**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.

API Reference

* jQuery Core
* Selectors
* Attributes
* Traversing
* Manipulation
* CSS
* Events
* Effects
* Ajax
* Utilities
* jQuery UI

**Core**

* [**jQuery.holdReady()**](http://api.jquery.com/jQuery.holdReady/)

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

* [**jQuery()**](http://api.jquery.com/jQuery/)

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

* [**jQuery.noConflict()**](http://api.jquery.com/jQuery.noConflict/)

[Setup Methods](http://api.jquery.com/category/miscellaneous/setup-methods/)

Relinquish jQuery's control of the $ variable.

* [**jQuery.sub()**](http://api.jquery.com/jQuery.sub/)

[Deprecated](http://api.jquery.com/category/deprecated/)

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

* [**jQuery.when()**](http://api.jquery.com/jQuery.when/)

[Deferred Object](http://api.jquery.com/category/deferred-object/)

Provides a way to execute callback functions based on one or more objects, usually [Deferred](http://api.jquery.com/category/deferred-object/) objects that represent asynchronous events.

**Selectors**

Borrowing from CSS 1–3, and then adding its own, jQuery offers a powerful set of tools for matching a set of elements in a document.

If you wish to use any of the meta-characters ( such as !"#$%&'()\*+,./:;<=>?@[\]^`{|}~ ) as a literal part of a name, you must escape the character with two backslashes: \\. For example, if you have an element with id="foo.bar", you can use the selector $("#foo\\.bar"). The W3C CSS specification contains the [complete set of rules regarding valid CSS selectors](http://www.w3.org/TR/CSS21/syndata.html#value-def-identifier). Also useful is the blog entry by Mathias Bynens on [CSS character escape sequences for identifiers](http://mathiasbynens.be/notes/css-escapes).

* [**All Selector (“\*”)**](http://api.jquery.com/all-selector/)

[Basic](http://api.jquery.com/category/selectors/basic-css-selectors/)

Selects all elements.

* [**:animated Selector**](http://api.jquery.com/animated-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

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

* [**Attribute Contains Prefix Selector [name|="value"]**](http://api.jquery.com/attribute-contains-prefix-selector/)

[Attribute](http://api.jquery.com/category/selectors/attribute-selectors/)

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"]**](http://api.jquery.com/attribute-contains-selector/)

[Attribute](http://api.jquery.com/category/selectors/attribute-selectors/)

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

* [**Attribute Contains Word Selector [name~="value"]**](http://api.jquery.com/attribute-contains-word-selector/)

[Attribute](http://api.jquery.com/category/selectors/attribute-selectors/)

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

* [**Attribute Ends With Selector [name$="value"]**](http://api.jquery.com/attribute-ends-with-selector/)

[Attribute](http://api.jquery.com/category/selectors/attribute-selectors/)

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"]**](http://api.jquery.com/attribute-equals-selector/)

[Attribute](http://api.jquery.com/category/selectors/attribute-selectors/)

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

* [**Attribute Not Equal Selector [name!="value"]**](http://api.jquery.com/attribute-not-equal-selector/)

[Attribute](http://api.jquery.com/category/selectors/attribute-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

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"]**](http://api.jquery.com/attribute-starts-with-selector/)

[Attribute](http://api.jquery.com/category/selectors/attribute-selectors/)

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

* [**:button Selector**](http://api.jquery.com/button-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all button elements and elements of type button.

* [**:checkbox Selector**](http://api.jquery.com/checkbox-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all elements of type checkbox.

* [**:checked Selector**](http://api.jquery.com/checked-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/)

Matches all elements that are checked.

* [**Child Selector (“parent > child”)**](http://api.jquery.com/child-selector/)

[Hierarchy](http://api.jquery.com/category/selectors/hierarchy-selectors/)

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

* [**Class Selector (“.class”)**](http://api.jquery.com/class-selector/)

[Basic](http://api.jquery.com/category/selectors/basic-css-selectors/)

Selects all elements with the given class.

* [**:contains() Selector**](http://api.jquery.com/contains-selector/)

[Content Filter](http://api.jquery.com/category/selectors/content-filter-selector/)

Select all elements that contain the specified text.

* [**Descendant Selector (“ancestor descendant”)**](http://api.jquery.com/descendant-selector/)

[Hierarchy](http://api.jquery.com/category/selectors/hierarchy-selectors/)

Selects all elements that are descendants of a given ancestor.

* [**:disabled Selector**](http://api.jquery.com/disabled-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/)

Selects all elements that are disabled.

* [**Element Selector (“element”)**](http://api.jquery.com/element-selector/)

[Basic](http://api.jquery.com/category/selectors/basic-css-selectors/)

Selects all elements with the given tag name.

* [**:empty Selector**](http://api.jquery.com/empty-selector/)

[Content Filter](http://api.jquery.com/category/selectors/content-filter-selector/)

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

* [**:enabled Selector**](http://api.jquery.com/enabled-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/)

Selects all elements that are enabled.

* [**:eq() Selector**](http://api.jquery.com/eq-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Select the element at index n within the matched set.

* [**:even Selector**](http://api.jquery.com/even-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects even elements, zero-indexed. See also [odd](http://api.jquery.com/Selectors/odd).

* [**:file Selector**](http://api.jquery.com/file-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all elements of type file.

* [**:first-child Selector**](http://api.jquery.com/first-child-selector/)

[Child Filter](http://api.jquery.com/category/selectors/child-filter-selectors/)

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

* [**:first Selector**](http://api.jquery.com/first-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects the first matched element.

* [**:focus selector**](http://api.jquery.com/focus-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/), [Form](http://api.jquery.com/category/selectors/form-selectors/)

Selects element if it is currently focused.

* [**:gt() Selector**](http://api.jquery.com/gt-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

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

* [**Has Attribute Selector [name]**](http://api.jquery.com/has-attribute-selector/)

[Attribute](http://api.jquery.com/category/selectors/attribute-selectors/)

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

* [**:has() Selector**](http://api.jquery.com/has-selector/)

[Content Filter](http://api.jquery.com/category/selectors/content-filter-selector/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

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

* [**:header Selector**](http://api.jquery.com/header-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

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

* [**:hidden Selector**](http://api.jquery.com/hidden-selector/)

[jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/), [Visibility Filter](http://api.jquery.com/category/selectors/visibility-filter-selectors/)

Selects all elements that are hidden.

* [**ID Selector (“#id”)**](http://api.jquery.com/id-selector/)

[Basic](http://api.jquery.com/category/selectors/basic-css-selectors/)

Selects a single element with the given id attribute.

* [**:image Selector**](http://api.jquery.com/image-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all elements of type image.

* [**:input Selector**](http://api.jquery.com/input-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all input, textarea, select and button elements.

* [**:last-child Selector**](http://api.jquery.com/last-child-selector/)

[Child Filter](http://api.jquery.com/category/selectors/child-filter-selectors/)

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

* [**:last Selector**](http://api.jquery.com/last-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects the last matched element.

* [**:lt() Selector**](http://api.jquery.com/lt-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

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

* [**Multiple Attribute Selector [name="value"][name2="value2"]**](http://api.jquery.com/multiple-attribute-selector/)

[Attribute](http://api.jquery.com/category/selectors/attribute-selectors/)

Matches elements that match all of the specified attribute filters.

* [**Multiple Selector (“selector1, selector2, selectorN”)**](http://api.jquery.com/multiple-selector/)

[Basic](http://api.jquery.com/category/selectors/basic-css-selectors/)

Selects the combined results of all the specified selectors.

* [**Next Adjacent Selector (“prev + next”)**](http://api.jquery.com/next-adjacent-Selector/)

[Hierarchy](http://api.jquery.com/category/selectors/hierarchy-selectors/)

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

* [**Next Siblings Selector (“prev ~ siblings”)**](http://api.jquery.com/next-siblings-selector/)

[Hierarchy](http://api.jquery.com/category/selectors/hierarchy-selectors/)

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

* [**:not() Selector**](http://api.jquery.com/not-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/)

Selects all elements that do not match the given selector.

* [**:nth-child() Selector**](http://api.jquery.com/nth-child-selector/)

[Child Filter](http://api.jquery.com/category/selectors/child-filter-selectors/)

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

* [**:odd Selector**](http://api.jquery.com/odd-selector/)

[Basic Filter](http://api.jquery.com/category/selectors/basic-filter-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects odd elements, zero-indexed. See also [even](http://api.jquery.com/Selectors/even).

* [**:only-child Selector**](http://api.jquery.com/only-child-selector/)

[Child Filter](http://api.jquery.com/category/selectors/child-filter-selectors/)

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

* [**:parent Selector**](http://api.jquery.com/parent-selector/)

[Content Filter](http://api.jquery.com/category/selectors/content-filter-selector/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

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

* [**:password Selector**](http://api.jquery.com/password-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all elements of type password.

* [**:radio Selector**](http://api.jquery.com/radio-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all elements of type radio.

* [**:reset Selector**](http://api.jquery.com/reset-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all elements of type reset.

* [**:selected Selector**](http://api.jquery.com/selected-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all elements that are selected.

* [**:submit Selector**](http://api.jquery.com/submit-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all elements of type submit.

* [**:text Selector**](http://api.jquery.com/text-selector/)

[Form](http://api.jquery.com/category/selectors/form-selectors/), [jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/)

Selects all elements of type text.

* [**:visible Selector**](http://api.jquery.com/visible-selector/)

[jQuery Extensions](http://api.jquery.com/category/selectors/jquery-selector-extensions/), [Visibility Filter](http://api.jquery.com/category/selectors/visibility-filter-selectors/)

Selects all elements that are visible.

**Attributes**

These methods get and set DOM attributes of elements.

* [**.addClass()**](http://api.jquery.com/addClass/)

[Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/), [CSS](http://api.jquery.com/category/css/)

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

* [**.attr()**](http://api.jquery.com/attr/)

[General Attributes](http://api.jquery.com/category/manipulation/general-attributes/)

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

* [**.hasClass()**](http://api.jquery.com/hasClass/)

[Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/), [CSS](http://api.jquery.com/category/css/)

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

* [**.html()**](http://api.jquery.com/html/)

[DOM Insertion, Inside](http://api.jquery.com/category/manipulation/dom-insertion-inside/)

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

* [**.prop()**](http://api.jquery.com/prop/)

[General Attributes](http://api.jquery.com/category/manipulation/general-attributes/)

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

* [**.removeAttr()**](http://api.jquery.com/removeAttr/)

[General Attributes](http://api.jquery.com/category/manipulation/general-attributes/)

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

* [**.removeClass()**](http://api.jquery.com/removeClass/)

[Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/), [CSS](http://api.jquery.com/category/css/)

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

* [**.removeProp()**](http://api.jquery.com/removeProp/)

[General Attributes](http://api.jquery.com/category/manipulation/general-attributes/)

Remove a property for the set of matched elements.

