How jQuery handles HTML attributes: problems and solutions

jQuery provides the `.attr()` method as a getter/setter for handling HTML attributes. When this method is used as a setter, it's simply behaves like a more sophisticated wrapper around the DOM's `setAttribute()` method. Problems arise when this method is used as a getter.

The DOM Level 2 specifications define the `getAttribute()` method for every HTML element. This method returns *the attribute's value as a string, or the empty string if that attribute does not have a specified or default value*.

Further, the same specifications define the `hasAttribute()` method, which tests if an element has a given attribute by returning a Boolean value:

```
var element = document.getElementById('test');
if(!element.hasAttribute('title')) {
        element.setAttribute('title', 'Test');
}
```

jQuery, instead, provides only the `.attr()` method. This method usually returns `undefined` when an element doesn't have a given attribute:

```
var noAttr = $('#test').attr('title');
if(typeof noAttr == 'undefined') {
    $('#test').attr('title', 'Test');
}
```

This approach, however, is error-prone, so the best way to handle this particular situation is to use the more reliable DOM's method:

```
(function($) {
        if(typeof $.fn.hasAttr !== 'function') {
                $.fn.hasAttr = function(attr) {
                        var $element = $(this),
                                element =
$element[0];

if(!element.hasAttribute(attr)) {
                                return false;
                        }

                        return true;
                };
        }
})(jQuery);
```

I've summarized the jQuery's behavior in the following test page.