**Manipulating Elements: Part 2**

**Moving, Copying, and Removing Elements**

There are a variety of ways to move elements around the DOM; generally, there are two approaches:

* Place the selected element(s) relative to another element
* Place an element relative to the selected element(s)

For example, jQuery provides $.fn.insertAfter and $.fn.after. The $.fn.insertAfter method places the selected element(s) after the element that you provide as an argument; the $.fn.after method places the element provided as an argument after the selected element. Several other methods follow this pattern: $.fn.insertBefore and $.fn.before;$.fn.appendTo and $.fn.append; and $.fn.prependTo and $.fn.prepend.The method that makes the most sense for you will depend on what elements you already have selected, and whether you will need to store a reference to the elements you're adding to the page. If you need to store a reference, you will always want to take the first approach -- placing the selected elements relative to another element -- as it returns the element(s) you're placing. In this case, $.fn.insertAfter, $.fn.insertBefore, $.fn.appendTo, and $.fn.prependTo will be your tools of choice.

**Example 3.27. Moving elements using different approaches**

// make the first list item the last list item

var $li = $('#myList li:first').appendTo('#myList');

// another approach to the same problem

$('#myList').append($('#myList li:first'));

// note that there's no way to access the

// list item that we moved, as this returns

// the list itself

**Cloning Elements**

When you use methods such as $.fn.appendTo, you are moving the element; sometimes you want to make a copy of the element instead. In this case, you'll need to use $.fn.clone first.

**Example 3.28. Making a copy of an element**

// copy the first list item to the end of the list

$('#myList li:first').clone().appendTo('#myList');

**Note**

If you need to copy related data and events, be sure to pass true as an argument to $.fn.clone.

**Removing Elements**

There are two ways to remove elements from the page: $.fn.remove and $.fn.detach. You'll use $.fn.remove when you want to permanently remove the selection from the page; while the method does return the removed element(s), those elements will not have their associated data and events attached to them if you return them to the page.

If you need the data and events to persist, you'll want to use $.fn.detach instead. Like $.fn.remove, it returns the selection, but it also maintains the data and events associated with the selection, so you can restore the selection to the page at a later time.

**Note**

The $.fn.detach method is extremely valuable if you are doing heavy manipulation to an element. In that case, it's beneficial to $.fn.detach the element from the page, work on it in your code, and then restore it to the page when you're done. This saves you from expensive "DOM touches" while maintaining the element's data and events.

If you want to leave the element on the page but simply want to remove its contents, you can use $.fn.empty to dispose of the element's inner HTML.