jQUERY 1.7
VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0–1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

✲ CORE

jQuery.holdReady(hold) *u*

Holds or releases the execution of jQuery's ready event.

jQuery() *jQ*

Accepts a string containing a CSS selector which is then used to match a set of elements.

jQuery(selector [, context]) - jQuery(element)
jQuery(object) - jQuery(elementArray)
jQuery(jQuery object)

jQuery.noConflict([removeAll]) *Obj*

Relinquish jQuery's control of the $ variable.

jQuery.sub() *jQ*

Creates a new copy of jQuery whose properties and methods can be modified without affecting the original jQuery object.

jQuery.when(deferreds) *pr*

Provides a way to execute callback functions based on one or more objects, usually Deferred objects that represent asynchronous events.

✲ SELECTORS

All Selector ("*")

Selects all elements.

:animated Selector

Select all elements that are in the progress of an animation at the time the selector is run.

Attribute Contains Prefix Selector [name|="value"]

Selects elements that have the specified attribute with a value either equal to a given string or starting with that string followed by a hyphen (-).

Attribute Contains Selector [name*="value"]

Selects elements that have the specified attribute with a value containing the a given substring.

Attribute Contains Word Selector [name~="value"]

Selects elements that have the specified attribute with a value containing a given word, delimited by spaces.

Attribute Ends With Selector [name $="value"]

Selects elements that have the specified attribute with a value ending exactly with a given string. The comparison is case sensitive.

Attribute Equals Selector [name="value"]

Selects elements that have the specified attribute with a value exactly equal to a certain value.

Attribute Not Equal Selector [name!="value"]

Select elements that either don't have the specified attribute, or do have the specified attribute but not with a certain value.

Attribute Starts With Selector [name^="value"]

Selects elements that have the specified attribute with a value beginning exactly with a given string.

:button Selector

Selects all button elements and elements of type button.

:checkbox Selector

Selects all elements of type checkbox.

:checked Selector

Matches all elements that are checked.

Child Selector ("parent > child")

Selects all direct child elements specified by "child" of elements specified by "parent".

Class Selector (".class")

Selects all elements with the given class.

:contains() Selector

Select all elements that contain the specified text.

Descendant Selector ("ancestor descendant")

Selects all elements that are descendants of a given ancestor.

:disabled Selector

Selects all elements that are disabled.

Element Selector ("element")

Selects all elements with the given tag name.

:empty Selector

Select all elements that have no children (including text nodes).

:enabled Selector

Selects all elements that are enabled.

:eq() Selector

Select the element at index n within the matched set.

:even Selector

Selects even elements, zero-indexed.

:file Selector

Selects all elements of type file.

:first-child Selector

Selects all elements that are the first child of their parent.

:first Selector

Selects the first matched element.

:focus selector

Selects element if it is currently focused.

:gt() Selector

Select all elements at an index greater than index within the matched set.

Has Attribute Selector [name]

Selects elements that have the specified attribute, with any value.

:has() Selector

Selects elements which contain at least one element that matches the specified selector.

:header Selector

Selects all elements that are headers, like h1, h2, h3 and so on.

:hidden Selector

Selects all elements that are hidden.

ID Selector ("#id")

Selects a single element with the given id attribute.

:image Selector

Selects all elements of type image.

:input Selector

Selects all input, textarea, select and button elements.

:last-child Selector

Selects all elements that are the last child of their parent.

:last Selector

Selects the last matched element.

:lt() Selector

Select all elements at an index less than index within the matched set.

Multiple Attribute Selector [name="value"][name2="value2"]

Matches elements that match all of the specified attribute filters.

Multiple Selector ("selector1, selector2, selectorN")

Selects the combined results of all the specified selectors.

Next Adjacent Selector ("prev + next")

Selects all next elements matching "next" that are immediately preceded by a sibling "prev".

Next Siblings Selector ("prev ~ siblings")

Selects all sibling elements that follow after the "prev" element, have the same parent, and match the filtering "siblings" selector.

:not() Selector

Selects all elements that do not match the given selector.

:nth-child() Selector

Selects all elements that are the nth-child of their parent.

:odd Selector

Selects odd elements, zero-indexed.

:only-child Selector

Selects all elements that are the only child of their parent.

:parent Selector

Select all elements that are the parent of another element, including text nodes.

:password Selector

Selects all elements of type password.

:radio Selector

Selects all elements of type password.

:reset Selector

Selects all elements of type reset.

jQUERY 1.7
VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0–1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

:selected Selector
Selects all elements that are selected.

:submit Selector
Selects all elements of type submit.

:text Selector
Selects all elements of type text.

:visible Selector
Selects all elements that are visible.

❊ ATTRIBUTES

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

.addClass(className) - .addClass(function(index, currentClass))

.attr() *Str*
Get the value of an attribute for the first element in the set of matched elements.

.attr(attributeName) - .attr(attributeName, value)

.hasClass(className) *Str*
Determine whether any of the matched elements are assigned the given class.

.html() *Str* *jQ*
Get the HTML contents of the first element in the set of matched elements.

