**Append Outside of Loops**

Touching the DOM comes at a cost. If you're appending a lot of elements to the DOM, you will want to append them all at once, rather than one at a time. This is a common problem when appending elements within a loop.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | $.each( myArray, **function**( i, item ) {    **var** newListItem = "<li>" + item + "</li>";    $( "#ballers" ).append( newListItem );    }); |

Another simple technique is to build up a string during each iteration of the loop. After the loop, just set the HTML of the DOM element to that string.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | **var** myHtml = "";    $.each( myArray, **function**( i, item ) {    myHtml += "<li>" + item + "</li>";    });    $( "#ballers" ).html( myHtml ); |

There are of course other techniques you could certainly test out. A great way to test the performance of these is through a site called [jsperf](http://jsperf.com/).

**Cache Length During Loops**

In a for loop, don't access the length property of an array every time; cache it beforehand.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | **var** myLength = myArray.length;    **for** ( **var** i = 0; i < myLength; i++ ) {    *// do stuff*    } |

**Don’t Act on Absent Elements**

jQuery won't tell you if you're trying to run a whole lot of code on an empty selection – it will proceed as though nothing's wrong. It's up to you to verify that your selection contains some elements.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | *// Bad: This runs three functions before it*  *// realizes there's nothing in the selection*  $( "#nosuchthing" ).slideUp();    *// Better:*  **var** $mySelection = $( "#nosuchthing" );    **if** ( $mySelection.length ) {    $mySelection.slideUp();    } |

**Optimize Selectors**

Selector optimization is less important than it used to be, as more browsers implement document.querySelectorAll() and the burden of selection shifts from jQuery to the browser. However, there are still some tips to keep in mind.

[link](http://learn.jquery.com/performance/optimize-selectors/#id-based-selectors)**ID-Based Selectors**

Beginning your selector with an ID is always best.

|  |  |
| --- | --- |
| 1  2  3  4  5 | *// Fast:*  $( "#container div.robotarm" );    *// Super-fast:*  $( "#container" ).find( "div.robotarm" ); |

The .find() approach is faster because the first selection is handled without going through the Sizzle selector engine – ID-only selections are handled using document.getElementById(), which is extremely fast because it is native to the browser.

**Use Stylesheets for Changing CSS on Many Elements**

If you're changing the CSS of more than 20 elements using .css(), consider adding a style tag to the page instead for a nearly 60% increase in speed.

|  |  |
| --- | --- |
| 1  2  3  4 | *// Fine for up to 20 elements, slow after that:*  $( "a.swedberg" ).css( "color", "#0769ad" );    *// Much faster:*  $( "<style type=\"text/css\">a.swedberg { color: #0769ad }</style>")  .appendTo( "head" ); |