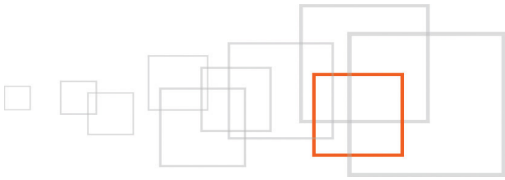
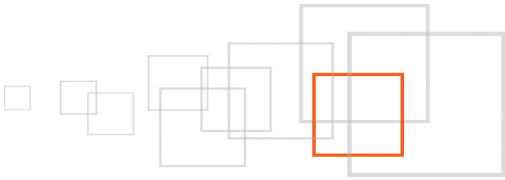
Transforming jQuery plugins into eZ publish extensions

By Thiago Campos Viana
http://thiagocamposviana.blogspot.com



Index



1 Goal description

In this tutorial we will see the main steps to transform a **jQuery** plugin into an eZ publish extension, in this case an **eZ flow block**.

2 Introduction

JQuery is a javascript library known for the huge amount of plugins available for download, which can be used to decrease the development time of a site and avoid having to re-invent the wheel where plugins could be re-used. Using these plugins with eZ publish, especially with the package eZ flow, usually requires a few modifications.

This tutorial gives an overview of how to transform **jQuery plugin** into an **eZ Flow block**, the steps described here can be used for other types of extensions such as datatypes, modules or design extensions.

3 Pre-requisites and target population

To understand this tutorial basic knowledge is required on developing extensions for eZ publish and javascript / jquery development. An indicated tutorial for learning how to develop extensions on eZ Publish is : http://share.ez.no/learn/ez-publish/an-introduction-to-developing-ez-publish-extensions

4 Step 1: Downloading and testing the plug-in

In this tutorial we'll use the **Pikachoose jQuery plugin** as a base, which is a **slideshow** with autoplay options, navigation buttons and carousel. The first step is to download the archive (zip), uncompress and test the plugin. If you like the plugin, please consider donating to the developer to eat some pizza and continue with the project.