.html(htmlString) - .html(function(index, oldhtml))

.prop() *Str* *jQ*
Get the value of a property for the first element in the set of matched elements.

.prop(propertyName) - .prop(propertyName, value)

★ .removeAttr(attributeName) *jQ*
Remove an attribute from each element in the set of matched elements.

.removeClass() *jQ*
Remove a single class, multiple classes, or all classes from each element in the set of matched elements.

.removeClass([className])
.removeClass(function(index, class))

.removeProp(propertyName) *jQ*
Remove a property for the set of matched elements.

.toggleClass(className) *jQ*
Add or remove one or more classes from each element in the set of matched elements, depending on either the class's presence or the value of the switch argument.

.val() *Str* *N* *aO*
Get the current value of the first element in the set of matched elements.

.val(value) - .val(function(index, value)) *jQ*

❊ TRAVERSING

.add() *jQ*
Add elements to the set of matched elements.

.add(selector) - .add(elements) - .add(html)
.add(jQuery object) - .add(selector, context)

.andSelf() *jQ*
Add the previous set of elements on the stack to the current set.

.children([selector]) *jQ*
Get the children of each element in the set of matched elements, optionally filtered by a selector.

.closest() *jQ* *aO*
Get the first element that matches the selector, beginning at the current element and progressing up through the DOM tree.

.closest(selector) - .closest(selector [, context])
.closest(jQuery object) - .closest(element)
.closest(selectors [, context])

.contents() *jQ*
Get the children of each element in the set of matched elements, including text and comment nodes.

.each(function(index, Element)) *jQ*
Iterate over a jQuery object, executing a function for each matched element.

.end() *jQ*
End the most recent filtering operation in the current chain and return the set of matched elements to its previous state.

.eq(index) *jQ*
Reduce the set of matched elements to the one at the specified index.

.filter() *jQ*
Reduce the set of matched elements to those that match the selector or pass the function's test.

.filter(selector) - .filter(function(index))
.filter(element) - .filter(jQuery object)

.find(selector) *jQ*
Get the descendants of each element in the current set of matched elements, filtered by a selector, jQuery object, or element.

.find(selector) - .find(jQuery object) - .find(element)

.first() *jQ*
Reduce the set of matched elements to the first in the set.

.has() *jQ*
Reduce the set of matched elements to those that have a descendant that matches the selector or DOM element.

.has(selector) - .has(contained)

★ .is() *0-1*
Check the current matched set of elements against a selector, element, or jQuery object and return true if at least one of these elements matches the given arguments.

.is(selector) - .is(function(index)) - .is(jQuery object)
.is(element)

.last() *jQ*
Reduce the set of matched elements to the final one in the set.

.map(callback(index, domElement)) *jQ*
Pass each element in the current matched set through a function, producing a new jQuery object containing the return values.

.next([selector]) *jQ*
Get the immediately following sibling of each element in the set of matched elements. If a selector is provided, it retrieves the next sibling only if it matches that selector.

.nextAll([selector]) *jQ*
Get all following siblings of each element in the set of matched elements, optionally filtered by a selector.

.nextUntil() *jQ*
Get all following siblings of each element up to but not including the element matched by the selector, DOM node, or jQuery object passed.

.nextUntil([selector] [, filter]) - .nextUntil([element] [, filter])

.not() *jQ*
Remove elements from the set of matched elements.

.not(selector) - .not(elements) - .not(function(index))

.offsetParent() *jQ*
Get the closest ancestor element that is positioned.

.parent([selector]) *jQ*
Get the parent of each element in the current set of matched elements, optionally filtered by a selector.

.parents([selector]) *jQ*
Get the ancestors of each element in the current set of matched elements, optionally filtered by a selector.

.parentsUntil() *jQ*
Get the ancestors of each element in the current set of matched elements, up to but not including the element matched by the selector, DOM node, or jQuery object.

.parentsUntil([selector] [, filter])
.parentsUntil([element] [, filter])

.prev([selector]) *jQ*
Get the immediately preceding sibling of each element in the set of matched elements, optionally filtered by a selector.

.prevAll([selector]) *jQ*
Get all preceding siblings of each element in the set of matched elements, optionally filtered by a selector.

.prevUntil() *jQ*
Get all preceding siblings of each element up to but not including the element matched by the selector, DOM node, or jQuery object.

.prevUntil([selector] [, filter])
.prevUntil([element] [, filter])

.siblings([selector]) *jQ*
Get the siblings of each element in the set of matched elements, optionally filtered by a selector.

.slice(start [, end]) *jQ*
Reduce the set of matched elements to a subset specified by a range of indices.

jQUERY 1.7
VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0-1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

✲ MANIPULATION

.addClass() *jQ*

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

.addClass(className) - .addClass(function(index, currentClass))

.after() *jQ*

Insert content, specified by the parameter, after each element in the set of matched elements.

.after(content [, content]) - .after(function(index))

.append() *jQ*

Insert content, specified by the parameter, to the end of each element in the set of matched elements.

