**Data Methods**

As your work with jQuery progresses, you'll find that there's often data about an element that you want to store with the element. In plain JavaScript, you might do this by adding a property to the DOM element, but you'd have to deal with memory leaks in some browsers. jQuery offers a straightforward way to store data related to an element, and it manages the memory issues for you.

**Example 4.2. Storing and retrieving data related to an element**

$('#myDiv').data('keyName', { foo : 'bar' });

$('#myDiv').data('keyName'); // { foo : 'bar' }

You can store any kind of data on an element, and it's hard to overstate the importance of this when you get into complex application development. For the purposes of this class, we'll mostly use $.fn.data to store references to other elements.

For example, we may want to establish a relationship between a list item and a div that's inside of it. We could establish this relationship every single time we interact with the list item, but a better solution would be to establish the relationship once, and then store a pointer to the div on the list item using $.fn.data:

**Example 4.3. Storing a relationship between elements using $.fn.data**

$('#myList li').each(function() {

var $li = $(this), $div = $li.find('div.content');

$li.data('contentDiv', $div);

});

// later, we don't have to find the div again;

// we can just read it from the list item's data

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

$firstLi.data('contentDiv').html('new content');

In addition to passing $.fn.data a single key-value pair to store data, you can also pass an object containing one or more pairs.