Using Event Bubbling to Your Advantage

Posted At : June 12, 2009 11:00 AM | Posted By : Jon Hartmann
Related Categories: jQuery, Javascript

Its happened to everyone working with Javascript: you build your test app that handles clicks on elements to do really neat things and everything's going great. You move it live, and as the data starts coming in your app gets slower and slower until finally the system starts to become unusable. Whats the problem? In testing you had maybe five or ten clickables, but your app has been growing, and you've now got 20, 60, 120 or more, and the time it takes to bind all of your click events is skyrocketing. What if you could just use a single handler for all of them?

Thanks to the magic of event bubbling, you can use just a single event handler to handle that even on any number of child elements. How does this work? Well when an event is registered, its passed up the chain of its ancestors until someone handles the event. This means that you can define a single handler on a higher level element to handle all of the events that happen within it.

The Example

Ok, to test this out, lets get a simple example bit of code:

```
<html>
    <head>
        <title>Test</title>
    </head>
    <body>
        <ul id="test">
            <li>One</li>
            <li>Two</li>
            <li>Three</li>
            <li>Four</li>
        </ul>
    </body>
</html>
```

Ok, so now lets say that we want to write some Javascript that alerts the contents of an element when you click on it. In this example you could define four separate click handlers handles like in the following jQuery example:

```
<script src="jquery-1.3.2.js"></script>
<script>


    $(document).ready(function () {
        $('#test li').click(function(event) {
            var elem = $(event.target);
            alert(elem.html())
        });
    });
</script>
```

Or, you could just drop that " li" from the selector and do it like this:

```
<script src="jquery-1.3.2.js"></script>
<script>
    $(document).ready(function () {
```

```
        $('#test').click(function(event) {
            var elem = $(event.target);
            alert(elem.html())
        });
    });
</script>
```

Both will work to alert you when you click on the element, but one uses only a single event handler, rather than four. This scales up so that you could have this single function working over any number of elements.

The one big "gotcha" with this technique is that the containing element's event handler is going to fire for every event of that type registered on any element inside it. Say you have a mix of types, and you only want some of them to alert their contents: to handle this you'll have to create some kind of filtering in the event handler itself, rather than just registering a handler on each element of that type. Its a little bit more overhead, but you should still be realizing massive savings.