.append(content [, content])
.append(function(index, html))

.appendTo(target) *jQ*

Insert every element in the set of matched elements to the end of the target.

.attr() *jQ*

Get the value of an attribute for the first element in the set of matched elements.

.attr(attributeName) - .attr(attributeName, value)

.before() *jQ*

Insert content, specified by the parameter, before each element in the set of matched elements.

.before(content [, content]) - .before(function)

.clone() *jQ*

Create a deep copy of the set of matched elements.

.clone([withDataAndEvents])
.clone([withDataAndEvents] [,deepWithDataAndEvents])

.css() *Str* *jQ*

Get the value of a style property for the first element in the set of matched elements.

.css(propertyName) - .css(propertyName, value)
.css(propertyName, function(index, value))
.css(map)

.detach([selector]) *jQ*

Remove the set of matched elements from the DOM.

.empty() *jQ*

Remove all child nodes of the set of matched elements from the DOM.

.hasClass(className) *Str*

Determine whether any of the matched elements are assigned the given class.

.html() *Str* *jQ*

Get the HTML contents of the first element in the set of matched elements.

.html(htmlString) - .html(function(index, oldhtml))

.innerHeight() *Int*

Get the current computed height for the first element in the set of matched elements, including padding but not border.

.innerWidth() *Int*

Get the current computed width for the first element in the set of matched elements, including padding but not border.

.insertAfter(target) *Int*

Insert every element in the set of matched elements after the target.

.insertBefore(target) *Int*

Insert every element in the set of matched elements before the target.

.offset() *Int*

Get the current coordinates of the first element in the set of matched elements, relative to the document.

.offset(coordinates) - .offset(function(index, coords))

.outerHeight([includeMargin]) *Int*

Get the current computed height for the first element in the set of matched elements, including padding, border, and optionally margin. Returns an integer (without "px") representation of the value or null if called on an empty set of elements.

.position() *Obj*

Get the current coordinates of the first element in the set of matched elements, relative to the offset parent.

.prepend(content [, content]) *jQ*

Insert content, specified by the parameter, to the beginning of each element in the set of matched elements.

.prependTo(target) *jQ*

Insert every element in the set of matched elements to the beginning of the target.

.prop() *Str* *jQ*

Get the value of a property for the first element in the set of matched elements.

.prop(propertyName) - .prop(propertyName, value)

.remove([selector]) *Str*

Remove the set of matched elements from the DOM.

.removeAttr(attributeName) *jQ*

Remove an attribute from each element in the set of matched elements.

.removeClass() *jQ*

Remove a single class, multiple classes, or all classes from each element in the set of matched elements.

.removeClass([className])
.removeClass(function(index, class))

.removeProp(propertyName) *jQ*

Remove a property for the set of matched elements.

.replaceAll(target) *jQ*

Replace each target element with the set of matched elements.

.replaceWith() *jQ*

Replace each element in the set of matched elements with the provided new content.

.replaceWith(newContent) - .replaceWith(function)

.scrollLeft() *jQ*

Get the current horizontal position of the scroll bar for the first element in the set of matched elements.

.scrollLeft(value)

.scrollTop() *jQ*

Get the current vertical position of the scroll bar for the first element in the set of matched elements.

.scrollTop value)

.text() *jQ*

Get the combined text contents of each element in the set of matched elements, including their descendants.

.text(textString) - .text(function(index, text))

.toggleClass() *jQ*

Add or remove one or more classes from each element in the set of matched elements, depending on either the class's presence or the value of the switch argument.

.toggleClass(className)
.toggleClass(className, switch)

.unwrap() *jQ*

Remove the parents of the set of matched elements from the DOM, leaving the matched elements in their place.

.val() *Str* *N* *a()* *jQ*

Get the current value of the first element in the set of matched elements.

.val(value) - .val(function(index, value))

.width() *Int*

Get the current computed width for the first element in the set of matched elements.

.width(value) - .width(function(index, width))

.wrap() *jQ*

Wrap an HTML structure around each element in the set of matched elements.

.wrap(wrappingElement) - .wrap(function(index))

.wrapAll(wrappingElement) *jQ*

Wrap an HTML structure around all elements in the set of matched elements.

.wrapInner() *jQ*

Wrap an HTML structure around the content of each element in the set of matched elements.

.wrapInner(wrappingElement)
.wrapInner(function(index))

✲ CSS

.addClass() *jQ*

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

.addClass(className) - .addClass(function(index, currentClass))

.css() *Str* *jQ*

Get the value of a style property for the first element in the set of matched elements.

.css(propertyName) - .css(propertyName, value)
.css(propertyName, function(index, value))
.css(map)

jQuery.cssHooks *Obj*

Hook directly into jQuery to override how particular CSS properties are retrieved or set, normalize CSS property naming, or create custom properties.

.hasClass(className) *Str*

Determine whether any of the matched elements are assigned the given class.

jQUERY 1.7
VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0–1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

.height() *Int*

Get the current computed height for the first element in the set of matched elements.

.height(value)
.height(function(index, height))

.innerHeight() *Int*

Get the current computed height for the first element in the set of matched elements, including padding but not border.

