D
Quick Reference

This appendix is intended to be a quick reference for the jQuery API, including selector expressions and methods. A more detailed discussion on this topic is available in this book's companion volume, *jQuery Reference Guide*, and on the jQuery documentation site, `http://docs.jquery.com`.

Selector expressions

The jQuery factory function `$()` is used to find elements on the page to work with. This function takes a string composed of CSS-like syntax, called a **selector expression**. Selector expressions are discussed in detail in Chapter 2.

Selector	Matches
`*`	All elements.
`#id`	The element with the given ID.
`element`	All elements of the given type.
`.class`	All elements with the given class.
`a, b`	Elements that are matched by `a` or `b`.
`a b`	Elements `b` that are descendants of `a`.
`a > b`	Elements `b` that are children of `a`.
`a + b`	Elements `b` that immediately follow `a`.
`a ~ b`	Elements `b` that are siblings of `a`.
`:first`	The first element in the result set.
`:last`	The last element in the result set.
`:not(a)`	All elements in the result set that are not matched by `a`.
`:even`	Even elements in the result set (0-based).
`:odd`	Odd elements in the result set (0-based).

Selector	Matches
`:eq(index)`	A numbered element in the result set (0-based).
`:gt(index)`	All elements in the result set after (greater than) the given index (0-based).
`:lt(index)`	All elements in the result set before (less than) the given index (0-based).
`:header`	Header elements (e.g. `<h1>`, `<h2>`).
`:animated`	Elements with an animation in progress.
`:contains(text)`	Elements containing the given text.
`:empty`	Elements with no child nodes.
`:has(a)`	Elements containing a descendant element matching `a`.
`:parent`	Elements that have child nodes.
`:hidden`	Elements that are hidden, either through CSS or because they are `<input type="hidden" />`.
`:visible`	The inverse of `:hidden`.
`[attr]`	Elements that have the attribute `attr`.
`[attr=value]`	Elements whose `attr` attribute is `value`.
`[attr!=value]`	Elements whose `attr` attribute is not `value`.
`[attr^=value]`	Elements whose `attr` attribute begins with `value`.
`[attr$=value]`	Elements whose `attr` attribute ends with `value`.
`[attr*=value]`	Elements whose `attr` attribute contains the substring `value`.
`:nth-child(index)`	Elements which are the `index`th child of their parent element (1-based).
`:nth-child(even)`	Elements which are an even child of their parent element (1-based).
`:nth-child(odd)`	Elements which are an odd child of their parent element (1-based).
`:nth-child(formula)`	Elements which are the `n`th child of their parent element (1-based). Formulas are of the form `an+b` for integers `a` and `b`.
`:first-child`	Elements which are the first child of their parent.
`:last-child`	Elements which are the last child of their parent.
`:only-child`	Elements which are the only child of their parent.
`:input`	All `<input>`, `<select>`, `<textarea>`, and `<button>` elements.
`:text`	`<input>` elements with `type="text"`.
`:password`	`<input>` elements with `type="password"`.
`:radio`	`<input>` elements with `type="radio"`.

Selector	Matches
`:checkbox`	`<input>` elements with `type="checkbox"`.
`:submit`	`<input>` elements with `type="submit"`.
`:image`	`<input>` elements with `type="image"`.
`:reset`	`<input>` elements with `type="reset"`.
`:button`	`<input>` elements with `type="button"`, and `<button>` elements.
`:file`	`<input>` elements with `type="file"`.
`:enabled`	Enabled form elements.
`:disabled`	Disabled form elements.
`:checked`	Checked checkboxes and radio buttons.
`:selected`	Selected `<option>` elements.

DOM traversal methods

After creating a jQuery object using `$()`, we can alter the set of matched elements we are working with by calling one of these **DOM traversal methods**. DOM traversal methods are discussed in detail in Chapter 2.

Traversal Method	Returns a jQuery object containing...
`.filter(selector)`	Selected elements that match the given selector.
`.filter(callback)`	Selected elements for which the `callback` function returns `true`.
`.eq(index)`	The selected element at the given 0-based index.
`.slice(start, [end])`	Selected elements in the given range of 0-based indices.
`.not(selector)`	Selected elements that do not match the given selector.
`.add(selector)`	Selected elements, plus any additional elements that match the given selector.
`.find(selector)`	Descendant elements that match the selector.
`.contents()`	Child nodes (including text nodes).
`.children([selector])`	Child nodes, optionally filtered by a selector.
`.next([selector])`	The sibling immediately following each selected element, optionally filtered by a selector.
`.nextAll([selector])`	All siblings following each selected element, optionally filtered by a selector.
`.prev([selector])`	The sibling immediately preceding each selected element, optionally filtered by a selector.

