

XML Slide Show V3

ABOUT

The XML Slide Show V3 is a flash component designed to help you load a sequence of slides (images or animations, from Library or external), in a sliding fashionable way. All these without writing any ActionScript, just write the XML file, drag the component to the stage, set it up and compile.

Compatibility

ActionScript: **2.0 only**

Flash IDE: **Flash 8, Flash CS3**

FlashPlayer: **FlashPlayer 8, FlashPlayer 9**

Features

- built from scratches based on the Jumpeye V3 Standards
- horizontal and vertical functionality
- 2 built-in animated preloaders
- 3 resize types: resize, scale, crop
- slides can have borders (customizable)
- automated slide show with interval
- script access to the contents
- true event handling implemented
- component content is loaded using XML
- component can be configured using XML

The story of the XML Slide Show V3

We wanted to help flash users to create sliding presentations of movieclips and images within flash and/or using external files also. We thought of those who would use this kind of component to speed up the development process, so we included options for borders, advanced pre-loading managing (with possibility to load all slides at the beginning or load them as you go, with adjacent priority).

The solution we came up with is as simple as a multiple advanced loader, masked by one rectangular customizable size mask that loads its content using a XML file and displays the contents in a sliding manner vertical or horizontal.

This component can be used in many ways, from showing many slides at one time, to one single slide per time, XML Slide Show V3 is simply the best slide show flash component. A slide is a movieclip, image, or external file, it may be another swf file that appears in a sliding way, or it may be a simple image, a simple swf containing a text ad, or it can be a menu item too.

Implemented methods allow the flash users to easy slide from code to a new slide, next, previous or any other slide.

Compatibility Note

XML Slide Show V3 (like all Jumpeye V3 Flash Components) is not compatible with AS3.0. However, once the AS3.0 version will be released (Jumpeye V3AS3 Release, 2007/Q3 prospect time), all AS2.0 V3 clients will receive FREE upgrades from AS2.0 to AS3.0 for their separately purchased components.

PROPERTIES

Property	Type Component Inspector	Description
xmlPath	String Yes	Optional parameter. Sets the path to a XML file that can configure the XML Slide Show V3. Parameters that are set from the <properties> node of the XML (xmlPath) will override parameters set from script or Component Inspector. The <data> node will contain the slides data, entering this node is essential for the slide show. You don't need to use xmlPath param, you can set the XML using the setXML method. Usage: componentInstance.xmlPath = "XMLSlideShowConfig.xml"
autoSlideShow	Boolean Yes	If true, the slide show will switch automatically between slides. Default: true Usage: componentInstance.autoSlideShow=false
border	Boolean Yes	If true, the slides will draw with borders. Default: true Usage: componentInstance.border = false
borderColor	Number Yes	Sets the color of the borders. Default: 0xFFFFFFFF Usage: componentInstance.borderColor = 0xFFC0CC
borderSize	Number Yes	Sets the size of the borders in pixels. Default: 3 Usage: componentInstance.borderSize = 4
builtInPreloader	String Yes	Sets the type of the built-in preloader/progress display. Values:[bar, circle, none] Default: none Usage: componentInstance.builtInPreloader="bar"
easeType	String Yes	Sets the ease type for the sliding transition tween. Values:[easeOut, easeIn, easeInOut, none] Default: easeOut Usage: componentInstance.easeType = "easeOut"
contentSize	Number Yes	Sets the width or height of the item in pixels (depending on the orientation, width on horizontal, height on vertical). Default: 250 Usage: componentInstance.contentSize = 100
loadVisibleItemsOnly	Boolean Yes	If true, the component loads only visible items in the sliding component, when an item appears to be visible it loads. This is very useful when loading many items. Default: false Usage: componentInstance.loadVisibleItemsOnly = true
orientation	String Yes	Sets the orientation of the slider, the invisible support that displays the slides. Values:[horizontal,vertical] Default:horizontal Usage: componentInstance.orientation = "horizontal"
preloaderColor	Number Yes	Sets the color of the built-in preloader and percentage. Default: 0xFFFFFFFF Usage: componentInstance.preloaderColor

resizeType	String Yes	Sets the type of the resize applied to the content of the slides. Values:[resize, scale, crop] Default: resize Usage: componentInstance.resizeType = "scale"
slideShowInterval	Number Yes	Sets the interval (in seconds) before the component will switch to another primary image. Default: 3 Usage: componentInstance.slideShowInterval = 5
spacing	Number Yes	Sets the space between items (slides). Default: 5 Usage: componentInstance.spacing = 2
tweening	Boolean Yes	If false, the switching between the items will be done without sliding. Default: true Usage: componentInstance.tweening = false
tweeningDuration	Number Yes	Sets the duration of the slide tweening in frames. Default: 20 Usage: componentInstance.tweeningDuration = 10
tweenType	String Yes	Sets the tweening type. Values:[Linear, Strong, Back, Elastic, Bounce] Default: Strong Usage: componentInstance.tweenType = "Back"
width	Number No	Used to resize the component's width in runtime without affecting the _xscale of the component. The component will than align and resize (if needed) the content in order to show the new size. Usage: componentInstance.width=200
height	Number No	Used to resize the component's height in runtime without affecting the _yscale of the component. The component will than align and resize (if needed) the content in order to show the new size. Usage: componentInstance.height=120