.innerWidth() *Int*

Get the current computed width for the first element in the set of matched elements, including padding but not border.

.offset() *Int*

Get the current coordinates of the first element in the set of matched elements, relative to the document.

.offset(coordinates) - .offset(function(index, coords))

.outerHeight([includeMargin]) *Int*

Get the current computed height for the first element in the set of matched elements, including padding, border, and optionally margin. Returns an integer (without "px") representation of the value or null if called on an empty set of elements.

.outerWidth([includeMargin]) *Int*

Get the current computed width for the first element in the set of matched elements, including padding and border.

.position() *Obj*

Get the current coordinates of the first element in the set of matched elements, relative to the offset parent.

.removeClass() *jQ*

Remove a single class, multiple classes, or all classes from each element in the set of matched elements.

.removeClass([className])
.removeClass(function(index, class))

.scrollLeft() *Int*

Get the current horizontal position of the scroll bar for the first element in the set of matched elements.

.scrollLeft(value)

.scrollTop() *Int*

Get the current vertical position of the scroll bar for the first element in the set of matched elements.

.scrollTop(value)

.toggleClass() *jQ*

Add or remove one or more classes from each element in the set of matched elements, depending on either the class's presence or the value of the switch argument.

.toggleClass(className)
.toggleClass(className, switch)

.width() *Int*

Get the current computed width for the first element in the set of matched elements.

.width(value) - .width(function(index, width))

✲ EVENTS

.bind() *jQ*

Attach a handler to an event for the elements.

.bind(eventType [, eventData], handler(eventObject))
.bind(eventType [, eventData], preventBubble)
.bind(events)

.blur() *jQ*

Bind an event handler to the "blur" JavaScript event, or trigger that event on an element.

.blur(handler(eventObject))
.blur([eventData], handler(eventObject))

.change() *jQ*

Bind an event handler to the "change" JavaScript event, or trigger that event on an element.

.change(handler(eventObject))
.change([eventData], handler(eventObject))

.click() *jQ*

Bind an event handler to the "click" JavaScript event, or trigger that event on an element.

.click(handler(eventObject))
.click([eventData], handler(eventObject))

.dblclick() *jQ*

Bind an event handler to the "dblclick" JavaScript event, or trigger that event on an element.

.dblclick(handler(eventObject))
.dblclick([eventData], handler(eventObject))

.delegate() *jQ*

Attach a handler to one or more events for all elements that match the selector, now or in the future, based on a specific set of root elements.

.delegate(selector, eventType, handler)
.delegate(selector, eventType, eventData, handler)
.delegate(selector, events)

.die() *jQ*

Remove all event handlers previously attached using .live() from the elements.

.die(eventType [, handler])
.die(eventTypes)

.error() *jQ*

Bind an event handler to the "error" JavaScript event.

.error(handler(eventObject))
.error([eventData], handler(eventObject))

event.currentTarget *jQ*

The current DOM element within the event bubbling phase.

event.data *Any*

The optional data passed to jQuery.fn.bind when the current executing handler was bound.

★ event.delegateTarget *el*

The element where the currently-called jQuery event handler was attached.

event.isDefaultPrevented() *0-1*

Returns whether event.preventDefault() was ever called on this event object.

event.isImmediatePropagationStopped() *0-1*

Returns whether event.stopImmediatePropagation() was ever called on this event object.

event.isPropagationStopped() *0-1*

Returns whether event.stopPropagation() was ever called on this event object.

event.namespace *Str*

The namespace specified when the event was triggered.

event.pageX *N*

The mouse position relative to the left edge of the document.

event.pageY *N*

The mouse position relative to the top edge of the document.

event.preventDefault() *u*

If this method is called, the default action of the event will not be triggered.

event.relatedTarget *el*

The other DOM element involved in the event, if any.

event.result *Obj*

The last value returned by an event handler that was triggered by this event, unless the value was undefined.

event.stopImmediatePropagation()

Prevents other event handlers from being called.

event.stopPropagation()

Prevents the event from bubbling up the DOM tree, preventing any parent handlers from being notified of the event.

event.target *el*

The DOM element that initiated the event.

event.timeStamp *N*

The difference in milliseconds between the time the browser created the event and January 1, 1970.

event.type *Str*

Describes the nature of the event.

event.which *N*

For key or button events, this attribute indicates the specific button or key that was pressed.

.focus() *jQ*

Bind an event handler to the "focus" JavaScript event, or trigger that event on an element

.focus(handler(eventObject))
.focus([eventData], handler(eventObject))

.focusin() *jQ*

Bind an event handler to the "focusin" event.

.focusin(handler(eventObject))
.focusin([eventData], handler(eventObject))

.focusout() *jQ*

Bind an event handler to the "focusout" JavaScript event.

.focusout(handler(eventObject))
.focusout([eventData], handler(eventObject))

.hover() *jQ*

Bind two handlers to the matched elements, to be executed when the mouse pointer enters and leaves the elements.