Traversal Method	Returns a jQuery object containing...
`.prevAll([selector])`	All siblings preceding each selected element, optionally filtered by a selector.
`.siblings([selector])`	All siblings, optionally filtered by a selector.
`.parent([selector])`	The parent of each selected element, optionally filtered by a selector.
`.parents([selector])`	All ancestors, optionally filtered by a selector.
`.closest selector`	The first element that matches the selector, starting at the selected element and moving up through its ancestors in the DOM tree.
`.offsetParent()`	The positioned parent (e.g. `relative`, `absolute`) of the **first** selected element.
`.andSelf()`	The selected elements, plus the previous set of selected elements on the internal jQuery stack.
`.end()`	The previous set of selected elements on the internal jQuery stack.
`.map(callback)`	The result of the `callback` function when called on each selected element.

Event methods

To react to user behavior, we need to register our handlers using these **event methods**. Note that many DOM events only apply to certain element types; these subtleties are not covered here. Event methods are discussed in detail in Chapter 3.

Event Method	Description
`.ready(handler)`	Bind `handler` to be called when the DOM and CSS are fully loaded.
`.bind(type, [data], handler)`	Bind `handler` to be called when the given type of event is sent to the element.
`.one(type, [data], handler)`	Bind `handler` to be called when the given type of event is sent to the element. Removes the binding when the handler is called.
`.unbind([type], [handler])`	Removes the bindings on the element (for an event type, a particular handler, or all bindings).
`.live(type, handler)`	Bind `handler` to be called when the given type of event is sent to the element, using event delegation.
`.die(type, [handler])`	Removes the bindings on the element previously bound with `.live()`.

Event Method	Description
`.blur(handler)`	Bind `handler` to be called when the element loses keyboard focus.
`.change(handler)`	Bind `handler` to be called when the element's value changes.
`.click(handler)`	Bind `handler` to be called when the element is clicked.
`.dblclick(handler)`	Bind `handler` to be called when the element is double-clicked.
`.error(handler)`	Bind `handler` to be called when the element receives an `error` event (browser-dependent).
`.focus(handler)`	Bind `handler` to be called when the element gains keyboard focus.
`.keydown(handler)`	Bind `handler` to be called when a key is pressed and the element has keyboard focus.
`.keypress(handler)`	Bind `handler` to be called when a keystroke occurs and the element has keyboard focus.
`.keyup(handler)`	Bind `handler` to be called when a key is released and the element has keyboard focus.
`.load(handler)`	Bind `handler` to be called when the element finishes loading.
`.mousedown(handler)`	Bind `handler` to be called when the mouse button is pressed within the element.
`.mouseenter(handler)`	Bind `handler` to be called when the mouse pointer enters the element. Not affected by event bubbling.
`.mouseleave(handler)`	Bind `handler` to be called when the mouse pointer leaves the element. Not affected by event bubbling.
`.mousemove(handler)`	Bind `handler` to be called when the mouse pointer moves within the element.
`.mouseout(handler)`	Bind `handler` to be called when the mouse pointer leaves the element.
`.mouseover(handler)`	Bind `handler` to be called when the mouse pointer enters the element.
`.mouseup(handler)`	Bind `handler` to be called when the mouse button is released within the element.
`.resize(handler)`	Bind `handler` to be called when the element is resized.
`.scroll(handler)`	Bind `handler` to be called when the element's scroll position changes.
`.select(handler)`	Bind `handler` to be called when text in the element is selected.

Event Method	Description
`.submit(handler)`	Bind `handler` to be called when the form element is submitted.
`.unload(handler)`	Bind `handler` to be called when the element is unloaded from memory.
`.hover(enter, leave)`	Bind `enter` to be called when the mouse enters the element, and `leave` to be called when the mouse leaves it.
`.toggle(handler1, handler2, ...)`	Bind `handler1` to be called when the mouse is clicked on the element, followed by `handler2` and so on for subsequent clicks.
`.trigger(type, [data])`	Trigger handlers for the event on the element, and execute the default action for the event.
`.triggerHandler(type, [data])`	Trigger handlers for the event on the element without executing any default actions.
`.blur()`	Trigger the `blur` event.
`.change()`	Trigger the `change` event.
`.click()`	Trigger the `click` event.
`.dblclick()`	Trigger the `dblclick` event.
`.error()`	Trigger the `error` event.
`.focus()`	Trigger the `focus` event.
`.keydown()`	Trigger the `keydown` event.
`.keypress()`	Trigger the `keypress` event.
`.keyup()`	Trigger the `keyup` event.
`.select()`	Trigger the `select` event.
`.submit()`	Trigger the `submit` event.

