# Prototype & script.aculo.us Lightbox

The combination of the Prototype and script.aculo.us JavaScript libraries provide a powerfull 1-2 punch of core functionality and effects. Yet niether provide a lightbox widget out of the box. We will look at how to create basic lightbox functionality using these libraries you may already have in your project.
UpdateDue to popular demand, I have attached a zip file that contains a working example of this lightbox implementation here: lightboxExample.zip. I look forward to your feedback!The End ResultModal dialog boxes are common place in the desktop world, however have been dificult to implement in the web world up until recently. We have always had javascript dialog boxes, however they have never exactly blended into any user experience well. Below is a screen shot of what we are attempting to acomplish:Project Setup

To keep things simple, we are going to impelement this example as a static HTML page. To begin, create a directory lightbox and create a file lightbox.html in it. You will also want to create a directory within lightbox called javascript. Within the javascript directory, place the Prototype and script.actulo.us javascript libraries (you can download the libraries from http://www.prototypejs.org and http://script.aculo.us respectively). Ok, let's get started coding.The HTML Markup

Open the lightbox.html file you created earlier in your favorite HTML editor. The above example consists of two main pieces of markup: a div to contain the contents of the lightbox popup and a div that we will use to contain the form the lightbox will go over. We will add the id containerDiv to the div used to contain the lightbox content. Place the message "This is my lightbox" in the div.

On the second div, we are going to ad the id body. Within the body div we will create an anchor tag with the id lightboxLink and an href of javascript:void(0). Make the text of the anchor Click Here. Below the anchor we just created, add a form with an input field and a select box. The reason we want the select box is to test an Internet Explorer bug that we will address later. Below is the HTML I used in my example (with styles):

```
<html>
    <head>
        #containerDiv {
            position: absolute;
            width: 30%;
            z-index: 99999;
            border: 1px blue solid;
            background-color: white;
        }

        #containerDiv p {
            position: relative;
            height: 10%;
            padding: 10%;
            text-align: center;
            z-index: 99999;
        }
    </head>
    <body>
        <div id="containerDiv">
            <p>This is the lightbox content.<br>
                <a id="closeLink" href="javascript:void(0);">Click here</a>
                to close the lightbox.
            </p>
        </div>
        <div id="body">
            <p>
                This is an example of a lightbox.
                Click the below link to launch the lightbox.
            </p>
            <a id="lightboxLink" href="javascript:void(0);">Launch lightbox</a>
            <div id="formContainer" style="margin: 10px 0 0 0;">
                <form>
                    <div>
                        <label>Name</label><input></input>
                    </div>
                    <div style="margin: 10px 0 0 0;">
                        <label>Select Box</label>
```

```
                    <select>
                        <option>Select One...</option>
                        <optiongroup>this is an option group</optiongroup>
                        <option>group1</option>
                        <option>group2</option>
                    </select>
                </div>
            </form>
        </div>
    </div>
  </body>
</html>
```

The Lightbox Library

In the javascript directory, create a file lightbox.js and open it in your favorite javascript editor. We're going to start by creating a Javacript class using the Prototype library:

```
var Lightbox = Class.create({});
```

In the class, we will create a number of methods. The first one being the constructor or initialize. In init we want to accomplish two things: store a reference to the lightbox container div (containerDiv) and set it's display to none. The following code accomplishes these two goals.

```
var Lightbox = Class.create({
    initialize : function(containerDiv) {
  this.container = containerDiv;
  this._hideLayer(this.container);
 },
 _hideLayer : function hideLayer(userAction){
  $(userAction).style.display="none";
 }
});
```

Now for the method to display the lightbox. The lightbox is actually made up of two elements, the containerDiv created in the markup and a background div used to create the grayed out effect over the rest of the page. We already created the containerDiv in our markup, however we will use javascript to create the background element.

Now why do we use markup for one part and javascript for the other? In this case, we want the javascript library to be reusable. The container div is used to create a specific lightbox and it's related content. The background element is shared from lightbox to lightbox. So in our case, if we had three lightboxes, we would have three containerDivs yet only one of the background elements will be created.

To make our lightbox visible, we create a method that accomplishes two things: if the background element has not been created previously, create it and set the apropriate style on the element. We create the new element, calling it bg_fade, using the Element constructor available in Prototype as seen below:

```
var Lightbox = Class.create({
    initialize : function(containerDiv) {
  this.container = containerDiv;
  this._hideLayer(this.container);
 },
 _hideLayer : function hideLayer(userAction){
  $(userAction).style.display="none";
 },
 _makeVisible : function makeVisible(){
  if($('bg_fade') == null) {
   shade = new Element('div', {'id': 'bg_fade',
                               'style': 'visibility=hidden;'});
   document.body.appendChild(shade);
  }
  $("bg_fade").setOpacity(0);
  $("bg_fade").style.visibility="visible";
  $("bg_fade").style.height='101%';
 }
});
```