.hover(handlerIn(eventObject)
.handlerOut(eventObject))
.hover(handlerInOut(eventObject))

jQUERY 1.7 VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0–1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

.keydown() *jQ*

Bind an event handler to the "keydown" JavaScript event, or trigger that event on an element.

.keydown(handler(eventObject))
.keydown([eventData], handler(eventObject))

.keypress() *jQ*

Bind an event handler to the "keypress" JavaScript event, or trigger that event on an element.

.keypress(handler(eventObject))
.keypress([eventData], handler(eventObject))

.keyup() *jQ*

Bind an event handler to the "keyup" JavaScript event, or trigger that event on an element.

.keyup(handler(eventObject))
.keyup([eventData], handler(eventObject))

.live() *jQ*

Attach an event handler for all elements which match the current selector, now and in the future.

.live(events, handler) - .live(events, data, handler)
.live(events-map)

.load() *jQ*

Bind an event handler to the "load" JavaScript event.

.load(handler(eventObject))
.load([eventData], handler(eventObject))

.mousedown() *jQ*

Bind an event handler to the "mousedown" JavaScript event, or trigger that event on an element.

.mousedown(handler(eventObject))
.mousedown([eventData], handler(eventObject))

.mouseenter() *jQ*

Bind an event handler to be fired when the mouse enters an element, or trigger that handler on an element.

.mouseenter(handler(eventObject))
.mouseenter([eventData], handler(eventObject))

.mouseleave() *jQ*

Bind an event handler to be fired when the mouse leaves an element, or trigger that handler on an element.

.mouseleave(handler(eventObject)
.mouseleave([eventData], handler(eventObject))

.mousemove() *jQ*

Bind an event handler to the "mousemove" JavaScript event, or trigger that event on an element.

.mousemove(handler(eventObject))
.mousemove([eventData], handler(eventObject))

.mouseout() *jQ*

Bind an event handler to the "mouseout" JavaScript event, or trigger that event on an element.

.mouseout(handler(eventObject))
.mouseout([eventData], handler(eventObject))

.mouseover() *jQ*

Get the current computed width for the first element in the set of matched elements.

.mouseover(handler(eventObject))
.mouseover([eventData], handler(eventObject))

.mouseup() *jQ*

Bind an event handler to the "mouseup" JavaScript event, or trigger that event on an element.

.mouseup(handler(eventObject))
.mouseup([eventData], handler(eventObject))

★ .off() *jQ*

Remove an event handler.

.off(events [, selector] [, handler])
.off(events-map [, selector])

★ .on() *jQ*

Attach an event handler function for one or more events to the selected elements.

.on(events [, selector] [, data], handler)
.on(events-map [, selector] [, data])

.one() *jQ*

Attach a handler to an event for the elements. The handler is executed at most once per element.

.one(events [, data], handler)
.one(events [, selector] [, data], handler)
.one(events-map [, selector] [, data])

jQuery.proxy() *jQ*

Takes a function and returns a new one that will always have a particular context.

jQuery.proxy(function, context)
jQuery.proxy(context, name)

.ready(handler) *jQ*

Specify a function to execute when the DOM is fully loaded.

.resize() *jQ*

Bind an event handler to the "resize" JavaScript event, or trigger that event on an element.

.resize(handler(eventObject))
.resize([eventData], handler(eventObject))

.scroll() *jQ*

Bind an event handler to the "scroll" JavaScript event, or trigger that event on an element.

.scroll(handler(eventObject))
.scroll([eventData], handler(eventObject))

.select() *jQ*

Bind an event handler to the "select" JavaScript event, or trigger that event on an element.

.select(handler(eventObject))
.select([eventData], handler(eventObject))

.submit() *jQ*

Bind an event handler to the "submit" JavaScript event, or trigger that event on an element.

.submit(handler(eventObject))
.submit([eventData], handler(eventObject))

.toggle() *jQ*

Bind two or more handlers to the matched elements, to be executed on alternate clicks.

handler(eventObject), handler(eventObject) [, handler(eventObject)]

.trigger() *jQ*

Execute all handlers and behaviors attached to the matched elements for the given event type.

.trigger(eventType, extraParameters)
.trigger(event)

.triggerHandler() *jQ*

Execute all handlers attached to an element for an event.

.triggerHandler(eventType, extraParameters)

.unbind() *jQ*

Remove a previously-attached event handler from the elements.

.unbind([eventType] [, handler(eventObject)])
.unbind(eventType, false)

.undelegate() *jQ*

Remove a handler from the event for all elements which match the current selector, based upon a specific set of root elements.

.undelegate(selector, eventType)
.undelegate(selector, eventType, handler)
.undelegate(selector, events)
.undelegate(namespace)

.undelegate() *jQ*

Remove a handler from the event for all elements which match the current selector, based upon a specific set of root elements.

.undelegate(selector, eventType)
.undelegate(selector, eventType, handler)
.undelegate(selector, events)
.undelegate(namespace)

.unload() *jQ*

Bind an event handler to the "unload" JavaScript event.

.unload(handler(eventObject))
.unload([eventData], handler(eventObject))

✲ EFFECTS

.animate() *jQ*

Perform a custom animation of a set of CSS properties.

