab interface contained in <div id="container">.

```
$('#container').triggerTab();
```

Example:

Activate a tab via its URL fragment identifier representation.

```
$('#container').triggerTab('fragment-2');
```

disableTab(String|Number)

`disableTab(String|Number tab) returns jQuery`

Disable a tab, so that clicking it has no effect.

Example:

Disable the second tab of the tab interface contained in <div id="container">.

```
$('#container').disableTab(2);
```

enableTab(String|Number)

`enableTab(String|Number tab) returns jQuery`

Enable a tab that has been disabled.

Example:

Enable the second tab of the tab interface contained in <div id="container">.

```
  $('#container').enableTab(2);
```

activeTab()

`activeTab() returns Number`

Get the position of the currently selected tab (no zero-based index).

Example:

Get the position of the currently selected tab of an interface contained in <div id="container">.

```
$('#container').activeTab();
```

bgiframe

bgiframe(Map)

`bgiframe(Map settings) returns jQuery`

The bgiframe is chainable and applies the iframe hack to get around zIndex issues in IE6. It will only apply itself in IE and adds a class to the iframe called 'bgiframe'. The iframe is appeneded as the first child of the matched element(s) with a tabIndex and zIndex of -1.
By default the plugin will take borders, sized with pixel units, into account. If a different unit is used for the border's width, then you will need to use the top and left settings as explained below.
NOTICE: This plugin has been reported to cause perfromance problems when used on elements that change properties (like width, height and opacity) a lot in IE6. Most of these problems have been caused by the expressions used to calculate the elements width, height and borders. Some have reported it is due to the opacity filter. All these settings can be changed if needed as explained below.

Example:

```
$('div').bgiframe();
```

HTML:

```
<div><p>Paragraph</p></div>
```

Result:

```
<div><iframe class="bgiframe".../><p>Paragraph</p></div>
```

Dimensions

height()

`height() returns Object`

If used on document, returns the document's height (innerHeight) If used on window, returns the viewport's (window) height See core docs on height() to see what happens when used on an element.

Example:

```
$("#testdiv").height()
```

Result:

```
200
```

Example:

```
$(document).height()
```

Result:

```
800
```

Example:

```
$(window).height()
```

Result:

```
400
```

width()

`width() returns Object`

If used on document, returns the document's width (innerWidth) If used on window, returns the viewport's (window) width See core docs on height() to see what happens when used on an element.

Example:

```
$("#testdiv").width()
```

Result:

```
200
```

Example:

```
$(document).width()
```

Result:

```
800
```

Example:

```
$(window).width()
```

Result:

```
400
```

innerHeight()

`innerHeight() returns Number`

Returns the inner height value (without border) for the first matched element. If used on document, returns the document's height (innerHeight) If used on window, returns the viewport's (window) height

Example:

```
$("#testdiv").innerHeight()
```

Result:

```
800
```

innerWidth()

`innerWidth() returns Number`

Returns the inner width value (without border) for the first matched element. If used on document, returns the document's Width (innerWidth) If used on window, returns the viewport's (window) width

Example:

```
$("#testdiv").innerWidth()
```

Result:

```
1000
```

outerHeight()

`outerHeight() returns Number`

Returns the outer height value (including border) for the first matched element. Cannot be used on document or window.

Example:

```
$("#testdiv").outerHeight()
```

Result:

```
1000
```

`outerHeight() returns Number`

Returns the outer width value (including border) for the first matched element. Cannot be used on document or window.

Example:

```
$("#testdiv").outerHeight()
```

Result:

```
1000
```

scrollLeft()

`scrollLeft() returns Number`

Returns how many pixels the user has scrolled to the right (scrollLeft). Works on containers with overflow: auto and window/document.

Example:

```
$("#testdiv").scrollLeft()
```

Result:

```
100
```

scrollLeft(Number)

`scrollLeft(Number value) returns jQuery`

Sets the scrollLeft property and continues the chain. Works on containers with overflow: auto and window/document.

