There were several goals to this code snippet.

1. To stay with just the jQuery-UI Library and not have to add an additional library for the effect.
2. To have code to the page that would add the dialog behavior to links of a specific class regardless of what page in the CMS the link was on.
3. To load the dialog by ajax so that each link, assigned an ajax loading url specific to that link, would create a dialog with its own content (They don't all have to be the same even though generated from one snippet).
4. And lastly, to be able to target a specific portion of the ajax loading page so the target could be a stand-alone page. In other words,only the relevant code would load in the dialog, but if javascript was disabled, a standalone page matching the site template would be loaded. By using the same page, the content did not necessarily have to either come from a specific ajax file which would require keeping the content of two files synced, not come from the database (though it could) requiring building a backend interface.

First, this requires jQuery and jQuery-ui to be loaded. I'm using a jQuery-ui library wih all the widgets included, so if that isn't what you're using, you'll need to add the dialog widget and any other libraries you would need for any effects you might add to the dialog.

I started with information I found on stackoverflow.com posted by Jek here .

The links for this code are formatted as My Link

Here the rel property is structured as [title]|[target]|[w,h] where the target can point to a page or a portion of the page, and the w,h is optional. Also, the title can be left out and a default title used, but then only the target should be in the tag. That is easy to work around, but this works for me.

Here is the code.

```
$().ready(function() {
```

```
  /*
  // Create the dialog element and initialize its functionality.
  // Rel is used to determine the target so portions of whole dosuments can be targeted.
*/
  $('a.openDialog').live('click', function() {
    var ajaxDialog = $('<div id="ajax-dialog" style="display:none;"></div>').appendTo('body');
      ajaxDialog.dialog({
          autoOpen: false,
          modal: true,
          close: function(event, ui)
          {
                $(this).dialog('destroy').remove();
          }
      });
  /*
  // Get and parse the rel attribute for a title, target, and optional width and height.
  // If the title is left off, using this code, the width and height must be left off since
  // it will assume that if any | symbol is included, the first item is the title.
  */
      var rel = $(this).attr("rel");
      if (rel.indexOf("|")) {
          rel = rel.split("|");
          var dialogTitle = rel[0];
          var url = rel[1];
          if (typeof(rel[2]) !== "undefined") {
              var dim = rel[2].split(",");
          }
      } else {
          var dialogTitle = "My Default Title";
          var url = rel;
      }
  /*
  // load remote content checking for whether a
  // specified width and height are included above.
  */
      $('#'+ajaxDialog.attr('id')).load(url, function() {
          ajaxDialog.dialog("option","title",dialogTitle);
          if (typeof(dim) !== "undefined") {
              ajaxDialog.dialog("option","width",dim[0]);
              ajaxDialog.dialog("option","height",dim[1]);
          }
          ajaxDialog.dialog("open");
      });

      //prevent the browser from following the link
      return false;
```

```
});
```

With this sort of method, I can apply one set of code and make any link a unique ui dialog on the page. It is also allows me to have a complete page on the target in case javascript is disabled and only get the content I want for the dialog off that page. That makes it easier to update the information in only one place without having to use a database.

The next post will be doing the same on the ui tooltip which was a bit more challenging.