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

.clearQueue([queueName]) *jQ*

Remove from the queue all items that have not yet been run.

.delay(duration [, queueName]) *jQ*

Set a timer to delay execution of subsequent items in the queue.

.dequeue([queueName]) *jQ*

Execute the next function on the queue for the matched elements.

.fadeIn() *jQ*

Display the matched elements by fading them to opaque.

.fadeIn([duration] [, callback])
.fadeIn([duration] [, easing] [, callback])

.fadeOut() *jQ*

Hide the matched elements by fading them to transparent.

.fadeOut([duration] [, callback])
.fadeOut([duration] [, easing] [, callback])

.fadeTo() *jQ*

Adjust the opacity of the matched elements.

.fadeTo(duration, opacity [, callback])
.fadeTo(duration, opacity [, easing] [, callback])

.fadeToggle([duration] [, easing] [, callback]) *jQ*

Display or hide the matched elements by animating their opacity.

jQUERY 1.7 VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0–1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

jQuery.fx.interval — *N*

The rate (in milliseconds) at which animations fire.

jQuery.fx.off — *o-1*

Globally disable all animations.

.hide() — *jQ*

Hide the matched elements.

.hide(duration [, callback])
.hide([duration] [, easing] [, callback])

.queue() — *aO* *jQ*

Show the queue of functions to be executed on the matched elements.

.queue([queueName])
.queue([queueName] , newQueue)
.queue([queueName] , callback(next))

.show() — *jQ*

Display the matched elements.

.show(duration [, callback])
.show([duration] [, easing] [, callback])

.slideDown() — *jQ*

Display the matched elements with a sliding motion.

.slideDown([duration] [, callback])
.slideDown([duration] [, easing] [, callback])

.slideToggle() — *jQ*

Display or hide the matched elements with a sliding motion.

.slideToggle([duration] [, callback])
.slideToggle([duration] [, easing] [, callback])

.slideUp() — *jQ*

Hide the matched elements with a sliding motion.

.slideUp([duration] [, callback])
.slideUp([duration] [, easing] [, callback])

★ .stop() — *jQ*

Stop the currently-running animation on the matched elements.

.stop([clearQueue] [, jumpToEnd])
.stop([queue] [, clearQueue] [, jumpToEnd])

.toggle() — *jQ*

Display or hide the matched elements.

.toggle([duration] [, callback])
.toggle([duration] [, easing] [, callback])
.toggle(showOrHide)

✲ AJAX

jQuery.ajax() — *jqXHR*

Perform an asynchronous HTTP (Ajax) request.

jQuery.ajax(url [, settings])
jQuery.ajax(settings)

.ajaxComplete() — *jQ*

Register a handler to be called when Ajax requests complete. This is an Ajax Event.

.ajaxComplete(handler(event, XMLHttpRequest, ajaxOptions))

.ajaxError() — *jQ*

Register a handler to be called when Ajax requests complete with an error. This is an Ajax Event.

.ajaxError(handler(event, jqXHR, ajaxSettings, thrownError))

jQuery.ajaxPrefilter() — *u*

Handle custom Ajax options or modify existing options before each request is sent and before they are processed by $.ajax().

jQuery.ajaxPrefilter([dataTypes] , handler(options, originalOptions, jqXHR))

.ajaxSend() — *jQ*

Attach a function to be executed before an Ajax request is sent. This is an Ajax Event.

.ajaxSend(handler(event, jqXHR, ajaxOptions))

jQuery.ajaxSetup(options)

Set default values for future Ajax requests.

.ajaxStart(handler()) — *jQ*

Register a handler to be called when the first Ajax request begins. This is an Ajax Event.

.ajaxStop(handler()) — *jQ*

Register a handler to be called when all Ajax requests have completed. This is an Ajax Event.

.ajaxSuccess() — *jQ*

Attach a function to be executed whenever an Ajax request completes successfully. This is an Ajax Event.

.ajaxSuccess(handler(event, XMLHttpRequest, ajaxOptions))

jQuery.get() — *jqXHR*

Load data from the server using a HTTP GET request.

jQuery.get(url [, data] [, success(data, textStatus, jqXHR)] [, dataType])

jQuery.getJSON() — *jqXHR*

Load JSON-encoded data from the server using a GET HTTP request.

jQuery.getJSON(url [, data] [, success(data, textStatus, jqXHR)])

jQuery.getScript() — *jqXHR*

Load a JavaScript file from the server using a GET HTTP request, then execute it.

jQuery.getScript(url [, success(data, textStatus)])

.load() — *jqXHR*

Load data from the server and place the returned HTML into the matched element.

.load(url [, data] [, complete(responseText, textStatus, XMLHttpRequest)])

jQuery.param() — *Str*

Create a serialized representation of an array or object, suitable for use in a URL query string or Ajax request.

jQuery.param(obj)
jQuery.param(obj, traditional)

jQuery.post() — *jqXHR*

Load data from the server using a HTTP POST request.

jQuery.post(url [, data] [, success(data, textStatus, jqXHR)] [, dataType])

.serialize() — *Str*

Encode a set of form elements as a string for submission.

