jQuery Reference Sheet

Selectors & Syntax - $ is a variable used to define/access jQuery API. (selector) is used to find and select HTML elements. The selector can find and select elements usually by CSS selectors, some of the supported selectors can be seen in the list below:

`$("*").action();`	Select and apply action to all elements
`$("p").action();`	Select all <p> tags and perform some action on each one
`$(".my_class").action();`	Select all elements that belong to my_class
`$("p.my_class").action();`	Select all <p> elements that belong to my_class
`$("#my_id").action()`	Select the element with id my_id
`$("[attribute]").action();`	Select any element with the attribute called attribute
`$("a[target='somevalue']") .action();`	Select all <a> tags with attribute target having value somevalue
`$("p:even").action()`	Select all even <p> elements
`$("p:odd").action()`	Select all odd <p> elements

DOM Traversal: You can use jQuery to traverse the elements in a document using the following methods:

`$(selector).first();`	Return the first element of the selected elements, order based the document
`$(selector).last();`	Opposite of .first(), this method returns the last element of the selected elements
`$(selector).next();`	Get the next sibling of the selected element
`$(selector).prev();`	Get the previous sibling of the selected element
`$(selector).siblings();`	Return all siblings of the selected element
`$(selector).find(selector2);`	Return all descendants of the selected element that match selector2
`$(selector).not(selector2);`	Filter out elements that match selector2
`$(selector).children();`	Return all direct children of the selected element

HTML Manipulation - jQuery allows for you to easily get, set, add, and remove content from an HTML document using the methods shown below:

Get Content	
`$(selector).text();`	Get the content of the selected element, HTML tags are not included
`$(selector).html();`	Get the the HTML contents of every matched element, this includes HTML tags, excluding the tags of the tags of the selected element
`$(selector).val();`	Get the current value of the set of matched element. Specifically, this method is used to get the value of form elements, like <input>, <select>, and <textarea>
`$(selector).attr(attrName);`	Get the value of an attribute of the selected element

HTML Manipulation Continued -

Set Content	
`$(selector).text(text);`	Clear the current content of the select element and replace it with text. Providing a string with HTML tags will escape the characters, and not create a new element. For example, calling $(selector).text("<p>A test</p>"), would set the content of the selected element to<p>A test</p>, a new paragraph will not be created.
`$(selector).html(html);`	Clear the current content of the selected element and repalce with html. HTML tags are not escaped. Thus calling $(selector).html("<p>A test</p>"), would create a new paragraph node.
`$(selector).val(val);`	Set the current value of a form element
`$(selector) .attr(attrName, value);`	Set the value of an attribute on an HTML element
Add New HTML Content	
`$(selector) .append(content [, content]);`	Add the provided content **to the end of the content** of the selected element. A special note, content can be a selector. Using a selector will move the selected element, so $(".container").append($("h2")); will grab any <h2> on the webpage and move it below the selected element
`$(selector) .prepend(content [, content]);`	Add the provided content **to the beginning of the content** of the selected element. This method is similar to append(), except it prepends to the beginning of the elements content
`$(selector) .after(content [, content]);`	Insert content **after** the selected HTML element
`$(selector) .before(content [, content]);`	Insert content **before** the selected HTML element
Remove HTML Content	
`$(selector).empty();`	Removes all descendants and text within the set of matched elements, leaving the matched element empty but still present
`$(selector).remove([selector2]);`	Similar to empty, but also removes the selected element itself, leaving no traces of the element. The optional argument allows us to filter which elements of the selected subset are removed. Anything that matches selector2 is removed.
`$(selector).detach([selector2]);`	Same as .remove() but all the data of removed elements is returned by the function

CSS Manipulation - jQuery has the ability to manipulate style properties. This is very helpful when the programmer wants to make their code visually dynamic when the user interacts with it. There are 5 methods that are commonly used for these purposes:

CSS Manipulation Methods:

`$("selector").css("property", "value");`	Sets a CSS property to selector
`$("selector").css("property");`	Gets the CSS property of selector
`$("selector").addClass("classname");`	Adds one or more classes to the selected element
`$("selector").removeClass("classname");`	Removes one or more classes from an element
`$("selector").hasClass("classname");`	Returns whether or not an element has class "classname"
`$("selector").toggleClass("classname");`	This method will switch between classes each time it is called

Event Handling - jQuery provides simplified methods for handling events on a webpage compared to base JavaScript. Listed below are some of the most commonly used methods for event handling, and following that are some common event types.