* [**.toggleClass()**](http://api.jquery.com/toggleClass/)

[Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/), [CSS](http://api.jquery.com/category/css/)

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()**](http://api.jquery.com/val/)

[Forms](http://api.jquery.com/category/forms/), [General Attributes](http://api.jquery.com/category/manipulation/general-attributes/)

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

**Traversing**

* [**.add()**](http://api.jquery.com/add/)

[Miscellaneous Traversing](http://api.jquery.com/category/traversing/miscellaneous-traversal/)

Add elements to the set of matched elements.

* [**.andSelf()**](http://api.jquery.com/andSelf/)

[Miscellaneous Traversing](http://api.jquery.com/category/traversing/miscellaneous-traversal/)

Add the previous set of elements on the stack to the current set.

* [**.children()**](http://api.jquery.com/children/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

Get the children of each element in the set of matched elements, optionally filtered by a selector.

* [**.closest()**](http://api.jquery.com/closest/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

Get the first element that matches the selector, beginning at the current element and progressing up through the DOM tree.

* [**.contents()**](http://api.jquery.com/contents/)

[Miscellaneous Traversing](http://api.jquery.com/category/traversing/miscellaneous-traversal/)

Get the children of each element in the set of matched elements, including text and comment nodes.

* [**.each()**](http://api.jquery.com/each/)

[Collection Manipulation](http://api.jquery.com/category/miscellaneous/collection-manipulation/)

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

* [**.end()**](http://api.jquery.com/end/)

[Miscellaneous Traversing](http://api.jquery.com/category/traversing/miscellaneous-traversal/)

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

* [**.eq()**](http://api.jquery.com/eq/)

[Filtering](http://api.jquery.com/category/traversing/filtering/)

Reduce the set of matched elements to the one at the specified index.

* [**.filter()**](http://api.jquery.com/filter/)

[Filtering](http://api.jquery.com/category/traversing/filtering/)

Reduce the set of matched elements to those that match the selector or pass the function's test.

* [**.find()**](http://api.jquery.com/find/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

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

* [**.first()**](http://api.jquery.com/first/)

[Filtering](http://api.jquery.com/category/traversing/filtering/)

Reduce the set of matched elements to the first in the set.

* [**.has()**](http://api.jquery.com/has/)

[Filtering](http://api.jquery.com/category/traversing/filtering/)

Reduce the set of matched elements to those that have a descendant that matches the selector or DOM element.

* [**.is()**](http://api.jquery.com/is/)

[Filtering](http://api.jquery.com/category/traversing/filtering/)

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.

* [**.last()**](http://api.jquery.com/last/)

[Filtering](http://api.jquery.com/category/traversing/filtering/)

Reduce the set of matched elements to the final one in the set.

* [**.map()**](http://api.jquery.com/map/)

[Filtering](http://api.jquery.com/category/traversing/filtering/)

Pass each element in the current matched set through a function, producing a new jQuery object containing the return values.

* [**.next()**](http://api.jquery.com/next/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

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()**](http://api.jquery.com/nextAll/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

Get all following siblings of each element in the set of matched elements, optionally filtered by a selector.

* [**.nextUntil()**](http://api.jquery.com/nextUntil/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

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

* [**.not()**](http://api.jquery.com/not/)

[Filtering](http://api.jquery.com/category/traversing/filtering/), [Miscellaneous Traversing](http://api.jquery.com/category/traversing/miscellaneous-traversal/)

Remove elements from the set of matched elements.

* [**.offsetParent()**](http://api.jquery.com/offsetParent/)

[Offset](http://api.jquery.com/category/offset/), [Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

Get the closest ancestor element that is positioned.

* [**.parent()**](http://api.jquery.com/parent/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

Get the parent of each element in the current set of matched elements, optionally filtered by a selector.

* [**.parents()**](http://api.jquery.com/parents/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

Get the ancestors of each element in the current set of matched elements, optionally filtered by a selector.

* [**.parentsUntil()**](http://api.jquery.com/parentsUntil/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

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.

* [**.prev()**](http://api.jquery.com/prev/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

Get the immediately preceding sibling of each element in the set of matched elements, optionally filtered by a selector.

* [**.prevAll()**](http://api.jquery.com/prevAll/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

Get all preceding siblings of each element in the set of matched elements, optionally filtered by a selector.

* [**.prevUntil()**](http://api.jquery.com/prevUntil/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

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

* [**.siblings()**](http://api.jquery.com/siblings/)

[Tree Traversal](http://api.jquery.com/category/traversing/tree-traversal/)

Get the siblings of each element in the set of matched elements, optionally filtered by a selector.

* [**.slice()**](http://api.jquery.com/slice/)

[Filtering](http://api.jquery.com/category/traversing/filtering/)

Reduce the set of matched elements to a subset specified by a range of indices.

**Manipulation**

All of the methods in this chapter manipulate the DOM in some manner. A few of them simply change one of the attributes of an element (also listed in the [Attributes category](http://api.jquery.com/category/attributes/)), while others set an element’s style properties (also listed in the [CSS category](http://api.jquery.com/category/css/)). Still others modify entire elements (or groups of elements) themselves—inserting, copying, removing, and so on. All of these methods are referred to as “setters,” as they change the values of properties.  
A few of these methods—such as .attr(), .html(), and .val()—also act as “getters,” retrieving information from DOM elements for later use.

* [**.addClass()**](http://api.jquery.com/addClass/)

[Attributes](http://api.jquery.com/category/attributes/), [Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/), [CSS](http://api.jquery.com/category/css/)

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

* [**.after()**](http://api.jquery.com/after/)

[DOM Insertion, Outside](http://api.jquery.com/category/manipulation/dom-insertion-outside/)

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

* [**.append()**](http://api.jquery.com/append/)

[DOM Insertion, Inside](http://api.jquery.com/category/manipulation/dom-insertion-inside/)

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

* [**.appendTo()**](http://api.jquery.com/appendTo/)

[DOM Insertion, Inside](http://api.jquery.com/category/manipulation/dom-insertion-inside/)

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

* [**.attr()**](http://api.jquery.com/attr/)

[Attributes](http://api.jquery.com/category/attributes/), [General Attributes](http://api.jquery.com/category/manipulation/general-attributes/)

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

* [**.before()**](http://api.jquery.com/before/)

[DOM Insertion, Outside](http://api.jquery.com/category/manipulation/dom-insertion-outside/)

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

* [**.clone()**](http://api.jquery.com/clone/)

[Copying](http://api.jquery.com/category/manipulation/copying/)

Create a deep copy of the set of matched elements.

* [**.css()**](http://api.jquery.com/css/)

[CSS](http://api.jquery.com/category/css/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.detach()**](http://api.jquery.com/detach/)

[DOM Removal](http://api.jquery.com/category/manipulation/dom-removal/)

Remove the set of matched elements from the DOM.

* [**.empty()**](http://api.jquery.com/empty/)

[DOM Removal](http://api.jquery.com/category/manipulation/dom-removal/)

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

* [**.hasClass()**](http://api.jquery.com/hasClass/)

[Attributes](http://api.jquery.com/category/attributes/), [Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/), [CSS](http://api.jquery.com/category/css/)

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

* [**.height()**](http://api.jquery.com/height/)

[CSS](http://api.jquery.com/category/css/), [Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.html()**](http://api.jquery.com/html/)

[Attributes](http://api.jquery.com/category/attributes/), [DOM Insertion, Inside](http://api.jquery.com/category/manipulation/dom-insertion-inside/)

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

* [**.innerHeight()**](http://api.jquery.com/innerHeight/)

[CSS](http://api.jquery.com/category/css/), [Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.innerWidth()**](http://api.jquery.com/innerWidth/)

[CSS](http://api.jquery.com/category/css/), [Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.insertAfter()**](http://api.jquery.com/insertAfter/)

[DOM Insertion, Outside](http://api.jquery.com/category/manipulation/dom-insertion-outside/)

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

* [**.insertBefore()**](http://api.jquery.com/insertBefore/)

[DOM Insertion, Outside](http://api.jquery.com/category/manipulation/dom-insertion-outside/)

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

* [**.offset()**](http://api.jquery.com/offset/)

[CSS](http://api.jquery.com/category/css/), [Offset](http://api.jquery.com/category/offset/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.outerHeight()**](http://api.jquery.com/outerHeight/)

[CSS](http://api.jquery.com/category/css/), [Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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()**](http://api.jquery.com/outerWidth/)

[CSS](http://api.jquery.com/category/css/), [Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.position()**](http://api.jquery.com/position/)

[CSS](http://api.jquery.com/category/css/), [Offset](http://api.jquery.com/category/offset/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.prepend()**](http://api.jquery.com/prepend/)

[DOM Insertion, Inside](http://api.jquery.com/category/manipulation/dom-insertion-inside/)

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

* [**.prependTo()**](http://api.jquery.com/prependTo/)

[DOM Insertion, Inside](http://api.jquery.com/category/manipulation/dom-insertion-inside/)

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

* [**.prop()**](http://api.jquery.com/prop/)

[Attributes](http://api.jquery.com/category/attributes/), [General Attributes](http://api.jquery.com/category/manipulation/general-attributes/)

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

* [**.remove()**](http://api.jquery.com/remove/)

[DOM Removal](http://api.jquery.com/category/manipulation/dom-removal/)

Remove the set of matched elements from the DOM.

* [**.removeAttr()**](http://api.jquery.com/removeAttr/)

[Attributes](http://api.jquery.com/category/attributes/), [General Attributes](http://api.jquery.com/category/manipulation/general-attributes/)

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

* [**.removeClass()**](http://api.jquery.com/removeClass/)

[Attributes](http://api.jquery.com/category/attributes/), [Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/), [CSS](http://api.jquery.com/category/css/)

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

* [**.removeProp()**](http://api.jquery.com/removeProp/)

[Attributes](http://api.jquery.com/category/attributes/), [General Attributes](http://api.jquery.com/category/manipulation/general-attributes/)

Remove a property for the set of matched elements.