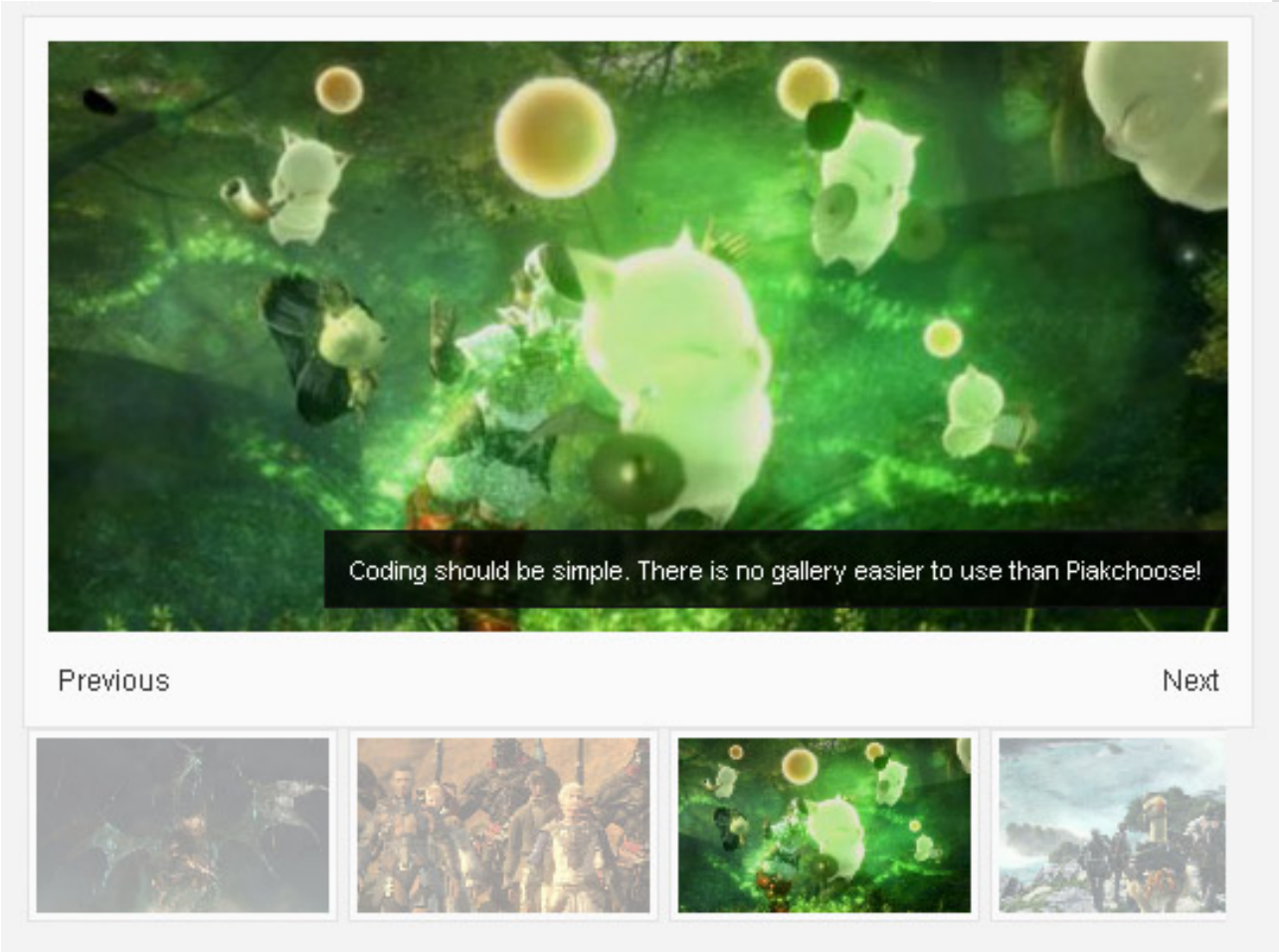


Figure 1 - Pikachoose plugin

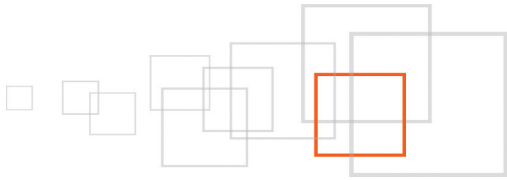
After performing the tests it is time to analyze the important plug-in files. Inside the plug-in folder there are the root assets folder, which in turn contains the images and js folders. We will need all the files from the images folder and only two js from the scripts folder, **jquery.jcarousel.min.js** and **jquery.pikachoose.js**.

Moreover, we must use some example, or more than one to create our eZ publish extension, so, you'll need to analyze the example we will use, in this case the "bottom with carousel " (root folder -> bottom -> with-carousel.html).

The first part is the code to import the necessary external files:
Code :

```
<link type="text/css" href="css/styles.css" rel="stylesheet" />
<script type="text/javascript" src="../assets/js/jquery.js"></script>
<script type="text/javascript" src="../assets/js/jquery.pikachoose.js"></script>
<script type="text/javascript" src="../assets/js/jquery.jcarousel.min.js"></script>
```

eZ Publish comes with the jquery.js script (shipped through the ezjscore extension), and we have marked the jquery.pikachoose.js and the jquery.jcarousel.min.js as necessary, but we need to mark the sample style sheet file as important : root folder / bottom / css / styles.css. We will have to modify this file in our extension.



The second code snippet is the Javascript example:
Code :

```
<script language="javascript">
	<!--
	$(document).ready(
		function (){
			$("#pikame").PikaChoose();


			$("#pikame").jcarousel({scroll:4,
				initCallback: function(carousel)
				{
					$(carousel.list).find('img').click(function() {
						//console.log($(this).parents('.jcarousel-
item').attr('jcarouselindex'));
						carousel.scroll(parseInt($
(this).parents('.jcarousel-item').attr('jcarouselindex')));
					});
				}
			});


		});


	-->
</script>
```

