1. Core

`$( String expr,  Element context)`

This function accepts a string containing a CSS or
basic XPath selector which is then used to match a set of elements.

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, $() looks for DOM elements within the context of the current HTML document.

Example:

This 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 html)

This function accepts a string of raw HTML.

This function accepts a string of raw HTML.
The HTML string is different from the traditional selectors in that it creates the DOM elements representing that HTML string, on the fly,
to be (assumedly) inserted into the document later.

Example:

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

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

$(Element elem)

Wrap jQuery functionality around a specific DOM Element.

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

Example:

```
$(document).find("div > p")
```

HTML:

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

Result:

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

Example:

Sets the background color of the page to black.

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

$(Array<Element> elems)

Wrap jQuery functionality around a set of DOM Elements.

Example:

Hides all the input elements within a form

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

$(Function fn)

A shorthand for $(document).

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
all of the other $() operations on your page. 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
});
```

$(jQuery obj)

A means of creating a cloned copy of a jQuery object.

A means of creating a cloned copy of a jQuery object. This function copies the set of matched elements from one jQuery object and creates
another, new, jQuery object containing the same elements.

Example:

Locates all p elements with all div elements, without disrupting the original jQuery object contained in 'div' (as would normally be the case if a simple div.find("p") was done).

```
var div = $("div");
$( div ).find("p");
```

length()

The number of elements currently matched.

Example:

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

HTML:

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

Result:

```
2
```

size()

The number of elements currently matched.

Example:

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

HTML:

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

Result:

```
2
```

get()

Access all matched elements.

Access all matched 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).

Example:

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

HTML:

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

Result:

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

get(Number num)

Access a single matched element.

Access a single matched element. num is used to access the Nth element matched.

Example:

```
$("img").get(1);
```

HTML:

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

Result:

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

each(Function fn)

Execute a function within the context of every matched element.

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 element.

Additionally, the function, when executed, is passed a single argument representing the position of the element in the matched set.

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(Object obj)

Searches every matched element for the object and returns the index of the element, if found, starting with zero.

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:

```
$("*").index(document.getElementById('foobar'))
```

HTML:

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

Result:

```
0
```

Example:

```
$("*").index(document.getElementById('foo'))
```

HTML:

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

Result:

2

Example:

```
$("*").index(document.getElementById('bar'))
```

HTML:

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

Result:

```
-1
```

$.extend(Object prop)

Extends the jQuery object itself.

Extends the jQuery object itself. Can be used to add functions into the jQuery namespace and to 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; }
});
```

eq(Number pos)

Reduce the set of matched elements to a single element.

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 pos)

Reduce the set of matched elements to all elements before a given position.

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 pos)

Reduce the set of matched elements to all elements after a given position.

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

attr(String name)

Access a property on the first matched element.

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

Example:

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

HTML:

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

Result:

```
test.jpg
```

attr(Hash prop)

Set a hash of key/value object properties to all matched elements.

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

Example:

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

HTML:

```
<img/>
```

Result:

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

attr(String key, Object value)

Set a single property to a value, on all matched elements.

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:

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

HTML:

```
<img/>
```

Result:

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

text()

Retrieve the text contents of all matched elements.

Retrieve 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:

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

HTML:

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

Result:

```
Test Paragraph.
```

removeAttr(String name)

Remove an attribute from each of the matched elements.

Example:

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

HTML:

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

Result:

```
<input/>
```

addClass(String class)

Adds the specified class to each of the set of matched elements.

Example:

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

HTML:

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

Result:

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

removeClass(String class)

Removes the specified class from the set of matched elements.