.serializeArray() — *aO*

Encode a set of form elements as an array of names and values.

✲ UTILITIES

jQuery.boxModel — *o-1*

Deprecated

jQuery.browser — *Map*

Deprecated

.clearQueue([queueName]) — *jQ*

Remove from the queue all items that have not yet been run.

jQuery.contains(container, contained) — *o-1*

Check to see if a DOM element is within another DOM element.

jQuery.data() — *Obj*

Store arbitrary data associated with the specified element. Returns the value that was set.

jQuery.data(element , key , value)
jQuery.data(element, key)
jQuery.data(element)

.dequeue([queueName]) — *jQ*

Execute the next function on the queue for the matched elements.

jQuery.each() — *Obj*

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

jQuery.each(collection, callback(indexInArray, valueOfElement))

jQuery.extend() — *Obj*

Merge the contents of two or more objects together into the first object.

jQuery.extend(target [, object1] [, objectN])
jQuery.extend([deep], target, object1 [, objectN])

jQuery.globalEval(code)

Execute some JavaScript code globally.

jQuery.each(collection, callback(indexInArray, valueOfElement))

jQuery.grep() — *aO*

Finds the elements of an array which satisfy a filter function. The original array is not affected.

jQuery.grep(array, function(elementOfArray, indexInArray) [, invert])

jQuery.inArray() — *N*

Search for a specified value within an array and return its index (or -1 if not found).

jQuery.inArray(value, array [, fromIndex])

jQuery.isArray(obj) — *o-1*

Determine whether the argument is an array.

jQuery.isEmptyObject(object) — *o-1*

Check to see if an object is empty (contains no properties).

jQuery.isFunction(object) — *o-1*

Determine if the argument passed is a Javascript function object.

jQUERY 1.7
VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0–1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

★ jQuery.isNumeric(object) — *o-1*

Determines whether its argument is a number.

jQuery.isPlainObject(object) — *o-1*

Check to see if an object is a plain object (created using "{}" or "new Object").

jQuery.isWindow(object) — *o-1*

Determine whether the argument is a window.

jQuery.isXMLDoc(object) — *o-1*

Check to see if a DOM node is within an XML document (or is an XML document).

jQuery.makeArray(object) — *a()*

Convert an array-like object into a true JavaScript array.

jQuery.map() — *a()*

Translate all items in an array or object to new array of items.

jQuery.map(array, callback(elementOfArray, indexInArray))
jQuery.map(arrayOrObject, callback(value, indexOrKey))

jQuery.merge(first, second) — *a()*

Merge the contents of two arrays together into the first array.

jQuery.noop() — *f(x)*

An empty function.

jQuery.now() — *N*

Return a number representing the current time.

jQuery.parseJSON(json) — *Obj*

Takes a well-formed JSON string and returns the resulting JavaScript object.

jQuery.parseXML(data) — *XMLdoc*

Parses a string into an XML document.

jQuery.proxy() — *f(x)*

Takes a function and returns a new one that will always have a particular context.

jQuery.proxy(function, context)
jQuery.proxy(context, name)

jQuery.queue() — *a() jQ*

Takes a function and returns a new one that will always have a particular context.

jQuery.queue(element [, queueName])
jQuery.queue(element , queueName , newQueue)

.queue() — *a() jQ*

Show the queue of functions to be executed on the matched elements.

.queue([queueName])
.queue([queueName] , newQueue)
.queue([queueName] , callback(next))

jQuery.removeData() — *jQ*

Remove a previously-stored piece of data

jQuery.removeData(element [, name])

jQuery.support — *obj*

A collection of properties that represent the presence of different browser features or bugs.

jQuery.trim(str) — *Str*

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

jQuery.type(obj) — *Str*

Determine the internal JavaScript [[Class]] of an object.

jQuery.unique(array) — *a()*

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.

❊ CALLBACKS OBJECT

★ jQuery.Callbacks(flag)

A multi-purpose callbacks list object that provides a powerful way to manage callback lists.

★ callbacks.add(callbacks) — *u*

Add a callback or a collection of callbacks to a callback list.

★ callbacks.disable() — *u*

Disable a callback list from doing anything more.

★ callbacks.empty() — *u*

Remove all of the callbacks from a list.

★ callbacks.fire(arguments) — *u*

Call all of the callbacks with the given arguments

★ callbacks.fired() — *o-1*

Determine if the callbacks have already been called at least once.

★ callbacks.fireWith() — *u*

Call all callbacks in a list with the given context and arguments.

callbacks.fireWith([context] [, args])

★ callbacks.has(callback) — *o-1*

Determine whether a supplied callback is in a list.

★ callbacks.lock() — *u*

Lock a callback list in its current state.

★ callbacks.locked() — *o-1*

Determine if the callbacks list has been locked.

★ callbacks.remove(callback) — *u*

Remove a callback or a collection of callbacks from a callback list.

