

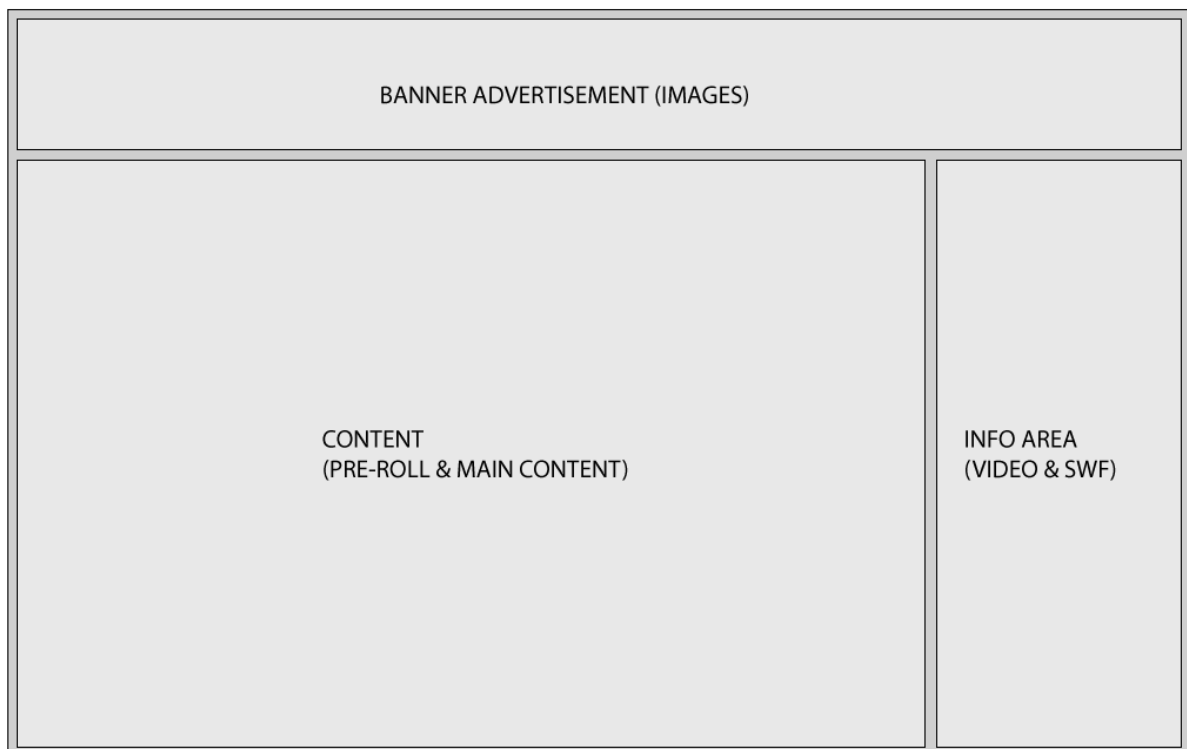


OSMF Release Samples

Walkthrough 7: Working With Container Layouts

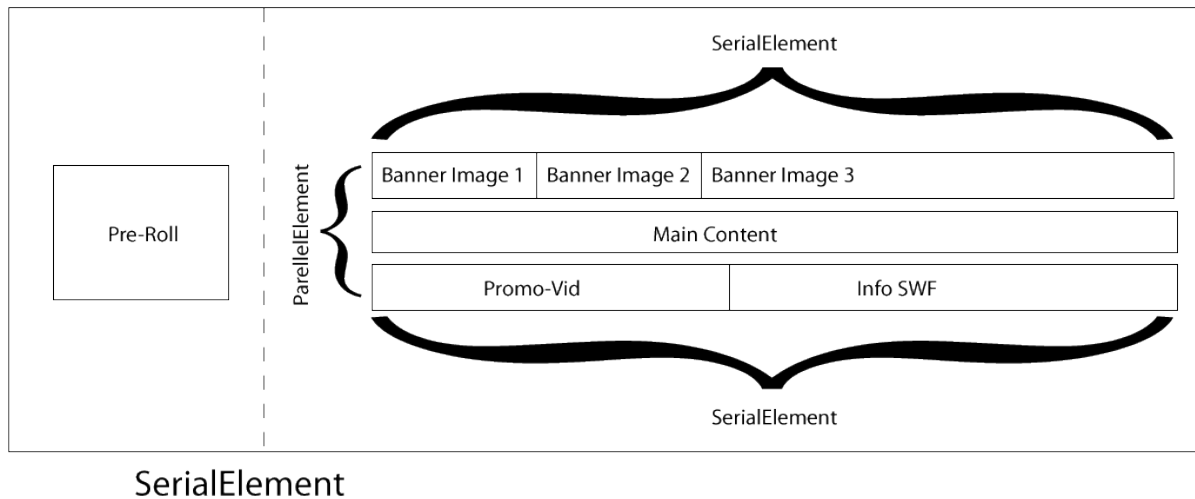
Overview:

The purpose of this walkthrough is to get a solid understanding of real-world applications of composition based media elements, and how to display them via the layout features of the OSMF. This walkthrough builds on-top of the previous examples and your knowledge of `SerialElement` and `ParallelElement`, as well as basic layout meta-data. The following diagram shows the layout that this example uses, and is achieved by using multiple `LayoutContainers` that are dynamically positioned by a primary linked `LayoutContainer`, and adding the targeted containers as layout targets of the primary container's `layoutRenderer` (exp: `mainContainer.layoutRenderer.addTarget(targetContainer)`).



Once a container has been added as a target for another container that will manage its position, the meta-data and basic layout properties of the target container will be used by the managing container to determine its size and/or position.

The diagram below illustrates the sequence and general structure of the complex composition this example will create.



The base of the composition is a serial element. It starts with a simple pre-roll video, then the second element of the composition is a ParallelElement - thus playing multiple MediaElements at the same time after the pre-roll. The parallel elements consist of another SerialElement for the top banners, which will display 3 images over a period of time, the main content, which will run for about 5 minutes, and a right-side info area, which is another SerialElement, and contains a video and a SWF.

An important item of note is when displaying images by default they do not have a set duration, and will display until manually changed. This can make it difficult to use them in a SerialElement, for instance, since it will not know when to advance. A simple solution to this is to wrap the ImageElement into a DurationElement. The DurationElement is a useful class for taking an existing MediaElement and applying a manual duration to it. The DurationElement constructor accepts a defined duration in seconds, and a MediaElement to apply it to.

Objectives:

- Create a real-world style advertising layout
- Compose complex media experience with nested serial and parallel compositions
- Understand the use of the DurationElement
- Use container properties and layout metadata to manage the display of the containers relative to each other
 - Understand the basics of layout renderer's and layout targets

Setup

1. Open the file WT07_ContainerLayout.as in the {SAMPLES_PROJECT}/src directory.

NOTE: This file has been provided as a starting point for these walkthroughs.

2. Set the class file as the application file to compile. There are two different ways of doing this depending on which program you are building your application in.

Flash Builder

Right-click the WT07_ContainerLayout.as file and select Set as Default Application from the context menu that appears. This will add the project to the list of compilable applications. A blue dot on the file icon indicates that the file is the default application file.

Flash Professional

Open the OSMF_SampleTemplate.fla and save it as WT07_ContainerLayout.fla.

Then change the document class for the file (in the Properties panel) to WT07_ContainerLayout.

Create the Main Composition

The first task is to create the main SerialElement that will contain a pre-roll clip. Then, play a ParallelElement that will have the main content and the advertising/info serial clips in unison.