Example:

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

HTML:

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

Result:

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

toggleClass(String class)

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> ]
```

3. CSS

`css( String name)`

Access a style property on the first matched element.

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. Note that for all style properties with a dash (like 'font-weight'), you have to
write it in camelCase. In other words: Every time you have a '-' in a property, remove it and replace the next character with an uppercase representation of itself. Eg. fontWeight, fontSize, fontFamily,

borderWidth,
borderStyle, borderBottomWidth etc.

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

HTML:

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

Result:

```
bold
```

css(Hash prop)

Set a hash of key/value style properties to all matched elements.

Set a hash of key/value 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:

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

HTML:

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

Result:

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

css(String key, Object value)

Set a single style property to a value, on all matched elements.

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>
```

width()

Get the current CSS width of the first matched element.

Example:

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

HTML:

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

Result:

```
"300px"
```

width(String val)

Set the CSS width of every matched element.

Set the CSS width of every matched element. Be sure to include the "px" (or other unit of measurement) after the number that you specify, otherwise you might get strange results.

Example:

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

HTML:

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

Result:

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

height()

Get the current CSS height of the first matched element.

Example:

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

HTML:

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

Result:

```
"14px"
```

height(String val)

Set the CSS height of every matched element.

Set the CSS height of every matched element. Be sure to include the "px" (or other unit of measurement) after the number that you specify, otherwise you might get strange results.

Example:

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

HTML:

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

Result:

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

top()

Get the current CSS top of the first matched element.

Example:

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

HTML:

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

Result:

```
"0px"
```

top(String val)

Set the CSS top of every matched element.

Set the CSS top of every matched element. Be sure to include the "px" (or other unit of measurement) after the number that you specify, otherwise you might get strange results.

Example:

```
$("p").top("20px");
```

HTML:

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

Result:

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

left()

Get the current CSS left of the first matched element.

Example:

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

HTML:

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

Result:

```
"0px"
```

left(String val)

Set the CSS left of every matched element.

Set the CSS left of every matched element. Be sure to include the "px" (or other unit of measurement) after the number that you specify, otherwise you might get strange results.

Example:

```
$("p").left("20px");
```

HTML:

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

Result:

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

position()

Get the current CSS position of the first matched element.

Example:

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

HTML:

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

Result:

```
"static"
```

position(String val)

Set the CSS position of every matched element.

Example:

```
$("p").position("relative");
```

HTML:

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

Result:

```
<p style="position:relative;">This is just a test.</p>
```

float()

Get the current CSS float of the first matched element.

Example:

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

HTML:

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

Result:

```
"none"
```

float(String val)

Set the CSS float of every matched element.

Example:

```
$("p").float("left");
```

HTML:

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

Result:

```
<p style="float:left;">This is just a test.</p>
```

overflow()

Get the current CSS overflow of the first matched element.

Example:

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

HTML:

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

Result:

```
"none"
```

overflow(String val)

Set the CSS overflow of every matched element.

Example:

```
$("p").overflow("auto");
```

HTML:

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

Result:

```
<p style="overflow:auto;">This is just a test.</p>
```

color()

Get the current CSS color of the first matched element.

Example:

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

HTML:

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

Result:

```
"black"
```

color(String val)

Set the CSS color of every matched element.

Example:

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

HTML:

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

Result:

```
<p style="color:blue;">This is just a test.</p>
```

background()

Get the current CSS background of the first matched element.

Example:

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

HTML:

```
<p style="background:blue;">This is just a test.</p>
```

Result:

```
"blue"
```

background(String val)

Set the CSS background of every matched element.

Example:

```
$("p").background("blue");
```

HTML:

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

Result:

```
<p style="background:blue;">This is just a test.</p>
```

4. Javascript

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

Extend one object with one or more others, returning the original, modified, 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 obj, Function fn)

A generic iterator function, which can be used to seemlessly iterate over both objects and arrays.

A generic iterator function, which can be used to seemlessly 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.

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){
 alert( "Item #" + i + ": " + this );
});
```

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){
 alert( "Name: " + i + ", Value: " + this );
});
```

$.trim(String str)

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

Example:

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

Result:

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

$.merge(Array first, Array second)

Merge two arrays together, removing all duplicates.