This is the code block that will be used and adapted in our extension. The third section of the html code is important for our extension:
Code :

```
<div class="pikachoose">
Basic example
	<ul id="pikame" class="jcarousel-skin-pika">
		<li><a href="http://www.pikachoose.com"><img
src="../1.jpg"/></a><span>Thanks to <a href="http://web.cara-jo.net">Cara Jo</a> for
the awesome new themes!</span></li>
		<li><a href="http://www.pikachoose.com"><img
src="../2.jpg"/></a><span>jCarousel is supported and integrated with PikaChoose!
</span></li>


		<li><a href="http://www.pikachoose.com"><img src="../3.jpg"/></a><span>Let
me know at jeremy.m.fry@gmail.com if you find any bugs!</span></li>
		<li><a href="http://www.pikachoose.com"><img
src="../4.jpg"/></a><span>Caption</span></li>
```

```
            <li><a href="http://www.pikachoose.com"><img
src="../5.jpg"/></a><span>Caption</span></li>

            <li><a href="http://www.pikachoose.com"><img
src="../1.jpg"/></a><span>Caption</span></li>

            <li><a href="http://www.pikachoose.com"><img
src="../2.jpg"/></a><span>Caption</span></li>

            <li><a href="http://www.pikachoose.com"><img
src="../3.jpg"/></a><span>Caption</span></li>


            <li><a href="http://www.pikachoose.com"><img
src="../4.jpg"/></a><span>Caption</span></li>

            <li><a href="http://www.pikachoose.com"><img
src="../5.jpg"/></a><span>Caption</span></li>


      </ul>

</div>
```

This is the part of the html that will be examined and put in our extension.

So we have an initial basic procedure for adapting a Jquery plug-in to eZ Publish :

- find the important extension files,
- look at an example and find some other imported files, in this case a css, a javascript and the main example html.

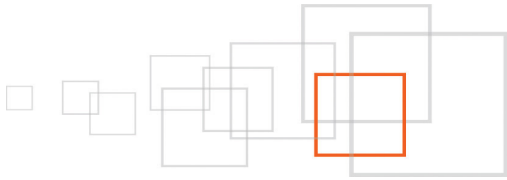
With all of these, it is time to move on and create our extension.

5 Step 2: Creating an eZ flow block

Now it's time to create a basic framework for our extension. The eZ flow documentation describes in detail how to create eZ flow blocks: http://doc.ez.no/Extensions/eZ-Flow/eZ-Flow-2.2/Configuration/Configuring-the-eZ-Flow-block -system/Defining-block-types

Creating the structure

Let's start by creating the folder structure and files for our initial extension, which will be called contentslider. Create the following initial files inside the eZ publish extension folder:



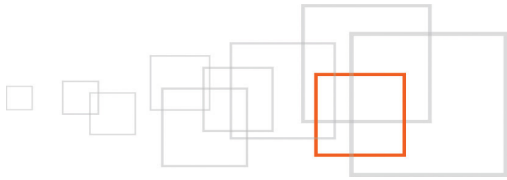
```
contentslider/
|-- design
|   `-- ezflow
|       |-- images ( copy pikachoose/assets/images )
|       |-- javascript
|       |   |-- jquery.jcarousel.min.js ( copy pikachoose/assets/js/ jquery.jcarousel.min.js )
|       |   `-- jquery.pikachoose.js ( copy pikachoose/assets/js/ jquery.pikachoose.js )
|       |-- override
|       |   `-- templates
|       |       `-- block
|       |           `-- contentslider.tpl
|       `-- stylesheets
|           `-- contentslider.css (copy pikachoose/bottom/css/styles.css, remove the first two lines)
`-- settings
    |-- block.ini.append.php
    |-- design.ini.append.php
    `-- override.ini.append.php
```

Before proceeding, check the contentslider.css, which has the same content as the pikachoose / bottom / css / styles.css file. Notice it makes image references using the original jQuery plugin file structure, you need to do some modifications on this file to reflect the new file structure.

So, open your code editor and do a find and replace, find "../../assets/images" and replace it with "../images".. The file is fixed.

We first need to tell eZ Publish that our newly created extension contains design elements (stylesheets, javascript files, templates, images). To do so, we need to modify the design.ini.append.php settings file:
Code :

```
<?php /* #?ini charset="utf-8"?
[ExtensionSettings]
DesignExtensions[]=contentslider
*/
?>
```



Creating the eZ Flow block

We will now declare our new eZ Flow block : the ContentSlider block. In order to do so, we will modify the block.ini.append.php settings file:

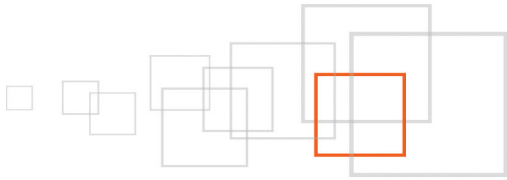
Code :

```
<?php /*
[General]
AllowedTypes[]=ContentSlider

[ContentSlider]
Name=Content Slider
NumberOfValidItems=9
NumberOfArchivedItems=5
ManualAddingOfItems=disabled
FetchClass=eZFlowMCFetch
FetchParameters[Source]=NodeID
FetchParametersSelectionType[Source]=single
FetchParametersIsRequired[Source]=true
FetchParameters[Classes]=string
ViewList[]=contentslider
ViewName[contentslider]=Content Slider
*/ ?>
```