* [**.replaceAll()**](http://api.jquery.com/replaceAll/)

[DOM Replacement](http://api.jquery.com/category/manipulation/dom-replacement/)

Replace each target element with the set of matched elements.

* [**.replaceWith()**](http://api.jquery.com/replaceWith/)

[DOM Replacement](http://api.jquery.com/category/manipulation/dom-replacement/)

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

* [**.scrollLeft()**](http://api.jquery.com/scrollLeft/)

[CSS](http://api.jquery.com/category/css/), [Offset](http://api.jquery.com/category/offset/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.scrollTop()**](http://api.jquery.com/scrollTop/)

[CSS](http://api.jquery.com/category/css/), [Offset](http://api.jquery.com/category/offset/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.text()**](http://api.jquery.com/text/)

[DOM Insertion, Inside](http://api.jquery.com/category/manipulation/dom-insertion-inside/)

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

* [**.toggleClass()**](http://api.jquery.com/toggleClass/)

[Attributes](http://api.jquery.com/category/attributes/), [Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/), [CSS](http://api.jquery.com/category/css/)

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.

* [**.unwrap()**](http://api.jquery.com/unwrap/)

[DOM Insertion, Around](http://api.jquery.com/category/manipulation/dom-insertion-around/), [DOM Removal](http://api.jquery.com/category/manipulation/dom-removal/)

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

* [**.val()**](http://api.jquery.com/val/)

[Attributes](http://api.jquery.com/category/attributes/), [Forms](http://api.jquery.com/category/forms/), [General Attributes](http://api.jquery.com/category/manipulation/general-attributes/)

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

* [**.width()**](http://api.jquery.com/width/)

[CSS](http://api.jquery.com/category/css/), [Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.wrap()**](http://api.jquery.com/wrap/)

[DOM Insertion, Around](http://api.jquery.com/category/manipulation/dom-insertion-around/)

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

* [**.wrapAll()**](http://api.jquery.com/wrapAll/)

[DOM Insertion, Around](http://api.jquery.com/category/manipulation/dom-insertion-around/)

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

* [**.wrapInner()**](http://api.jquery.com/wrapInner/)

[DOM Insertion, Around](http://api.jquery.com/category/manipulation/dom-insertion-around/)

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

**CSS**

These methods get and set CSS-related properties of elements.

* [**.addClass()**](http://api.jquery.com/addClass/)

[Attributes](http://api.jquery.com/category/attributes/), [Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/)

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

* [**.css()**](http://api.jquery.com/css/)

[Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**jQuery.cssHooks**](http://api.jquery.com/jQuery.cssHooks/)

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

* [**.hasClass()**](http://api.jquery.com/hasClass/)

[Attributes](http://api.jquery.com/category/attributes/), [Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/)

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

* [**.height()**](http://api.jquery.com/height/)

[Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.innerHeight()**](http://api.jquery.com/innerHeight/)

[Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.innerWidth()**](http://api.jquery.com/innerWidth/)

[Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.offset()**](http://api.jquery.com/offset/)

[Offset](http://api.jquery.com/category/offset/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.outerHeight()**](http://api.jquery.com/outerHeight/)

[Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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()**](http://api.jquery.com/outerWidth/)

[Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.position()**](http://api.jquery.com/position/)

[Offset](http://api.jquery.com/category/offset/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.removeClass()**](http://api.jquery.com/removeClass/)

[Attributes](http://api.jquery.com/category/attributes/), [Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/)

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

* [**.scrollLeft()**](http://api.jquery.com/scrollLeft/)

[Offset](http://api.jquery.com/category/offset/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.scrollTop()**](http://api.jquery.com/scrollTop/)

[Offset](http://api.jquery.com/category/offset/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

* [**.toggleClass()**](http://api.jquery.com/toggleClass/)

[Attributes](http://api.jquery.com/category/attributes/), [Class Attribute](http://api.jquery.com/category/manipulation/class-attribute/)

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.

* [**.width()**](http://api.jquery.com/width/)

[Dimensions](http://api.jquery.com/category/dimensions/), [Style Properties](http://api.jquery.com/category/manipulation/style-properties/)

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

**Events**

These methods are used to register behaviors to take effect when the user interacts with the browser, and to further manipulate those registered behaviors.

* [**.bind()**](http://api.jquery.com/bind/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

Attach a handler to an event for the elements.

* [**.blur()**](http://api.jquery.com/blur/)

[Form Events](http://api.jquery.com/category/events/form-events/), [Forms](http://api.jquery.com/category/forms/)

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

* [**.change()**](http://api.jquery.com/change/)

[Form Events](http://api.jquery.com/category/events/form-events/), [Forms](http://api.jquery.com/category/forms/)

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

* [**.click()**](http://api.jquery.com/click/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.dblclick()**](http://api.jquery.com/dblclick/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.delegate()**](http://api.jquery.com/delegate/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

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.

* [**.die()**](http://api.jquery.com/die/)

[Deprecated](http://api.jquery.com/category/deprecated/), [Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

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

* [**.error()**](http://api.jquery.com/error/)

[Browser Events](http://api.jquery.com/category/events/browser-events/)

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

* [**event.currentTarget**](http://api.jquery.com/event.currentTarget/)

[Event Object](http://api.jquery.com/category/events/event-object/)

The current DOM element within the event bubbling phase.

* [**event.data**](http://api.jquery.com/event.data/)

[Event Object](http://api.jquery.com/category/events/event-object/)

An optional data map passed to an event method when the current executing handler is bound.

* [**event.delegateTarget**](http://api.jquery.com/event.delegateTarget/)

[Event Object](http://api.jquery.com/category/events/event-object/)

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

* [**event.isDefaultPrevented()**](http://api.jquery.com/event.isDefaultPrevented/)

[Event Object](http://api.jquery.com/category/events/event-object/)

Returns whether [event.preventDefault()](http://api.jquery.com/event.preventDefault) was ever called on this event object.

* [**event.isImmediatePropagationStopped()**](http://api.jquery.com/event.isImmediatePropagationStopped/)

[Event Object](http://api.jquery.com/category/events/event-object/)

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

* [**event.isPropagationStopped()**](http://api.jquery.com/event.isPropagationStopped/)

[Event Object](http://api.jquery.com/category/events/event-object/)

Returns whether [event.stopPropagation()](http://api.jquery.com/event.stopPropagation) was ever called on this event object.

* [**event.namespace**](http://api.jquery.com/event.namespace/)

[Event Object](http://api.jquery.com/category/events/event-object/)

The namespace specified when the event was triggered.

* [**event.pageX**](http://api.jquery.com/event.pageX/)

[Event Object](http://api.jquery.com/category/events/event-object/)

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

* [**event.pageY**](http://api.jquery.com/event.pageY/)

[Event Object](http://api.jquery.com/category/events/event-object/)

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

* [**event.preventDefault()**](http://api.jquery.com/event.preventDefault/)

[Event Object](http://api.jquery.com/category/events/event-object/)

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

* [**event.relatedTarget**](http://api.jquery.com/event.relatedTarget/)

[Event Object](http://api.jquery.com/category/events/event-object/)

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

* [**event.result**](http://api.jquery.com/event.result/)

[Event Object](http://api.jquery.com/category/events/event-object/)

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

* [**event.stopImmediatePropagation()**](http://api.jquery.com/event.stopImmediatePropagation/)

[Event Object](http://api.jquery.com/category/events/event-object/)

Prevents other event handlers from being called.

* [**event.stopPropagation()**](http://api.jquery.com/event.stopPropagation/)

[Event Object](http://api.jquery.com/category/events/event-object/)

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

* [**event.target**](http://api.jquery.com/event.target/)

[Event Object](http://api.jquery.com/category/events/event-object/)

The DOM element that initiated the event.

* [**event.timeStamp**](http://api.jquery.com/event.timeStamp/)

[Event Object](http://api.jquery.com/category/events/event-object/)

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

* [**event.type**](http://api.jquery.com/event.type/)

[Event Object](http://api.jquery.com/category/events/event-object/)

Describes the nature of the event.

* [**event.which**](http://api.jquery.com/event.which/)

[Event Object](http://api.jquery.com/category/events/event-object/)

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

* [**.focus()**](http://api.jquery.com/focus/)

[Form Events](http://api.jquery.com/category/events/form-events/), [Forms](http://api.jquery.com/category/forms/)

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

* [**.focusin()**](http://api.jquery.com/focusin/)

[Keyboard Events](http://api.jquery.com/category/events/keyboard-events/), [Mouse Events](http://api.jquery.com/category/events/mouse-events/)

Bind an event handler to the "focusin" event.

* [**.focusout()**](http://api.jquery.com/focusout/)

[Keyboard Events](http://api.jquery.com/category/events/keyboard-events/), [Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.hover()**](http://api.jquery.com/hover/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.keydown()**](http://api.jquery.com/keydown/)

[Keyboard Events](http://api.jquery.com/category/events/keyboard-events/)

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

* [**.keypress()**](http://api.jquery.com/keypress/)

[Keyboard Events](http://api.jquery.com/category/events/keyboard-events/)

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

* [**.keyup()**](http://api.jquery.com/keyup/)

[Keyboard Events](http://api.jquery.com/category/events/keyboard-events/)

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

* [**.live()**](http://api.jquery.com/live/)

[Deprecated](http://api.jquery.com/category/deprecated/), [Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

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

* [**.load()**](http://api.jquery.com/load-event/)

[Document Loading](http://api.jquery.com/category/events/document-loading/)

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

* [**.mousedown()**](http://api.jquery.com/mousedown/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.mouseenter()**](http://api.jquery.com/mouseenter/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.mouseleave()**](http://api.jquery.com/mouseleave/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.mousemove()**](http://api.jquery.com/mousemove/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.mouseout()**](http://api.jquery.com/mouseout/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.mouseover()**](http://api.jquery.com/mouseover/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.mouseup()**](http://api.jquery.com/mouseup/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.off()**](http://api.jquery.com/off/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

Remove an event handler.