EVENTS

Event	Description
onLoadXML	Event is triggered after the XML is loaded, this happens before loading the content. Usage: myEventListener = new Object(); myEventListener.onLoadXML = function(args){ trace("onLoadXML "+ args.target + " : " + args.success); } componentInstance.addEventListener("onLoadXML",myEventListener);
onLoad	Event is triggered after the content is loaded. Usage: myEventListener = new Object(); myEventListener.onLoad = function(args){ } componentInstance.addEventListener("onLoad",myEventListener);
onDrawComplete	Event is triggered after the content has been visually drawn. Usage: myEventListener = new Object(); myEventListener.onDrawComplete = function(args){ } componentInstance.addEventListener("onDrawComplete",myEventListener);
onRelease	Event is triggered when "onRelease" event is invoked on one item. Usage: myEventListener = new Object(); myEventListener.onRelease= function(args){ trace("onRelease event on item: " + args.item); } componentInstance.addEventListener("onRelease",myEventListener);
onPress	Event is triggered when "onPress" event is invoked on one item. Usage:

	<pre> myEventListener = new Object(); myEventListener.onPress= function(args){ trace("onPress event on item: " + args.item); } componentInstance.addEventListener("onPress",myEventListener); </pre>
onRollOver	Event is triggered when "onRollOver" event is invoked on one item. Usage: <pre> myEventListener = new Object(); myEventListener.onRollOver = function(args){ trace("onRollOver event on item: " + args.item); } componentInstance.addEventListener("onRollOver",myEventListener); </pre>
onRollOut	Event is triggered when "onRollOut" event is invoked on one item. Usage: <pre> myEventListener = new Object(); myEventListener.onRollOut = function(args){ trace("onRollOut event on item: " + args.item); } componentInstance.addEventListener("onRollOut",myEventListener); </pre>
onReleaseOutside	Event is triggered when "onReleaseOutside" event is invoked on one item. Usage: <pre> myEventListener = new Object(); myEventListener.onReleaseOutside = function(args){ trace("onReleaseOutside event on item: " + args.item); } componentInstance.addEventListener("onReleaseOutside",myEventListener); </pre>
onDragOut	Event is triggered when "onDragOut" event is invoked on one item. Usage: <pre> myEventListener = new Object(); myEventListener.onDragOut = function(args){ trace("onDragOut event on item: " + args.item); } componentInstance.addEventListener("onDragOut",myEventListener); </pre>
onChange	Event is triggered when "onChange" event is invoked on one item. Usage: <pre> myEventListener = new Object(); myEventListener.onChange = function(args){ trace("onChange event on item: " + args.item); } componentInstance.addEventListener("onChange",myEventListener); </pre>
onMotionFinished	Event is triggered when the tweening is over. Usage: <pre> myEventListener = new Object(); myEventListener.onMotionFinished = function(args){ trace("onMotionFinished event on item: " + args.item); } componentInstance.addEventListener("onMotionFinished",myEventListener); </pre>
onProgress	Event is triggered when the items are loaded into the component Usage: <pre> myEventListener = new Object(); myEventListener.onProgress= function(args){ trace(args.numberOfItemsLoaded + ":" + args.numberOfTotalItems); } componentInstance.addEventListener("onProgress",myEventListener); </pre>

METHODS

Method	Description
nextImage	Switches to the next image (slide). Usage: <pre>componentInstance.nextImage()</pre>
previousImage	Switches to the previous image (slide). Usage: <pre>componentInstance.previousImage()</pre>

setSize	Sets the size of the Slide Show Pro V3 component in real time. Syntax: setSize(width:Number,height:Number):Void Usage: componentInstance.setSize(120,200)
startSlideShow	Starts the automated sliding show. Usage: componentInstance.startSlideShow()
stopSlideShow	Stops the automated sliding show. Usage: componentInstance.stopSlideShow()
setXML	This method is used to set the XML of a component in real time. Usage: componentInstance.setXML(myXML)
getXML	This method is used to get the XML of a component in real time. Returns: XML object Usage: var myXML:XML = componentInstance.getXML()
get	This method is used to get the MovieClip reference to a slide by item id. Returns: MovieClip Usage: var itemMc:MovieClip = componentInstance.get(itemId)
getByProperty	This method is used to get the MovieClip reference to a slide by a property (attribute) found in the XML node of the item. This property can be either user-defined or a predefined. Returns: MovieClip Usage: var itemMc:MovieClip = componentInstance.getByProperty(myProp,myValue)
slideTo	This method allows you to slide to a specific item, by item id. Usage: componentInstance.slideTo(itemId)
slideToItemByProperty	This method allows you to slide to a specific item, by a property. Usage: componentInstance.slideToItemByProperty(myProp,myValue)