jQuery: add public properties and methods to plugins using the fn object

In a previous post I've showed you how to extend jQuery plugins by adding methods and properties directly to the element object wrapped within the plugin. This approach has some drawbacks, so it's better to add methods and properties which need to be public by using the `fn` object.

The `fn` object is an alias of the jQuery's `prototype` object. When you define a plugin you're actually augmenting the jQuery's `prototype` object. We can follow the prototypical chain to make our plugin's methods and properties public:

```
(function($) {
    $.fn.plugin = function(options) {
        options = $.extend({
            test: 'Test'
        }, options);
        $.fn.plugin.method = function(text) {
            return '<strong>' + text + '</strong>';
        };

        $.fn.plugin.settings = options;

        return this.each(function() {
            var $element = $(this);

$element.html($.fn.plugin.method(options.test));
```

```
        });
    };
})(jQuery);
```

A simple test:

```
alert($('#test').plugin.method('Foo')); //
'<strong>Foo</strong>'
alert($('#test').plugin.settings.test); // 'Test'
```

Now our methods and properties belong to the plugin namespace and are perfectly visible from outside the plugin's code. This is the same approach followed by some popular plugins, including Nivo Slider and Cycle.

You can see the demo on this page.