* [**.on()**](http://api.jquery.com/on/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

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

* [**.one()**](http://api.jquery.com/one/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

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

* [**jQuery.proxy()**](http://api.jquery.com/jQuery.proxy/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/), [Utilities](http://api.jquery.com/category/utilities/)

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

* [**.ready()**](http://api.jquery.com/ready/)

[Document Loading](http://api.jquery.com/category/events/document-loading/)

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

* [**.resize()**](http://api.jquery.com/resize/)

[Browser Events](http://api.jquery.com/category/events/browser-events/)

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

* [**.scroll()**](http://api.jquery.com/scroll/)

[Browser Events](http://api.jquery.com/category/events/browser-events/)

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

* [**.select()**](http://api.jquery.com/select/)

[Form Events](http://api.jquery.com/category/events/form-events/), [Forms](http://api.jquery.com/category/forms/)

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

* [**.submit()**](http://api.jquery.com/submit/)

[Form Events](http://api.jquery.com/category/events/form-events/), [Forms](http://api.jquery.com/category/forms/)

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

* [**.toggle()**](http://api.jquery.com/toggle-event/)

[Mouse Events](http://api.jquery.com/category/events/mouse-events/)

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

* [**.trigger()**](http://api.jquery.com/trigger/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

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

* [**.triggerHandler()**](http://api.jquery.com/triggerHandler/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

Execute all handlers attached to an element for an event.

* [**.unbind()**](http://api.jquery.com/unbind/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

Remove a previously-attached event handler from the elements.

* [**.undelegate()**](http://api.jquery.com/undelegate/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

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

* [**.unload()**](http://api.jquery.com/unload/)

[Document Loading](http://api.jquery.com/category/events/document-loading/)

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

**Effects**

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. In this chapter, we’ll closely examine each of the effect methods, revealing all of the mechanisms jQuery has for providing visual feedback to the user.

* [**.animate()**](http://api.jquery.com/animate/)

[Custom](http://api.jquery.com/category/effects/custom-effects/)

Perform a custom animation of a set of CSS properties.

* [**.clearQueue()**](http://api.jquery.com/clearQueue/)

[Custom](http://api.jquery.com/category/effects/custom-effects/), [Data](http://api.jquery.com/category/data/), [Utilities](http://api.jquery.com/category/utilities/)

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

* [**.delay()**](http://api.jquery.com/delay/)

[Custom](http://api.jquery.com/category/effects/custom-effects/)

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

* [**.dequeue()**](http://api.jquery.com/dequeue/)

[Custom](http://api.jquery.com/category/effects/custom-effects/), [Data](http://api.jquery.com/category/data/), [Utilities](http://api.jquery.com/category/utilities/)

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

* [**.fadeIn()**](http://api.jquery.com/fadeIn/)

[Fading](http://api.jquery.com/category/effects/fading/)

Display the matched elements by fading them to opaque.

* [**.fadeOut()**](http://api.jquery.com/fadeOut/)

[Fading](http://api.jquery.com/category/effects/fading/)

Hide the matched elements by fading them to transparent.

* [**.fadeTo()**](http://api.jquery.com/fadeTo/)

[Fading](http://api.jquery.com/category/effects/fading/)

Adjust the opacity of the matched elements.

* [**.fadeToggle()**](http://api.jquery.com/fadeToggle/)

[Fading](http://api.jquery.com/category/effects/fading/)

Display or hide the matched elements by animating their opacity.

* [**jQuery.fx.interval**](http://api.jquery.com/jQuery.fx.interval/)

[Custom](http://api.jquery.com/category/effects/custom-effects/), [Properties of the Global jQuery Object](http://api.jquery.com/category/properties/global-jquery-object-properties/)

The rate (in milliseconds) at which animations fire.

* [**jQuery.fx.off**](http://api.jquery.com/jQuery.fx.off/)

[Custom](http://api.jquery.com/category/effects/custom-effects/), [Properties of the Global jQuery Object](http://api.jquery.com/category/properties/global-jquery-object-properties/)

Globally disable all animations.

* [**.hide()**](http://api.jquery.com/hide/)

[Basics](http://api.jquery.com/category/effects/basics/)

Hide the matched elements.

* [**.queue()**](http://api.jquery.com/queue/)

[Custom](http://api.jquery.com/category/effects/custom-effects/), [Data](http://api.jquery.com/category/data/), [Utilities](http://api.jquery.com/category/utilities/)

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

* [**.show()**](http://api.jquery.com/show/)

[Basics](http://api.jquery.com/category/effects/basics/)

Display the matched elements.

* [**.slideDown()**](http://api.jquery.com/slideDown/)

[Sliding](http://api.jquery.com/category/effects/sliding/)

Display the matched elements with a sliding motion.

* [**.slideToggle()**](http://api.jquery.com/slideToggle/)

[Sliding](http://api.jquery.com/category/effects/sliding/)

Display or hide the matched elements with a sliding motion.

* [**.slideUp()**](http://api.jquery.com/slideUp/)

[Sliding](http://api.jquery.com/category/effects/sliding/)

Hide the matched elements with a sliding motion.

* [**.stop()**](http://api.jquery.com/stop/)

[Custom](http://api.jquery.com/category/effects/custom-effects/)

Stop the currently-running animation on the matched elements.

* [**.toggle()**](http://api.jquery.com/toggle/)

[Basics](http://api.jquery.com/category/effects/basics/)

Display or hide the matched elements.

**Ajax**

The jQuery library has a full suite of AJAX capabilities. The functions and methods therein allow us to load data from the server without a browser page refresh.

* [**jQuery.ajax()**](http://api.jquery.com/jQuery.ajax/)

[Low-Level Interface](http://api.jquery.com/category/ajax/low-level-interface/)

Perform an asynchronous HTTP (Ajax) request.

* [**.ajaxComplete()**](http://api.jquery.com/ajaxComplete/)

[Global Ajax Event Handlers](http://api.jquery.com/category/ajax/global-ajax-event-handlers/)

Register a handler to be called when Ajax requests complete. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

* [**.ajaxError()**](http://api.jquery.com/ajaxError/)

[Global Ajax Event Handlers](http://api.jquery.com/category/ajax/global-ajax-event-handlers/)

Register a handler to be called when Ajax requests complete with an error. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

* [**jQuery.ajaxPrefilter()**](http://api.jquery.com/jQuery.ajaxPrefilter/)

[Low-Level Interface](http://api.jquery.com/category/ajax/low-level-interface/)

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

* [**.ajaxSend()**](http://api.jquery.com/ajaxSend/)

[Global Ajax Event Handlers](http://api.jquery.com/category/ajax/global-ajax-event-handlers/)

Attach a function to be executed before an Ajax request is sent. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

* [**jQuery.ajaxSetup()**](http://api.jquery.com/jQuery.ajaxSetup/)

[Low-Level Interface](http://api.jquery.com/category/ajax/low-level-interface/)

Set default values for future Ajax requests.

* [**.ajaxStart()**](http://api.jquery.com/ajaxStart/)

[Global Ajax Event Handlers](http://api.jquery.com/category/ajax/global-ajax-event-handlers/)

Register a handler to be called when the first Ajax request begins. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

* [**.ajaxStop()**](http://api.jquery.com/ajaxStop/)

[Global Ajax Event Handlers](http://api.jquery.com/category/ajax/global-ajax-event-handlers/)

Register a handler to be called when all Ajax requests have completed. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

* [**.ajaxSuccess()**](http://api.jquery.com/ajaxSuccess/)

[Global Ajax Event Handlers](http://api.jquery.com/category/ajax/global-ajax-event-handlers/)

Attach a function to be executed whenever an Ajax request completes successfully. This is an [Ajax Event](http://docs.jquery.com/Ajax_Events).

* [**jQuery.get()**](http://api.jquery.com/jQuery.get/)

[Shorthand Methods](http://api.jquery.com/category/ajax/shorthand-methods/)

Load data from the server using a HTTP GET request.

* [**jQuery.getJSON()**](http://api.jquery.com/jQuery.getJSON/)

[Shorthand Methods](http://api.jquery.com/category/ajax/shorthand-methods/)

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

* [**jQuery.getScript()**](http://api.jquery.com/jQuery.getScript/)

[Shorthand Methods](http://api.jquery.com/category/ajax/shorthand-methods/)

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

* [**.load()**](http://api.jquery.com/load/)

[Shorthand Methods](http://api.jquery.com/category/ajax/shorthand-methods/)

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

* [**jQuery.param()**](http://api.jquery.com/jQuery.param/)

[Collection Manipulation](http://api.jquery.com/category/miscellaneous/collection-manipulation/), [Forms](http://api.jquery.com/category/forms/), [Helper Functions](http://api.jquery.com/category/ajax/helper-functions/)

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

* [**jQuery.post()**](http://api.jquery.com/jQuery.post/)

[Shorthand Methods](http://api.jquery.com/category/ajax/shorthand-methods/)

Load data from the server using a HTTP POST request.

* [**.serialize()**](http://api.jquery.com/serialize/)

[Forms](http://api.jquery.com/category/forms/), [Helper Functions](http://api.jquery.com/category/ajax/helper-functions/)

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

* [**.serializeArray()**](http://api.jquery.com/serializeArray/)

[Forms](http://api.jquery.com/category/forms/), [Helper Functions](http://api.jquery.com/category/ajax/helper-functions/)

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

**Utilities**

* [**jQuery.boxModel**](http://api.jquery.com/jQuery.boxModel/)

[Deprecated](http://api.jquery.com/category/deprecated/)

**Deprecated in jQuery 1.3 (see** [**jQuery.support**](http://api.jquery.com/jQuery.support)**)**. States if the current page, in the user's browser, is being rendered using the [W3C CSS Box Model](http://www.w3.org/TR/REC-CSS2/box.html).

* [**jQuery.browser**](http://api.jquery.com/jQuery.browser/)

[Deprecated](http://api.jquery.com/category/deprecated/), [Properties of the Global jQuery Object](http://api.jquery.com/category/properties/global-jquery-object-properties/)

Contains flags for the useragent, read from navigator.userAgent. **We recommend against using this property; please try to use feature detection instead (see jQuery.support). jQuery.browser may be moved to a plugin in a future release of jQuery.**

* [**.clearQueue()**](http://api.jquery.com/clearQueue/)

[Custom](http://api.jquery.com/category/effects/custom-effects/), [Data](http://api.jquery.com/category/data/)

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

* [**jQuery.contains()**](http://api.jquery.com/jQuery.contains/)

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

* [**jQuery.data()**](http://api.jquery.com/jQuery.data/)

[Data](http://api.jquery.com/category/data/)

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

* [**.dequeue()**](http://api.jquery.com/dequeue/)

[Custom](http://api.jquery.com/category/effects/custom-effects/), [Data](http://api.jquery.com/category/data/)

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

* [**jQuery.dequeue()**](http://api.jquery.com/jQuery.dequeue/)

[Data](http://api.jquery.com/category/data/)

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

* [**jQuery.each()**](http://api.jquery.com/jQuery.each/)

A generic iterator function, which can be used to seamlessly iterate over both objects and arrays. Arrays and array-like objects with a length property (such as a function's arguments object) are iterated by numeric index, from 0 to length-1. Other objects are iterated via their named properties.