Merge two arrays together, removing all duplicates. The final order or the new array is: All the results from the first array, followed by the unique results from the second array.

Example:

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

Result:

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

Example:

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

Result:

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

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

Filter items out of an array, by using a filter function.

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 should return 'true' if you wish to keep the item in
the array, false if it should be removed.

Example:

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

Result:

```
[1, 2]
```

$.map(Array array, Function fn)

Translate all items in an array to another array of items.

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:

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

Result:

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

Example:

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

Result:

```
[2, 3]
```

Example:

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

Result:

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

$.browser()

Contains flags for the useragent, read from navigator.

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. Effects

show()

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> ]
```

hide()

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> ]
```

```
var pass = true, div = $("div");
div.hide().each(function(){
 if ( this.style.display != "none" ) pass = false;
});
ok( pass, "Hide" );
```

toggle()

Toggles each of the set of matched elements.

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> ]
```

6. Events

bind(String type, Function fn)

Binds a handler to a particular event (like click) for each matched element.

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.

Example:

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

HTML:

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

Result:

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

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();
} )
```

unbind(String type, Function fn)

The opposite of bind, removes a bound event from each of the matched
elements.

The opposite of bind, removes a bound event from each of the matched elements. You must pass the identical function that was used in the original
bind method.

Example:

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

HTML:

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

Result:

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

unbind(String type)

Removes all bound events of a particular type from each of the matched elements.

Example:

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

HTML:

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

Result:

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

unbind()

Removes all bound events from each of the matched elements.

Example:

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

HTML:

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

Result:

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

trigger(String type)

Trigger a type of event on every matched element.

Example:

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

HTML:

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

Result:

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

toggle(Function even, Function odd)

Toggle between two function calls every other click.

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.

Example:

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

hover(Function over, Function out)

A method for simulating hovering (moving the mouse on, and off, an object).

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("over");
},function(){
 $(this).addClass("out");
});
```

ready(Function fn)

Bind a function to be executed whenever the DOM is ready to be traversed and manipulated.

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 exactly what 99.99% of all Javascript code needs to run.

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... });
```

7. AJAX

`loadIfModified( String url,  Hash params, Function callback)`

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 url, Object params, Function callback)

Load HTML from a remote file and inject it into the DOM.

Load HTML from a remote file and inject it into the DOM.
Note: Avoid to use this to load scripts, instead use $.getScript.

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()

Serializes a set of input elements into a string of data.

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 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 callback)

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 callback)

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 callback)

Attach a function to be executed whenever an AJAX request completes.

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 callback)

Attach a function to be executed whenever an AJAX request completes
successfully.

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 callback)

Attach a function to be executed whenever an AJAX request fails.

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 callback)

Attach a function to be executed before an AJAX request is send.

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

Example:

Show a message before an AJAX request is send.

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

$.get(String url, Hash params, Function callback)

Load a remote page using an HTTP GET request.

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 url, Hash params, Function callback)

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 url, Function callback)

Loads, and executes, a remote JavaScript file using an HTTP GET request.

Loads, and executes, a remote JavaScript file using an HTTP GET request.
Warning: Safari <= 2.0.x is unable to evalulate 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 url, Hash params, Function callback)

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 url, Hash params, Function callback)

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 time)

Set the timeout of all AJAX requests to a specific amount of time.

Set the timeout 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.

Example:

Make all AJAX requests timeout after 5 seconds.

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

$.ajax(Hash prop)

Load a remote page using an HTTP request.

Load a remote page using an HTTP request.
This is jQuery's low-level AJAX implementation. See $.get, $.post etc. for
higher-level abstractions.

$.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: Make sure the server sends the right mimetype (eg. xml as "text/xml"). Sending the wrong mimetype will get you into serious trouble that jQuery can't solve.

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 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 send 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 with others means when synchronization is
necessary, instead to block the complete browser.

```
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
});
```