MODX PhotoSwipe Gallery

Combine MODX Gallery Extra and PhotoSwipe Gallery to create a Gallery Album and when clicked on a thumb inside the album the picture is shown with PhotoSwipe Gallery.

Required things

We need the following to implement that.

- MODX Gallery Extra
- MODX phpThumbOf Extra
- getImageProperties Snippet ↓
- PhotoSwipe Gallery
- photoswipeIntegration Snippet (optional) ↓

Before beginning

The first thing to do is installing the MODX Gallery Extra and make sure that at least one album was created and a few pictures were loaded into it. To avoid confusion, give the uploaded images a short description. Any questions? A quick reference for setting up your gallery could be found in the documentation.

Usage

Step 1: Creating a template based overview of the albums. In this case is use the Gallery Snippet and the GalleryAlbums Snippet wich both comes with the MODX Gallery Extra.

MODX Gallery Extra

The Gallery Snippet and the GalleryAlbums Snippet can be called using this tags. To make things easier you can also put in there the optional photoswipeIntegration Snippet.

Create a new Resource or Template and add following.

```
[[!GalleryAlbums? &toPlaceholder=`my-galleries`]]
<div>
  <h2>Galleries</h2>
  <ul>
    [[+my-galleries]]
  </ul>
</div>

[[!Gallery? &toPlaceholder=`my-gallery` &thumbTpl=`getImagePropertiesTpl`]]
```

```
[[!+my-gallery:notempty=`
  <!-- Combine MODX Gallery Extra and PhotoSwipe Gallery -->
  <h2>[[+my-gallery.name]]</h2>
  <div class="my-gallery" itemscope itemtype="http://schema.org/ImageGallery">
    [[+my-gallery]]
  </div>
`]]

[[!photoswipeIntegration]]
```

Templating

For templating create the following two Chunks.

Chunk getImagePropertiesTpl

Create Chunk "getImagePropertiesTpl" and add the following.

```
[[!getImageProperties? &ia=`[[+image_absolute]]`]]
```

Chunk albumItemTpl

This is to build an array of slides from a list of links. For resizing the image items i use the MODX phpThumbOf Extra, make sure you installed it.

Create Chunk "albumItemTpl" and add the following.

```
<figure>
  <a href="[[+image_absolute]]" data-size="[[+image_absolute_width]]x[[+image_absolute_height]]">
    <img title="[[+description]]" src="[[+image_absolute:phpthumbof=`w=768&h=576&zc=1&q=98`]]" alt="[[+description]]"/></a>
  <figcaption>[[+description]]</fi2gcaption>
</figure>
```

Step 2: PhotoSwipe integration

Snippet getImageProperties

The getImageProperties Snippet fetch the width and height from each Gallery Album Item. It's necessary because PhotoSwipe requires predefined image dimensions for html data-size attribute.

Create Snippet "getImageProperties" and add the following.

```
<?php
/**
 * getImageProperties
 *
 * DESCRIPTION
 *
 * This Snippet fetch the width and height from a Gallery Album Item using the "image_absolute" Placeholder
 *
 * It's necessary to combine MODX Gallery Extra and PhotoSwipe gallery script
 * PhotoSwipe requires predefined image dimensions for html data-size attribute
 *
 * PROPERTIES:
 *
 * &ia string is required
 *
 * USAGE:
 *
 * [[!getImageProperties? &ia=`[[+image_absolute]]`]]
 *
 */

$ia = $modx->getOption('ia', $scriptProperties);

if (!isset($scriptProperties['ia'])) {
  $modx->log(modX::LOG_LEVEL_ERROR, '[getImageProperties] missing required properties &ia!');
  return;
}

// to use PHP getimagesize properly in some cases it is necessary to remove the leading slash
$l = strlen($ia)-1;
$ian=substr($ia,1,$l);

list($iawidth, $iaheight) = getimagesize($ian);

// templating
$tpl = $modx->getOption('tpl',$scriptProperties,'albumItemTpl'); // default property
$output = $modx->getChunk($tpl,array(
  'image_absolute_width' => $iawidth,
  'image_absolute_height' => $iaheight
));

return $output;
```

PhotoSwipe Gallery

Next add the PhotoSwipe code to your Resource or Template. For getting startet with PhotoSwipe please visit the documentation. Alternatively you can use the optional photoswipeIntegration Snippet, which you can find below.

Snippet photoswipeIntegration

This Snippet includes PhotoSwipe JS and CSS files, add PhotoSwipe (.pswp) element to DOM and initialize and execute (pure Vanilla JS implementation). It's optional, but it is to make things easier. Integration can of course also be done manually.