Example:

```
$("#testdiv").scrollLeft(10).scrollLeft()
```

Result:

```
10
```

scrollTop()

`scrollTop() returns Number`

Returns how many pixels the user has scrolled to the bottom (scrollTop). Works on containers with overflow: auto and window/document.

Example:

```
$("#testdiv").scrollTop()
```

Result:

```
100
```

scrollTop(Number)

`scrollTop(Number value) returns jQuery`

Sets the scrollTop property and continues the chain. Works on containers with overflow: auto and window/document.

Example:

```
$("#testdiv").scrollTop(10).scrollTop()
```

Result:

```
10
```

position(Map,Object)

`position(Map options, Object returnObject) returns Object`

Returns the top and left positioned offset in pixels. The positioned offset is the offset between a positioned parent and the element itself.

Example:

```
$("#testdiv").position()
```

Result:

```
{ top: 100, left: 100 }
```

offset(Map,Object)

`offset(Map options, Object returnObject) returns Object`

Returns the location of the element in pixels from the top left corner of the viewport.
For accurate readings make sure to use pixel values for margins, borders and padding.
Known issues: - Issue: A div positioned relative or static without any content before it and its parent will report an offsetTop of 0 in Safari
  Workaround: Place content before the relative div ... and set height and width to 0 and overflow to hidden

Example:

```
$("#testdiv").offset()
```

Result:

```
{ top: 100, left: 100, scrollTop: 10, scrollLeft: 10 }
```

Example:

```
$("#testdiv").offset({ scroll: false })
```

Result:

```
{ top: 90, left: 90 }
```

Example:

```
var offset = {}
$("#testdiv").offset({ scroll: false }, offset)
```

Result:

```
offset = { top: 90, left: 90 }
```

offsetLite(Map,Object)

`offsetLite(Map options, Object returnObject) returns Object`

Returns the location of the element in pixels from the top left corner of the viewport. This method is much faster than offset but not as accurate. This method can be invoked by setting the lite option to true in the offset method.

Tooltip

Tooltip(Object)

`Tooltip(Object settings) returns jQuery`

Display a customized tooltip instead of the default one for every selected element. The tooltip behaviour mimics the default one, but lets you style the tooltip and specify the delay before displaying it. In addition, it displays the href value, if it is available.
Requires dimensions plugin.
When used on a page with select elements, include the bgiframe plugin. It is used if present.
To style the tooltip, use these selectors in your stylesheet:
#tooltip - The tooltip container
#tooltip h3 - The tooltip title
#tooltip div.body - The tooltip body, shown when using showBody
#tooltip div.url - The tooltip url, shown when using showURL

Example:

Shows tooltips for anchors, inputs and images, if they have a title

```
$('a, input, img').Tooltip();
```

Example:

Shows tooltips for labels with no delay, tracking mousemovement, displaying the tooltip when the label is clicked.

```
$('label').Tooltip({
 delay: 0,
 track: true,
 event: "click"
});
```

Example:

This example starts with modifying the global settings, applying them to all following Tooltips; Afterwards, Tooltips for anchors with class pretty are created with an extra class for the Tooltip: "fancy" for anchors, "fancy-img" for images

```
// modify global settings
$.extend($.fn.Tooltip.defaults, {
 track: true,
 delay: 0,
 showURL: false,
 showBody: " - ",
fixPNG: true
});
// setup fancy tooltips
$('a.pretty').Tooltip({
  extraClass: "fancy"
});
$('img.pretty').Tooltip({
  extraClass: "fancy-img",
});
```

$.Tooltip.blocked()

`$.Tooltip.blocked() returns Boolean`

A global flag to disable all tooltips.

Example:

```
$("button.openModal").click(function() {
 $.Tooltip.blocked = true;
 // do some other stuff, eg. showing a modal dialog
 $.Tooltip.blocked = false;
});
```

$.Tooltip.defaults()

`$.Tooltip.defaults() returns Map`

Global defaults for tooltips. Apply to all calls to the Tooltip plugin after modifying the defaults.

Example:

```
$.extend($.Tooltip.defaults, {
 track: true,
 delay: 0
});
```