Event Handling Methods:

`$(selector).on(events[,selector] [,data], handler);`	The current jQuery standard for event binding to an element.
`$(selector).one(events[,selector] [,data], handler);`	Binds an event to an element that can only be activated one time. Useful for setup events that only need to be run once
`$(selector).off(events[,selector] [,data]);`	Removes any event handlers attached using .on()
`$(selector).trigger(eventType [,extraParameters]);`	Takes an event type as a parameter and triggers that event on the element the method is being called on
`event .stopPropagation();`	Keeps the occurring event from propagating up the DOM and triggering event handlers of ancestors
`event .preventDefault();`	Prevents the browser default action of the event (eg a link being clicked)
`event.pageX;`	Returns the number of pixels the cursor is from the left side of the screen
`event.pageY;`	Returns the number of pixels the cursor is from the top side of the screen
`event.target;`	Returns the DOM element that triggered the target event
`event.which;`	Returns the specific user input that triggered the event (mouse click, keyboard press, etc)

Common Event Types:

`click`	Triggers upon an element being primary(left) clicked
`dbclick`	Triggers upon an element being primary(left) clicked twice in quick succession
`mousedown`	Triggers upon the mouse button being pressed down on an element
`mouseup`	Triggers upon a mouse button being released
`mouseover`	Triggers upon the user hovering the mouse over an element
`mouseout`	Triggers upon the user's mouse leaving an element
`mousemove`	Triggers upon any movement of the mouse on the window
`keypress`	Triggers upon a key on the keyboard being pressed and released
`keydown`	Triggers upon the initial downpress of a key on the keyboard
`keyup`	Triggers upon the release of a key on the keyboard
`load`	Triggers upon the webpage finishing loading
`unload`	Triggers upon a user leaving the webpage/the page unloading
`error`	Triggers upon the browser receiving a JavaScript error
`resize`	Triggers upon the browser's window size being adjusted
`scroll`	Triggers upon the user scrolling up or down on the window

Animations - The jQuery library provides several techniques for adding animation to a web page. These include simple, standard animations that are frequently used, and the ability to craft sophisticated custom effects. This is done with the .animate method:

```
.animate( properties [, duration ] [, easing ] [, complete ] )

.animate( properties, options )
```

The only required parameter is properties, a plain object of CSS properties. CSS properties can also be assigned values 'show', 'hide' or 'toggle'. For example, if the height attribute is set to 'toggle', and the object is originally visible, then the animation will decrease the height to 0. The second function signature differs from the first in that all options are passed to .animate() using a map/object. The second function signature has more flexibility in terms of the options we can apply with 10 options listed on jQuery API.

Effects - jQuery provides several easy to use effects that can be attached to elements to make a page come alive. Shown below are the methods to apply these effects to elements in your HTML document.

`$(selector).fadeIn( [duration][, callback]);`	Fades an object into view, and completes an optional callback function upon completion
`$(selector).fadeOut([ duration][, callback]);`	Similar to fadeIn, but object is faded out of view
`$(selector).fadeTo( duration, opacity[, callback]);`	Similar to functions above, but fades an object to a specified opacity
`$(selector).slideDown( [duration][, callback]);`	Slides an object down into view
`$(selector).slideUp( [duration][, callback]);`	Slides an object up into view
`$(selector).hide( [duration][, callback]);`	Prevents the browser default action of the event (eg a link being clicked)

Animation and Effects Queue - There is also functionality to queuing up sequences of animations and effect methods. Each element with an animation has a default queue attached to it called 'fx'. This queue allows a sequence of actions to be called on an element asynchronously. Below is a list of useful queue methods.

`$(selector).queue( [queueName]);`	Show the queue of functions/animations to be executed on an element. An element's queue name is "fx" by default
`$(selector).queue( [queueName], newQueue);`	Manipulate the queue of functions to be executed on an element. An elements queue name is "fx" by default. New functions can be appended to the queue to add to the queued animation
`$(selector).clearQueue();`	Removes all queued animation items
`$(selector).dequeue();`	The next function is removed from the queue and executed. All functions on queue should also call dequeue to continue running the sequences
`$(selector).stop();`	Stops a currently running animation on an object

AJAX (Asynchronous Javascript And XML) **-** AJAX is mainly used for exchanging data with a server and updating parts of a web page without needing to reload the page. AJAX is used to load data in the background to be displayed without needing a reload, and uses XML, plain text, or JSON text. With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post. You can load the external data directly into the selected HTML elements of your web page.