Create Snippet "photoswipeIntegration" and add the following.

```
<?php
/**
 * photoswipeIntegration
 *
 * DESCRIPTION
 *
 * This Snippet includes PhotoSwipe JS and CSS core files, add PhotoSwipe (.pswp) element to DOM and initialize.
 * Based on PhotoSwipe - v4.1.1
 *
 * REMARKS:
 *
 * In Step 3: Initialize - To work with the JS in this Snippet some parts of the original code must have been escaped.
 * So it is not a one-to-one copy of the PhotoSwipe code.
 *
 * USAGE:
 *
 * [[!photoswipeIntegration]]
 *
 */

/**************************************************************************************/

/* Step 1: Include JS and CSS files */

$modx->regClientStartupHTMLBlock('
<!-- Core CSS file -->');
$modx->regClientCSS(MODX_ASSETS_URL.'photoswipe/photoswipe.css');
$modx->regClientStartupHTMLBlock(' ');

$modx->regClientStartupHTMLBlock('<!-- Skin CSS file (styling of UI - buttons, caption, etc.)
     In the folder of skin CSS file there are also:
     - .png and .svg icons sprite,
     - preloader.gif (for browsers that do not support CSS animations) -->');
$modx->regClientCSS(MODX_ASSETS_URL.'photoswipe/default-skin/default-skin.css');

$modx->regClientStartupHTMLBlock('<!-- Core JS file -->');

$modx->regClientStartupHTMLBlock('<script src="'.MODX_ASSETS_URL.'photoswipe/photoswipe.min.js"></script>');

$modx->regClientStartupHTMLBlock('<!-- UI JS file -->');
$modx->regClientStartupHTMLBlock('<script src="'.MODX_ASSETS_URL.'photoswipe/photoswipe-ui-default.min.js"></script>');

/* Step 2: Add PhotoSwipe (.pswp) element to DOM */

$modx->regClientHTMLBlock('
<!-- Root element of PhotoSwipe. Must have class pswp. -->
<div class="pswp" tabindex="-1" role="dialog" aria-hidden="true">
    <div class="pswp__bg"></div>
    <div class="pswp__scroll-wrap">
        <div class="pswp__container">
            <div class="pswp__item"></div>
            <div class="pswp__item"></div>
            <div class="pswp__item"></div>
        </div>
        <div class="pswp__ui pswp__ui--hidden">
            <div class="pswp__top-bar">
                <div class="pswp__counter"></div>
                <button class="pswp__button pswp__button--close" title="Close (Esc)"></button>
                <button class="pswp__button pswp__button--share" title="Share"></button>
                <button class="pswp__button pswp__button--fs" title="Fullscreen"></button>
                <button class="pswp__button pswp__button--zoom" title="Zoom in/out"></button>
                <div class="pswp__preloader">
                    <div class="pswp__preloader__icn">
                      <div class="pswp__preloader__cut">
                        <div class="pswp__preloader__donut"></div>
                      </div>
                    </div>
                </div>
            </div>
            <div class="pswp__share-modal pswp__share-modal--hidden pswp__single-tap">
                <div class="pswp__share-tooltip"></div>
            </div>
            <button class="pswp__button pswp__button--arrow--left" title="Prev">
            </button>
            <button class="pswp__button pswp__button--arrow--right" title="Next">
            </button>
            <div class="pswp__caption">
                <div class="pswp__caption__center"></div>
            </div>
          </div>
        </div>
</div>');

/* Step 3: Initialize and execute function */

// pure Vanilla JS implementation (standard for this snippet)

$modx->regClientHTMLBlock('<!-- Initialize -->');

$modx->regClientHTMLBlock('<script>
var initPhotoSwipeFromDOM = function(gallerySelector) {

    // parse slide data (url, title, size ...) from DOM elements
    // (children of gallerySelector)
    var parseThumbnailElements = function(el) {
        var thumbElements = el.childNodes,
            numNodes = thumbElements.length,
            items = [],
            figureEl,
            linkEl,
            size,
            item;

        for(var i = 0; i < numNodes; i++) {

            figureEl = thumbElements[i]; // <figure> element

            // include only element nodes
            if(figureEl.nodeType !== 1) {
                continue;
            }

            linkEl = figureEl.children[0]; // <a> element

            size = linkEl.getAttribute(\'data-size\').split(\'x\');

            // create slide object
            item = {
                src: linkEl.getAttribute(\'href\'),
                w: parseInt(size[0], 10),
                h: parseInt(size[1], 10)
            };



            if(figureEl.children.length > 1) {
                // <figcaption> content
                item.title = figureEl.children[1].innerHTML;
            }

            if(linkEl.children.length > 0) {
                // <img> thumbnail element, retrieving thumbnail url
                item.msrc = linkEl.children[0].getAttribute(\'src\');
            }

            item.el = figureEl; // save link to element for getThumbBoundsFn
            items.push(item);
        }

        return items;
    };

    // find nearest parent element
    var closest = function closest(el, fn) {
        return el && ( fn(el) ? el : closest(el.parentNode, fn) );
