

Core		Selectors	
<code>\$(html) / \$(element) / \$(selector [,context])</code>	<code>\$(func) === \$(document).ready(func)</code>	<code>#id / .className</code>	<code>:visible / :hidden</code>
<code>each (func)</code>	<code>\$("img").each(function(i){</code> this.src = "test" + i + ".jpg"; return false; // stop looping over each });	<code>:enabled / :disabled</code>	<code>:checked / :selected</code>
<code>\$(i) === \$.get(i)</code>		<code>:nth-child(n) / :first-child / :last-child / :only-child / :gt(n) / :lt(n)</code>	
<code>\$(0).length / \$(0).size() / index(obj)</code>		<code>:first / :last</code>	<code>:eq(0) / :nth(0)</code>
<code>\$(0).selector() / \$(0).context()</code>		<code>:header / :animated</code>	<code>:even / :odd</code>
Data		AJAX	
<code>.data(key) / .data(key,val) / .removeData(key)</code>	Store/retrieve/remove arbitrary data tied to elements	<code>E.empty</code>	has no children (including text nodes)
<code>.queue(name) / .queue(name, fn queue)</code>	Retrieve an element's queue or add to/replace existing queue	<code>E.not(s)</code>	does not match simple selector s
Attributes		<code>E F</code>	F element descendant of an E element
<code>attr(name) / attr(name,val) / attr({name:val})</code>	<code>attr(name, func)</code>	<code>E > F</code>	F element child of an E element
<code>removeAttr(name)</code>	<code>\$("img").attr("title", function() { return this.src });</code>	<code>E + F</code>	F element immediately preceded by an E element
<code>addClass(c) / hasClass(c) / removeClass(c) / toggleClass(c[,switch])</code>		<code>E ~ F</code>	F element preceded by an E element
<code>html() / html(content) / text() / text(content) / val() / val(value)</code>		<code>E[foo]</code>	contains a "foo" attribute
Traversing		<code>E[foo=bar]</code>	"foo" attribute value is exactly equal to "bar"
<code>add(expr) / add(html) / add(Element)</code>	Append more elements to the set of matched elements	<code>E[foo^=bar]</code>	"foo" attribute value begins exactly with "bar"
<code>contains('text')</code>	<code>\$("div").contains('text') === \$("div :contains('text')"</code>	<code>E[foo\$=bar]</code>	"foo" attribute value ends exactly with "bar"
<code>filter(expr) / filter(func)</code>	Leave elements matching expr or func returning true	<code>E[foo*=bar]</code>	"foo" attribute value contains the substring "bar"
<code>find(expr)</code>	<code>\$("p").find("span") === \$("p span"</code>	<code>E[foo!=bar]</code>	"foo" attribute is not equal to "bar"
<code>is(expr [,expr])</code>	Returns true if any in set matches expr. Complex selectors ok	<code>E[foo~=bar]</code>	space-delimited "foo" attribute contains "bar"
<code>next([expr]) / prev([expr])</code>	Only immediate next/prev sibling [if expr matches]	<code>E[foo=bar][baz=bop]</code>	Match multiple attributes
<code>nextAll([expr]) / prevAll([expr])</code>	All next/previous siblings [if expr matches]	<code>:parent</code>	elements which have child elements (including text)
<code>not(expr) / not(Element)</code>	Removes matched elements from list	<code>:contains('test')</code>	elements which contain the specified text.
<code>parent([expr])</code>	Immediate parent, if matches expr	<code>:input</code>	All form elements, not just type=input
<code>parents([expr])</code>	All parent elements matching expr	Types= <code>:password :radio :checkbox :submit :image :reset :button :file</code>	
<code>offsetParent()</code>	Positioning offset parent	AJAX	
<code>closest(expr) === parents(expr :first)</code>	Find closest parent element matching expr	<code>\$.ajax(properties)</code>	async: true
<code>siblings([expr]) / children([expr])</code>	<code>andSelf() / end()</code>	<code>\$.ajaxSetup(properties)</code>	beforeSend: func(xhr)
DOM Manipulation		<code>\$.get (url, properties, fn(data))</code>	cache: true (false=no caching)
<code>before(content) / after(content)</code>	Creates a new sibling before/after element	<code>\$.getJSON (url,props,fn(json))</code>	complete: func(xhr, textStatus)
<code>insertBefore(expr) / insertAfter(expr)</code>	Attach selected elements as new sibling to others	<code>\$.getScript (url, callback)</code>	contentType: String
<code>prepend(content) / append(content)</code>	Creates a new child node at the beginning/end	<code>\$.post (url, props, fn(data))</code>	data: {obj} String
<code>prependTo(expr) / appendTo(expr)</code>	Attach selected elements to others, return attached	<code>ajaxComplete (fn(xhr,props))</code>	dataFilter: func(data,type) - return sanitized data
