**Increasing Performance with Event Delegation**

You'll frequently use jQuery to add new elements to the page, and when you do, you may need to bind events to those new elements -- events you already bound to similar elements that were on the page originally. Instead of repeating your event binding every time you add elements to the page, you can use event delegation. With event delegation, you bind your event to a container element, and then when the event occurs, you look to see which contained element it occurred on. If this sounds complicated, luckily jQuery makes it easy with its $.fn.live and $.fn.delegate methods.

While most people discover event delegation while dealing with elements added to the page later, it has some performance benefits even if you never add more elements to the page. The time required to bind event handlers to hundreds of individual elements is non-trivial; if you have a large set of elements, you should consider delegating related events to a container element.

**Note**

The $.fn.live method was introduced in jQuery 1.3, and at that time only certain event types were supported. As of jQuery 1.4.2, the $.fn.delegate method is available, and is the preferred method.

**Example 5.10. Event delegation using $.fn.delegate**

$('#myUnorderedList').delegate('li', 'click', function(e) {

var $myListItem = $(this);

// ...

});

**Example 5.11. Event delegation using $.fn.live**

$('#myUnorderedList li').live('click', function(e) {

var $myListItem = $(this);

// ...

});

**Unbinding Delegated Events**

If you need to remove delegated events, you can't simply unbind them. Instead, use $.fn.undelegate for events connected with $.fn.delegate, and $.fn.die for events connected with $.fn.live. As with bind, you can optionally pass in the name of the bound function.

**Example 5.12. Unbinding delegated events**

$('#myUnorderedList').undelegate('li', 'click');

$('#myUnorderedList li').die('click');