This is how one can define how an eZ Flow block works. For more information about eZ flow blocks configuration, visit : http://doc.ez.no/Extensions/eZ-Flow/eZ-Flow-2.2/Configuration/Configuring-the-eZ-Flow-block-system/Defining-block-types

We now need to give our new block a template to express himself. This is achieved through an override rule, written in our extension's override.ini.append.php file :
Code :

```
<?php /* #?ini charset="utf-8"?

[block_contentslider]
Source=block/view/view.tpl
MatchFile=block/contentslider.tpl
Subdir=templates
Match[type]=ContentSlider
Match[view]=contentslider

*/ ?>
```



This file specifies which template file is used to display the eZ Flow html content, in our case the file that will be called by eZ is "contentslider / design / ezflow / override / templates / block / contentslider.tpl", in this file we need to put the html and javascript from the example, making minor adjustments, so, here is our base the javascript (from the Jquery plug-in):
Code :

```
<script language="javascript">
	<!--
	$(document).ready(
		function (){
			$("#pikame").PikaChoose();

			$("#pikame").jcarousel({scroll:4,
				initCallback: function(carousel)
				{
					$(carousel.list).find('img').click(function() {
						//console.log($(this).parents('.jcarousel-
item').attr('jcarouselindex'));
						carousel.scroll(parseInt($
(this).parents('.jcarousel-item').attr('jcarouselindex')));
					});
				}
			});

		});

	-->
</script>
```

As the characters '{' and '}' are used as eZ template language delimiters, we have to open the editor and find and replace the character '{' for 'ldelim' and '} ' for 'rdelim', then do a find and replace again for 'ldelim' by '{ldelim}' and 'rdelim' by '{rdelim}', fixing the javascript that now can be used inside the ez template code, thus:
Code :

```
<script language="javascript">
	<!--
	$(document).ready(
		function (){ldelim}
			$("#pikame").PikaChoose();

			$("#pikame").jcarousel({ldelim}scroll:4,
				initCallback: function(carousel)
```

```
                        {ldelim}
                                $(carousel.list).find('img').click(function() {ldelim}
                                        //console.log($(this).parents('.jcarousel-
item').attr('jcarouselindex'));
                                        carousel.scroll(parseInt($
(this).parents('.jcarousel-item').attr('jcarouselindex')));
                                {rdelim});
                        {rdelim}
                {rdelim});


        {rdelim});


    -->
</script>
```

Copy this code to the "contentslider / design / ezflow / override / templates / block / contentslider.tpl" file. Let's now check the html shipped with the jquery plug-in:

Code :

```
<div class="pikachoose">
Basic example
    <ul id="pikame" class="jcarousel-skin-pika">
        <li><a href="http://www.pikachoose.com"><img
src="../1.jpg"/></a><span>Thanks to <a href="http://web.cara-jo.net">Cara Jo</a> for
the awesome new themes!</span></li>
        <li><a href="http://www.pikachoose.com"><img
src="../2.jpg"/></a><span>jCarousel is supported and integrated with PikaChoose!
</span></li>

        <li><a href="http://www.pikachoose.com"><img src="../3.jpg"/></a><span>Let
me know at jeremy.m.fry@gmail.com if you find any bugs!</span></li>
        <li><a href="http://www.pikachoose.com"><img
src="../4.jpg"/></a><span>Caption</span></li>
        <li><a href="http://www.pikachoose.com"><img
src="../5.jpg"/></a><span>Caption</span></li>
        <li><a href="http://www.pikachoose.com"><img
src="../1.jpg"/></a><span>Caption</span></li>
        <li><a href="http://www.pikachoose.com"><img
src="../2.jpg"/></a><span>Caption</span></li>
        <li><a href="http://www.pikachoose.com"><img
src="../3.jpg"/></a><span>Caption</span></li>

        <li><a href="http://www.pikachoose.com"><img
src="../4.jpg"/></a><span>Caption</span></li>
        <li><a href="http://www.pikachoose.com"><img
src="../5.jpg"/></a><span>Caption</span></li>
```

```
      </ul>
</div>
```

One can note that the HTML follows the following structure:

Code :

```
<div class="pikachoose">
Basic example
      <ul id="pikame" class="jcarousel-skin-pika">
      <!-- LOOP OBJECTS -->
            <li><a href="URL"><img src="IMG_URL"/></a><span>MESSAGE</span></li>
      <!-- END LOOP OBJECTS -->
      </ul>
</div>
```

We will reflect this loop in the display template for our block, using the {foreach} template language construct.

