Connecting Events to Elements

jQuery offers convenience methods for most common events, and these are the methods you will see used most often. These methods -- including $.fn.click, $.fn.focus, $.fn.blur, $.fn.change, etc. -- are shorthand for jQuery's $.fn.bind method. The bind method is useful for binding the same hadler function to multiple events, and is also used when you want to provide data to the event hander, or when you are working with custom events.

Example 5.1. Event binding using a convenience method

$('p').click(function() {

console.log('click');

});

Example 5.2. Event biding using the $.fn.bind method

$('p').bind('click', function() {

console.log('click');

});

Example 5.3. Event binding using the $.fn.bind method with data

$('input').bind(

'click change', // bind to multiple events

{ foo : 'bar' }, // pass in data

function(eventObject) {

console.log(eventObject.type, eventObject.data);

// logs event type, then { foo : 'bar' }

}

);

Connecting Events to Run Only Once

Sometimes you need a particular handler to run only once -- after that, you may want no handler to run, or you may want a different handler to run. jQuery provides the $.fn.one method for this purpose.

Example 5.4. Switching handlers using the $.fn.one method

$('p').one('click', function() {

$(this).click(function() { console.log('You clicked this before!'); });

});

The $.fn.one method is especially useful if you need to do some complicated setup the first time an element is clicked, but not subsequent times.

Disconnecting Events

To disconnect an event handler, you use the $.fn.unbind method and pass in the event type to unbind. If you attached a named function to the event, then you can isolate the unbinding to that named function by passing it as the second argument.

Example 5.5. Unbinding all click handlers on a selection

$('p').unbind('click');

Example 5.6. Unbinding a particular click handler

var foo = function() { console.log('foo'); };

var bar = function() { console.log('bar'); };

$('p').bind('click', foo).bind('click', bar);

$('p').unbind('click', bar); // foo is still bound to the click event

Namespacing Events

For complex applications and for plugins you share with others, it can be useful to namespace your events so you don't unintentionally disconnect events that you didn't or couldn't know about.

Example 5.7. Namespacing events

$('p').bind('click.myNamespace', function() { /\* ... \*/ });

$('p').unbind('click.myNamespace');

$('p').unbind('.myNamespace'); // unbind all events in the namespace