3. Locate the comment that begins "//Marker 1:" in the initPlayer() method.
4. Create a local SerialElement variable named rootElement and set it equal to a new SerialElement object.

```
//Marker 1: The root element  
var rootElement:SerialElement = new SerialElement();
```

5. Create a local var named preRollElement of type MediaElement and set it equal to a generated a MediaElement by using the mediaFactory instance and the PRE_ROLL static const already defined.

```
var preRollElement:MediaElement = mediaFactory.createMediaElement(  
new URLResource( PRE_ROLL ) );
```

6. Create a local var called parallelComp of type ParallelComp and instantiate it accordingly.

```
var parallelComp:ParallelElement = new ParallelElement();
```

7. Add both the preRollElement and the parallelComp to the rootElement to create a sequence of the pre-roll then the parallel composition.

```
rootElement.addChild( preRollElement );  
rootElement.addChild( parallelComp );
```

Create the Parallel Composition

8. Under the comment that begins "//Marker 2:" create a new MediaElement named mainElement and use the mediaFactory to create a new MediaElement with the STREAMING_MP4_PATH static const variable.
9. Add the mainElement as a child of the rootElement.

```
//Marker 2: Main video element  
var mainElement:MediaElement = mediaFactory.createMediaElement( new  
URLResource( STREAMING_MP4_PATH ) );  
rootElement.addChild( mainElement );
```

10. Under the comment that begins "//Marker 3:" create a new SerialElement named topBannerElement.

```
//Marker 3: Top banner - serial element consisting of multiple  
images
```

```
var topBannerElement:SerialElement = new SerialElement();
```

11. Create two DurationElements using the TOP_BANNER_1 and TOP_BANNER_2 static constant variables named imageBanner1 and imageBanner2. DurationElements allow you to specify a time for a MediaElement that by default may not have a duration, such as an image.

```
//Marker 3: Top banner - serial element consisting of multiple
images
var topBannerElement:SerialElement = new SerialElement();
var imageBanner1:DurationElement = new DurationElement(
BANNER_DURATION, mediaFactory.createMediaElement( new URLResource(
TOP_BANNER_1 ) ) );
var imageBanner2:DurationElement = new DurationElement(
BANNER_DURATION, mediaFactory.createMediaElement( new URLResource(
TOP_BANNER_2 ) ) );
```

12. Create a new MediaElement variable named imageBanner3 using the static constant TOP_BANNER_3.

```
var imageBanner2:DurationElement = new DurationElement(
BANNER_DURATION, mediaFactory.createMediaElement( new URLResource(
TOP_BANNER_2 ) ) );
var imageBanner3:MediaElement = mediaFactory.createMediaElement( new
URLResource( TOP_BANNER_3 ) );
```

13. Add each imageBanner variable as a child of the topBannerElement.

```
var imageBanner3:MediaElement = mediaFactory.createMediaElement( new
URLResource( TOP_BANNER_3 ) );
topBannerElement.addChild( imageBanner1 );
topBannerElement.addChild( imageBanner2 );
topBannerElement.addChild( imageBanner3 );
```

14. Add the topBannerElement as a child of the pallelComp.

```
topBannerElement.addChild( imageBanner1 );
topBannerElement.addChild( imageBanner2 );
topBannerElement.addChild( imageBanner3 );
pallelComp.addChild( topBannerElement );
```

15. Under the comment that begins "//Marker 4:" create a SerialElement named rightBannerElement.

```
//Marker 4: Right banner element
var rightBannerElement:SerialElement = new SerialElement();
```

16. Add two children directly to the rightBannerElement using the mediaFactory and the static constant variables RIGHT_BANNER_1 and RIGHT_BANNER_2.

```
var rightBannerElement:SerialElement = new SerialElement();
rightBannerElement.addChild( mediaFactory.createMediaElement( new
URLResource( RIGHT_BANNER_1 ) ) );
rightBannerElement.addChild( mediaFactory.createMediaElement( new
```

```
URLResource( RIGHT_BANNER_2 ) ) );
```

17. Add the rightBannerElement as a child of the pallelComp.

```
rightBannerElement.addChild( mediaFactory.createMediaElement( new  
URLResource( RIGHT_BANNER_1 ) ) );  
rightBannerElement.addChild( mediaFactory.createMediaElement( new  
URLResource( RIGHT_BANNER_2 ) ) );  
pallelComp.addChild( rightBannerElement );
```

Creating the Containers

Working with containers and the advanced layout capabilities can be tricky. Generally speaking it is best to apply the primary layout constraints to the container and let them automatically handle the layout of the elements. You can also apply layout meta data to the elements, but you must take into consideration the constraints of its parent container. By using both the direct properties of the MediaContainer and the properties available on the layoutMetadata of the MediaContainer, both positioning of the child elements of the containers, and the relative positioning of the containers to other layout targets or containers, can easily be achieved.