When using our block, we will associate to it a list of content objects (all placed in one folder, content container), for display. This constitutes the dynamic content These objects can be accessed within the block's view template like this:
Code :

```
{def $valid_nodes = $block.valid_nodes}
```

New Content Class to finalize our block

We will create a content class called Slide with the following attributes (name and datatype are described):

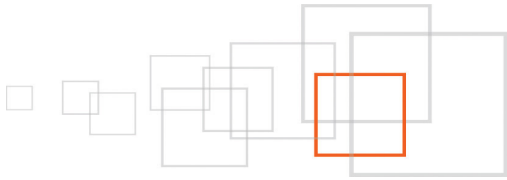
- Title: Text line,
- Message: Text line,
- Image: Image,
- Object: Related Objects.

Assuming we will only associate “Slide” objects to our block, here is how our contentslider.tpl file looks so far:

Code :

```
{def $valid_nodes = $block.valid_nodes}

<!-- BLOCK: START -->

<div class="pikachoose">
```

```
Basic example
      <ul id="pikame" class="jcarousel-skin-pika">
            {foreach $valid_nodes as $vnode}


            <li><a href={fetch( 'content', 'node', hash( 'node_id',
$vnode.data_map.object.content.relation_list[0].node_id ) ).url_alias|
ezurl}>{attribute_view_gui image_class=large
attribute=$vnode.data_map.image}</a><span>{$vnode.object.data_map.message.data_text}</s
pan></li>


            {/foreach}
            </ul>
</div>


<script language="javascript">
      <!--
      $(document).ready(
            function (){ldelim}
                  $("#pikame").PikaChoose();


                  $("#pikame").jcarousel({ldelim}scroll:4,
                        initCallback: function(carousel)
                        {ldelim}
                              $(carousel.list).find('img').click(function() {ldelim}
                                    //console.log($(this).parents('.jcarousel-
item').attr('jcarouselindex'));
                                    carousel.scroll(parseInt($
(this).parents('.jcarousel-item').attr('jcarouselindex')));
                              {rdelim});
                        {rdelim}
                  {rdelim});


            {rdelim});


      -->
</script>
```



We forgot a key aspect : including the required external elements : javascript and css files, required by the Jquery plug-in. Here is how, relying on eZJSCore's template operators, we can easily include these external elements :

Code :

```
{ezcss_require( 'contentslider.css' )}

{ezscript_require( array('ezjsc::jquery','jquery.jcarousel.min.js',
'jquery.pikachoose.js') )}
```

Here is how our updated template code looks :

Code :

```
{def $valid_nodes = $block.valid_nodes}


<!-- BLOCK: START -->


<div class="pikachoose">

Basic example

      <ul id="pikame" class="jcarousel-skin-pika">

            {foreach $valid_nodes as $vnode}


            <li><a href={fetch( 'content', 'node', hash( 'node_id',
$vnode.data_map.object.content.relation_list[0].node_id ) ).url_alias|
ezurl}>{attribute_view_gui image_class=large
attribute=$vnode.data_map.image}</a><span>{$vnode.object.data_map.message.data_text}</s
pan></li>


            {/foreach}

            </ul>

</div>


{ezcss_require( 'contentslider.css' )}


{ezscript_require( array('ezjsc::jquery','jquery.jcarousel.min.js',
'jquery.pikachoose.js') )}


<script language="javascript">

      <!--

      $(document).ready(

            function (){ldelim}

                  $("#pikame").PikaChoose();


                  $("#pikame").jcarousel({ldelim}scroll:4,

                        initCallback: function(carousel)

                        {ldelim}
```

```
                                $(carousel.list).find('img').click(function() {ldelim}
                                        //console.log($(this).parents('.jcarousel-
item').attr('jcarouselindex'));
                                        carousel.scroll(parseInt($
(this).parents('.jcarousel-item').attr('jcarouselindex')));
                                {rdelim});
                        {rdelim}
                {rdelim});


        {rdelim});


    -->
</script>
```

6 Step 3: Polishing your extension

Our extension now is functional, but needs some polishing. First realize that images can appear in different sizes, while our slider only accepts one specific size of image. We need to fix that.