* [**jQuery.extend()**](http://api.jquery.com/jQuery.extend/)

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

* [**jQuery.globalEval()**](http://api.jquery.com/jQuery.globalEval/)

Execute some JavaScript code globally.

* [**jQuery.grep()**](http://api.jquery.com/jQuery.grep/)

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

* [**jQuery.inArray()**](http://api.jquery.com/jQuery.inArray/)

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

* [**jQuery.isArray()**](http://api.jquery.com/jQuery.isArray/)

Determine whether the argument is an array.

* [**jQuery.isEmptyObject()**](http://api.jquery.com/jQuery.isEmptyObject/)

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

* [**jQuery.isFunction()**](http://api.jquery.com/jQuery.isFunction/)

Determine if the argument passed is a Javascript function object.

* [**jQuery.isNumeric()**](http://api.jquery.com/jQuery.isNumeric/)

Determines whether its argument is a number.

* [**jQuery.isPlainObject()**](http://api.jquery.com/jQuery.isPlainObject/)

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

* [**jQuery.isWindow()**](http://api.jquery.com/jQuery.isWindow/)

Determine whether the argument is a window.

* [**jQuery.isXMLDoc()**](http://api.jquery.com/jQuery.isXMLDoc/)

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

* [**jQuery.makeArray()**](http://api.jquery.com/jQuery.makeArray/)

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

* [**jQuery.map()**](http://api.jquery.com/jQuery.map/)

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

* [**jQuery.merge()**](http://api.jquery.com/jQuery.merge/)

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

* [**jQuery.noop()**](http://api.jquery.com/jQuery.noop/)

An empty function.

* [**jQuery.now()**](http://api.jquery.com/jQuery.now/)

Return a number representing the current time.

* [**jQuery.parseJSON**](http://api.jquery.com/jQuery.parseJSON/)

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

* [**jQuery.parseXML()**](http://api.jquery.com/jQuery.parseXML/)

Parses a string into an XML document.

* [**jQuery.proxy()**](http://api.jquery.com/jQuery.proxy/)

[Event Handler Attachment](http://api.jquery.com/category/events/event-handler-attachment/)

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

* [**.queue()**](http://api.jquery.com/queue/)

[Custom](http://api.jquery.com/category/effects/custom-effects/), [Data](http://api.jquery.com/category/data/)

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

* [**jQuery.queue()**](http://api.jquery.com/jQuery.queue/)

[Data](http://api.jquery.com/category/data/)

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

* [**jQuery.removeData()**](http://api.jquery.com/jQuery.removeData/)

[Data](http://api.jquery.com/category/data/)

Remove a previously-stored piece of data.

* [**jQuery.support**](http://api.jquery.com/jQuery.support/)

[Properties of the Global jQuery Object](http://api.jquery.com/category/properties/global-jquery-object-properties/)

A collection of properties that represent the presence of different browser features or bugs. Primarily intended for jQuery's internal use; specific properties may be removed when they are no longer needed internally to improve page startup performance.

* [**jQuery.trim()**](http://api.jquery.com/jQuery.trim/)

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

* [**jQuery.type()**](http://api.jquery.com/jQuery.type/)

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

* [**jQuery.unique()**](http://api.jquery.com/jQuery.unique/)

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.

**UI**

Interactions:

Draggable

Droppable

Resizable

Selectable

Sortable

Widgets:

Accordion

Autocomplete

Button

Datepicker

Dialog

Progressbar

Slider

Tabs

Utilities:

Position

Effects **Types**

1. String

1.1 Quoting

1.2 Built-in Methods

1.3 Length Property

1.4 Boolean Default

2. Number

2.1 Boolean Default

2.2 Math

2.3 Parsing Numbers

2.4 Numbers to Strings

2.5 NaN and Infinity

2.6 Integer

2.7 Float

3. Boolean

4. Object

4.1 Dot Notation

4.2 Array Notation

4.3 Iteration

4.4 Boolean default

4.5 Prototype

5. Options

6. Array

6.1 Iteration

6.2 Boolean Default

6.3 Array<Type> Notation

7. Map

8. Function

8.1 Arguments

8.2 Context, Call and Apply

8.3 Scope

8.4 Closures

8.5 Proxy Pattern

9. Callback

10. Selector

11. Event

12. Element

13. jQuery

14. XMLHttpRequest

**String**

"I'm a String in JavaScript!"

'So am I!'

A string in JavaScript is an immutable object that contains none, one or many characters.

The type of a string is "string".

typeof "some string"; // "string"

**Quoting**

A string can be defined using single or double quotes. You can nest single quotes inside of double quotes, and the other way round. To mix double quotes with double quotes, the nested ones have to be escaped with a backslash.

"You make 'me' sad."

'Holy "cranking" moses!'

"<a href=\"home\">Home</a>"

**Built-in Methods**

A string in JavaScript has some built-in methods to manipulate the string, though the result is always a new string - or something else, eg. split returns an [array](http://docs.jquery.com/Types#Array).

"hello".charAt(0) // "h"

"hello".toUpperCase() // "HELLO"

"Hello".toLowerCase() // "hello"

"hello".replace(/e|o/g, "x") // "hxllx"

"1,2,3".split(",") // ["1", "2", "3"]

**Length Property**

All strings have a length property.

"Hello".length // 5

"".length // 0

**Boolean Default**

An empty string defaults to false:

!"" // true

!"hello" // false

!"true" // false

!new Boolean(false) // false

**Number**

12

3.543

Numbers in JavaScript are double-precision 64-bit format IEEE 754 values. They are immutable, just as [strings](http://docs.jquery.com/Types#String). All operators common in c-based languages are available to work with numbers (+, -, \*, /, %, =, +=, -=, \*=, /=, ++, --).

The type of a number is "number".

typeof 12 // "number"

typeof 3.543 // "number"

**Boolean Default**

If a number is zero, it defaults to false:

!0 // true

!1 // false

!-1 // false

Due to the implementation of numbers as double-precision values, the following result is not an error:

0.1 + 0.2 // 0.30000000000000004

**Math**

JavaScript provides utilities to work with numbers in the Math object:

Math.PI // 3.141592653589793

Math.cos(Math.PI) // -1

**Parsing Numbers**

parseInt and parseFloat help parsing strings into numbers. Both do some implicit conversion if the base isn't specified:

parseInt("123") = 123 (implicit decimal)

parseInt("010") = 8 (implicit octal)

parseInt("0xCAFE") = 51966 (implicit hexadecimal)

parseInt("010", 10) = 10 (explicit decimal)

parseInt("11", 2) = 3 (explicit binary)

parseFloat("10.10") = 10.1

**Numbers to Strings**

When appending numbers to string, the result is always a string. The operator is the same, so be careful: If you want to add numbers and then append them to a string, put parentheses around them:

"" + 1 + 2; // "12"

"" + (1 + 2); // "3"

"" + 0.0000001; // "1e-7"

parseInt(0.0000001); // 1 (!)

Or you use the String class provided by javascript, which try to parse a value as string:

String(1) + String(2); //"12"

String(1 + 2); //"3"

**NaN and Infinity**

Parsing something that isn't a number results in NaN. isNaN helps to detect those cases:

parseInt("hello", 10) // NaN

isNaN(parseInt("hello", 10)) // true

Division by zero results in Infinity:

1 / 0 // Infinity

Both NaN and Infinity are of type "number":

typeof NaN // "number"

typeof Infinity // "number"

Note that NaN compares in a strange way:

NaN == NaN // false (!)

But:

Infinity == Infinity // true

**Integer**

An integer is a plain Number type, but whenever explicitly mentioned, indicates that a non-floating-point number is expected.

**Float**

A float is a plain Number type, just as Integer, but whenever explicitly mentioned, indicates that a floating-point number is expected.

**Boolean**

A boolean in JavaScript can be either true or false:

if ( true ) console.log("always!")

if ( false ) console.log("never!")

When an [option](http://docs.jquery.com/Types#Options) is specified as a boolean, it often looks like this:

$("...").somePlugin({

hideOnStartup: true,

onlyOnce: false

});

**Object**

Everything in JavaScript is an object, though some are more objective (haha). The easiest way to create an object is the object literal:

var x = {};

var y = {

name: "Pete",

age: 15

};

The type of an object is "object":

typeof {} // "object"

**Dot Notation**

You can write and read properties of an object using the dot notation:

y.name // "Pete"

y.age // 15

x.name = y.name + " Pan" // "Pete Pan"

x.age = y.age + 1 // 16

**Array Notation**

Or you write and read properties using the array notation, which allows you to dynamically choose the property:

var operations = {

increase: "++",

decrease: "--"

}

var operation = "increase";

operations[operation] // "++";

operations["multiply"] = "\*"; // "\*"

**Iteration**

Iterating over objects is easy with the for-in-loop:

var obj = {

name: "Pete",

age: 15

};

for(key in obj) {

alert("key is "+[key]+", value is "+obj[key]);

}

Note that for-in-loop can be spoiled by extending Object.prototype (see [Object.prototype is verboten](http://erik.eae.net/archives/2005/06/06/22.13.54)) so take care when using other libraries.

jQuery provides a generic [each-function](http://docs.jquery.com/Utilities/jQuery.each) to iterate over properties of objects, as well as elements of arrays:

jQuery.each(obj, function(key, value) {

console.log("key", key, "value", value);

});

The drawback is that the callback is called in the context of each value, therefore you lose the context of your own object if applicable. More on this below at Functions.

**Boolean default**

An object, no matter if it has properties or not, never defaults to false:

!{} // false

**Prototype**

All objects have a prototype property. Whenever the interpreter looks for a property, it also checks the prototype. jQuery uses that extensively to add methods to jQuery instances.

var form = $("#myform");

form.clearForm; // undefined

form.fn.clearForm = function() {

return this.find(":input").each(function() {

this.value = "";

}).end();

};

form.clearForm() // works for all instances of jQuery objects, because the new method was added to the prototype

(This example needs clarification: how does it modify the prototype when the word "prototype" doesn't appear anywhere? The implication is that form.fn is simply an alias for form.prototype, but if that's the case then it should be explained. :-?)

