Preventing Multiple Submissions with Prototype and jQuery

Posted At : June 11, 2009 6:56 PM | Posted By : Jon Hartmann
Related Categories: User Interface Design, jQuery, Javascript

If there is one problem that plagues me across applications, its users that are too antsy to wait for the page to come back after hitting submit on a form. Even worse, some people just instinctually double and triple click. How do you keep these pesky users from duplicating records or charging themselves three times for that item in your e-shop? Click "more" to see how I do it in Prototype and jQuery.

So, how do we user Javascript to prevent multiple submissions? The easiest way is to disable the submit button on the page, but some browser's don't play nice if we simply set the element to disabled, so we have to get a little fancier than that. We also want to let the user know that we're doing something, and that its not just that their button broke. In order to find a really nice solution, we need to hide the original button and put in it's place a look-alike button that is disabled and has an appropriate name.

The Base

Both of these examples will use the same base code and the same **Javascript gerundize function** that I've written about in previous posts. That code looks like this:

test.cfm

```
<html>
    <head>
        <title>Test</title>
        <InvalidTag>
            function gerundize (x) {
                var len = x.length;
                switch(x.slice(len-1, len)) {
                    case 'e':
                        return x.slice(0, len-1) + 'ing';
                    default:
                        return x + 'ing';
                }
            };
        </script>
    </head>
    <body>
        <cfset sleep(1000)>
        <form method="post">
            <input type="hidden" value="x" name="test" />
            <input type="submit" name="test2" value="Test" />
        </form>
        <cfdump var="#form#">
    </body>
</html>
```

And into that framework I'll add my Prototype and jQuery based solutions. The and are not strictly necessary, but they help make sure you can see whats going on by making the response take a little while and by showing you the form values.

Prototype

Since its my old stand-by, Prototype is up first. The first thing you'll want to do is **grab a copy of prototype-1.6.0.3.js** from

PrototypeJS.org, and include it in your HTML thusly:

```
...
<title>Test</title>
<InvalidTag src="prototype-1.6.0.3.js"></script>
...
```

And now we'll want to create a function to handle creating the new input and hiding the old one and bind it to the click events for every submit button once the DOM is ready.

```
<InvalidTag>
    function preventMultipleSubmission (event) {
        // We will fill this in next.
    }

    document.observe('dom:loaded', function () {
        $$('[type=submit]').invoke('observe', 'click', preventMultipleSubmission.bindAsEventListener());
    });
</script>
```

Basically that creates an empty function called preventMultipleSubmission() that we will use later, and then sets up an event handler for when the DOM is ready. Once it runs, we get all elements that have a type value of submit, and then set the observe function to watch for all clicks and call preventMultipleSubmission(). The "bindAsEventListener()" part tells Prototype that it needs to pass in the appropriate event objects too. Next up we fill in preventMultipleSubmission().

```
function preventMultipleSubmission (event) {
    var button = $(Event.element(event));
    button.insert({
        'after': new Element('input', {
            type: 'button',
            value: gerundize(button.value) + '...'
        }).disable()
    }).wrap(new Element('div').hide());
}
```

What does that do? Well its a little complicated due to chaining, but the first line gets the element that we clicked on (the button), and the next bit creates a new input element with a value based on our original button's text, disables it, and then inserts it right next to our original button. The last line creates a new

, makes it hidden, and then wraps it around the original button to hide it. Pretty cool, huh?

jQuery

Ok, so I'm new to jQuery, so if this example looks sloppy, please just point it out in the comments and I'll clean things up :) Like before, we need to snag a current version, so head on over to jQuery.com and snag yourself a copy of jQuery-1.3.2.js, and pug it into our base testing code:

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

And now lets get our first block of code filled in:

```
<InvalidTag>
    function preventMultipleSubmission (event) {
        // Not yet...
    }

    $(document).ready(function() {
        $(':submit').bind('click', preventMultipleSubmission);
    });
</script>
```

This is doing the same thing as our Prototype version, just with a slightly different syntax. We wait until the DOM is ready, then get all submit items and then bind preventMultipleSubmission() to their click event. Now to make it do the rest.

```
function preventMultipleSubmission (event) {
    var button = $(event.target);


    button.after(
        $('<input />')
            .attr('type', 'button')
            .val(gerundize(button.val()))
            .attr('disabled', 'disabled')
    ).wrap(
        $('<div />').css('display', 'none')
    );
}
```

As before, this code gets the current event target, then creates new elements and appends and wraps them where needed to make a new dummy element that is disabled and then to hide the original. About the only thing that I couldn't do exactly the same were the disable() and hide() methods from Prototype. Despite thinking I'd seen them in jQuery, I couldn't find an equivalent.

The End

Either way you pull this off, it looks great to the user and keeps them from submitting multiple times. The gerundize() function gives you some nice flexibility, as it will handle changing a number of different "save" type words into their active forms.