The way is to create an image class to generate images of a fixed size. Create the contentslider/settings/image.ini.append.php file:

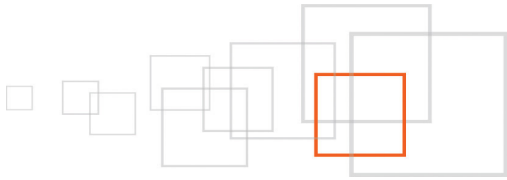
Code :

```
<?php /* #?ini charset="utf-8"?

[AliasSettings]
AliasList[]=slider

[slider]
Filters[]
Filters[]=geometry/scaleexact=445;300

*/ ?>
```

In our block's template, change the following code snippet:

Code :

```
<li>
<a href={fetch( 'content', 'node', hash( 'node_id',
$vnode.data_map.object.content.relation_list[0].node_id ) ).url_alias|ezurl}>
{attribute_view_gui image_class=large attribute=$vnode.data_map.image}
</a>
<span>{$vnode.object.data_map.message.data_text}</span>
</li>
```



To:

Code :

```
<li>
<a href={fetch( 'content', 'node', hash( 'node_id',
$vnode.data_map.object.content.relation_list[0].node_id ) ).url_alias|
ezurl}>{attribute_view_gui image_class=slider attribute=$vnode.data_map.image}
</a>
<span>{$vnode.object.data_map.message.data_text}</span>
</li>
```

To use multiple blocks in one page, you must change the id pikaname to a single value, so that we can use the variable {$block.id}, our final template code looks like this:

Code :

```
{def $valid_nodes = $block.valid_nodes}


<!-- BLOCK: START -->


<div class="pikachoose">
            <ul id="container{$block.id}" class="jcarousel-skin-pika">
            {foreach $valid_nodes as $vnode}


            <li><a href={fetch( 'content', 'node', hash( 'node_id',
$vnode.data_map.object.content.relation_list[0].node_id ) ).url_alias|
ezurl}>{attribute_view_gui image_class=slider
attribute=$vnode.data_map.image}</a><span>{$vnode.object.data_map.message.data_text}</s
pan></li>


            {/foreach}
            </ul>
</div>



{ezcss_require( 'contentslider.css' )}


{ezscript_require( array('ezjsc::jquery','jquery.jcarousel.min.js',
'jquery.pikachoose.js') )}



<script type="text/javascript">
<!--


                  $(document).ready(
                        function (){ldelim}
```

```
                                    $
("#container{$block.id}").PikaChoose({ldelim}transition:[-1]{rdelim});


                                    $("#container{$block.id}").jcarousel({ldelim}scroll:5,
wrap: 'circular',
                                                initCallback: function(carousel) {ldelim}
                                                        $
(carousel.list).find('img').click(function() {ldelim}
                                                                //console.log($
(this).parents('.jcarousel-item').attr('jcarouselindex'));
                                                                carousel.scroll(parseInt($
(this).parents('.jcarousel-item').attr('jcarouselindex')));
                                                        {rdelim});
                                                {rdelim}
                                    {rdelim});


                            {rdelim});


-->
</script>


<!-- BLOCK: END -->


{undef $valid_nodes}
```

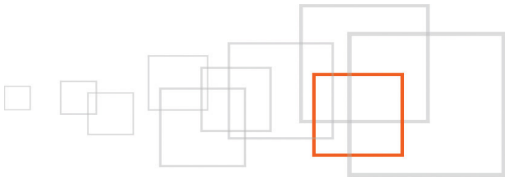
This code uses **ezjscore** functions, there's already a good tutorial about this extension, so, I'll not explain here how it works, you just need to know that it is including the css and js files.