❊ DATA & MISCELLANEOUS

.data() — *jQ*

Store arbitrary data associated with the matched elements.

data(key , value), data(key)

jQuery.dequeue() — *jQ*

Execute the next function on the queue for the matched element.

jQuery.dequeue(element [, queueName])

jQuery.hasData(element) — *o-1*

Determine whether an element has any jQuery data associated with it.

★ .removeData() — *jQ*

Remove a previously-stored piece of data.

.removeData([name]), .removeData([list])

.each() — *jQ*

Iterate over a jQuery object, executing a function for each matched element.

.each(function(index, Element))

.get([index]) — *el a()*

Retrieve the DOM elements matched by the jQuery object.

.each(function(index, Element))

.index — *Num*

Search for a given element from among the matched elements.

.index(selector), .index(element)

jQuery.noConflict([removeAll]) — *Obj*

Relinquish jQuery's control of the $ variable.

.size() — *Num*

Return the number of elements in the jQuery object.

.toArray() — *a()*

Retrieve all the DOM elements contained in the jQuery set, as an array.

❊ DEFERRED

deferred.always() — *def*

Add handlers to be called when the Deferred object is either resolved or rejected.

deferred.always(alwaysCallbacks [, alwaysCallbacks])

deferred.done() — *def*

Add handlers to be called when the Deferred object is resolved.

deferred.done(doneCallbacks [, doneCallbacks])

deferred.fail() — *def*

Add handlers to be called when the Deferred object is rejected.

deferred.fail(failCallbacks [, failCallbacks])

deferred.isRejected() — *o-1*

Determine whether a Deferred object has been rejected.

deferred.isResolved() — *o-1*

Determine whether a Deferred object has been resolved.

deferred.notify(args) — *o-1*

Call the progressCallbacks on a Deferred object with the given args.

jQUERY 1.7
VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0-1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

deferred.notifyWith() *def*

Call the progressCallbacks on a Deferred object with the given context and args.

deferred.notifyWith(context [, args])

deferred.pipe() *promise*

Utility method to filter and/or chain Deferreds.

deferred.pipe([doneFilter] [, failFilter])
deferred.pipe([doneFilter] [, failFilter] [, progressFilter])

deferred.progress() *def*

Add handlers to be called when the Deferred object generates progress notifications.

deferred.progress(progressCallbacks)

deferred.promise() *promise*

Return a Deferred's Promise object.

deferred.promise([target])

deferred.reject(args) *def*

Reject a Deferred object and call any failCallbacks with the given args.

deferred.rejectWith() *def*

Reject a Deferred object and call any failCallbacks with the given context and args.

deferred.rejectWith(context [, args])

deferred.resolve(args) *def*

Resolve a Deferred object and call any doneCallbacks with the given args.

deferred.resolveWith() *def*

Resolve a Deferred object and call any doneCallbacks with the given args.

deferred.resolveWith(context [, args])

deferred.state() *Str*

Determine the current state of a Deferred object.

deferred.then() *def*

Add handlers to be called when the Deferred object is resolved or rejected.

deferred.then(doneCallbacks, failCallbacks)
deferred.then(doneCallbacks, failCallbacks [, progressCallbacks])

.promise() *promise*

Return a Promise object to observe when all actions of a certain type bound to the collection, queued or not, have finished.

.promise([type] [, target])

jQuery.when(deferreds) *promise*

Provides a way to execute callback functions based on one or more objects, usually Deferred objects that represent asynchronous events.

❊

New or Changed in jQuery 1.7.1

WHAT'S NEW?

Aspects of the API that were changed in the corresponding version of jQuery. API changes in jQuery 1.7.0 dealt primarily with the new Event APIs: .on() and .off()

Better Support for HTML5 in IE6/7/8

jQuery.Callbacks()

Toggling Animations Work Intuitively

jQuery.Callbacks()
callbacks.add()
callbacks.disable()
callbacks.empty()
callbacks.fire()
callbacks.fired()
callbacks.fireWith()
callbacks.has()
callbacks.lock()
callbacks.locked()
callbacks.remove()
deferred.notify()
deferred.notifyWith()
deferred.pipe()
deferred.progress()
deferred.state()
deferred.then()
event.delegateTarget
.is()
jQuery.isNumeric()
.off()
.on()
.removeAttr()
.removeData()
.stop()

About jQuery

THE WRITE LESS, DO MORE, JAVASCRIPT LIBRARY

jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.

jQuery Official Web Site http://jquery.com
Download http://docs.jquery.com/Downloading_jQuery
Documentation http://docs.jquery.com/Main_Page
Tutorials http://docs.jquery.com/Tutorials
Bug Traker http://bugs.jquery.com/newticket
Discussion http://docs.jquery.com/Discussion

Credits

JQUERY 1.7 VISUAL CHEAT SHEET

This jQuery visual cheat sheet is designed by Antonio Lupetti and is distributed for free.

Update as of december 2011, jQuery current release v 1.7.1

My Blog http://woorkup.com
Twitter @woork - http://twitter.com/woork
Facebook http://facebook.com/antoniolupetti
Google+ https://plus.google.com/u/0/112930363515083491757