To hide the background element, we remove the related styling with the _makeInvisible method below:

```
var Lightbox = Class.create({
    initialize : function(containerDiv) {
  this.container = containerDiv;
  this._hideLayer(this.container);
```

```
 },
 _hideLayer : function hideLayer(userAction){
  $(userAction).style.display="none";
 },
 _makeVisible : function makeVisible(){
  if($('bg_fade') == null) {
   shade = new Element('div', {'id': 'bg_fade',
                               'style': 'visibility=hidden;'});
   document.body.appendChild(shade);
  }
  $("bg_fade").setOpacity(0);
  $("bg_fade").style.visibility="visible";
  $("bg_fade").style.height='101%';
 },
 _makeInvisible : function makeInvisible(){
  $("bg_fade").setOpacity(0);
  $("bg_fade").style.visibility="hidden";
  $("bg_fade").style.height='2px';
 }
});
```

The methods above address the gray out layer, now let's address the lightbox itself. We will add two methods, showLayer and hideLayer to show and hide the lightbox above the grayed out background. In order to toggle the display, we will flip the display attribute of containerDiv to none to hide and block to display. var Lightbox = Class.create({

```
      initialize : function(containerDiv) {
  this.container = containerDiv;
  this._hideLayer(this.container);
 },
 _hideLayer : function hideLayer(userAction){
  $(userAction).style.display="none";
 },
 _makeVisible : function makeVisible(){
  if($('bg_fade') == null) {
   shade = new Element('div', {'id': 'bg_fade',
                               'style': 'visibility=hidden;'});
   document.body.appendChild(shade);
  }
  $("bg_fade").setOpacity(0);
  $("bg_fade").style.visibility="visible";
  $("bg_fade").style.height='101%';
 },
 _makeInvisible : function makeInvisible(){
  $("bg_fade").setOpacity(0);
  $("bg_fade").style.visibility="hidden";
  $("bg_fade").style.height='2px';
 },
 _showLayer : function showLayer(userAction){
  $(userAction).style.display="block";
 },
 _hideLayer : function hideLayer(userAction){
  $(userAction).style.display="none";
  }
});
```

Finally, we want to use the script.aculo.us Opacity effect to have our lightbox fade in for added punch. To do that, we will create a fade method on our Lightbox class. This method will either fade the lightbox in or out for half a second based on if it is currently displayed or not. We use the after and before callbacks that script.aculo.us provides to call the functions to display the lightbox after the fade in and hide it before the fade out. var Lightbox = Class.create({

```
      initialize : function(containerDiv) {
  this.container = containerDiv;
  this._hideLayer(this.container);
 },
 _hideLayer : function hideLayer(userAction){
  $(userAction).style.display="none";
 },
```

```
 _makeVisible : function makeVisible(){
  if($('bg_fade') == null) {
   shade = new Element('div', {'id': 'bg_fade',
                               'style': 'visibility=hidden;'});
   document.body.appendChild(shade);
  }
  $("bg_fade").setOpacity(0);
  $("bg_fade").style.visibility="visible";
  $("bg_fade").style.height='101%';
 },
 _makeInvisible : function makeInvisible(){
  $("bg_fade").setOpacity(0);
  $("bg_fade").style.visibility="hidden";
  $("bg_fade").style.height='2px';
 },
 _showLayer : function showLayer(userAction){
  $(userAction).style.display="block";
 },
 _hideLayer : function hideLayer(userAction){
  $(userAction).style.display="none";
 },
 _fade : function fadeBg(userAction,whichDiv){
  if(userAction=='close'){
   new Effect.Opacity('bg_fade',
       {duration:.5,
        from:0.5,
        to:0,
        afterFinish:this._makeInvisible(),
        afterUpdate:this._hideLayer(whichDiv)});
  }else{
   new Effect.Opacity('bg_fade',
       {duration:.5,
        from:0,
        to:0.5,
        beforeUpdate:this._makeVisible(),
        afterFinish:this._showLayer(whichDiv)});
  }
 }
});
```

Since we have all the methods defined that do all the "work", we create public open and close methods to be called upon the correct event.

```
  var Lightbox = Class.create({
      initialize : function(containerDiv) {
 this.container = containerDiv;
 this._hideLayer(this.container);
 },
 _hideLayer : function hideLayer(userAction){
  $(userAction).style.display="none";
 },
 _makeVisible : function makeVisible(){
  if($('bg_fade') == null) {
   shade = new Element('div', {'id': 'bg_fade',
                               'style': 'visibility=hidden;'});
   document.body.appendChild(shade);
  }
  $("bg_fade").setOpacity(0);
  $("bg_fade").style.visibility="visible";
  $("bg_fade").style.height='101%';
 },
 _makeInvisible : function makeInvisible(){
  $("bg_fade").setOpacity(0);
  $("bg_fade").style.visibility="hidden";
```

```
 $("bg_fade").style.height='2px';
},
_showLayer : function showLayer(userAction){
 $(userAction).style.display="block";
},
_hideLayer : function hideLayer(userAction){
 $(userAction).style.display="none";
},
_fade : function fadeBg(userAction,whichDiv){
 if(userAction=='close'){
  new Effect.Opacity('bg_fade',
      {duration:.5,
       from:0.5,
       to:0,
       afterFinish:this._makeInvisible(),
       afterUpdate:this._hideLayer(whichDiv)});
 }else{
  new Effect.Opacity('bg_fade',
      {duration:.5,
       from:0,
       to:0.5,
       beforeUpdate:this._makeVisible(),
       afterFinish:this._showLayer(whichDiv)});
 }
},
open : function () {
 this._fade('open', this.container);
},
close : function () {
 this._fade('close', this.container);
}
});
```