You will also need to make some changes to the file contentslider / design / ezflow / stylesheets / contentslider.css, so it will look a little better in your site:

Code :

```
.pika-wrap {width: 520px; margin: 0 auto;}


.pika-image {position: relative; height: 300px; width: 445px; background: #fafafa;
border: 1px solid #e5e5e5; padding: 10px;}
      /*position image holders */
      .pika-image .animation, .pika-image .main-image {position: absolute; top: 10px;
left: 10px;}
      .pika-image .animation {display: none;z-index:2;}
      .pika-image img {border:0;}


.pika-image .caption {position: absolute; background: url(../images/75-black.png);
border: 1px solid #141414; font-size: 11px; color: #fafafa; padding: 10px; text-align:
right; bottom: 10px; right: 10px;}
```

```
      .pika-image .caption p {padding: 0; margin: 0; line-height: 14px;}


.pika-imgnav a {position: absolute; text-indent: -5000px; display: block;z-index:3;}

      .pika-imgnav a.previous {background: url(../images/prev.png) no-repeat left 50%;
height: 300px; width: 50px; top: 10px; left: 10px;cursor:pointer;}

      .pika-imgnav a.next {background: url(../images/next.png) no-repeat right 50%;
height: 300px; width: 50px; top: 10px; right: 10px;cursor:pointer;}

      .pika-imgnav a.play {background: url(../images/play.png) no-repeat 50% 50%;
height: 100px; width: 40px;top:0;left:50%;display: none;cursor:pointer;}

      .pika-imgnav a.pause {background: url(../images/pause.png) no-repeat 50% 50%;
height: 100px; width: 40px;top:0;left:50%;display:none;cursor:pointer;}


.pika-textnav {overflow: hidden; margin: 10px 0 0 0;}

.pika-textnav a {font-size: 12px; text-decoration: none; font-family:  helvetica,
arial, sans-serif; color: #333; padding: 4px;}

            .pika-textnav a:hover {background: #e5e5e5; color: #0065B2;}

      /*

      .pika-textnav a.previous {float: left; width: auto; display: block;}

      .pika-textnav a.next {float: right; width: auto; display: block;}

      */

      .pika-textnav a.previous, .pika-textnav a.next {display:none;}


.pika-thumbs {margin: 10px 0 0 0; padding: 0; overflow: hidden; }

      .pika-thumbs li {float: left; list-style-type: none; width: 74px; padding: 3px;
margin: 0 2px; background: #fafafa; border: 1px solid #e5e5e5; cursor: pointer;}

            .pika-thumbs li:last {margin: 0;}

            .pika-thumbs li .clip {position:relative;width: 74px; height: 74px; text-
align: center; vertical-align: center; overflow: hidden;}


.clip span{background-color:black;position:absolute;top:5px;left:5px;display:block;}

ul#pikame{width:570px;}



/* jCarousel Styles */

/*if you're not using the carousel you can delete everything below this */

.jcarousel-skin-pika .jcarousel-container-horizontal { padding: 15px 0px;}

.jcarousel-skin-pika .jcarousel-container-vertical { width: 90px; height: 350px;
padding: 20px 20px;}

.jcarousel-skin-pika .jcarousel-clip-horizontal {height: 90px; width: 465px;}

.jcarousel-skin-pika .jcarousel-clip-vertical { width:  90px; height: 350px;}

.jcarousel-skin-pika .jcarousel-item-horizontal { margin-right: 10px;}

.jcarousel-skin-pika .jcarousel-item-vertical {margin-bottom: 10px;}

.jcarousel-skin-pika .jcarousel-item-placeholder {background: #fff; color: #000;}
```



7 Conclusion

It is not hard to transform jQuery extensions into eZ publish extensions, hopefully with the help of this tutorial the process will be clear and faster for you. I tried to expose an as generic as possible method, relying on an eZ Flow block. Bear in mind that not all Jquery plug-ins are suited for transformation into an eZ Flow block, but that 100% of them can be integrated into eZ Publish.

8 Resources

- Jquery : http://jquery.com
- Jquery's Pikachoose plug-in : http://pikachoose.com
- eZ Flow configuration documentation : http://doc.ez.no/Extensions/eZ-Flow/eZ-Flow-2.2/Configuration
- Another example of Jquery usage with eZ Publish : http://share.ez.no/learn/ez-publish/encapsulating-e-mails-for-usability-and-spam-protection
- An Introduction to developing eZ Publish extensions : http://share.ez.no/learn/ez-publish/an-introduction-to-developing-ez-publish-extensions
- eZ JS Core, eZ Publish Ajax & Javascript framework, a tutorial : http://share.ez.no/learn/ez-publish/ezjscore-ez-publish-javascript-and-ajax-framework

9 About the author : Thiago Campos Viana

Thiago Campos Viana is a web developer and eZ Publish enthusiast from Brazil.

10 License choice

http://creativecommons.org/licenses/by-sa/3.0