Effect methods

These **effect methods** may be used to perform animations on DOM elements. Effect methods are discussed in detail in Chapter 4.

Effect Method	Description
`.show()`	Display the matched elements.
`.hide()`	Hide the matched elements.
`.show(speed, [callback])`	Display the matched elements by animating height, width, and opacity.
`.hide(speed, [callback])`	Hide the matched elements by animating height, width, and opacity.

Effect Method	Description
`.toggle([speed], [callback])`	Display or hide the matched elements.
`.slideDown([speed], [callback])`	Display the matched elements with a sliding motion.
`.slideUp([speed], [callback])`	Hide the matched elements with a sliding motion.
`.slideToggle([speed], [callback])`	Display or hides the matched elements with a sliding motion.
`.fadeIn([speed], [callback])`	Display the matched elements by fading them to opaque.
`.fadeOut([speed], [callback])`	Hide the matched elements by fading them to transparent.
`.fadeTo(speed, opacity, [callback])`	Adjust the opacity of the matched elements.
`.animate(attributes, [speed], [easing], [callback])`	Perform a custom animation of the specified CSS attributes.
`.animate(attributes, options)`	A lower-level interface to `.animate()`, allowing control over the animation queue.
`.stop([clearQueue], [jumpToEnd])`	Stop the currently running animation, then start queued animations, if any.
`.queue()`	Retrieve the queue of animations on the first matched element.
`.queue(callback)`	Add `callback` to the end of the queue.
`.queue(newQueue)`	Replace the queue with a new one.
`.dequeue()`	Execute the next animation on the queue.

DOM manipulation methods

DOM manipulation methods are discussed in detail in Chapter 5.

Method	Description
`.attr(key)`	Get the attribute named `key`.
`.attr(key, value)`	Set the attribute named `key` to `value`.
`.attr(key, fn)`	Set the attribute named `key` to the result of `fn` (called separately on each matched element).
`.attr(map)`	Set attribute values, given as key-value pairs.
`.removeAttr(key)`	Remove the attribute named `key`.

Method	Description
`.addClass(class)`	Add the given class to each matched element.
`.removeClass(class)`	Remove the given class from each matched element.
`.toggleClass(class)`	Remove the given class if present, and adds it if not, for each matched element.
`.hasClass(class)`	Return `true` if any of the matched elements has the given class.
`.html()`	Get the HTML content of the first matched element.
`.html(value)`	Set the HTML content of each matched element to `value`.
`.text()`	Get the textual content of all matched elements as a single string.
`.text(value)`	Set the textual content of each matched element to `value`.
`.val()`	Get the value attribute of the first matched element.
`.val(value)`	Set the value attribute of each element to `value`.
`.css(key)`	Get the CSS attribute named `key`.
`.css(key, value)`	Set the CSS attribute named `key` to `value`.
`.css(map)`	Set CSS attribute values, given as key-value pairs.
`.offset()`	Get the top, and left, pixel coordinates of the first matched element, relative to the viewport.
`.position()`	Get the top, and left, pixel coordinates of the first matched element, relative to the element returned by `.offsetParent()`.
`.scrollTop()`	Get the vertical scroll position of the first matched element.
`.scrollTop(value)`	Set the vertical scroll position of all matched elements to `value`.
`.scrollLeft()`	Get the horizontal scroll position of the first matched element.
`.scrollLeft(value)`	Set the horizontal scroll position of all matched elements to `value`.
`.height()`	Get the height of the first matched element.
`.height(value)`	Set the height of all matched elements to `value`.
`.width()`	Get the width of the first matched element.
`.width(value)`	Set the width of all matched elements to `value`.
`.innerHeight()`	Get the height of the first matched element, including padding, but not border.