In javascript:the definitive guide 5 edition,dont add attibute to Object.prototype

**Options**

Options in jQuery are plain JavaScript objects. Whenever Options is mentioned as a type, that object and also all of its properties should be optional. There are exceptions where at least one option is required. jQuery's most prominent use of Options is its [AJAX](http://docs.jquery.com/Ajax/jQuery.ajax)-method. Nearly all jQuery plugins provide an Options-based API: They work without any configuration, but allow users to specify whatever customization they need.

Let's look at an example from the form plugin. It allows you to submit a form via AJAX with this simple line of code:

$("#myform").ajaxForm();

In that mode, it uses the form's action-attribute as the AJAX-URL and the form's method-attribute to determine whether to GET or POST the form. You can override both defaults by specifying them as options:

$("#myform").ajaxForm({

url: "mypage.php",

type: "POST"

});

**Array**

Arrays in JavaScript are mutable lists with a few built-in methods. You can define arrays using the array literal:

var x = [];

var y = [1, 2, 3];

The type of an array is "object":

typeof []; // "object"

typeof [1, 2, 3]; // "object"

Reading and writing elements to an array uses the array-notation:

x[0] = 1;

y[2] // 3

**Iteration**

An array has a length property that is useful for iteration:

for (var i = 0; i < a.length; i++) {

// Do something with a[i]

}

When performance is critical, reading the length property only once can help to speed things up. This should be used only when a performance bottleneck was discovered:

for (var i = 0, j = a.length; i < j; i++) {

// Do something with a[i]

}

Another variation defines a variable that is filled for each iteration, removing the array-notation from the loop-body. It does not work when the array contains 0 or empty strings!

for (var i = 0, item; item = a[i]; i++) {

// Do something with item

}

jQuery provides a generic [each-function](http://docs.jquery.com/Utilities/jQuery.each) to iterate over element of arrays, as well as properties of objects:

var x = [1, 2, 3];

jQuery.each(x, function(index, value) {

console.log("index", index, "value", value);

});

The drawback is that the callback is called in the context of each value, therefore you lose the context of your own object if applicable. More on this below at Functions.

The length property can also be used to add elements to the end of an array. That is equivalent to using the push-method:

var x = [];

x.push(1);

x[x.length] = 2;

x // 1, 2

You'll see both variations a lot when looking through JavaScript library code.

Other built-in methods are reverse, join, shift, unshift, pop, slice, splice and sort:

var x = [0, 3, 1, 2];

x.reverse() // [2, 1, 3, 0]

x.join(" – ") // "2 - 1 - 3 - 0"

x.pop() // [2, 1, 3]

x.unshift(-1) // [-1, 2, 1, 3]

x.shift() // [2, 1, 3]

x.sort() // [1, 2, 3]

x.splice(1, 2) // [2, 3]

Note: .unshift() method does not return a length property in Internet Explorer.

**Boolean Default**

An array, no matter if it has elements or not, never defaults to false:

![] // false

**Array<Type> Notation**

In the jQuery API you'll often find the notation of Array<Type>:

dragPrevention Array<String>

This indicates that the method doesn't only expect an array as the argument, but also specifies the expected type. The notation is borrowed from Java 5's generics notation (or C++ templates).

**Map**

The map type is used by the AJAX function to hold the data of a request. This type could be a string, an array<form elements>, a jQuery object with form elements or an object with key/value pairs. In the last case, it is possible to assign multiple values to one key by assigning an array.

{'key[]':['valuea','valueb']}

becomes on the server-side (in PHP):

$\_REQUEST['key'][0]="valuea";

$\_REQUEST['key'][1]="valueb";

in Rails or Merb:

params[:key] = ["valuea", "valueb"]

**Function**

A function in JavaScript can be either named or anonymous. Any function can be assigned to a variable or passed to a method, but passing member functions this way can cause them to be called in the context of another object (i.e. with a different "this" object).

function named() {}

var handler = function() {}

You see a lot of anonymous functions in jQuery code:

$(document).ready(function() {});

$("a").click(function() {});

$.ajax({

url: "someurl.php",

success: function() {}

});

The type of a function is "function".

**Arguments**

Inside a function a special variable "arguments" is always available. It's similar to an array in that it has a length property, but it lacks the built-in methods of an array. The elements of the pseudo-array are the argument of the function call.

function log(x) {

console.log(typeof x, arguments.length);

}

log(); // "undefined", 0

log(1); // "number", 1

log("1", "2", "3"); // "string", 3

The arguments object also has a callee property, which refers to the function you're inside of. For instance:

var awesome = function() { return arguments.callee }

awesome() == awesome // true

**Context, Call and Apply**

In JavaScript, the variable "this" always refers to the current context. By default, "this" refers to the window object. Within a function this context can change, depending on how the function is called.

All event handlers in jQuery are called with the handling element as the context.

$(document).ready(function() {

// this refers to window.document

});

$("a").click(function() {

// this refers to an anchor DOM element

});

You can specify the context for a function call using the function-built-in methods call and apply. The difference between them is how they pass arguments. Call passes all arguments through as arguments to the function, while apply accepts an array as the arguments.

function scope() {

console.log(this, arguments.length);

}

scope() // window, 0

scope.call("foobar", [1,2]); // "foobar", 1

scope.apply("foobar", [1,2]); // "foobar", 2

**Scope**

In JavaScript, all variables defined inside a function are only visible inside that function scope. Consider the following example:

// global

var x = 0;

(function() {

// private

var x = 1;

console.log(x); // 1

})();

console.log(x); // 0

It defines a variable *x* in the global scope, then defines an anonymous function and executes it immediately (the additional parentheses are required for immediate execution). Inside the function another variable *x* is defined with a different value. It is only visible within that function and doesn't overwrite the global variable.

**Closures**

Closures are created whenever a variable that is defined outside the current scope is accessed from within some inner scope. In the following example, the variable *counter* is visible within the create, increment, and print functions, but not outside of them.

function create() {

var counter = 0;

return {

increment: function() {

counter++;

},

print: function() {

console.log(counter);

}

}

}

var c = create();

c.increment();

c.print(); // 1

The pattern allows you to create objects with methods that operate on data that isn't visible to the outside—the very basis of object-oriented programming.

**Proxy Pattern**

Combining the above knowledge gives you as a JavaScript developer quite a lot of power. One way to combine that is to implement a proxy pattern in JavaScript, enabling the basics of aspect-oriented programming (AOP):

(function() {

// log all calls to setArray

var proxied = jQuery.fn.setArray;

jQuery.fn.setArray = function() {

console.log(this, arguments);

return proxied.apply(this, arguments);

};

})();

The above wraps its code in a function to hide the "proxied"-variable. It saves jQuery's setArray-method in a closure and overwrites it. The proxy then logs all calls to the method and delegates the call to the original. Using apply(this, arguments) guarantees that the caller won't be able to notice the difference between the original and the proxied method.

**Callback**

A callback is a plain JavaScript function passed to some method as an argument or option. Some callbacks are just events, called to give the user a chance to react when a certain state is triggered. jQuery's event system uses such callbacks everywhere:

$("body").click(function(event) {

console.log("clicked: " + event.target);

});

Most callbacks provide arguments and a context. In the event-handler example, the callback is called with one argument, an Event. The context is set to the handling element, in the above example, document.body.

Some callbacks are required to return something, others make that return value optional. To prevent a form submission, a submit event handler can return false:

$("#myform").submit(function() {

return false;

});

Instead of always returning false, the callback could check fields of the form for validity, and return false only when the form is invalid.

**Selector**

A selector is used in jQuery to select DOM elements from a DOM document. That document is, in most cases, the DOM document present in all browsers, but can also be a XML document received via AJAX.

The selectors are a composition of CSS and custom additions. XPath selectors are available as a plugin.

All selectors available in jQuery are documented on the [Selectors API page](http://docs.jquery.com/Selectors).

There are lot of plugins that leverage jQuery's selectors in other ways. The validation plugin accepts a selector to specify a dependency, whether an input is required or not:

emailrules: {

required: "#email:filled"

}

This would make a checkbox with name "emailrules" required only if the user entered an email address in the email field, selected via its id, filtered via a custom selector ":filled" that the validation plugin provides.

If Selector is specified as the type of an argument, it accepts everything that the jQuery constructor accepts, eg. Strings, Elements, Lists of Elements.

**[Event](http://docs.jquery.com/Events/jQuery.Event" \o "Events/jQuery.Event)**

jQuery's event system normalizes the event object according to W3C standards. The event object is guaranteed to be passed to the event handler (no checks for window.event required). It normalizes the target, relatedTarget, which, metaKey and pageX/Y properties and provides both stopPropagation() and preventDefault() methods.

Those properties are all documented, and accompanied by examples, on the [Event](http://docs.jquery.com/Events/jQuery.Event) page.

The standard events in the Document Object Model are: blur, focus, load, resize, scroll, unload, beforeunload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress, and keyup. Since the DOM event names have predefined meanings for some elements, using them for other purposes is not recommended. jQuery's event model can trigger an event by any name on an element, and it is propagated up the DOM tree to which that element belongs, if any.

**Element**

An element in the Document Object Model (DOM) has attributes, text and children. It provides methods to traverse the parent and children and to get access to its attributes. Due to a lot of flaws in DOM API specifications and implementations, those methods are no fun to use. jQuery provides a wrapper around those elements to help interacting with the DOM. But often enough you will be working directly with DOM elements, or see methods that (also) accept DOM elements as arguments.

Whenever you use jQuery's each-method, the context of your callback is set to a DOM element. That is also the case for event handlers.

Some properties of DOM elements are quite consistent among browsers. Consider this example of a simple on-blur-validation:

$(":text").blur(function() {

if(!this.value) {

alert("Please enter some text!");

}

});

You could replace this.value with $(this).val() to access the value of the text input via jQuery, but in that case you don't gain anything.

**jQuery**

A jQuery object contains a collection of Document Object Model (DOM) elements that have been created from an HTML string or selected from a document. Since jQuery methods often use CSS selectors to match elements from a document, the set of elements in a jQuery object is often called a set of "matched elements" or "selected elements".

The jQuery object itself behaves much like an array; it has a length property and the elements in the object can be accessed by their numeric indices [0] to [length-1]. Note that a jQuery object is not actually a Javascript Array object, so it does not have all the methods of a true Array object such as join().