    };

    // triggers when user clicks on thumbnail
    var onThumbnailsClick = function(e) {
        e = e || window.event;
        e.preventDefault ? e.preventDefault() : e.returnValue = false;

        var eTarget = e.target || e.srcElement;

        // find root element of slide
        var clickedListItem = closest(eTarget, function(el) {
            return (el.tagName && el.tagName.toUpperCase() === \'FIGURE\');
        });

        if(!clickedListItem) {
            return;
        }

        // find index of clicked item by looping through all child nodes
        // alternatively, you may define index via data- attribute
        var clickedGallery = clickedListItem.parentNode,
            childNodes = clickedListItem.parentNode.childNodes,
            numChildNodes = childNodes.length,
            nodeIndex = 0,
            index;

        for (var i = 0; i < numChildNodes; i++) {
            if(childNodes[i].nodeType !== 1) {
                continue;
            }

            if(childNodes[i] === clickedListItem) {
                index = nodeIndex;
                break;
            }
            nodeIndex++;
        }



        if(index >= 0) {
            // open PhotoSwipe if valid index found
            openPhotoSwipe( index, clickedGallery );
        }
        return false;
    };

    // parse picture index and gallery index from URL (#&pid=1&gid=2)
    var photoswipeParseHash = function() {
        var hash = window.location.hash.substring(1),
        params = {};

        if(hash.length < 5) {
            return params;
        }

        var vars = hash.split(\'&\');
        for (var i = 0; i < vars.length; i++) {
            if(!vars[i]) {
                continue;
            }
            var pair = vars[i].split(\'=\');
            if(pair.length < 2) {
                continue;
            }
            params[pair[0]] = pair[1];
        }

        if(params.gid) {
            params.gid = parseInt(params.gid, 10);
        }

        return params;
    };

    var openPhotoSwipe = function(index, galleryElement, disableAnimation, fromURL) {
        var pswpElement = document.querySelectorAll(\'.pswp\')[0],
            gallery,
            options,
            items;

        items = parseThumbnailElements(galleryElement);

        // define options (if needed)
        options = {

            // define gallery index (for URL)
            galleryUID: galleryElement.getAttribute(\'data-pswp-uid\'),

            getThumbBoundsFn: function(index) {
                // See Options -> getThumbBoundsFn section of documentation for more info
                var thumbnail = items[index].el.getElementsByTagName(\'img\')[0], // find thumbnail
                    pageYScroll = window.pageYOffset || document.documentElement.scrollTop,
                    rect = thumbnail.getBoundingClientRect();

                return {x:rect.left, y:rect.top + pageYScroll, w:rect.width};
            }

        };

        // PhotoSwipe opened from URL
        if(fromURL) {
            if(options.galleryPIDs) {
                // parse real index when custom PIDs are used
                // http://photoswipe.com/documentation/faq.html#custom-pid-in-url
                for(var j = 0; j < items.length; j++) {
                    if(items[j].pid == index) {
                        options.index = j;
                        break;
                    }
                }
            } else {
                // in URL indexes start from 1
                options.index = parseInt(index, 10) - 1;
            }
        } else {
            options.index = parseInt(index, 10);
        }

        // exit if index not found
        if( isNaN(options.index) ) {
            return;
        }

        if(disableAnimation) {
            options.showAnimationDuration = 0;
        }

        // Pass data to PhotoSwipe and initialize it
        gallery = new PhotoSwipe( pswpElement, PhotoSwipeUI_Default, items, options);
        gallery.init();
    };

    // loop through all gallery elements and bind events
    var galleryElements = document.querySelectorAll( gallerySelector );

    for(var i = 0, l = galleryElements.length; i < l; i++) {
        galleryElements[i].setAttribute(\'data-pswp-uid\', i+1);
        galleryElements[i].onclick = onThumbnailsClick;
    }

    // Parse URL and open gallery if it contains #&pid=3&gid=1
    var hashData = photoswipeParseHash();
    if(hashData.pid && hashData.gid) {
        openPhotoSwipe( hashData.pid ,  galleryElements[ hashData.gid - 1 ], true, true );
    }
};

// execute above function
initPhotoSwipeFromDOM(\'.my-gallery\');
</script>');

$modx->regClientHTMLBlock('');
```

Required PhotoSwipe files

Complete the integration with upload the required PhotoSwipe files, get them at GitHub. Create "photoswipe" directory in your assets directory and put the css, js and skin files in there.

That should be it. Note: This is not a reference, it should only illustrate what is possible when you combine the MODX Gallery Extra with other third party components like PhotoSwipe Gallery.