# How to Create Your Own jQuery Plugin

## Why Create a jQuery Plugin?

* Reuse, Reuse, & Reuse
* Encapsulation
* Prevent Collisions using Namespaces
* They are easy to create
* Because they are soooo cool

## How Does a jQuery Plugin Work?

* Plugins are passed the **this** object which references the jQuery object
* Plugin manipulates the DOM
* When finished, the Plugin should return **this** to facilitate chaining

## Plugin Declaration

* jQuery.prototype.tooltip = function(options) {
* $.fn.tooltip = function(options) {
* Self Invoking Anonymous Function

(function($){

$.fn.tooltip =

function(options) { ... }

})(jQuery);

## Plugin Parameters

* + You could pass individual parameters to your jQuery Plugin

$.fn.tooltip =

function(**text, color**) {};

  $('#helloWorld')

   .tooltip(**'Hello World!',**

**'#cccccc'**);

* + However, I am more a proponent of passing a JSON object...

* + Provide a set of Public Accessible Default Plugin Settings for User to Manipulate
  + Accept Custom Settings from the User
  + Merge the Default Settings and Custom Settings and use the Union within the Plugin

  $.fn.tooltip = function(**options**){};

  $('#helloWorld').tooltip**({**

**text: 'Hello World!',**

**color: '#cccccc'**

**}**);

## For Each

* + Note: The Plugin may have been passed one or more elements

* + A common technique is to use jQuery's each method to enumerate through the elements

**this.each(function() {**    
    var $this = $(this);    
    //etc...

**}**

## Public Functions

$.fn.tooltip = function(options) {  
  return this.each(function() {  
    var $this = $(this);  
    **var message =**

**$.fn.tooltip.public();**  
  });  
 };  
  
 **$.fn.tooltip.public = function(){**

**return 'Hello World!';  
 };**

## Private Functions

(function($) {  
  $.fn.tooltip = function(options) {

**var message = private();**  
  };  
    
**function private() {**

**return 'Hello World!';  
  };**  
 })(jQuery);

## Support the Metadata Plugin

* + Provide Additional Information to a particular element…

<li

class="someclass **{some: 'data'}**">

</li>  
  
 <li **data="{some:'random',**

**json: 'data'}"**></li>  
  
Etc…

  $.fn.tooltip = function(options) {   
    var opts = $.extend({},

      $.fn.tooltip.defaults, options);  
    return this.each(function() {  
      var $this = $(this);  
**//Element Specific Options  
      var o = $.meta ? $.extend({},**

**opts, $this.data()): opts;**

//Plugin Logic Here…

    }

  }

## Allow for Chaining

* + As with most jQuery functions and Plugins, it is desirable to return the jQuery object to facilitate chaining

**$(‘a’).css(‘color’, ‘red’).fadeIn();**

  (function($){

    $.fn.tooltip = function(options){   
      **return this; //chainability**    
    }    
  })(jQuery);