Most frequently, you will use the jQuery() function to create a jQuery object. jQuery() can also be accessed by its familiar single-character alias of $(), unless you have called jQuery.noConflict() to disable this option. Many jQuery methods return the jQuery object itself, so that method calls can be chained:

$("p").css("color", "red").find(".special").css("color", "green");

Whenever you use a "destructive" jQuery method that potentially changes the set of elements in the jQuery object, such as .filter() or .find(), that method actually returns a new jQuery object with the resulting elements. To return to the previous jQuery object, you use the .end() method.

A jQuery object may be empty, containing no DOM elements. You can create an empty jQuery object with $() (that is, passing no arguments at all). A jQuery object may also be empty if a selector doesn't select any elements, or if a chained method filters out all the elements. It is not an error; any further methods called on that jQuery object simply have no effect since they have no elements to act upon. So, in this example if there are no bad entries on the page then no elements will be colored red:

$(".badEntry").css({color: "red"});

**XMLHttpRequest**

Some of jQuery's AJAX functions return the native XMLHttpRequest (XHR) object, or pass it as an argument to success/error/complete handlers, so that you can do additional processing or monitoring on the request. Note that AJAX functions only return or pass an XHR object when an XHR object is actually used in the request. For example, JSONP requests and cross-domain GET requests use a script element rather than an XHR object.

Although the XHR object is a standard, there are variations in its behavior on different browsers. Refer to the W3C site and browsers' documentation for more information:

* [W3C standard](http://www.w3.org/TR/XMLHttpRequest/)
* [Apple (Safari)](http://developer.apple.com/internet/webcontent/xmlhttpreq.html)
* [Mozilla (Firefox)](http://developer.mozilla.org/en/xmlhttprequest)
* [Microsoft (Internet Explorer)](http://msdn.microsoft.com/en-us/library/ms535874%28VS.85%29.aspx)
* [Opera](http://www.opera.com/docs/specs/opera9/xhr/)

Google does not appear to have an official page for their XHR documentation for Chrome. As of version 5, Chrome does not support the use of the file protocol for XHR requests.

**jQuery: The Basics**

This is a basic tutorial, designed to help you get started using jQuery. If you don't have a test page set up yet, start by creating a new HTML page with the following contents:

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Demo</title>

</head>

<body>

<a href="http://jquery.com/">jQuery</a>

<script src="jquery.js"></script>

<script>

</script>

</body>

</html>

Edit the src attribute in the script tag to point to your copy of jquery.js. For example, if jquery.js is in the same directory as your HTML file, you can use:

<script src="jquery.js"></script>

You can download your own copy of jQuery from the [Downloading jQuery](http://docs.jquery.com/Downloading_jQuery) page

**Launching Code on Document Ready**

The first thing that most Javascript programmers end up doing is adding some code to their program, similar to this:

window.onload = function(){ alert("welcome"); }

Inside of which is the code that you want to run right when the page is loaded. Problematically, however, the Javascript code isn't run until all images are finished downloading (this includes banner ads). The reason for using window.onload in the first place is that the HTML 'document' isn't finished loading yet, when you first try to run your code.

To circumvent both problems, jQuery has a simple statement that checks the document and waits until it's ready to be manipulated, known as the [ready event](http://docs.jquery.com/Events#ready.28_fn_.29):

$(document).ready(function(){

// Your code here

});

Inside the ready event, add a click handler to the link:

$(document).ready(function(){

**$("a").click(function(event){**

**alert("Thanks for visiting!");**

**});**

});

Save your HTML file and reload the test page in your browser. Clicking the link on the page should make a browser's alert pop-up, before leaving to go to the main jQuery page.

For click and most other [events](http://docs.jquery.com/Events), you can prevent the default behaviour - here, following the link to jquery.com - by calling event.preventDefault() in the event handler:

$(document).ready(function(){

$("a").click(function(**event**){

alert("As you can see, the link no longer took you to jquery.com");

**event.preventDefault();**

});

});

**Complete Example**

The following is an example of what the complete HTML file might look like if you were to use the script in your own file. Note that it links to Google's [CDN](http://code.google.com/apis/libraries/) to load the jQuery core file. Also, while the custom script is included in the <head>, it is generally preferable to place it in a separate file and refer that file with the script element's src attribute

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>jQuery demo</title>

</head>

<body>

<a href="<http://jquery.com/>">jQuery</a>

<script src="<http://ajax.googleapis.com/ajax/libs/jquery/1.5/jquery.min.js>"></script>

<script>

$(document).ready(function(){

$("a").click(function(event){

alert("As you can see, the link no longer took you to jquery.com");

event.preventDefault();

});

});

</script>

</body>

</html>

**Adding and Removing an HTML Class**

**Important:** *The remaining jQuery examples will need to be placed inside the ready event so that they are executed when the document is ready to be worked on. See* [*Launching Code on Document Ready*](http://docs.jquery.com/How_jQuery_Works#Launching_Code_on_Document_Ready) *above for details.*

Another common task is adding (or removing) a class.

First, add some style information into the <head> of your document, like this:

<style>

a.test { font-weight: bold; }

</style>

Next, add the [addClass](http://docs.jquery.com/Attributes/addClass) call to your script:

$("a").addClass("test");

All your a elements will now be bold.

To remove the class, use [removeClass](http://docs.jquery.com/Attributes/removeClass)

$("a").removeClass("test");

* (HTML allows multiple classes to be added to an element.)

**Special Effects**

In jQuery, a couple of handy [effects](http://docs.jquery.com/Effects) are provided, to really make your web site stand out. To put this to the test, change the click that you added earlier to this:

$("a").click(function(event){

event.preventDefault();

$(this).hide("slow");

});

Now, if you click any link, it should make itself slowly disappear.

**Callback and Functions**

A callback is a function that is passed as an argument to another function and is executed after its parent function has completed. The special thing about a callback is that functions that appear after the "parent" can execute before the callback executes. Another important thing to know is how to properly pass the callback. This is where I have often forgotten the proper syntax.

**Callback *without* arguments**

For a callback with no arguments you pass it like this:

$.get('myhtmlpage.html', myCallBack);

**Note** that the second parameter here is simply the function name (but *not* as a string and without parentheses). Functions in Javascript are 'First class citizens' and so can be passed around like variable references and executed at a later time.

**Callback *with* arguments**

"What do you do if you have arguments that you want to pass?", you might ask yourself.

**Wrong**

The Wrong Way (will **not** work!)

$.get('myhtmlpage.html', myCallBack('foo', 'bar'));

This will not work because it calls

myCallBack('foo', 'bar')

and then passes the return value as the second parameter to [$.get()](http://docs.jquery.com/Ajax)

**Right**

The problem with the above example is that myCallBack('foo', 'bar') is evaluated before being passed as a function. Javascript and by extension jQuery expects a function pointer in cases like these. I.E. setTimeout function.

In the below usage, an anonymous function is created (just a block of statements) and is registered as the callback function. Note the use of 'function(){'. The anonymous function does exactly one thing: calls myCallBack, with the values of 'foo' and 'bar'.

$.get('myhtmlpage.html', function(){

myCallBack('foo', 'bar');

});

myCallBack is invoked when the '$.get' is done getting the page.

**How do I ... ?**

**How do I select an item using class or ID?**

This code selects an element with an ID of "myDivId". Since IDs are unique, this expression always selects either zero or one elements depending upon whether or not an element with the specified ID exists.

$('#myDivId')

This code selects an element with a class of "myCssClass". Since any number of elements can have the same class, this expression will select any number of elements.

$('.myCssClass')

A jQuery object containing the selected element can be assigned to a JavaScript variable like normal:

var myDivElement = $('#myDivId');

Usually, elements in a jQuery object are acted on by other jQuery functions:

var myValue = $('#myDivId').val(); // get the value of a form input

$('#myDivId').val("hello world"); // set the value of a form input

**How do I select elements when I already have a DOM element?**

If you have a variable containing a DOM element, and want to select elements related to that DOM element, simply wrap it in a jQuery object.

var myDomElement = document.getElementById('foo'); // a plain DOM element

$(myDomElement).find('a'); // finds all anchors inside the DOM element

Many people try to concatenate a DOM element or jQuery object with a CSS selector, like so:

$(myDomElement + '.bar'); // WRONG! equivalent to $("[object HTMLElement].bar")

This is wrong. You cannot concatenate strings to objects.

**How do I test whether an element has a particular class?**

[hasClass](http://docs.jquery.com/Traversing/hasClass) (added in version 1.2) handles this common use case:

$("div").click(function(){

if ( $(this).hasClass("protected") )

$(this)

.animate({ left: -10 })

.animate({ left: 10 })

.animate({ left: -10 })

.animate({ left: 10 })

.animate({ left: 0 });

});

You can also use the [is()](http://docs.jquery.com/DOM/Traversing#is.28_expr_.29) method along with an appropriate selector for more advanced matching:

if ( $('#myDiv').is('.pretty.awesome') )

$('#myDiv').show();

Note that this method allows you to test for other things as well. For example, you can test whether an element is hidden (by using the custom [:hidden](http://docs.jquery.com/Selectors#Custom_Selectors_2) selector):

if ( $('#myDiv').is(':hidden') )

$('#myDiv').show();

**How do I test whether an element exists?**

Use the [length](http://docs.jquery.com/Core#length) property of the jQuery collection returned by your selector:

if ( $('#myDiv').length )

$('#myDiv').show();

Note that it isn't always necessary to test whether an element exists. The following code will show the element if it exists, and do nothing (with no errors) if it does not:

$('#myDiv').show();

**How do I determine the state of a toggled element?**

You can determine whether an element is collapsed or not by using the :visible and :hidden selectors.

var isVisible = $('#myDiv').is(':visible');

var isHidden = $('#myDiv').is(':hidden');

If you're simply acting on an element based on its visibility, just include ":visible" or ":hidden" in the selector expression. For example:

$('#myDiv:visible').animate({left: '+=200px'}, 'slow');

**How do I select an element by an ID that has characters used in CSS notation?**

Because jQuery uses CSS syntax for selecting elements, some characters are interpreted as CSS notation. For example, ID attributes, after an initial letter (a-z or A-Z), may also use periods and colons, in addition to letters, numbers, hyphens, and underscores (see W3C [Basic HTML Data Types](http://www.w3.org/TR/html4/types.html#type-id)). The colon (":") and period (".") are problematic within the context of a jQuery selector because they indicate a pseudo-class and class, respectively.

In order to tell jQuery to treat these characters literally rather than as CSS notation, they must be "escaped" by placing two backslashes in front of them.

// Does not work

$("#some:id")

// Works!

$("#some\\:id")

// Does not work

$("#some.id")

// Works!