Method	Description
`.innerWidth()`	Get the width of the first matched element, including padding, but not border.
`.outerHeight(includeMargin)`	Get the height of the first matched element, including padding, border, and optional margin.
`.outerWidth(includeMargin)`	Get the width of the first matched element, including padding, border, and optional margin.
`.append(content)`	Insert `content` at the end of the interior of each matched element.
`.appendTo(selector)`	Insert the matched elements at the end of the interior of the elements matched by `selector`.
`.prepend(content)`	Insert `content` at the beginning of the interior of each matched element.
`.prependTo(selector)`	Insert the matched elements at the beginning of the interior of the elements matched by `selector`.
`.after(content)`	Insert `content` after each matched element.
`.insertAfter(selector)`	Insert the matched elements after each of the elements matched by `selector`.
`.before(content)`	Insert `content` before each matched element.
`.insertBefore(selector)`	Insert the matched elements before each of the elements matched by `selector`.
`.wrap(content)`	Wrap each of the matched elements within `content`.
`.wrapAll(content)`	Wrap all of the matched elements as a single unit within `content`.
`.wrapInner(content)`	Wrap the interior contents of each of the matched elements within `content`.
`.replaceWith(content)`	Replace the matched elements with `content`.
`.replaceAll(selector)`	Replace the elements matched by `selector` with the matched elements.
`.empty()`	Remove the child nodes of each matched element.
`.remove([selector])`	Remove the matched nodes (optionally filtered by `selector`) from the DOM.
`.clone([withHandlers])`	Make a copy of all matched elements, optionally also copying event handlers.
`.data(key)`	Get the data item named `key` associated with the first matched element.
`.data(key, value)`	Set the data item named `key` associated with each matched element to `value`.
`.removeData(key)`	Remove the data item named `key` associated with each matched element.

AJAX methods

We can retrieve information from the server without requiring a page refresh by calling one of these **AJAX methods**. AJAX methods are discussed in detail in Chapter 6.

AJAX Method	Description
`$.ajax(options)`	Make an AJAX request using the provided set of options. This is a low-level method that is usually called via other convenience methods.
`.load(url, [data], [callback])`	Make an AJAX request to `url`, and place the response into the matched elements.
`$.get(url, [data], [callback], [returnType])`	Make an AJAX request to `url` using the GET method.
`$.getJSON(url, [data], [callback])`	Make an AJAX request to `url`, interpreting the response as a JSON data structure.
`$.getScript(url, [callback])`	Make an AJAX request to `url`, executing the response as JavaScript.
`$.post(url, [data], [callback], [returnType])`	Make an AJAX request to `url` using the POST method.
`.ajaxComplete(handler)`	Bind `handler` to be called when any AJAX transaction completes.
`.ajaxError(handler)`	Bind `handler` to be called when any AJAX transaction completes with an error.
`.ajaxSend(handler)`	Bind `handler` to be called when any AJAX transaction begins.
`.ajaxStart(handler)`	Bind handler to be called when any AJAX transaction begins, and no others are active.
`.ajaxStop(handler)`	Bind `handler` to be called when any AJAX transaction ends, and no others are still active.
`.ajaxSuccess(handler)`	Bind `handler` to be called when any AJAX transaction completes successfully.
`$.ajaxSetup(options)`	Set default options for all subsequent AJAX transactions.
`.serialize()`	Encode the values of a set of form controls into a query string.
`.serializeArray()`	Encode the values of a set of form controls into a JSON data structure.
`$.param(map)`	Encode an arbitrary map of values into a query string.

Miscellaneous methods

These utility methods do not fit neatly into the above categories, but are often very useful when writing scripts using jQuery.

Method or Property	Description
`$.support`	Return a map of properties indicating whether the browser supports various features and standards
`$.each(collection, callback)`	Iterate over `collection`, executing `callback` for each item.
`$.extend(target, addition, ...)`	Modify the object `target` by adding properties from the other supplied objects.
`$.grep(array, callback, [invert])`	Filter `array` by using `callback` as a test.
`$.makeArray(object)`	Convert `object` into an array.
`$.map(array, callback)`	Construct a new array consisting of the result of `callback` being called on each item.
`$.inArray(value, array)`	Determine whether `value` is in `array`.
`$.merge(array1, array2)`	Combine the contents of `array1` and `array2`.
`$.unique(array)`	Remove any duplicate DOM elements from `array`.
`$.isFunction(object)`	Determine whether `object` is a function.
`$.trim(string)`	Remove whitespace from the ends of `string`.
`$.noConflict([extreme])`	Revert `$` to its pre-jQuery definition.
`.hasClass(className)`	Determine whether any matched element has the given class.
`.is(selector)`	Determine whether any matched element is matched by the given selector expression.
`.each(callback)`	Iterate over the matched elements, executing `callback` for each element.
`.length`	Get the number of matched elements.
`.get()`	Get an array of DOM nodes corresponding to the matched elements.
`.get(index)`	Get the DOM node corresponding to the matched element at the given index.
`.index(element)`	Get the index of the given DOM node within the set of matched elements.