18. Under the comment that begins "//Marker 5:", create a MediaContainer variable named rootContainer.

```
//Marker 5: Root container  
var rootContainer:MediaContainer = new MediaContainer();
```

19. Set the following properties and values on the rootContainer object.
 1. width: 960
 2. height: 488
20. Add the rootContainer as a child of the class.

```
var rootContainer:MediaContainer = new MediaContainer();  
rootContainer.width = 960;  
rootContainer.height = 488;  
addChild( rootContainer );
```

21. Under the comment that begins "//Marker 6:" create a new MediaContainer named topContainer then set the following properties and values on the topContainer object.
 1. backgroundColor: 0x000000
 2. backgroundAlpha: 1
22. Set the percentWidth property of the topContainer's layoutMetadata property to 100
23. Set the height property of the topContainer's layoutMetadata property to 60.
24. Add the topContainer as a child of the class.

```
//Marker 6: Top container  
var topContainer:MediaContainer = new MediaContainer();  
topContainer.backgroundColor = 0x666666;  
topContainer.backgroundAlpha = .2;
```

```
topContainer.layoutMetadata.percentWidth = 100;
topContainer.layoutMetadata.height = 60;
addChild( topContainer );
```

25. Under the comment that begins "//Marker 7:" create a new MediaContainer named rightContainer then set the following properties and values on the rightContainer object.
 1. width: 192
 2. height: 428
 3. backgroundColor: 0x000000
 4. backgroundAlpha: 1
26. Set the horizontalAlign property of the rightContainer's layoutMetadata property to HorizontalAlign.RIGHT.
27. Set the verticalAlign property of the rightContainer's layoutMetadata property to VerticalAlign.BOTTOM.
28. Add the rightContainer as a child of the class.

```
//Marker 7: Right container
var rightContainer:MediaContainer = new MediaContainer();
rightContainer.width = 192;
rightContainer.height = 428;
rightContainer.backgroundColor = 0x000000;
rightContainer.backgroundAlpha = 1;
rightContainer.layoutMetadata.horizontalAlign =
HorizontalAlign.RIGHT;
rightContainer.layoutMetadata.verticalAlign = VerticalAlign.BOTTOM;
addChild( rightContainer );
```

29. Under the comment that begins "//Marker 8:", create a new MediaContainer named mainContainer.
30. Set the following properties and values on the mainContainer object.
 1. backgroundColor: 0x999999
 2. backgroundAlpha: 1
 3. width: 768
 4. height: 428
 5. layoutMetadata.verticalAlign: VerticalAlign.BOTTOM
31. Add the mainContainer as a child of the class

```
//Marker 8: Main content container
var mainContainer:MediaContainer = new MediaContainer();
mainContainer.backgroundColor = 0x999999;
mainContainer.backgroundAlpha = 1;
mainContainer.width = 768;
mainContainer.height = 428;
mainContainer.layoutMetadata.verticalAlign = VerticalAlign.BOTTOM;
addChild( mainContainer );
```

Adding the MediaElements and Layout Targets

In this section, the media elements will be linked with their respective containers so they are directed where to properly display, and the containers will be linked to a single master container that will manage their relative layouts based on the layout data applied earlier.

32. Under the comment that begins "//Marker 9:" add the rootElement object as a MediaElement of the mainContainer by calling the addMediaElement() method of the mainContainer passing the rootElement as the only parameter.