<code>empty()</code> - Removes all child nodes and content	<code>remove()</code>	<code>ajaxError (fn(xhr,props))</code>	dataType: [xml,html,script,json,jsonp,text]
<code>wrap(html)</code>	<code>\$("p").wrap("<div class='wrap'></div>");</code>	<code>ajaxSend (fn(xhr,props))</code>	error: func(xhr, textStatus, exception)
<code>wrapAll(html elem) / wrapInner(html elem)</code>	<code>replaceWith(content) / replaceWith(expr)</code>	<code>ajaxStart (fn(xhr,props))</code>	global: true (fire global events)
<code>clone([boolean])</code>	Clone event handlers if passed true	<code>ajaxStop (fn(xhr,props))</code>	ifModified: false
CSS		<code>ajaxSuccess (fn(xhr,props))</code>	jsonp: String
<code>css(name) - get val from first element in list only</code>	<code>css(key,val) / css({key:val, key:val})</code>	<code>.serialize()</code>	processData:true
<code>offset() / position()</code>	Returns {top,left} relative to doc or offset parent	<code>.serializeArray()</code>	scriptCharset: String
<code>scrollTop([num]) / scrollLeft([num])</code>	Scroll position of first element only	<code>.load (url, props, fn(responseText,status,xhr))</code>	success: func(data, textStatus)
<code>height() / height(val) / width() / width(val)</code>	<code>innerHeight() / innerWidth()</code>	partial content using selector in url:	timeout: Number
<code>outerHeight([bool]) / outerWidth([bool])</code>	Pass false to ignore margins	<code>\$("#feed").load("feeds.php .results",</code> {limit: 25}, function(text,status,xhr) { alert("Loaded 25!");	type: [POST,GET]
Events (return false from handlers to cancel default action)		<code>);</code>	url: string
<code>bind (type,data,func) / unbind (type,func)</code>	function handler(event) { alert(event.data.foo);	Feature/Browser Detection	
<code>one (type, data, func) - execute only once</code>		<code>\$.support.property</code>	Check for browser support of features:
<code>hover (overfunc, outfunc)</code>		boxModel, cssFloat, hrefNormalized, htmlSerialize, leadingWhitespace, noCloneEvent, objectAll, opacity, scriptEval, style, tbody	
<code>toggle (evenfunc, oddfunc)</code>	<code>\$("p").bind("click", {foo: "bar"}, handler)</code>	<code>\$.browser.version</code>	<code>\$.browser</code> : [safari, opera, msie, mozilla]
<code>trigger (type, data)</code>	Executes browser's default action also	Misc	
Trigger an event: <code>event()</code>	Bind a function to an event: <code>event(fn)</code>	<code>\$.each (obj, func)</code>	<code>\$.trim (str) / \$.unique (array)</code>
<code>EventTypes: blur, change, click, dblclick, error, focus, keydown, keypress, keyup, load, mousedown, mouseenter, mouseleave, mousemove, mouseout, mouseover, mouseup, resize, scroll, select, submit, unload</code>		<code>\$.each([0,1,2], function(i, n){</code> alert("Item #" + i + ": " + n); });	<code>\$.extend (target, prop1, propN)</code>
<code>trigger (type, data)</code>	Executes browser's default action also	<code>\$.each({name:"John",lang:"JS"},function(i,n){</code> alert("Name: " + i + ", Value: " + n); });	var options = { name: "bar" }; \$.extend({validate:false,name:'foo'}, options); Result == { validate: false, name: "bar" }
<code>live (type,fn)</code>	Handle event using event delegation	<code>\$.grep (array, func, invert)</code>	<code>\$.map (array, func)</code>
<code>die (type,fn)</code>	Remove events added w/ live().	<code>\$.grep([0,1,2], function(n,i){ return n>0; }) == [1,2]</code>	<code>\$.map([0,1,2], function(n){ return n + 4; });</code>
Effects		<code>\$.makeArray (obj) === [obj]</code>	<code>\$.merge (array, array) - removes dupes</code>
Speeds: slow / normal / fast / #ms	<code>toggle ([boolean]) / toggle (speed,fn)</code>	<code>\$.isArray (obj) / \$.isFunction (obj)</code>	<code>\$.merge([0,1,2], [2,3,4]) === [0,1,2,3,4]</code>
<code>animate (params, options)</code>	<code>animate (params, speed, easing, callback)</code>		<code>\$.inArray (value, array)</code>
<code>hide/show (speed [,callback])</code>	<code>stop (clearQueue,gotoEnd)</code>		<code>\$.inArray('y',['x','y','z']) == 1 (-1 if not found)</code>
<code>fadeIn/fadeOut (speed, callback)</code>	<code>fadeTo (speed, opacity, callback)</code>		
<code>slideDown/slideUp (speed, callback)</code>	<code>slideToggle (speed,callback)</code>		
Set <code>jQuery.fx.off=true</code> to disable all current and queued animations			