$("#some\\.id")

The following function takes care of escaping these characters and places a "#" at the beginning of the ID string:

function jq(myid) {

return '#' + myid.replace(/(:|\.)/g,'\\$1');

}

The function can be used like so:

$( jq('some.id') )

**How do I disable/enable a form element?**

There are two ways to disable/enable form elements.

Set the 'disabled' attribute to true or false:

// Disable #x

$('#x').attr('disabled', true);

// Enable #x

$('#x').attr('disabled', false);

Add or remove the 'disabled' attribute:

// Disable #x

$("#x").attr('disabled', 'disabled');

// Enable #x

$("#x").removeAttr('disabled');

You can try an example of enabling/disabling with the following demo:



and here's the source code to the demo:

<select id="x" style="width:200px;">

<option>one</option>

<option>two</option>

</select>

<input type="button" value="Disable" onclick="$('#x').attr('disabled','disabled')"/>

<input type="button" value="Enable" onclick="$('#x').removeAttr('disabled')"/>

**How do I check/uncheck a checkbox input or radio button?**

There are two ways to check/uncheck a checkbox/radio button.

Set the 'checked' attribute to true or false.

// Check #x

$('#x').attr('checked', true);

// Uncheck #x

$('#x').attr('checked', false);

Add or remove the 'checked' attribute:

// Check #x

$("#x").attr('checked', 'checked');

// Uncheck #x

$("#x").removeAttr('checked');

You can try an example of checking/unchecking with the following demo:

I'll be checked/unchecked.

and here's the source code to the demo:

<label><input type="checkbox" id="c"/> I'll be checked/unchecked.</label>

<input type="button" value="Check" onclick='$("#c").attr("checked","checked")'/>

<input type="button" value="Uncheck" onclick='$("#c").removeAttr("checked")'/>

**How do I get the text value of a selected option?**

Select elements typically have two values that you want to access. First there's the value to be sent to the server, which is easy:

$("#myselect").val();

// => 1

The second is the text value of the select. For example, using the following select box:

<select id="myselect">

<option value="1">Mr</option>

<option value="2">Mrs</option>

<option value="3">Ms</option>

<option value="4">Dr</option>

<option value="5">Prof</option>

</select>

If you wanted to get the string "Mr" if the first option was selected (instead of just "1"), you would do that in the following way:

$("#myselect option:selected").text();

// => "Mr"

You can see this in action in the following demo:



and here's the full source code to the demo:

<select id="myselect">

<option value="1">Mr</option>

<option value="2">Mrs</option>

<option value="3">Ms</option>

<option value="4">Dr</option>

<option value="5">Prof</option>

</select>

<input type="button" value="Get Value" onclick="alert($('#myselect').val())"/>

<input type="button" value="Get Text Value" onclick="alert($('#myselect option:selected').text())"/>

**How do I replace text from the 3rd element of a list of 10 items?**

Either the :eq() selector or the .eq() method will allow you to select the proper item. However, to replace the text, you must *get* the value before you *set* it:

// This doesn't work; text() returns a string, not the jQuery object

$(this).find('li a').eq(2).text().replace('foo','bar');

// This works

var $thirdLink = $(this).find('li a').eq(2);

var linkText = $thirdLink.text().replace('foo','bar');

$thirdLink.text(linkText);

The first example just discards the modified text. The second example saves the modified text and then replaces the old text with the new modified text. Remember, .text() *gets*; .text("foo") *sets*.

**How do I compress my code?**

Generally the best way to do it is to use the [Google Closure Compiler](http://code.google.com/closure/compiler/) (used by jQuery) or [YUI compressor](http://developer.yahoo.com/yui/compressor/). jQuery provides a pre-[minified version of jQuery](http://code.jquery.com/jquery-latest.min.js) for your convenience.

Packing JavaScript using [Dean Edwards' Packer](http://dean.edwards.name/packer/) (specifically using the base64 encode) is not recommended, as the client-side decoding has significant overhead that outweighs the file-size benefits.

If compressing your JavaScript breaks it, try running the code through [JSLint](http://jslint.com/). This will detect minor errors that can cause packed JavaScript to fail where the unpacked version works fine.

**How do I submit a bug report?**

You can [submit a bug report](http://dev.jquery.com/newticket) through the [jQuery bug tracker](http://dev.jquery.com/).

Any information you can provide will help, such as:

* A detailed bug report,
* An online demo page, showing the problem,
* A specific piece of code that is affected, or
* A pointer to the area in jQuery where the bug occurs.

The more information a bug report has, the more likely it will be to get fixed. If a long period of time has gone by without an update to your bug, please bring it up for discussion on the [jQuery Dev List](http://docs.jquery.com/Discussion).

**How do I get and use the server response from an AJAX request?**

The 'A' in AJAX stands for asynchronous. When invoking functions that have asynchronous behavior you must provide a callback function to capture the desired result. This is especially important with AJAX in the browser because when a remote request is made, it is indeterminate when (or even if) the response will be received.

The following snippet shows an example of making an AJAX call and alerting the response (or error):

$.ajax({

url: 'myPage.php',

success: function(response) {

alert(response);

},

error: function(xhr) {

alert('Error! Status = ' + xhr.status);

}

});

But how can the response be used in context of a function? Consider this flawed example where we try to update some status information on the page:

function updateStatus() {

var status;

$.ajax({

url: 'getStatus.php',

success: function(response) {

status = response;

}

});

// update status element? this will not work as expected

$('#status').html(status);

}

The code above does not work as desired due to the nature of asynchronous programming. The provided success handler is not invoked immediately, but rather at some time in the future when the response is received from the server. So when we use the 'status' variable immediately after the $.ajax call, its value is still undefined. The next snippet shows how we can rewrite this function to behave as desired:

function updateStatus() {

$.ajax({

url: 'getStatus.php',

success: function(response) {

// update status element

$('#status').html(response);

}

});

}

But how can I return the server response from an AJAX call? Here again we show a flawed attempt. In this example we attempt to alert the http status code for the url of 'getStatus.php':

//...

alert(getUrlStatus('getStatus.php'));

//...

function getUrlStatus(url) {

$.ajax({

url: url,

complete: function(xhr) {

return xhr.status;

}

});

}

The code above will not work because you cannot 'return' data from a function that is called asynchronously. Instead, it must be rewritten to use a callback:

//...

getUrlStatus('getStatus.php', function(status) {

alert(status);

});

// ...

function getUrlStatus(url, callback) {

$.ajax({

url: url,

complete: function(xhr) {

callback(xhr.status);

}

});

}

**How do I pull a native DOM element from a jQuery object?**

A jQuery object is an array-like wrapper around one or more DOM elements. To get a reference to the actual DOM elements (instead of the jQuery object), you have two options. The first (and fastest) method is to use array notation:

$('#foo')[0]; // equivalent to document.getElementById('foo')

The second method is to use the [get](http://docs.jquery.com/action/edit/Miscellaneous/get) function:

$('#foo').get(0); // identical to above, only slower

You can also call [get](http://docs.jquery.com/action/edit/Miscellaneous/get) without any arguments to retrieve a true array of DOM elements.

**Why do ... ?**

**Why do my events stop working after an AJAX request?**

Frequently, when you've added a click (or other event) handler to all links using $('a').click(fn), you'll find that the events no longer work after you've loaded new content into a page using an AJAX request.

When you call $('a'), it returns all the links on the page *at the time it was called*, and .click(fn) adds your handler to only those elements. When new links are added, they are not affected. See the [AJAX and Events Tutorial](http://docs.jquery.com/Tutorials:AJAX_and_Events) for a longer discussion.

You have two ways of handling this:

**Using event delegation**

[Event](http://www.cherny.com/webdev/70/javascript-event-delegation-and-event-hanlders) [delegation](http://icant.co.uk/sandbox/eventdelegation/) is a technique that exploits [event bubbling](http://www.quirksmode.org/js/events_order.html#link3) to capture events on elements anywhere in the DOM.

As of **jQuery 1.3**, you can use the [live](http://docs.jquery.com/Events/live) and [die](http://docs.jquery.com/Events/die) methods for event delegation with a subset of event types. As of **jQuery 1.4**, you can use these methods (along with [delegate](http://docs.jquery.com/action/edit/Events/delegate) and [undelegate](http://docs.jquery.com/action/edit/Events/undelegate) starting in 1.4.2) for event delegation with pretty much any event type.

For earlier versions of jQuery, take a look at the [Live Query plugin](http://plugins.jquery.com/project/livequery/) by [Brandon Aaron](http://brandonaaron.net/). You may also manually handle event delegation by binding to a common container and listening for events from there. For example:

$('#mydiv').click(function(e){

if( $(e.target).is('a') )

fn.call(e.target,e);

});

$('#mydiv').load('my.html');

This example will handle clicks on any <a> element within #mydiv, even if they do not exist yet when the click handler is added.

**Using event rebinding**

This method requires you to call the [bind](http://docs.jquery.com/Events/bind) method on new elements as they are added. For example:

$('a').click(fn);

$('#mydiv').load('my.html',function(){

$('#mydiv a').click(fn);

});

**Beware!** As of jQuery 1.4.2, binding the same handler to the same element multiple times will cause it to execute more than once. This differs from previous versions of jQuery as well as the DOM 2 Events spec (which normally ignores duplicate event handlers).

To learn more about event rebinding [read this article](http://www.learningjquery.com/2008/05/working-with-events-part-2) on the [Learning jQuery](http://www.learningjquery.com/) blog.

**Why doesn't an event work on a new element I've created?**

As explained in the previous question about AJAX, events are bound only to elements that exist at the time when you issue your initial jQuery call. When you create a new element, you must bind the event to it separately, or use event delegation.

**Why do animations set the display style to block?**

Only block-level elements can have a custom width or height. When you do an animation on an element that animates the height or width (such as show, hide, slideUp, or slideDown), the display CSS property will be set to 'block' for the duration of the animation. The display property will be reverted to its original value after the animation completes. (This does not work properly for inline-block elements.)

There are two common workarounds:

If you want the element to stay inline, but you just want it to animate in or out, you can use the fadeIn or fadeOut animations instead (which only affect the opacity of an element).

// Instead of this:

$("span").show("slow");

// do this:

$("span").fadeIn("slow");

The other option is to use a block-level element, but to add a float such that it appears to stay inline with the rest of the content around it. The result might looks something like this:

// A floated block element

<div style="float:left;">...</div>

// Your code:

$("div").show("slow");