NOTE: This is a critical step. This defines which content is displayed in which container.

```
//Marker 9: Give the containers their media elements  
mainContainer.addMediaElement( rootElement );
```

33. Add the topContainer object as a MediaElement of the topContainer.

34. Add the rightBanner object as a MediaElement of the rightContainer.

```
//Marker 9: Give the containers their media elements  
mainContainer.addMediaElement( rootElement );  
topContainer.addMediaElement( topBannerElement );  
rightContainer.addMediaElement( rightBannerElement );
```

35. Under the comment that begins "//Marker 10:", Call the addtarget() method on the layoutRenderer property of the rootContainer passing it the rightContainer object to add it as a layout target of the rootContainer. This links the content containers to the root container and allows the root container to dynamically position the content containers relative to each other based on their layout data.

```
//Marker 10: Set the layout targets for the rootContainer  
rootContainer.layoutRenderer.addTarget( rightContainer );
```

36. Add the topContainer as a layout target of the rootContainer.

37. Add the mainContainer as a layout target of the rootContainer.

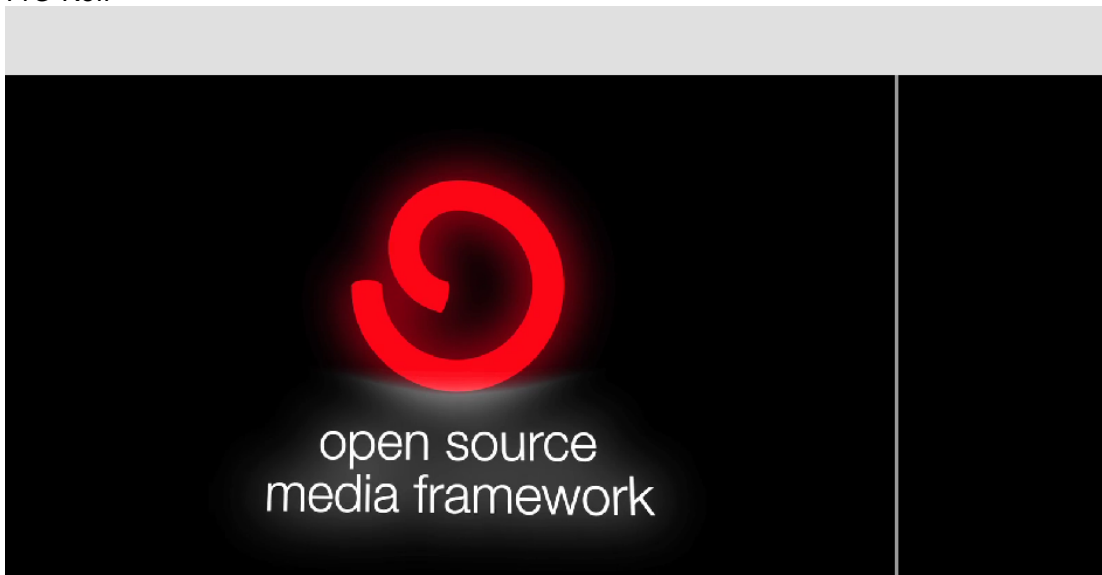
```
//Marker 10: Set the layout targets for the rootContainer  
rootContainer.layoutRenderer.addTarget( rightContainer );  
rootContainer.layoutRenderer.addTarget( topContainer );  
rootContainer.layoutRenderer.addTarget( mainContainer );
```

38. Save the file and run the application. You should see the player layout with 3 main areas: a top bar with banner images, a right bar with a video, then a swf and the main video display area.