That's it for Lightbox version .1.  To use it, we add links to the libraries (Prototype, script.aculo.us, and our lightbox library) to our HTML page.  We then add the following javascript to our page to create an instance of Lightbox and associate the open and close methods with the apropriate click methods:<script>

```
    var test;

    Event.observe(window, 'load', function () {
        test = new Lightbox('containerDiv');
    });

    Event.observe('lightboxLink', 'click', function () {
        test.open();
    });

    Event.observe('closeLink', 'click', function () {
        test.close();
    });
</script>
```

Centering

Once you try out the above code, you'll notice a few issues, the centering of the background (gray) div, the centering of the lightbox and the fact that the select box isn't covered by our lightbox in Internet Explorer.  We'll get to the select box in a minute.  To start, let's look at centering.

Centering an element poses two challenges, determining the size of the element and determining the size of the window.  To determine the size of the element, Prototype provides a greate set of cross browser methods getHeight and getWidth.  We will use both of these to determine the height and width of the lightbox.

Although Prototype also provides support for determining the size of the viewport (the area of the document within the browser that you actually can see), it has been my experience that they don't work in IE 6.  Because of this, we have to do a bit of checking before we use these methods.

To provide centering functionality, we will write a centerWindow method. It will take an element as an arguement and center it accordingly. Below is the code (It is added to the lightbox.js file):function centerWindow(element) {

```
    if($(element) != null) {
        if(typeof window.innerHeight != 'undefined') {
            $(element).style.top =
                Math.round(document.viewport.getScrollOffsets().top +
                ((window.innerHeight - $(element).getHeight()))/2)+'px';
            $(element).style.left =
                Math.round(document.viewport.getScrollOffsets().left +
                ((window.innerWidth - $(element).getWidth()))/2)+'px';
        } else {
            $(element).style.top =
                Math.round(document.body.scrollTop +
                (($$('body')[0].clientHeight - $(element).getHeight()))/2)+'px';
            $(element).style.left =
                Math.round(document.body.scrollLeft +
                (($$('body')[0].clientWidth - $(element).getWidth()))/2)+'px';
        }
    }
}
```

Essentially the above code reads as follows

if element exists
    if Prototype's viewport functionality will work
        element's top offset = the top scroll offset from viewport + (the height of the window - half the height of the element)
        element's left offset = the left scroll offset from viewport + (the width of the window - half the width of the element)
    else
        element's top offset = the top scroll offset
+ (the height of the window - half the height of the element)
        element's left offset = the left scroll offset + (the width of the window - half the width of the element)
The Select Box

Great, we now have a lightbox that shows up centered. All is good, right? Well, it is if you only work with Firefox, however if you check out our example in Internet Explorer, we have a problem. The problem is the select box. In Internet Explorer, select boxes are considered window level elements. This causes them to not be covered by a div like the one we use for graying out the background. In order to fix this, we need to put another window level element over the select box. In this case, we will put an empty iframe over the select box. Essentially, the way the final layout will be layered is

Lightbox
bg_fade
iframe
parent document

To do this, we will change _makeVisible and _makeInvisible in the following way:_makeVisible : function makeVisible(){

```
    if($('dummyFrame') == null) {
        dummyFrame =
            new Element('iFrame',
                {'src': '', 'id': 'dummyFrame',
                  'scrolling': 'no', 'style': 'background-color:' +
                  ' transparent; border: 0px;filter:progid:' +
                  'DXImageTransform.Microsoft.Alpha(style=0,opacity=0);' +
                  'position:absolute; top: 0px; left:0px; z-index: 999;' +
                  'width: 100%;height:100%;'});
        document.body.appendChild(dummyFrame);
    }
    if($('bg_fade') == null) {
        shade = new Element('div', {'id': 'bg_fade',
                        'style': 'visibility=hidden;'});
        document.body.appendChild(shade);
    }
    centerWindow('dummyFrame');
    $("bg_fade").setOpacity(0);
    $("bg_fade").style.visibility="visible";
    $("bg_fade").style.height='101%';
```

```
    centerWindow('bg_fade');
},

_makeInvisible : function makeInvisible(){
    $("bg_fade").setOpacity(0);
    $("bg_fade").style.visibility="hidden";
    $("bg_fade").style.height='2px';
    Element.remove('dummyFrame');
},
```

As you can see, when we show the lightbox we use the Prototype constructor to create a new iframe element over the entire document.  This will hide the select box.  When we hide the lightbox, we remove the iframe to redisplay it.
Conclusion

So that's it.  I hope this gives you some insight not only in how to create a modal lightbox with javascript libraries you most likely are already using, but a more indepth view of those libraries for future use. Update

Feel free to download an example of the lightbox code here: lightboxExample.zip