Basic AJAX Steps:

1. An event occurs on a web page, anything from a button to a page being loaded
2. JavaScript creates a JSON HTTP Request object
3. The request is sent to a web server
4. The request is processed by the server
5. A response is sent by the server to the web page
6. The response is read by JavaScript
7. A background page update occurs

AJAX Methods:

`$(selector).load(URL [, data][, callback]);`	Loads data from a server and puts the returned data into the selected element
`$(selector).get(URL, callback);`	Requests data from a server with an HTTP GET request
`$(selector).post(URL [, data][, callback]);`	Requests data from a server using an HTTP POST request

jQuery Utility Methods - jQuery utility methods are a miscellaneous collection of general purpose methods. Below a detailed table of methods, description, and the parameters is listed.

Method Name	Description	Parameter
`$(selector).clearQueue();`	Removes items from queue that have not been executed	
`$(selector).dequeue();`	Removes the next function on the queue and executes it	
`$(selector).contains( container, contained);`	Checks to see if a DOM element is a descendant of another DOM element, returns a boolean.	container: DOM element that may contain the other element contained: DOM element that may be a descendant of the container
`$(selector).data( element, key, value);`	Stores data associated with the element, returns value that was set	element: DOM element to associate with the data key: String naming the data to set value: New data value (cannot be undefined)
`$.(selector)each(array, callback);`	A generic iterator function that can be used to iterative over objects and arrays	array: Array or array-like object to iterate over callback: Function that will be executed on every value
`$(selector).extend( target, object1, [objectN]);`	Merges the contents of two or more objects together into the first object	target: Object that will receive the new properties object1: Object containing additional properties to merge in objectN: Additional objects containing properties to merge in
`$(selector).fn.extend(object);`	Merges the contents of an object onto the jQuery prototype to provide new jQuery instance methods	object: an object to merge onto the jQuery prototype
`$(selector).globalEval(code);`	Executes some JavaScript code globally	code: JavaScript code to execute
`$(selector).grep( array, function[, invert]);`	Finds the elements of an array which satisfy a filter function (original array is unaffected)	array: Array-like object to search through function: Function to process each item against (should take the item as first argument and index as the second argument and return a boolean) invert: Boolean type. If false, the function returns an array consisting of elements for which callback returns true. If true, returns array of elements for which callback returns false.
`$(selector).inArray( value, array[, fromIndex]);`	Searches for a specified value in an array and returns its index (or -1 if not found)	value: Value to search for array: Array to search fromIndex: Index of array from which to begin searching (default is 0)
`$(selector).isEmptyObject(object);`	Checks to see if an object is empty, returns a boolean	object: The object to be checked
`$(selector).isPlainObject(object);`	Check to see if an object is a plain object (created using {} or new Object)	object: The object that will be tested
`$(selector).isXMLDoc(node);`	Checks to see if a DOM node is within an XML document (or is an XML document)	node: DOM node that will be tested
`$(selector).makeArray(object);`	Converts an array-like object into a true JavaScript array	object: Any object to turn into a native Array
`$(selector).map(array, callback);`	Translates all items in an array or object to new array of items	array: The array to translate callback: Function to process each item against (should take the value as first argument and the key of the object property as the second argument and returns any value to add to the array)
`$(selector).noop();`	An empty function	
`$(selector).parseHTML( data[, context][, keepScripts]);`	Parses a string into an array of DOM nodes	data: HTML string to be parsed context: Element to serve as the context in which the HTML fragment will be created (defaults to document) keepScripts: Boolean indicating whether to include scripts passed in the HTML string (defaults to false)
`$(selector).parseXML(data);`	Parses a string into an XML document	data: XML string to be parsed
`$(selector).queue( element[, queueName]);`	Show the queue of functions to be executed on the matched element	element: DOM element to inspect for an attached queue queueName: String containing name of queue (defaults to fx)
`$(selector).removeData( element[, name]);`	Remove a previously-stored piece of data	element: DOM element from which to remove data name: String naming the piece of data to remove
`$(selector).uniqueSort(arr);`	Sorts an array of DOM elements, in place, with the duplicates removed. Note that this only works on arrays of DOM elements, not strings or numbers	arr: Array of DOM elements