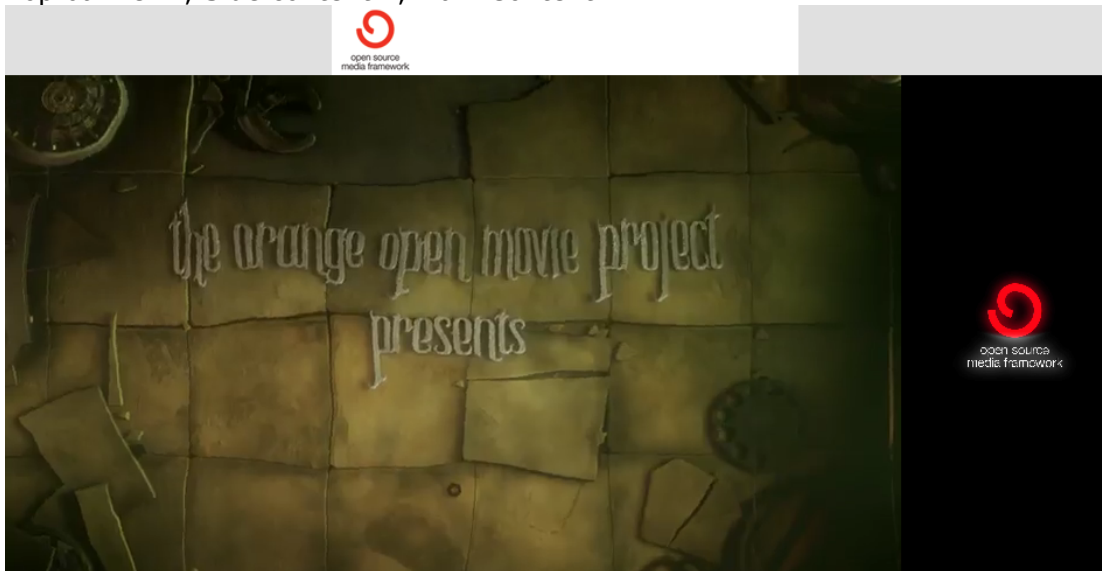
The top bar uses a SerialElement to cycle through 3 images. The first two were assigned a duration and the last should show for the remaining time of the video.

The right bar displays the OSMF logo video. When the logo video is completed, a SWF is loaded.

Pre-Roll



Top banner 1, Side content 1, Main Content

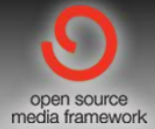
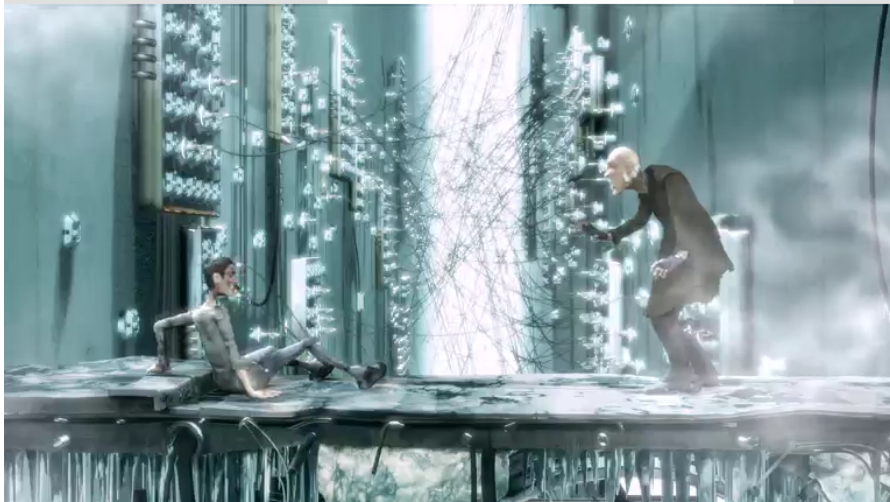


Top banner 2, Side content 2, Main Content



OSMF reduces the complexity of player development, allowing the developer more time to focus on the overall user experience. OSMF's flexible architecture allows the developer to easily customize their player for the browser, incorporating plug-ins for advertising, reporting, and content delivery along with standard player features such as play/pause, seek, volume/mute, download progress, buffering, and bitrate switching.

Top banner 3, Side content 2, Main Content



OSMF reduces the complexity of player development, allowing the developer more time to focus on the overall user experience. OSMF's flexible architecture allows the developer to easily customize their player for the browser, incorporating plug-ins for advertising, reporting, and content delivery along with standard player features such as play/pause, seek, volume/mute, download progress, buffering, and bitrate switching.